Distribution Category:
Mathematics and Computer
Science (UC-405)

ANL--88-10

DE89 001324

ANL-88-10

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439-4801

Searching for Fixed Point Combinators by Using Automated Theorem Proving: A Preliminary Report

by

Larry Wos and William McCune

Mathematics and Computer Science Division

September 1988

This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Contents

Searching for Fixed Point Combinators by Using Automated Theorem Proving: A Preliminary Report

Larry Wos and William McCune

Abstract

In this report, we establish that the use of an automated theorem-proving program to study deep questions from mathematics and logic is indeed an excellent move. Among such problems, we focus mainly on that concerning the construction of fixed point combinators—a problem considered by logicians to be significant and difficult to solve, and often computationally intensive and arduous. To be a fixed point combinator, Θ must satisfy the equation $\Theta x = x(\Theta x)$ for all combinators x. The specific questions on which we focus most heavily ask, for each chosen set of combinators, whether a fixed point combinator can be constructed from the members of that set.

For answering questions of this type, we present a new, sound, and efficient method, called the *kernel method*, which can be applied quite easily by hand and very easily by an automated theorem-proving program. For the application of the kernel method by a theorem-proving program, we illustrate the vital role that is played by both paramodulation and demodulation—two of the powerful features frequently offered by an automated theorem-proving program for treating equality as if it is "understood". We also state a conjecture that, if proved, establishes the completeness of the kernel method. From what we can ascertain, this method—which relies on the introduced concepts of *kernel* and *superkernel*—offers the first systematic approach for searching for fixed point combinators.

We successfully apply the new kernel method to various sets of combinators and, for the set consisting of the combinators B and W, construct an infinite set of fixed point combinators such that no two of the combinators are equal even in the presence of extensionality—a law that asserts that two combinators are equal if they behave the same. This result sharply extends the earlier success obtained by the logician Statman in February 1986. Perhaps more significant, we also apply the new method to yield a distinctly different solution to the problem of constructing fixed point combinators from B and one other regular combinator, a solution that involves the combinator N, with $((Nx)y)z = ((xz)y)z$, rather than the combinator W. The combinator N apparently has not been studied before. To further establish the value of automated theorem proving for solving problems from combinatory logic, we adapt various techniques from automated theorem proving to answer two previously open questions posed by the logician Smullyan.

1. Introduction

"A formalized proof can be checked mechanically but cannot be mechanically discovered." The preceding quote is taken from a paper by the outstanding Polish logician Lukasiewicz, a paper written in 1948 [Lukasiewicz48]. Obviously—and most fortunately—as we repeatedly demonstrate in this report, he was mistaken. That such a famous logician held this position is completely understandable; indeed, to have believed otherwise in 1948 might have required clairvoyance. To predict with great accuracy how much progress will occur and what power will be available 40 years in the future is simply an awesome task.

Indeed, for a striking example of what can occur, 40 years after the original pronouncement, we have its refutation—not only do computer programs "discover" proofs, but often the discovery occurs in astoundingly little computer time. Even more satisfying, occasionally a proof found by a computer program answers a deep question. If the number of successes achieved with the assistance of a computer program continues to increase at the rate evidenced by the preceding ten years, perhaps ten years from now, many of the currently open questions will have been attacked and answered—or be under attack—by a team consisting of a scientist and a computer program. As many already know—and as many will learn from this report—the key to these successes in a substantial number of cases is automated theorem proving.

Automated theorem proving is a relatively young and challenging field concerned with computer programs that *reason* logically. Such programs, called "automated theorem-proving programs", can be used to prove theorems from mathematics, design logic circuits, verify computer code, solve puzzles, and assist in many other activities that require reasoning. Our general objective in this report is to establish the value of automated theorem proving for research focusing on deep questions from mathematics and logic. As evidence, we present a systematic and uniform method—called the *kernel method*—for attacking a set of hard problems from combinatory logic. The main objective of this new method is to search for *fixed point combinators*. The kernel method relies heavily on the use of an automated theorem-proving program and heavily on the concepts of *kernel*, *superkernel*, and methodology introduced in this report. Although we focus on the specific field of combinatory logic, the material presented here illustrates—even for those not interested in this field—how research objectives were identified, pursued, and eventually reached. For example, we show how research can and did lead to the discovery of the new and significant concepts just mentioned, and how further experimentation led to the appropriate refinements of those concepts.

To complement our general objective of establishing the value of automated theorem proving for research, our specific objective is to discuss in detail and then apply the new systematic kernel method to search for fixed point combinators. By definition, a combinator Θ is a fixed point combinator if $\Theta x = x(\Theta x)$ for all combinators x. Fixed point combinators are of especial interest in combinatory logic because of their relevance to various paradoxes. Of such paradoxes, perhaps the best known is the Russell Paradox which asks one to consider the set S whose members are all sets that are *not* members of themselves; if S is a member of itself, then S is *not* a member of itself, and if S is *not* a member of itself, then S is a member of itself. The Russell Paradox and other paradoxes can in fact be formulated in terms of fixed point combinators; indeed, such combinators were known as paradoxical combinators in the early days of

combinatory logic. However, the interest in fixed point combinators is not confined to their relation to paradoxes. The interest in such combinators also rests in part with the fact that Gödel's self-referential sentence and Kleene's recursion theorem can be interpreted as applications of fixed point combinators [Barendregt81].

In this report, we shall give numerous examples of fixed point combinators and of the concepts of kernel and superkernel which play such a vital role in the new kernel method. From what we can ascertain, the kernel method is the first of its kind; no other method exists for systematically constructing (when they exist) fixed point combinators when given some arbitrarily chosen set of combinators. The value of this new method rests in part with the fact that logicians consider the search for fixed point combinators often to require arduous and intensive computation, which contrasts sharply with the ease and efficiency offered to a person or a theorem-proving program by the kernel method. The value of this new method also rests in part with the fact that, even when the set of combinators under consideration offers sufficient power for the desired construction, a search for fixed point combinators can easily meet with failure if one does not rely on the kernel method.

To demonstrate the power of the kernel method, we apply it first to the successful construction (from the combinators B and W alone, defined in Section 1.3) of an infinite set of fixed point combinators such that no two of the combinators are equal even in the presence of extensionality. This result sharply extends an earlier success obtained by the logician Statman in February 1986 [Statman86]. The existence of this infinite set is indeed interesting and unexpected, for—until Statman solved the problem—the question concerning the existence of a single regular combinator to be used with B for the construction of a fixed point combinator was still open. Even further, one can imagine how startled we were when we succeeded in later applying our kernel method to the construction (from B and W alone) of an infinite class of infinite sets of fixed point combinators.

Perhaps more significant—and still later in our research—we applied the new method to finding a distinctly different solution to the problem of constructing fixed point combinators from B and one other regular combinator, a solution that does not involve the combinator W. This second solution relies on the combinator N with

$$(\forall x \forall y \forall z)\ ((Nx)y)z = ((xz)y)z.$$

The combinator N apparently has not been studied before—at least, our preliminary examination of the literature indicates that this is the case. In addition to this second solution, we have obtained solutions that depend on H rather than on N as a replacement for W, and that depend on other regular combinators in three variables to replace W.

To further illustrate how valuable automated theorem proving can be for studying aspects of combinatory logic, we adapt some of the techniques from automated theorem proving to answer the following two open questions posed by the logician Smullyan [Smullyan87]. First, does the set consisting of the combinators B and L alone satisfy the strong fixed point property (defined in Section 1.3)? Second, does the set consisting of the combinators L and Q alone satisfy that property?

$$(\forall x \forall y \forall z)\ ((Bx)y)z = x(yz)$$
$$(\forall x \forall y)\ (Lx)y = x(yy)$$

$(\forall x \forall y \forall z)\ ((Qx)y)z = y(xz)$

We also use techniques adapted from automated theorem proving to give a simple proof of a theorem Statman proved earlier [Statman86] in response to yet another question posed by Smullyan [Smullyan84]. The theorem asserts that no single regular combinator can, by itself, satisfy the strong fixed point property.

1.1. Significance for the Bigger Picture

The material contained in this report represents the realization of a vision, a vision shared for many years by researchers in the field of automated theorem proving. Simply stated, the vision concerns finding some specific area of mathematics or logic that is very amenable to attack with an automated theorem-proving program. For an area or set of problems to be correctly considered as very amenable to attack, one essential property that must be present concerns the likelihood of success. In particular, when given a randomly selected question from the specified area of mathematics or logic, the theorem-proving program should succeed in answering that question at least half of the time. Of course, answers must be returned in no more than a few CPU hours of the computer's time, and not in CPU days. Of especial satisfaction, therefore, is the fact that the kernel method, when applied by our theorem-proving program OTTER (Other Theorem-proving Techniques for Effective Research), returns the sought-after answers to the selected questions more than 70% of the time, and returns them usually within a few CPU seconds on a Sun 3 workstation. The randomly selected questions are taken from combinatory logic, and focus on the presence or absence of fixed point properties (defined in Section 1.3).

The next essential property that must be present—for the specified area to be correctly considered as very amenable to attack—focuses on the power of the automated theorem-proving program versus the power of the mathematician or logician. In particular, if viewed as a contest—in the ideal case—the theorem-proving program should win at least 80% of the time. Again we can report some early success; in the few examples we have tried, when OTTER applies the kernel method to a question about fixed point properties possessed by a set of combinators selected by some person in an audience of experts, in almost all cases OTTER answers the question far more quickly than does a logician. Were a lengthy contest held and won by a theorem-proving program applying the kernel method—rather than in any way denigrating the skills of a mathematician or logician—we would simply have the desired evidence that one vision of researchers in automated theorem proving had finally been realized.

The third essential property that must be present concerns the speed of execution, more specifically, the source of that speed. For example, no doubt exists concerning the ability of a computer to dwarf the performance of a person if the task that is assigned is simply arithmetic; the source of such increased power rests with the fact that arithmetic tasks are ordinarily algorithmic. In contrast, until now, the theorem-proving community has not had an example of consistent high performance when the assigned task or set of tasks depends on logical reasoning—reasoning that is clearly *not* algorithmic in nature. Again we can report satisfaction since the important fact is that the kernel method is *not* an algorithm for constructing fixed point combinators; instead, the method provides a systematic approach for reasoning about their construction. In other words, when the program OTTER applies the kernel method to a given question, in no way is the program imitating what occurs for arithmetic problems or the like—in no way

is the program applying an algorithm. Instead, the kernel method can correctly be thought of as a global strategy for searching for fixed point combinators, a strategy that sharply restricts the search that might ordinarily be made.

We are not limited to the discussion of fixed point properties when our goal is to provide evidence supporting our view that the vision of the researcher in automated theorem proving has now been realized. Indeed, as further evidence—when we complete the book that will contain the material in this report as a proper subset—we shall include material focusing on the use of an automated theorem-proving program for attempting to construct a specific type of combinator from a given set of combinators. In particular, for any chosen set of combinators, the object is to determine whether one can produce an expression from the members of that set such that the expression behaves identically to some given combinator not in the set under consideration.

Since we have completed our brief and not totally precise discussion of why the results presented in this report are significant for the big picture—when the emphasis is on automated theorem proving—we now turn to a discussion of the significance of those results for a second big picture, one focusing on combinatory logic. Such a second discussion—even if it appeals only to one's intuition—is particularly useful when a report or book focuses on fields that are apparently unrelated; the fields of automated theorem proving and combinatory logic, which are central to this report, certainly appear to be unrelated. Of course, since the kernel method can be readily applied by an automated theorem-proving program to answer—often with startling ease—deep questions from combinatory logic, from our viewpoint, the two fields share an important connection.

Indeed, for the bigger picture as viewed from the perspective of combinatory logic, the significance of our results rests with the suggestion that a researcher's chance of succeeding can be substantially increased when the researcher relies on the assistance of an automated theorem-proving program. Of course, the entire picture would be incomplete if we did not touch on the importance, to the rest of science, of combinatory logic. Therefore, although the precise meaning of the relevant concepts must wait its turn, let us turn to the importance and usefulness of combinatory logic.

If all goes as some researchers intend, perhaps the most important use of combinatory logic will be for computer programming. Just as one can use a Turing machine as a programming language, one can also use combinatory logic for that purpose. Of course, for serious or efficient programming by a person rather than by a computer, neither is a particularly good choice; indeed, were one to program in either language, the experience would be tedious and painful. However, well-known and well-respected researchers, such as David Turner, are most interested in programming languages that compile code into combinators—combinators that are the machine instructions executed by the computer. In fact, such computers have been designed and built. In the context of compiled code, the use of an automated theorem-proving program to produce one type of combinator from a given set of combinators might be of substantial value; by contacting the appropriate researchers, we are attempting to determine how valuable, and will report what we learn in the book we mentioned earlier. To provide a more complete background, we shall also include in that book examples of computer programs written in terms of combinators.

With regard to the potential and general interest in the kernel method and fixed point combinators with various properties, the significance of such combinators rests with their relevance to various

paradoxes. Therefore, perhaps the kernel method can be put to good use in some context that focuses on paradoxical statements. Rather than attempting to clarify and amplify this remark and preceding remarks—since this report is accurately labeled "preliminary"—we shall simply, and rather abruptly, turn to other matters, leaving the added material to be included in the proposed book. However, we note that combinatory logic in general is of interest because every effectively computable function can be obtained with this logic, and every recursive function is an effectively computable function.

1.2. A Map

At this point, we in effect give a map to aid one in deciding how to read this long report. Since the audience for whom this report may be of interest is diverse, we include a review of the pertinent aspects of automated theorem proving and a short introduction to combinatory logic. In particular, especially for those unfamiliar with theorem-proving programs, in Section 2 we review (by example) problem representation, the inference rule that was used for making deductions, various strategies employed for controlling the search for interesting deductions, a procedure for canonicalization, a mechanism for discarding trivial corollaries, and a test for assignment completion.

In the introduction to combinatory logic given in Section 3, we focus on an intuitive understanding of combinators, appropriate definitions, and a variety of questions that can be attacked with the assistance of an automated theorem-proving program. However, one should not expect a deep and thorough treatment of combinatory logic; in fact, we discuss only those aspects that are pertinent to our research. Nevertheless, we do include sufficient material to tantalize and intrigue one.

With regard to combinatory logic, we cannot include much more than we do, for we are at best novices in this field; our expertise lies in the area of automated theorem proving. We also note that, since this is a preliminary report, the definitions and notation found here are subject to slight modification; in particular, we may make a number of small changes for our final report on this study, that which we expect will appear as a book within a year. In addition, certain promises we make here concerning results and observations may be only incompletely fulfilled; for the full story, we recommend the final report on this study.

With both our general and our specific objectives in mind, we give in Section 4 a chronological account of the role that an automated theorem-proving program played in our research, the research that culminated in the formulation of the systematic kernel method, so central to this report. By giving a full account of our study, we intend to provide—especially for those who are new to research—valuable clues to be used in various other studies, many of which will be totally unrelated to combinatory logic. We shall, therefore, include comments explaining why certain decisions were made and why certain paths of inquiry were pursued. Although such accounts typically do not include material focusing on research that did not succeed, we shall nevertheless discuss why various attempts to answer a given question failed, and why various answers that appear promising fail to fulfill their promise.

In addition, our account will show that many of the questions in this area of logic can readily be viewed as intriguing puzzles (see Section 1.3)—puzzles that one can attempt to solve knowing relatively little about mathematics or logic. Despite the fact that many questions from combinatory logic—some of which we answer in this report—can be studied without much expertise, their answers may indeed prove

significant.

Rather than learning about the kernel method as we did, which we chronicle in Sections 4.1 through 4.4, one may instead wish to immediately see how the kernel method works. If that is the case, one can turn to Section 4.5 where we present a full treatment of the systematic kernel method for searching for fixed point combinators. If one chooses this path, then the earlier material can be used as reference material, providing examples and illustrations of the various aspects on which we focus in Section 4.5 and in its subsections. However, to learn about how research can occur—at least, how it did for us—one must read Section 4 from its beginning.

If one prefers a more rapid treatment (than that given in Section 4) of what resulted from our research, one can turn to Section 5, entitled "Highlights", where we give the most significant results of our research. That section contains three major subsections, focusing, respectively, on answers to previously open questions and related material, on various unanswered questions that include some arising directly from our research and that merit further research, and on conjectures that might be challenging to study.

Even further, if one wishes just the headlines describing our new results, then Section 5.1, which is currently entitled "Summary of Highlights", is the recommendation. Perhaps, in the spirit of poking fun at ourselves, we should have called Section 5.1 "Highlights of Highlights", especially since Section 5 as a whole is rather lengthy for a section called "Highlights".

Finally, because of the nature of the various choices for reading this report, one should expect to encounter a fair amount of repetition. However, if one chooses to begin at the beginning and read straight through, one will benefit in a number of ways. In particular, as compensation for coping with the repetition, the person who chooses the straight route will learn how research can and did occur and will gain a fuller understanding of the basic concepts and the methodology we present here. In addition, that person will also experience directly the development of the kernel method from its birth to its relative maturity.

1.3. An Intriguing Puzzle

To see how intriguing questions from combinatory logic are—and to sample some of the results of our research—let us consider a hard puzzle that can be solved by applying equality-oriented reasoning, reasoning that is typical of mathematics and logic. To obtain its solution—which we give in this section to further illustrate the mysteries and challenges offered by this field—one need not be familiar with combinatory logic. However, since a little background is needed, let us supply what is sufficient for attacking and solving the puzzle, and then state the puzzle precisely.

The puzzle focuses on four combinators—B, W, S, and L—arranged in a square.

B W

S L

Their respective behavior is given by equations (1) through (4).

(1) $(\forall x \forall y \forall z)\ ((Bx)y)z = x(yz)$

(2) $(\forall x \forall y)\ (Wx)y = (xy)y$

(3) $(\forall x \forall y \forall z)\ ((Sx)y)z = (xz)(yz)$

(4) $(\forall x \forall y)\,(Lx)y = x(yy)$

Next, we need the definition of a fixed point combinator; Θ is a *fixed point combinator* if and only if Θ satisfies

(5) $\Theta x = x(\Theta x)$

for all combinators x. Finally, we need to know what is meant by *constructing a fixed point combinator* from a set P of combinators. To construct a fixed point combinator from a set P of combinators, one must find an expression Θ satisfying equation (5), where Θ is expressed purely in terms of the elements of the set P.

Perhaps the most natural way to find such an expression is to consider the equations for the elements of the set P under study—equations such as (1) through (4)—and begin drawing conclusions that are justified by those equations. As an example of such reasoning, from equation (2), one can conclude that

$(WW)y = (Wy)y,$

and also conclude from equation (2) that

$(Wy)y = (yy)y.$

If one then applies transitivity to the two conclusions, one discovers how the combinator WW behaves; in particular, one finds that

$(WW)y = (yy)y$

for all combinators y.

With the preceding information in hand, let us turn to the precise statement of the puzzle. The first rule of the game requires one to consider the square

$B\ W$
$S\ L$

and select a pair of combinators, one from column 1 of the square, and one from column 2. The second rule requires one to form the other three combinator pairs that can be formed by applying rule 1. The puzzle asks which, if any, of the four two-element sets that result from applying the two rules of the game can be used to construct a fixed point combinator. Since, by definition, a set with this capacity satisfies the *strong fixed point property*, one can view the puzzle as asking which, if any, of the four two-element sets under discussion satisfies the strong fixed point property.

As the following results show, the consideration of the four combinators—B, W, S, and L, whose behavior is given by equations (1) through (4)—is indeed sufficient for illustrating how subtle, complex, and challenging the search for a fixed point combinator Θ can be, where Θ satisfies equation (5) and where Θ is expressed purely in terms of the elements of P. Of course, if one wishes to attempt to solve the given puzzle oneself, one must avoid reading the next few sentences—since they provide the answers to the puzzle—and one must instead move immediately to Section 1.4.

In particular, from B and W alone, one can construct such a Θ (see Section 4.1.4); but from B and L alone, one cannot (see Corollary 3 of Section 5.3.6.2). In contrast, from S and W alone, one cannot construct such a Θ (see Corollary 2 of Section 6.3.1); but from S and L alone, one can (see Section 4.5.4).

We shall discuss the corresponding four sets of combinators in detail in Section 4, and prove the appropriate theorems. However, in contrast to relying on the natural and straightforward approach discussed earlier—that exemplified by drawing the conclusion that $(WW)y = (yy)y$—we show how one can solve the four subpuzzles by relying directly and indirectly on the kernel method, which we introduce in this report.

Of the two subsets that admit the construction of a fixed point combinator, by far the simplest to study is the set consisting of S and L alone. The only fixed point combinator, known to us, that is constructible from S and L alone is $(SL)L$. Immediately, we encounter what may be an open question—a question that asks one to prove that, other than $(SL)L$, one cannot construct a fixed point combinator from S and L alone. As for the other set that satisfies the strong fixed point property—that consisting of B and W alone—the first combinator that was found that satisfies

$$\Theta x = x(\Theta x)$$

is the combinator we call Θ_4 where

$$\Theta_4 = (B(WW))((BW)((BB)B)).$$

This fixed point combinator was discovered in 1986 by the logician Statman [Statman86]. As one will learn later in this report, the combinators B and W are exceedingly prolific—one can construct many other fixed point combinators from just those two. At this point, we cannot easily show—even intuitively—why neither the set consisting of B and L nor the set consisting of S and W satisfies the strong fixed point property; for proofs of those results, see Section 5.3.6.

For now, by considering the interconnections of the four cases, we can immediately see that combinatory logic offers many challenges to meet and many intriguing puzzles to solve. Specifically, if one either replaces B by S or replaces W by L, the strong fixed point property—possessed by the set consisting of B and W alone—is lost. Given this fact, one might naturally conjecture that making both replacements would certainly prevent the strong fixed point property from holding, but such is not the case. Instead, we have something like the double negative in the English language or like a two-position switch—either replacement alone removes the strong fixed point property, but the sequence of the two replacements in either order restores it.

In addition to focusing on the strong fixed point property, we shall also focus in this report on the *weak fixed point property*. The weak fixed point property asserts that, for all combinators x, there exists a y such that

$$y = xy.$$

We shall see—and one may already be able to prove without our assistance—that the strong fixed point property implies the weak fixed point property.

1.4. The Basic Problem to Solve

In one sense, constructing a fixed point combinator Θ from, say, B and W alone means using equations (1) and (2) and mathematical reasoning to produce a proof of the existence of an instance of equation (5) such that Θ is expressed purely in terms of B and W (see Section 4.3.3). Equivalently, but from the viewpoint of automated theorem proving, constructing a fixed point combinator from B and W can

mean using paramodulation [RobinsonG69,Wos73] (see Section 2.2)—an inference rule used for equality-oriented reasoning—to derive an appropriate instance of equation (5) starting with equations (1) and (2).

Alternatively, as in our earlier paper [McCune87], such a construction might consist of extracting an appropriate instance of equation (5) from a paramodulation-based proof by contradiction derivable from equations (1) and (2) and the assumption that equation (5) is false. Such an assumption asserts that, for each y, there exists some x depending on y violating equation (5). To make such an assertion, one can write

(6) $a(y,f(y)) \neq a(f(y),a(y,f(y)))$

where the function f conveys the corresponding dependence, and the function a is used for the function that occurs implicitly in equations such as (1) and (2) in which one combinator is *applied* to another. We shall discuss problem representation in Section 2.1, elaborating on the use of functions such as f and a, and showing that the preceding expression is actually an abbreviation for the commonly used notation in which EQUAL or ¬EQUAL precedes the corresponding statement. In Section 4, we shall give examples of paramodulation proofs of both types.

In contrast to each of the alternatives for obtaining an instance of equation (5)—that of reasoning forward from the appropriate axioms, and that of reasoning backward from the assumption that the strong fixed point property fails to hold—in this report, we offer a third choice which, as it turns out, is far more attractive and efficient to apply. Specifically, we show how the kernel method can be used effectively to search for fixed point combinators when the method is given a chosen set of combinators. Indeed, we show how the method succeeded in various cases, and succeeded in astoundingly little computer time.

1.5. More on the Content of This Report

In Section 4, we also revisit the five fixed point combinators Θ_1 through Θ_5, constructible from B and W alone, that were presented in our earlier paper on the subject [McCune87]. In that paper, those combinators were named J through N; the reason for the new notation will become clear later in this report when we enumerate certain sets of combinators (see Section 4.4.3). We then discuss how the use of kernels and superkernels markedly improves the search for such objects regardless of the choice of combinators to be used in the construction. For pedagogical reasons, we begin with cases far less complicated than that focusing on B and W; among those cases, we study the set consisting of S and L alone, which, of the four subpuzzles given in Section 1.3, is the simplest to solve.

When we do turn to the consideration of B and W, we clearly show why the study of kernels and superkernels holds so much interest for us. We do so by focusing on the results mentioned earlier and on other related results. For example, we discover that the research we reported earlier in [McCune87]—which surprised and interested Smullyan, who posed the original question that prompted our study of combinatory logic—is merely a hint of the richness offered by B and W. Specifically, rather than five fixed point combinators that can be constructed and that one might reasonably suspect exhaust the class, the use of kernels leads to finding an infinite set of such combinators. Then we discover that, in contrast to the fact that the five previously presented combinators are all provably equal in the presence of extensionality—a law that says that two combinators are equal if they behave the same when applied to

all other combinators—the newly found set contains an infinite subset such that no two of its elements are equal even when extensionality *is* present. This result is particularly pleasing, for it substantially strengthens our earlier results concerning Θ_1 through Θ_5 and, as mentioned earlier, sharply extends Statman's 1986 result. Later discoveries include answers to questions concerning smaller fixed point combinators constructible from S and K, and answers to questions focusing on the relationship of the weak fixed point property to the strong fixed point property.

Of course, as so often occurs when open questions are answered, new questions—and even conjectures—arise to take their place. Throughout this report, we pose such questions and conjectures, some of which focus on the possible completeness of the kernel method for searching for fixed point combinators. We collect and present most of the open questions in Section 5.4, and most of the conjectures in Section 5.5. Perhaps the most interesting conjecture—to us—asserts that, to every fixed point combinator, there corresponds a kernel. The questions we pose are equally relevant to combinatory logic and to automated theorem proving, although for different reasons. With regard to combinatory logic, their answers are important for proving that certain sets of combinators—when considered as the only ones that can be used—possess the sufficient properties for the construction of fixed point combinators, where other sets lack those properties. Thus, in this report, we address both sides of the question concerning the existence of fixed point combinators—that concerned with the sufficiency and that concerned with the insufficiency of various sets of combinators for constructing a fixed point combinator.

Let us, therefore, set the stage for the detailed discussion of the kernel method by supplying the appropriate background in Sections 2 and 3. We can then turn in Section 4 to the chronological account of how the method was formulated, which will illustrate how valuable to research in mathematics and logic the use of an automated theorem-proving program can be. In particular, we shall show precisely where our computer programs were of use and what role they played. We shall also show that the problem of finding fixed point combinators is a difficult problem indeed. As we proceed, we shall answer the question mathematicians and logicians most often ask when the subject of automated theorem proving arises: When, if ever, will such programs be of real and substantial use as research assistants for answering hard questions? In this report, we give evidence that that time has arrived.

1.6. Notation and Conventions

We need little notation, and few conventions. With one exception, we use uppercase letters to denote combinators and lowercase letters, u through z, to denote variables. The exception is the letter f, which we use to denote an arbitrarily chosen combinator; the significance of this notation will be made clear at the beginning of Section 3. For general discussions, we use Θ to denote a fixed point combinator, Γ to denote a kernel, and Ω to denote a generator of a kernel. The use of a subscript usually denotes the corresponding position in some enumeration—for example, Θ_5 denotes that the fifth element of some sequence is under discussion. Finally, in most cases, the use of a subscript/superscript pair i,j denotes that the i-th element in some sequence has length j—for example, Θ_5^9 denotes that the fifth element of some sequence is under discussion, and the element has length 9. One exception occurs when we write, for example, Γ_1^9 to refer to an expression that in turn points to an element that is the first element in some sequence, where the element pointed to has length 9.

2. A Review of Automated Theorem Proving

At this point, we turn to a brief review of the pertinent aspects of automated theorem proving, and show how an automated theorem-proving program can provide valuable assistance for various types of research. This review also provides the needed background for a clear understanding of how an automated theorem-proving program can be used to apply the systematic kernel method (given in Section 4.5) for searching for fixed point combinators. Although the method does have the nice property that it can be applied by hand, reliance on a computer program contributes materially to ease and accuracy. In particular, the probability of overlooking an interesting kernel or combinator is sharply reduced when the application of the method is assigned to the computer. The elements to be covered in our review are representation of information (by using the *clause language*), reasoning (the specific inference rule *paramodulation* for building in equality), strategies (including the *set of support strategy*) of various types for controlling the reasoning, a procedure (*demodulation*) for rewriting information into a canonical form, a procedure (*subsumption*) for discarding trivial corollaries, and a means (*proof by contradiction*) for the program to determine that the given assignment has been completed.

As the name implies, the field of automated theorem proving focuses on the formulation and implementation of techniques that permit a computer program to prove theorems. The fields from which a theorem may be taken include algebra, geometry, topology, set theory, or—as is the case in this report—combinatory logic. The most widely used approach for proving theorems with a computer program focuses on searching for a proof by contradiction. Therefore, to enable a program using such an approach to attempt to prove a proposed theorem of the form **if** P **then** Q, one is required to define the field from which the theorem is taken, to give the special (or added) hypothesis of the theorem, and to assume that the theorem is in fact false. To fulfill this requirement, one writes a set of statements of the form P **and (not** Q**)**, where P corresponds to the axioms of the underlying theory, the special hypothesis of the theorem, and any lemmas one might wish to supply, and where Q corresponds to the conclusion of the theorem. This set of statements is then given to the automated theorem-proving program in use, which then applies logical reasoning in an attempt to complete the assignment of finding a proof by contradiction.

To reason logically—and thus begin its attack on the given problem—the theorem-proving program draws conclusions by applying one or more *sound* inference rules chosen by the researcher. The application of each rule is directed by one type of strategy and restricted by another type. When a conclusion is drawn, the program rewrites it into a canonical form by applying rules (*demodulators* [Wos67]) supplied by the researcher or discovered by the program. The result is then tested (with the use of subsumption [RobinsonJ65]) to see whether it is a trivial corollary of information the program already has and should, therefore, be discarded. The result can also be tested to see whether it satisfies criteria, supplied by the researcher, for measuring significance; if the conclusion fails the test, it is discarded. The goal is to deduce some new conclusion that contradicts a conclusion drawn earlier or contradicts some input statement. When such a discovery is made, a proof by contradiction has been found, and the automated theorem-proving program in use "knows" that the given assignment has been completed.

Using the preceding summary of the program's actions as an outline for this section, let us sample each of the main areas, beginning with a discussion of how one specifies the problem to be considered.

Of course, for those who are familiar with automated theorem proving or automated reasoning, the remainder of this section can be skipped or can be read very quickly to become familiar with our notation.

2.1. Representation

In this section, we mainly use the notation of the *clause language*, the language in which one usually converses with a theorem-proving program and which is used in our two reference books [Wos84,Wos87]. We take this action even though, without doubt, the various languages of mathematics an logic are usually easier to read than the clause language is. Indeed, in later sections, we frequently employ the notation of combinatory logic, which we give in Section 3.2, so that one can concentrate on the logic itself and rely on previous exposure to mathematics and logic. However, we provide here a limited treatment of the clause language since one of our objectives is to introduce automated theorem proving to mathematicians and logicians. Our first examples will be taken from ordinary arithmetic, and then we shall turn to examples from the less familiar field of combinatory logic, a field we introduce in Section 3.

The conventions observed in the recommended source books [Wos84,Wos87] require variables to be chosen from among lowercase u through z, functions and constants to be written in lower case, and relations (predicates) to be written in upper case. Also by convention, each variable is implicitly universally quantified (meaning "for all"), and each variable is relevant only to the *clause* in which it occurs. In addition, by writing EQUAL for the equality relation, the program is able to treat equality as "understood" or "built in". Finally, the symbol $\neg$ means **not**, and the symbol | means **or**. We shall only infrequently have need of statements in which **or** occurs; however, when we do, the expressions separated by | are called *literals*. As we shall see, existence is not represented with variables; but, instead, appropriate functions and constants are employed. In addition, we shall see that the logical operator **and** never occurs within a clause, and is always assumed to be present implicitly between clauses. As for the operator **if-then**, its place is taken by the use of **not** and **or**, using the equivalence of **if** P **then** Q with **not** P **or** q. To be consistent with the material presented later and to prepare for the subject of combinators, we occasionally violate one of the given conventions. Specifically, we use uppercase letters for the constants that correspond to various specific combinators, kernels, and the like. Let us now turn to some examples.

To express the fact that

$$0 + x = x$$

for all x, we write

EQUAL(sum(0,x),x)

where x is treated implicitly as a universally quantified variable ranging over all numbers, and 0 is treated as a *constant*; formally, 0 is a *Skolem constant*. To express the existence of an additive (right) inverse, for all x there exists a y such that

$$x + y = 0,$$

we write the clause

EQUAL(sum(x,minus(x)),0)

where x implicitly ranges over all numbers, and where *minus* is an appropriate *Skolem function*.

A *Skolem constant* is used for expressing the existence of some element that does *not* depend on the universally quantified variables—for example, there exists a y such that for all x $y + x = x$, which is, of course, closely related to the earlier example regarding 0. A *Skolem function* is used for existentially quantified variables that *do* depend on other universally quantified variables—for example, for all x there exists a y such that $x + y = 0$. A Skolem constant names the element in question, and a Skolem function names a function and, by its arguments, expresses the appropriate dependence. The respective equivalents—given earlier in which the predicate EQUAL occurs—of the two arithmetic laws are called *clauses*.

For the next examples, let us turn to combinatory logic itself and, specifically, to a theorem that follows from a more general result cited in Section 1 and proved in Section 4—the result that asserts that Θ_4 is a fixed point combinator constructible from B and W. The theorem to prove says that, in the presence of

$$(\forall x \forall y \forall z)\ (((Bx)y)z) = (x(yz))$$

and

$$(\forall x \forall y)\ ((Wx)y) = ((xy)y),$$

there exists a Θ such that for all x

$$\Theta x = x(\Theta x).$$

The existence theorem under consideration is logically weaker than the corresponding theorem for Θ_4 because the latter implies the former. After all, if one proves that

$$\Theta_4 x = x(\Theta_4 x),$$

then one has obviously found a Θ of the desired type, namely, $\Theta = \Theta_4$. However, as so often occurs in mathematics, the more general theorem is actually easier to prove than the less general because, as in the case under discussion, one can focus directly on Θ_4 and its properties rather than searching for Θ_4 or some other combinator with the desired properties. In other words, concentrating on Θ_4 provides clues about how one might prove that it is a fixed point combinator. In Section 4.1, when we discuss the attempts to prove these two theorems with the aid of the automated theorem-proving program ITP, we shall shed additional light on why the more general result is actually easier to prove than the less general is.

The first step in presenting this theorem for consideration by an automated theorem-proving program is to produce clauses that correspond to the required axioms. In addition to clauses to capture the actions of B and W, we must include clause (1) for the axiom of reflexivity of equality.

(1) EQUAL(x,x)

Because of our choice of inference rule, which we discuss in the next section, no other equality axioms are needed. Since the equations for B and W implicitly mention a binary function to show that one combinator is *applied* to another, we must choose a function symbol to employ explicitly. We choose *a* for *apply* as our function, and, observing the conventions of using upper case letters for predicates (relations) and of using EQUAL for equality, we write two clauses

(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

(3) EQUAL(a(a(W,x),y),a(a(x,y),y))

to capture the actions of the corresponding equations, where $a(x,y)$ means xy—which, in combinatory logic means apply x to y. No other clauses are needed to characterize the theorem under study and to permit the program to attempt to prove that theorem.

However, one additional step regarding representation is required if we wish the theorem-proving program to "know" that it has succeeded in its attempt; that step enables the program to use the standard means—finding a proof by contradiction—for determining assignment completion. The second and final step required for presenting the problem is to produce the clause or clauses that arise from assuming (the conclusion of) the theorem false. Since the conclusion is, formally, there exists a y such that for all x

$yx = x(yx)$,

the clause

(4) ¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))

is included.

To understand intuitively how clause (4) captures the assumed falseness of the conclusion of the theorem, one notes that such an assumption implies that, for each chosen combinator y intended to satisfy the conclusion, there must be some combinator $f(y)$ dependent on the chosen combinator y for which the theorem fails to hold. In other words, a function f exists such that, for each chosen combinator y intended to satisfy the theorem, f assigns an $f(y)$ to y violating the desired conclusion. On the other hand, to see formally how clause (4) is obtained, first one notes that the assumption that the conclusion of the theorem is false yields

not exists y such that for all x, $yx = x(yx)$,

which is logically equivalent to

for all y exists x, $yx \neq x(yx)$.

Finally, since the existentially quantified x is a function of the universally quantified y in the assumption that the conclusion is false, one employs a new Skolem function, f, that depends on y to get clause (4).

2.2. An Inference Rule, Paramodulation

Although a general-purpose theorem-proving program offers a wide variety of inference rules, we need introduce only one of those rules for the study in this report. Use of that rule, *paramodulation* [RobinsonG69,Wos73], enables an automated theorem-proving program to treat equality as if it is "understood" in the sense that no axioms other than reflexivity ($x = x$) need to be included when a problem is submitted for consideration. In particular, when paramodulation is applied to two clauses satisfying the appropriate conditions, a conclusion is obtained in the spirit of equality substitution. Actually, as we shall see, paramodulation generalizes the usual notion of equality substitution.

For the actions of paramodulation to be easily understood, we need one concept, that of *unification* [RobinsonJ65,Wos84,Wos87]. This concept, which plays a vital role for our attack on problems in combinatory logic, is also required for other aspects of automated theorem proving presented later in this section.

To *unify* two expressions, the program must find terms to replace the variables in each such that the two resulting expressions are identical (possibly with the exception of sign). When unification is applied to two expressions both of which are clauses, the program always treats the two expressions as having no variables in common; the program is allowed to take this action since a variable is relevant only to the clause in which it occurs. The most general replacement for variables is always preferred, as evidenced in the third example of paramodulation given shortly. The object is to maintain generality where possible, which adds markedly to the efficiency of the theorem-proving program.

An automated theorem-proving program's preference for generality is but one difference between computer-oriented and person-oriented reasoning. Since people often prefer to use a specific instance of a general fact rather than the fact itself, we see that a typical theorem-proving program does *not* usually reason as a person does. Paramodulation, as we shall see, is an excellent example of the kind of reasoning used by theorem-proving programs that can hardly be said to imitate a person's reasoning. On the other hand, for the kind of reasoning a person often uses but that a typical theorem-proving program does not, we can turn to that area of mathematics known as group theory. For example, where the identity of a group is denoted by e, the reasoning that permits deducing

$(yz)(yz) = e$

from the hypothesis that

$xx = e$

is not used by such programs, although such reasoning is quite typical of mathematics. The view of many experts in the field of automated theorem proving is that using this type of reasoning, called *instantiation*, would markedly decrease the efficiency of theorem-proving programs, even though mathematicians obviously use it very effectively.

2.2.1. Paramodulation Defined with Examples

Rather than giving a formal definition of paramodulation, we shall, as in the preceding section, rely on examples. The first example illustrates the use of the simplest and most familiar form of equality substitution. The second illustrates a slightly more complex form of substitution in that variables occur in one of the hypotheses. The third example, substantially more complex, is by far the most interesting, for it shows how paramodulation generalizes the standard notion of equality substitution.

For the first example, paramodulation applied to both the equation $a + (-a) = 0$ and the statement $a + (-a)$ is congruent to b yields in a single step the statement (conclusion) 0 is congruent to b. In clause form, *from*

EQUAL(sum(a,minus(a)),0)

into

CONGRUENT(sum(a,minus(a)),b)

one obtains the clause

CONGRUENT(0,b)

by paramodulation.

For the second example, since variables such as x mean "for all x", one can show that the following example of equality-oriented reasoning is logically correct, *logically sound*. Paramodulation applied to both the equation $a + (-a) = 0$ and the statement $x + (-a)$ is congruent to x yields in a single step the statement (conclusion) 0 is congruent to a. In clause form, *from*

EQUAL(sum(a,minus(a)),0)

into

CONGRUENT(sum(x,minus(a)),x)

the clause

CONGRUENT(0,a)

is obtained. This example illustrates some of the complexity of paramodulation; in particular, the second occurrence of the variable x in the *into clause* becomes the constant a in the conclusion, but the term containing the first occurrence of x becomes the constant 0 in the conclusion. Although the unification of the first argument of the *from clause* with the first argument of the *into clause* temporarily requires both occurrences of the variable x to be replaced by the constant a, paramodulation then requires an additional term replacement justified by equality substitution. In particular, in this second example, paramodulation requires the replacement of $a + (-a)$ by 0.

Finally, for the third and complex example, paramodulation applied to both the equation $x + (-x) = 0$ and the equation $y + (-y + z) = z$ yields in a single step the conclusion $y + 0 = -(-y)$. In clause form, *from*

EQUAL(sum(x,minus(x)),0)

into

EQUAL(sum(y,sum(minus(y),z)),z)

the clause

EQUAL(sum(y,0),minus(minus(y)))

is obtained by paramodulation.

To see that this last clause is in fact a logical consequence of its two parents, one unifies the argument *sum(x,minus(x))* with the term *sum(minus(y),z)*, applies the corresponding substitution to both the *from* and *into clauses*, and then makes the appropriate term replacement justified by the typical use of equality. The substitution found by the attempt to unify the given argument and given term requires substituting *minus(y)* for x and *minus(minus(y))* for z. To prepare for the (standard) use of equality in this third example, a nontrivial substitution for variables in both the *from* and the *into clauses* is required, which illustrates how paramodulation generalizes the usual notion of equality substitution. In contrast, in the standard use of equality substitution, one does *not* apply a nontrivial replacement for variables in both the *from* and the *into* statements.

Summarizing, a successful use of paramodulation combines in a single step the process of finding the (in an obvious sense) most general common domain for which both the *from* and *into clauses* are relevant and applying standard equality substitution to that common domain.

The complexity of this inference rule rests in part with its unnaturalness if viewed from the type of reasoning people employ, in part with the fact that the rule is permitted to apply nontrivial variable replacement to both of the statements under consideration, and in part with the fact that different occurrences of the same expression can be transformed differently. As an illustration of the third cited property, in the third example of paramodulation, the (term containing the) first occurrence of z is transformed to 0, but the second occurrence of z is transformed to $minus(minus(y))$.

2.2.2. Paramodulation Used for Combinatory Logic

In the spirit of the third example of paramodulation, we can now consider an example from combinatory logic in which we focus on the combinator W. By definition, W satisfies the equation

(1) $(Wx)y = (xy)y$

for all combinators x and y. If we search for conclusions that follow from considering W with itself, we encounter the combinator WW and discover how it behaves. In particular, by applying mathematical reasoning to the case in which the variable x is set to W, we have

(2) $(WW)y = (Wy)y = (yy)y$

for all combinators y. The equality of the first two expressions of (2) is justified by merely substituting W for x in (1) or, alternatively, by applying (1) to the first expression. The equality of the second and third expressions of (2) is justified by applying (1) to the second or by choosing an appropriate instance of (1). The equality of the first and third expressions—which is the interesting result because it gives the behavior of the combinator WW—is justified, of course, by transitivity.

Instead of reasoning as the mathematician or the logician does—which we just illustrated—we can apply paramodulation to determine how the combinator WW behaves. We take two copies of (1) with no variables in common so that the use of paramodulation can be followed more easily, write the corresponding clauses, and call them, respectively, (3a) and (3b). *From*

(3a) EQUAL(a(a(W,x),y),a(a(x,y),y))

into

(3b) EQUAL(a(a(W,u),v),a(a(u,v),v))

we obtain in one paramodulation step

(4) EQUAL(a(a(W,W),y),a(a(y,y),y))

as the result.

To see precisely how a single application of paramodulation produces (4) *from* (3a) *into* (3b), first one notes that the program attempts to find an argument of clause (3a) and a term in clause (3b) such that the two expressions unify. Among the successful unifications that occur, the program succeeds in unifying the second argument of clause (3a) with the first argument of clause (3b), which produces the substitution W for x, y for u, and y for v. Then the program deduces clause (4) by next applying the given substitution to produce the temporary clauses (3a′) and (3b′). Finally the program replaces the first argument of (3b′) by the first argument of (3a′).

As in the third example of paramodulation given earlier, the usual notion of equality substitution does not apply here—to clauses (3a) and (3b); in particular, since a nontrivial replacement of variables by terms in both (3a) and (3b) is required to deduce clause (4), generalized equality substitution is being used rather than the usual notion of equality substitution. In (3a), all occurrences of x are replaced by W to obtain (3a′); in (3b), all occurrences of u and v are replaced by y to obtain (3b′). To stress the point, no way exists to obtain clause (4) by merely making an appropriate replacement of variables in exactly one of (3a) or (3b) followed by an ordinary use of equality substitution.

2.2.3. The Power of Paramodulation

We thus have a simple but striking illustration of the potential value offered by an automated theorem-proving program for studies in mathematics and logic. In particular, a single application of paramodulation leads to the deduction of an interesting conclusion that the usual mathematical reasoning requires more than one step to yield. It is important to note the contrast between the two types of reasoning used to yield the information contained in clause (4). The mathematician or logician picks the instances—the substitutions of terms for variables—that might be of interest, and then applies (the usual notion of) equality substitution. Paramodulation, on the other hand, picks the (maximal) instances of the *from* and *into clauses* automatically and, rather than deducing intermediate conclusions by applying the corresponding replacement of terms for variables, instead combines the instance picking with a standard application of equality substitution. Avoiding the retention of intermediate conclusions can contribute markedly to the performance of a theorem-proving program. The instances that a person chooses may be far less general than those chosen by the theorem-proving program in its attempt to apply paramodulation. As a result, the program may deduce a conclusion that is far more general than a person might. Again the program benefits, for its efficiency often increases by having access to general conclusions rather than specific ones; people do not require such generality in most cases.

Summarizing, the inference rule instantiation—replacing variables by terms in a statement to deduce a corollary—is frequently used, and used wisely, in mathematics and logic. Instantiation is *not* an inference rule offered by the typical theorem-proving program, for no known strategy exists for wisely choosing appropriate instances even though mathematicians and logicians do have that ability.

To give a second illustration of the power offered by paramodulation—if combined with a procedure (demodulation, which is discussed in Section 2.4) used by theorem-proving programs to simplify expressions—we consider the following example so typical of mathematics. In the example, the conclusion that is reached with a string of equalities—if one applies the usual type of reasoning that occurs in mathematics—is drawn by a theorem-proving program in a single deduction step.

$$z = 0 + z = (x + (-x)) + z = x + ((-x) + z)$$

This string of equalities is a proof of an obvious but useful lemma in arithmetic and in group theory, the lemma that asserts the equality of the first and last expressions in the string of equalities. In clause notation, we have *from* right inverse

EQUAL(sum(x,minus(x)),0)

into associativity

EQUAL(sum(sum(u,v),z),sum(u,sum(v,z)))

the program can deduce

EQUAL(sum(0,z),sum(x,sum(minus(x),z)))

which it can then automatically simplify (see Section 2.4) to

EQUAL(z,sum(x,sum(minus(x),z)))

without producing any intermediate results.

If we return to the example from combinatory logic and continue to reason from (1) with itself or, equivalently, from (3a) and (3b), we discover that focusing on a single axiom can produce a myriad of conclusions. In fact, as we shall immediately see, the potential for drawing a large number of conclusions is often so great that we see why there exists a need for using *strategy* to control the reasoning, a need that we address in more detail in Section 2.3. The reasoning we shall apply is similar, although not identical, to that (applied in Section 2.2.2) which yields (2)—or yields its equivalent (4) in clause notation.

Relying on the convention (commonly used in combinatory logic) that expressions are left associated unless otherwise indicated, the following conclusions can be correctly drawn. The most interesting result in each case is the equality of the first and third expressions. Clauses expressing the following eight equalities—one of which we have already discussed—can be obtained by paramodulating *from* the right side of (3a) *into* various terms of (3b).

$WWy = Wyy = yyy$
$W(Wxy)v = W(xyy)v = xyyvv$
$Wu(Wxy) = Wu(xyy) = u(xyy)(xyy)$
$W(xyy)v = xyyvv = Wxyvv$
$Wu(xyy) = u(xyy)(xyy) = u(Wxy)(xyy)$
$Wu(xyy) = u(xyy)(xyy) = u(xyy)(Wxy)$
$W(xy)y = xyyy = Wxyy$
$Wxy = xyy = Wxy$

On the other hand, by paramodulating *from* the left side of (3a) *into* various terms of (3b), we obtain the following eight equalities.

$W(xyy)v = W(Wxy)v = Wxyvv$
$Wu(xyy) = Wu(Wxy) = u(Wxy)(Wxy)$
$W(Wxy)v = Wxyvv = xyyvv$
$Wu(Wxy) = u(Wxy)(Wxy) = u(xyy)(Wxy)$
$Wu(Wxy) = u(Wxy)(Wxy) = u(Wxy)(xyy)$
$xyy = Wxy = xyy$
$W(Wx)y = Wxyy = xyyy$
$WWv = Wvv = vvv$

Fortunately, the consideration by an automated theorem-proving program of clauses (3a) and (3b)—technically, of the equation for W in clause form with itself—will not be hindered by all sixteen conclusions we have just listed. Instead, typically, the clausal equivalent of the second through the sixth

equality of the first set and the first five of the second set will not be deduced by the theorem-proving program because, as we shall see in the next section which focuses on strategy, paramodulation is almost always restricted from considering *into terms* that are variables. Of the remaining conclusions, the program will discard as trivial corollaries the last one of the first set and the sixth and eighth of the last set. Trivial corollaries are discarded by using a procedure called *subsumption*, discussed in Section 2.5. The program will, therefore, retain only clauses equivalent to the following three equalities.

$WWy = yyy$
$W(xy)y = Wxyy$
$W(Wx)y = xyyy$

2.3. Strategy

In the preceding section (Section 2.23), we saw that the presence of a single axiom enables one to draw 16 conclusions, each with one application of paramodulation. Each of those conclusions is obtained by considering the axiom W with itself. One can imagine how many additional conclusions would be drawn were we to continue reasoning by focusing on pairs of those 16 conclusions—which are the first generation of descendants of W with itself—and then reason by focusing on the second generation. To avoid the flood of conclusions that can result if reasoning is uncontrolled, an automated theorem-proving program can use various types of strategy. On the one hand, some types of strategy are designed to direct the reasoning; on the other, some types are designed to restrict it. From a different perspective, some types of strategy are general in the sense that they apply to all inference rules, and others are specific in the sense that they apply only to a single inference rule.

2.3.1. The Set of Support Strategy

Of those of a general nature, the only strategy we used for our study of combinatory logic reported here is called the *set of support strategy* [Wos65]. That strategy restricts the program's reasoning with the intention of sharply retarding the generation of irrelevant conclusions. To use the set of support strategy, one is required to choose a subset T of the set S of clauses given to the program to describe the problem to be studied. The program then draws a conclusion only if it is recursively traceable to T. Therefore, at the beginning of the program's attack on a problem—from a procedural viewpoint—an inference rule is applied to a pair of clauses from S only if at least one of them is in T. Any clause that is retained is automatically given the power of the clauses that are members of T. For example, an inference rule can be applied to a pair of clauses if one of them is a clause not in S. Stated another way, with the set of support strategy, the program is not allowed to apply an inference rule to a pair of clauses both of which are in $S - T$.

Of those of a specific nature—in addition to the set of support strategy, which can be used to restrict the application of any inference rule—we also used a number of strategies designed with the express purpose of restricting the application of paramodulation. First, the program was not allowed to paramodulate *from* or *into* a variable. This restriction prevents a myriad of conclusions from being drawn, as illustrated in the preceding section. Such a restriction strategy is, in most cases, mandatory. After all, paramodulating *from* or *into* a variable will always yield a conclusion since the corresponding unification can never

fail. Second, in many of our attempts to prove various theorems, the program was allowed to paramodulate only from a specified side of each equality. In some cases, all paramodulations that were allowed were from the left side only; in others, all were from the right only. Third, some attempts to complete an assignment were restricted by requiring that the *into term* for each application of paramodulation have a *position vector* (see Section 2.3.2) consisting of all 1's; in Section 4, we show how useful this strategy is.

2.3.2. Position Vectors

The *position vector* of a term is a k-tuple that gives the relative position of the term within a literal. For example, the position vector [2,3,1] says that the corresponding term is the first subterm of the third subterm of the second argument. For a concrete illustration—taken from the results of our research—of the use of position vectors, the third occurrence of the constant W in the equality

EQUAL(a(a(B,a(a(B,a(a(B,a(W,W)),W)),B)),B),Θ_1)

has the position vector [1,1,2,1,2,2]. (The first argument of this equality is in fact a fixed point combinator, constructible from B and W, that plays a vital role in the study of such combinators.) For a second example, if we write

FIXED(a(a(B,a(a(B,a(a(B,a(W,W)),W)),B)),B))

to mean that the given expression is a fixed point combinator, then the entire expression—the argument of the predicate FIXED—has a position vector of 1.

2.3.3. Strategy and Types of Proof

In addition to the strategies just discussed, we also use three strategies to restrict the program to searching for a specific type of proof; all three types of proof are still required to be a proof by contradiction. In the first strategy—designed to restrict the program to searching for *input proofs* only—the program is required to consider paramodulation steps in which at least one of the two clauses is an *input clause*, one of the clauses given in the problem description. In the second, the program is required to search for *forward proofs* only. In other words, the program is not allowed to attempt a paramodulation step in which one of the two clauses is an input clause that is included as a result of assuming the proposed theorem false, an assumption that is typically made to enable the program to seek a proof by contradiction (see Section 2.1). In the third, the program is required to search for *backward proofs* only. The program applies paramodulation to a pair of clauses only if at least one of the pair is an input clause that is present as a result of assuming the theorem false, or is a descendant of such a deduction.

A forward proof should not be confused with what one might call a positive proof, a proof in which paramodulation is applied only to pairs of positive clauses—clauses free of the negation symbol $\neg$. In particular, the assumption that the conclusion of some theorem under study is false might indeed introduce a positive clause. Such a clause cannot be used for a paramodulation step in a forward proof. Similarly, a backward proof should not be confused with a negative proof, a proof that consists of steps in which paramodulation is applied to a pair of clauses at least one of which is negative—a clause all of whose literals are preceded by $\neg$. The input set of clauses may indeed contain a negative clause as one of the axioms for the field from which the problem under study has been selected. In Section 4, we shall give examples of each of the three types of proof (input, forward, backward) by giving examples of the

corresponding strategy used to search for each type of proof. One could, of course, use a strategy that is a combination of two of the given three strategies; for example, one might restrict the program to searching for proofs required to be both input and forward.

Regardless of which strategy or strategies one chooses, the important point to note is that, for most assignments given to an automated theorem-proving program, some form of strategy is essential for completing the assignment in a reasonable amount of computer time. What is not always recognized is the fact that a corresponding remark holds when an assignment is given to a person rather than to a computer program. Part of the confusion in this regard can be traced to the realization that we often are unable to identify the specific strategy or strategies a person is using to produce one success after another.

2.4. Demodulation

The next topic for this review of automated theorem proving is that of canonicalization, which is available to automated theorem-proving programs by the use of demodulation [Wos67]. In many areas of mathematics and logic, one often rewrites each new item of information into a canonical form. In automated theorem proving, a similar action can occur. A sharp difference, however, exists between mathematics and automated theorem proving. In the former, one uses canonicalization when it seems convenient and appropriate. In the latter, one cannot in most cases exercise such freedom for, without automatically requiring each conclusion to be rewritten into a canonical form, the program will drown in information. Of course, some conclusions will be unaffected by the particular choice of demodulators [Wos67] in a given problem.

One other difference exists between the use of canonicalization by a mathematician or logician and its use by a theorem-proving program. When a person uses various equalities for canonicalization, the equalities are applied only to the expressions chosen by the person; when a program applies such equalities, all expressions to which the equalities are applicable are in fact rewritten into a canonical form. Now, let us examine some typical uses of canonicalization.

One might, for example, uniformly replace $-(-t)$ by t for any term t, or replace $r(s+t)$ by $rs+rt$ for terms r, s, and t. For such term replacements, many automated theorem-proving programs offer a mechanism, *demodulation*, referred to earlier that automatically rewrites expressions into a canonical form based on some set of rewrite rules called *demodulators*. The demodulators can be included among the input clauses, or the program can be asked to find potentially useful demodulators. For the two given examples, the demodulators

EQUAL(minus(minus(x)),x)

and

EQUAL(prod(x,sum(y,z)),sum(prod(x,z),prod(y,z)))

can be used, respectively.

As one might predict, care must be taken in the choice of which equality clauses to attempt to use as demodulators. In particular, an unwise choice can cause the program to loop. To see how looping can occur, let us consider what will happen when the equation for W is used as a demodulator and when the program encounters a term of a certain type. In particular, when the clause

(W) EQUAL(a(a(W,x),y),a(a(x,y),y))

is used as a demodulator and the term *WWW* is encountered, this term will be transformed to itself repeatedly without ever terminating. For a more interesting example, one that will be studied in Section 4, let us consider the expression

$\Gamma = a(a(W,a(B,x)),a(W,a(B,x)))$

and what occurs when the two demodulators

(W) EQUAL(a(a(W,x),y),a(a(x,y),y))

and

(B) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

are applied to Γ with the object of producing a canonical form. Demodulation begins by successfully applying *W*, then *B*, to obtain

a(x,Γ)

which leads to another successful application of *W* followed by *B* to obtain

a(x,a(x,Γ))

which of course leads to yet another success and, in fact, to a nonterminating sequence of successful applications of *W* and *B*. (As an aside, for those who enjoy a preview designed to increase one's curiosity, Γ is a kernel—in other words, is an example of the key to unlocking the puzzle of searching for fixed point combinators in an efficient manner.)

Nevertheless, as long as one is careful about the choice of demodulators, the use of demodulation can sharply increase the effectiveness of a theorem-proving program in its search for assignment completion. Even further, for many studies, without the use of demodulation the program will simply require an inordinate amount of computer time to complete rather trivial assignments.

2.5. Subsumption

We now come to the next-to-the-last topic of this section, a mechanism that is almost always essential for a theorem-proving program to be effective. Although the use of various types of strategy does indeed contribute markedly to the efficiency of an automated theorem-proving program, such a program continues to deduce conclusions that are trivial and unneeded corollaries of existing clauses. For example, if the program paramodulates *from* the second argument of the clause equivalent of

$Wxy = xyy$

into the first subargument of the second argument of the clause equivalent of

$W(Wx)y = xyyy,$

the program deduces the clause equivalent of

$W(Wx)y = Wxyy,$

which is a trivial corollary, in fact an instance, of

$Wxy = xyy$

and is, therefore, immediately discarded. The procedure for discarding such deductions is *subsumption* [RobinsonJ65]. The clause *A* ***subsumes*** the clause *B* if there exists a uniform replacement of terms for variables in *A* that produces a clause that is a subclause of *B*. For our purposes in this study—since we focus mainly on clauses free of |, the symbol for logical **or**—we can almost always use the more restricted definition asserting that the clause *A* subsumes the clause *B* if some replacement of terms for variables in *A* produces a copy of *B*. As an aside, if we wished to have our program use instantiation as an inference rule to permit the program to imitate the corresponding type of reasoning (discussed in Section 2.2) that is often used in mathematics, we would be forced to protect the instances that were deduced so that subsumption would not immediately discard them.

Of especial use is the role of subsumption for removing clauses that assert the equality of a term with itself; such an assertion is, needless to say, usually of little interest and seldom useful. To remove such trivial clauses, the clause

EQUAL(x,x)

for reflexivity of equality subsumes many clauses that would otherwise be kept. This clause, which must be included in the input if paramodulation is the chosen rule of inference, is also important in that a proof often terminates with the deduction of a clause of the form

¬EQUAL(t,t)

for some term *t*.

2.6. The ANSWER Literal and Proof by Contradiction

We close our review of automated theorem proving by showing how one can easily extract, from a proof by contradiction, information that may be hidden in such a proof. To be completely accurate, we show how one can have the program itself extract a certain type of hidden information. Two existence theorems will serve nicely for our purpose. For our two examples, to make it easier to see what occurs, we choose the variables so that no variable appears in more than one clause.

The first theorem asserts that if one considers the set *P* of combinators such that *P* contains the single combinator *L* with

$(Lu)v = u(vv)$,

then one can prove that *P* satisfies a property known as the *weak fixed point property*—for all combinators *x*, there exists a *y* such that

$y = xy$.

As usual, when we submit a (conjectured) theorem to an automated theorem-proving program and assign the program the task of finding a proof, we assume that the theorem is false. We therefore assume that there exists an *x* such that for all *y*

$y \neq xy$.

If we were simply seeking a proof and using the inference rule paramodulation, we would give the program the following three clauses.

(1) EQUAL(x,x)

(2) EQUAL(a(a(L,u),v),a(u,a(v,v)))

(3) ¬EQUAL(y,a(f,y))

To see how clause (3) is obtained, recall that we must use a Skolem constant since the x that is assumed to exist does not depend on any variables; we choose to use f.

As it turns out, clauses (2) and (3) contradict each other; to see this, one substitutes f for u, $a(L,f)$ for v, and $a(a(L,f),a(L,f))$ for y. In other words, the program would succeed in proving the theorem and would have no need of paramodulation. However, as is transparent, to prove almost any theorem requires some conclusions to be drawn—requires the use of paramodulation, or the use of whatever inference rule has been chosen. More important, when attacking an existence theorem, one often wishes to have more than a proof; in particular, one often wishes to find an object that possesses the property or properties of interest. If in fact the goal is to find such an object, a proof of the type just completed does not immediately display the desired object. Therefore, one is forced to analyze the proof—as we do in similar proofs in Section 4—to produce the appropriate construct.

Far more satisfying and less error prone is the approach that assigns the construction of the desired object to the program. If one chooses this approach, one simply appends to the appropriate clause or clauses a literal, the ANSWER literal, and assigns its arguments to be the variable or variables corresponding to the object to be constructed. In our case, we replace clause (3), therefore, with clause

(3a) ¬EQUAL(y,a(f,y)) | ANSWER(y)

and again assign the program the task of finding a proof by contradiction.

When seeking such a proof, the program behaves as if the ANSWER literal does not occur in any clause. Almost identical to the preceding proof, the program immediately discovers that clause (2) contradicts clause (. However, in contrast to the first given proof, this time the program returns the clause

ANSWER(a(a(L,f),a(L,f)))

knowing that a proof by contradiction has been completed since a clause whose only literals are ANSWER literals has been deduced. To find the y that must exist for the weak fixed point property to hold for P consisting of L alone, one merely replaces all occurrences of f in the argument of the ANSWER literal by x, which is justified since f occurs because of assuming the theorem false.

Although the first existence theorem was so simple to prove that one might find no need for using the ANSWER literal, most other existence theorems—as one would expect—are far more complicated to prove. In those cases—if the ANSWER literal is not used—to obtain the desired construct, the researcher might be forced to execute a far more involved procedure to extract the object from a proof by contradiction. To illustrate this point, let us consider our second existence theorem, a theorem whose proof is somewhat more complicated than that of the first theorem. The second theorem asserts that if one considers the set P of combinators consisting of B and W alone with

$((Bx)y)z = x(yz)$

and

$(Wx)y = (xy)y$,

one can show that this set P also satisfies the weak fixed point property.

The following four clauses can be used when assigning a theorem-proving program the task of proving this second theorem, if we choose paramodulation as the inference rule and wish to have the program construct the appropriate y.

(1) EQUAL(x,x)
(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))
(3) EQUAL(a(a(W,x),y),a(a(x,y),y))
(4) ¬EQUAL(y,a(f,y)) | ANSWER(y)

If we wish to restrict the program to searching for a backward proof—which forces the program in this case to deduce inequalities only—we place clause (4) only in the set of support. We note that, in contrast to the discussion of the first theorem, here we are writing the clauses as we would ordinarily write them when submitting a problem of this type to a theorem-proving program. In particular, we are not using different variables in different clauses as we did earlier; instead, we let the program worry about treating variables as being relevant only to the clause in which they occur. On the one hand, continually choosing new variables to avoid confusion is a nuisance and a burden; on the other hand, not to do so increases the likelihood of error and makes it more complicated to follow the deductive process.

The program finds the following proof by contradiction.

from clause (2) *into* clause (4)
(5) ¬EQUAL(a(y,z),a(a(a(B,f),y),z)) | ANSWER(a(y,z))

from clause (3) *into* clause (5)
(6) ¬EQUAL(a(y,y),a(a(W,a(B,f)),y)) | ANSWER(a(y,y))

from clause (1) and clause (6), as the result of testing to see whether a contradiction has been found,
(7) ANSWER(a(a(W,a(B,f)),a(W,a(B,f))))

One simply replaces all occurrences of f in the ANSWER literal by x to finish the construction of the desired y that satisfies

$$y = xy$$

that is required by the weak fixed point property.

For constructing some object—as one often wishes to do in an existence theorem—a glance at this example shows how much easier it is to use the ANSWER literal than to look through a standard proof by contradiction to extract the desired object. The action of replacing all occurrences of f by x to obtain the desired y brings us again into contact with the kernel mentioned earlier, $W(Bx)(W(Bx))$. Is this a sign that we are about to embark on the important voyage, the voyage whose object is to become familiar with the systematic method—the *kernel method*—for searching for fixed point combinators?

Summarizing, we see that an automated theorem-proving program can—in addition to proving various proposed theorems that one might wish to prove—automatically extract hidden information of a certain type. Specifically, when the proposed theorem under consideration is one concerning the existence of some object, the program can be assigned the task of displaying or constructing the object.

3. Introduction to Combinatory Logic

In this brief treatment of the deep subject of combinatory logic, we shall mainly confine our attention to those aspects and definitions that are pertinent to the research presented in this report. Indeed, we can do little more than that, for, as commented in the introduction, we are novices in this area of logic. Nevertheless, the general observations we give in Section 3.2 explain why studies of the type reported here may have substantial significance.

To prepare the way for those observations and, even more, for the material we present in Section 4 and later, we pause to address one technical fine point concerning the use of the symbol f to denote an arbitrarily chosen combinator. By addressing this point here, we can fulfill the promise we made in the introduction when we said that we would explain why we use f to denote a combinator, which is the exception to our rule of using uppercase letters for that purpose. The corresponding discussion focuses on statements such as

$$((Bx)y)z = x(yz),$$

which implicitly are true for all combinators x, y, and z, versus statements such as the following. A kernel is a set of combinators Γ_f each of which reduces to $f\Gamma_f$, where f is an arbitrarily chosen combinator. (The concept of reduction is defined in Section 3.4; for the simpler and syntactic definition of kernel, one can turn to Section 4.3; for the full and semantic definition of kernel, one can turn to Section 4.5.) In the most technical sense, the first of the two given statements—that focusing on the actions of the combinator B—is an example of compact and shorthand notation. In contrast, the second of the two statements—that focusing on the definition of a kernel—is a rigorous statement, and not shorthand. Depending on one's background, this distinction is somewhat enlightening or totally obvious. Since we wish that all who read this report understand it fully, and since we are writing for a diverse audience—many of whom may never have considered the distinction in question before—we make the following detailed comments showing why one statement is shorthand and the other is not.

For example, in group theory, one writes

$$ex = x$$

to mean that the identity element e multiplied by x on its right is x, for any element x. Similarly, in ordinary arithmetic, one writes

$$0 + x = x,$$

and no confusion arises. However, in the most technical—and one might say pedantic—sense, one can neither multiply e by the variable x in a group, nor add 0 to the variable x in ordinary arithmetic. After all, neither ex nor $0 + x$ is actually defined. Rather, in a group, multiplication is defined for each pair of elements, and not for an element and a variable. Similarly, in arithmetic, addition is defined for pairs of numbers, and not for a number and a variable. Nevertheless, one is permitted to write ex and $0 + x$ as a shorthand notation to mean that, if one replaces x by an appropriate object, the resulting equation is true.

In combinatory logic, we encounter the same situation. We write

$$((Bx)y)z = x(yz)$$

even though, technically, one cannot *apply* a combinator such as B to a variable. Strictly speaking, one

can apply a combinator to another expression only if that expression is a combinator. Nevertheless, we write such equations as shorthand for saying that if one substitutes variable-free expressions for each of x, y, and z, then the resulting equation is true. No confusion results.

The connection to the concept of kernel rests with our use of the symbol f to denote an arbitrarily chosen combinator. For example, when we say that a kernel is a set of combinators Γ_f such that Γ_f reduces to $f\Gamma_f$, where Γ_f contains occurrences of f and therefore is a function of f, we are speaking precisely—one cannot, technically, reduce an expression that contains variables. When we say that Γ (containing occurrences of x) reduces to $x\Gamma$, we are using shorthand, in the sense we have just discussed. Such a statement means that, if one uniformly replaces x by any variable-free expression, then the first term reduces to the second. The justification for this shorthand notation rests with the fact that f is an arbitrarily chosen combinator. Indeed, if we prove that some expression Γ reduces to $f\Gamma$, then one can replace f by any arbitrarily chosen combinator and the proof is still valid.

Therefore, when we make various statements in which f denotes an arbitrarily chosen combinator, we could also write x in place of f since such statements are true for all x. For example, we expect that no confusion will arise when we occasionally switch from writing Γ reduces to $f\Gamma$ to writing Γ reduces to $x\Gamma$, and conversely. Equally, we may write $\Theta f = f(\Theta f)$ or $\Theta x = x(\Theta x)$ interchangeably when referring to the equation for the strong fixed point property. Because our use of f is a convention, and because of the importance of the fact that one can at any point replace f by any arbitrarily chosen combinator, we shall—perhaps far more frequently than might seem necessary—include the phrase "where f is an arbitrarily chosen combinator" so that the significance of a statement will not be missed.

Our use of f also meshes well with automated theorem proving, for we often wish to reason backward from assuming that some property fails to hold. Such an assumption frequently introduces a constant, such as f, as when denying that the weak fixed point property holds—which is in fact the assumption we make, as we shall see, when we search for a kernel. In the case where we succeed in obtaining a proof by contradiction—if one takes into account the preceding remarks—we can then say without confusion that, for example, we have proved that $\Gamma = x\Gamma$ or have proved that Γ reduces to $x\Gamma$.

Summarizing, one might continually keep in mind that f is a reserved symbol, reserved to denote an arbitrarily chosen combinator. We thus see that the use of f serves us well for the case in which we are being very careful—as in the (semantic) definition of a kernel, for example—and also for the case in which we are focusing on the denial of some property, and the focus therefore shifts to involve an appropriate constant.

3.1. Description of Combinatory Logic

Combinatory logic can be viewed as an alternative foundation for mathematics—which was Curry's proposal [Curry58,Curry72]—or viewed as a programming language. On the one hand, although Curry's original plan of producing a self-contained foundation for all of mathematics and logic was doomed to fail, nevertheless the logic offers the same generality and power as set theory in the sense that essentially all of mathematics can be embedded in it. On the other hand, any computable function can be expressed in combinatory logic, and the logic can be used as an alternative to the Turing machine. In fact, combinators have been used as the basis for the design of computers. This logic is concerned with the abstract

notion of *applying* one function to another.

For a more formal definition, we can borrow from Barendregt [Barendregt81] who defines combinatory logic as a system satisfying the combinators *S* and *K* (defined shortly) with *S* and *K* as constants, and satisfying reflexivity, symmetry, transitivity, and two equality substitution axioms for the function that exists implicitly for *applying* one combinator to another.

A quick review of Section 2.2 leads one to the correct conclusion that, especially when paramodulation is used as the inference rule, combinatory logic is clearly within the province of automated theorem proving. In particular, to study combinatory logic with the assistance of an automated theorem-proving program, we need only choose an appropriate function symbol—such as *a* to denote that one combinator is *applied* to another—and supply the axiom for reflexivity of equality and those for *S* and *K*.

```
EQUAL(x,x)
EQUAL(a(a(a(S,x),y),z),a(a(x,z),a(y,z)))
EQUAL(a(a(K,x),y),x)
```

Finally, we can give the view currently held by Hindley and Seldin [Hindley86]: "The theories of combinators and lambda calculus both have the same purpose; to describe some of the most primitive and general properties of operators and combinations of operators. In effect, they are abstract programming languages. They contain the concept of computation in full generality and strength, but in a pure distilled form with syntactic complications kept to a minimum."

In fact, when extensionality—which is defined shortly and is distantly related to, but different in a vital way from, right cancellation—is present, lambda calculus and combinatory logic are provably the same. When restricted to the study of some chosen set of combinators, such as *B* and *W*, such systems are called (by Smullyan) applicative systems [Smullyan84]. Although, technically, we shall be concerned in most cases with small applicative systems, we shall, nevertheless, simply refer to each study as combinatory logic. Let us, therefore, turn to the specific details that set the stage for Section 4.

3.2. Combinators Classified by Their Actions

One of the activities in combinatory logic is that of defining a class of combinators and then studying that class. Each combinator—more precisely, each class of combinators—is defined by writing an equation that specifies its actions on all other combinators. All combinators within a given class act the same. For example, with the convention that expressions are considered to be left associated unless otherwise indicated, each combinator in the class containing *L* has an equation essentially identical to the equation for *L*,

$$Lxy = x(yy)$$

for all combinators x and y. The only difference in the equations for the combinators in a class is the occurrence of the specific combinator whose actions are being given. In this logic, one talks about a class of combinators that act like *L* or like *S* or like *K*, just as one might talk about a group, a ring, or a vector space. Indeed, just as one can talk about some class of objects in terms of an element of that class, one can refer to combinators in the same class by a common name. For example, Smullyan [Smullyan84] refers to combinators that behave as *L* does as larks. We sometimes find it convenient—since terms like

"lark" are not widely used—to talk about an L when referring to a combinator that is in the class containing L. Such references can sometimes be confusing.

For an illustration of how one might speak, let us examine the combinator BWB with

$$Bxyz = x(yz)$$

and

$$Wxy = xyy$$

for all combinators x, y, and z. To see how the combinator BWB behaves, one applies the respective equations for B, W, and B again and finds that

$$BWBxy = W(Bx)y = Bxyy = x(yy)$$

which, by taking the first and last expressions in this sequence of equalities, yields an equation identical to the one we just gave for L except that BWB appears where L did. In combinatory logic, to express this result, one says that L has been defined with B and W. In the terms that Smullyan uses, one says that BWB is a lark. More casually, we might say that BWB is an L, or at least call BWB L, or, informally, say that $BWB = L$. In contrast, in the strict sense of equality,

$$WWB = WBB = BBB,$$

which we can see by applying the equation for W given earlier. As the following definition shows, the distinction between the two uses of equal disappears when *extensionality* is present.

Definition. *Extensionality*: for all y and for all z (if for all x, $yx = zx$), then $y = z$.

The law of extensionality asserts that two combinators are equal, in the strict sense of the term, if they behave the same—if their respective equations, when they apply, say that they have precisely the same effect on all other combinators. Therefore, when extensionality is present, saying

$$BWB = L$$

is a correct statement even with the strict interpretation of equality. To prove this, we start with

$$BWBxy = x(yy) = Lxy,$$

apply (with the appropriate renaming of variables) the definition of extensionality to conclude that

$$BWBx = Lx,$$

and again apply the definition to get

$$BWB = L$$

to complete the proof.

Extensionality can be viewed as a weakening of right cancellation—for all y, z, and x, if $yx = zx$, then $x = y$. Indeed, the law for right cancellation implies the law for extensionality. To see this, we consider the following two clauses which are, respectively, the equivalent of right cancellation and extensionality.

¬EQUAL(a(y,x),a(z,x)) | EQUAL(y,z)
¬EQUAL(a(y,h(y,z)),a(z,h(y,z))) | EQUAL(y,z)

In the second clause, the Skolem function h is used to reflect the fact that extensionality asserts the existence of an x that depends on y and z for all y and all z. To see that the second clause is implied by the first, we merely replace x in the first by $h(y,z)$ to obtain the second. Although we do not ordinarily assume extensionality to hold, we shall consider one interesting case in which it does.

3.3. Combinators Classified by Their Properties

Since our research focuses on various combinators based on certain properties—the properties of a combinator are not to be confused with the actions of a combinator—we need the following definitions. In particular, we are interested in combinators that are termed *proper*, *regular*, *eliminating*, *replicating*, *permuting*, and *isolating*.

Definition. A combinator is called *proper* if and only if (1) the left side of its equation is left associated and consists of the combinator followed by some nonempty list of distinct variables and (2) the right side consists of some or all of the variables that occur on the left side.

Definition. A combinator is called *regular* if and only if (1) the combinator is proper and (2) the first variable on the right side of its equation appears precisely once and is identical to the first variable occurring on the left side of its equation.

For example, the combinator M with

$$Mx = xx$$

is proper but not regular; the combinator T with

$$Txy = yx$$

is also proper but not regular; but the combinator L with

$$Lxy = x(yy)$$

is proper and regular. On the other hand, one paramodulates *from* the second argument of B with

$$Bxyz = x(yz)$$

into the entire first argument of W with

$$Wxy = xyy$$

to obtain

$$B(Wx)yz = x(yz)(yz),$$

the result does not satisfy the requirements for an equation of a proper combinator—its left side is not fully left associated, for example—and therefore does not satisfy the requirements for a regular combinator either. Indeed, if we were given this expression, we would say that the expression fails to establish that the combinator BW is proper, and therefore fails to establish that BW is regular. We say "fails to establish" because the given equality in which B and W occur does not settle the issue; in particular, a different paramodulation step or some other approach might yield the sought-after equation for BW.

In fact, with the following method, such questions appear to be decidable, when termination is guaranteed; by decidable, we mean that, when a question of the type is posed, a method exists that always correctly returns either a yes or a no answer. One method—that appears to show that these questions are

decidable—for answering such questions consists of writing the combinator under consideration followed by a sequence of distinct variables, applying the left sides of the corresponding combinators one at a time, and then, when all combinators have vanished, discarding the variables that have not participated in the computation.

For the case of BW, if we guess that five variables are sufficient, we get

$$BWxyzuv = W(xy)zuv = xyzzuv,$$

which leads to the conclusion that

$$BWxyz = xyzz$$

since both of the variables u and v did not participate, establishing that BW is in fact both proper and regular. The same bit of information can be found with a single application of paramodulation; simply paramodulate *from* the second argument of the equation for B *into* the term Wx of the equation for W. We again have an illustration of the use of an automated theorem-proving program as an assistant for research; in particular, one can use paramodulation instead of applying the given algorithm.

As an aside, to illustrate how one can discover additional information, we return to the equation

$$B(Wx)yz = x(yz)(yz)$$

and apply the right side of the combinator B to it to obtain

$$BBWxyz = x(yz)(yz),$$

which shows that the combinator BBW is also a regular combinator.

For a third example—an example that illustrates how carefully one must apply the requirements for the definition of regular—we consider the combinator K with

$$Kxy = x$$

and see that K is regular even though not all of the variables appearing on the left side of its equation appear on the right side. According to the following definition, such combinators, in addition to being regular, are also called *eliminators* and are termed *eliminating*. As the definition also shows, a combinator need not be regular to be eliminating.

Definition. A combinator is called an *eliminator* if and only if (1) the combinator is proper and (2) there exists at least one variable appearing on the left side of its equation that does not appear on the right side. Such a combinator is termed *eliminating*.

Next we have combinators such as W, L, and M which are called *replicators* and are termed *replicating*, combinators such as T which are called *permuters* and are termed *permuting*, and combinators such as B, L, and M which are called *isolators* and are termed *isolating*.

Definition. A combinator is called a *replicator* if and only if (1) the combinator is proper and (2) a variable appears more than once on the right side of its equation. Such combinators are termed *replicating*.

Definition. A combinator is called a *permuter* if and only if (1) the combinator is proper and (2) the order of some subsequence of variables on the right side of the equation for the combinator is a nontrivial permutation of the corresponding sequence of variables on the left side. Such combinators are termed

permuting.

Definition. A combinator is called an *isolator* if and only if (1) it is proper and (2) the right side of its equation consists of a single variable or is of the form *vt* for some variable *v* and some term *t*. Such combinators are termed *isolating*.

Included among isolators are *B*, *L*, *M*, and *K*, where

$Kxy = x$;

K satisfies the definition of an isolator because no other variables appear on its right side except for the first variable that appears there.

To illustrate how important is the difference between combinators that do and that do not isolate, we can consider *L* and *W* whose equations are identical except for the association of the variables on their respective right sides. In contrast to *L* with

$Lxy = x(yy)$,

which *is* is an isolator, the combinator *W* with

$Wxy = xyy$

is *not* an isolator; the first variable, *x*, on the right side of *W* is attached to the first occurrence of *y*.

One can use a simple syntactic test to determine whether a given combinator is isolating, a test that an automated theorem-proving program can easily use. One simply writes the equation for the given combinator in the notation illustrated in Section 2.1, using the function *a*. If the second argument of the corresponding positive equality unit clause—the right side of the combinator's equation—begins with one or fewer occurrences of the function *a*, then the combinator is isolating, and conversely.

For example, when one examines

EQUAL(a(a(W,x),y),a(a(x,y),y))

one sees that *W* is not isolating. In contrast, when one examines

EQUAL(a(a(L,x),y),a(x,a(y,y)))

one sees that *L* is isolating.

For our study of fixed point combinators, which we begin in earnest in Section 4 and which is central to this report, combinators that are replicating and combinators that are isolating play an essential role. Of course, a combinator can have more than one property from among the properties of replicating, eliminating, permuting, and isolating. For example, the combinator *L* with

$Lxy = x(yy)$

is both replicating and isolating, in contrast to *W* with

$Wxy = xyy$

which is merely replicating.

3.4. Key Concepts, Properties, and Theorems

To complete the background for the discussion of our study given in Section 4, we need to define a few additional concepts and properties. The three most important properties are, respectively, the *strong fixed point property*, the *weak fixed point property*, and the *P-reducible weak fixed point property*. The concepts to be defined are *reduction*, *standard reduction*, *generalized reduction*, *expansion*, *generalized expansion*, *fixed point combinator*, *a fixed point of a combinator*, and *a P-reducible fixed point of a combinator*. With these additions and the statement of two key theorems, we shall be ready to present the question that motivated the research we present in this report, and to discuss how it was attacked—and solved. Then we turn to the topic so central to this report, the *kernel method*.

Definition. Where P is a given set of combinators, the *strong fixed point property* holds for P if and only if there exists a combinator y such that, for all combinators x, $yx = x(yx)$, where y is expressed purely in terms of combinators in P. Any y with this property is called a *fixed point combinator*.

The simplest example we can give of a fixed point combinator is UU with

$$Uxy = y(xxy).$$

To see that UU is in fact such a combinator—although this shortcut seldom is useful—first replace all occurrences, in the equation for U, of the variable x by U, and then rename the variable y to x to get

$$UUx = x(UUx),$$

which is clearly an instance of the defining equation for fixed point combinators. Since $y = UU$ is expressed purely in terms of the combinator U, the strong fixed point property, therefore, holds for the set P consisting of U alone. The combinator U is another example of a combinator that is both replicating and isolating, but U is not regular.

Definition. Where P is a given set of combinators, the *weak fixed point property* holds for P if and only if for all combinators x there exists a combinator y such that $y = xy$, where y is expressed purely in terms of the combinators in P and the variable x.

We can immediately give two examples of sets for which the weak fixed point property holds. If P consists of U alone, and if we let y be UUx, then from the equation for U we have

$$UUx = x(UUx),$$

which shows that the weak fixed point property holds for this choice of P since y is expressed purely in terms of the combinators in P and x. Even further, we see that y does indeed depend on x—as allowed in the definition of the weak fixed point property—since different values for x produce different values for y. Of course, since one can prove, as we shall, that the strong fixed point property implies the weak fixed point property, we could have immediately applied that result to show that P satisfies the weak fixed point property since we showed earlier that it satisfies the strong.

For our second example, we let P consist of L alone and y be $Lx(Lx)$. From

$$Lxy = x(yy),$$

we have

$$Lx(Lx) = x(Lx(Lx)),$$

and again the requirements of the definition are satisfied and, as usual, y depends on x. If one now reverses the order of the questions by asking if P satisfies the strong fixed point property, one sees that the expression $Lx(Lx)$ is of no use in this context because the expression depends on x. In particular, to prove that the strong fixed point property holds for a set, one must find an expression that depends only on the elements in that set. As an application of Theorem 5 of Section 4.6, we shall prove that the set consisting of L alone fails to satisfy the strong fixed point property—one cannot construct a fixed point combinator from L alone. In Section 4, when we present our method—known as the kernel method—for searching for fixed point combinators, we shall revisit both expressions UUx and $Lx(Lx)$ as examples of the key concept of kernel on which, of course, the kernel method rests.

Definition. Where f and g are given combinators, g is a *fixed point of* f if and only if $g = fg$.

For example, since $UUB = B(UUB)$, UUB is a fixed point of the combinator B. For a second example, $LW(LW)$ is a fixed point of the combinator W. By using the concept of a fixed point of a combinator, we can give the following alternative definition for the weak fixed point property holding for a set of combinators.

Definition. Where P is a given set of combinators, the *weak fixed point property* holds for P if and only if for all combinators f there exists a fixed point g of f such that the fixed point g is expressed purely in terms of the combinators in P and f.

If the set P of combinators satisfies the strong fixed point property, then, for each combinator f, we can select a fixed point g_f of f such that the members of the resulting set of fixed points are very tightly coupled. In particular, there must exist a Θ such that Θ is a fixed point combinator expressed purely in terms of the elements of P. Then, since $\Theta x = x(\Theta x)$ for all combinators x, if f and g are two distinct combinators, their respective fixed points Θf and Θg in the set under discussion are identical except that the last symbol in one is f whereas the last symbol in the other is g. For a glimpse of what is to come when we discuss kernels and the kernel method, we shall discover that, when the weak fixed point property is possessed by one of the sets P of combinators we study, we can again construct a set of fixed points that are very tightly coupled.

Definition. The sequence $C_0, C_1, \ldots, C_k$ is a *reduction*, relative to the proper combinators $A_1, A_2, \ldots, A_k$, of the expression C to the expression D if and only if (1) $C = C_0$, (2) C_i is obtained from C_{i-1} by applying (in the spirit of equality substitution) the left side of A_i, (3) $C_k = D$, and (4) the terms that are replaced are each free of variables. We say that A_i *reduces* C_{i-1} to C_i. If the sequence $C_0, C_1, \ldots, C_k$ is a *reduction*, relative to the proper combinators $A_1, A_2, \ldots, A_k$, of the expression C to the expression D, then, for any expression B, the sequence $BC_0, BC_1, \ldots, BC_k$ is a reduction of BC to BD, and the sequence $C_0B, C_1B, \ldots, C_kB$ is a reduction of CB to DB. If, for any expressions C and D, there exists a reduction of C to D, then C is *reducible* to D, or C *reduces* to D. Therefore, for any expressions A, B, C, and D, if A reduces to B and C to D, then AC reduces to BD and there exists a reduction from AC to BD.

Definition. Where f and g are combinators, and where P is a given set of combinators, g is a *P-reducible fixed point of* f if and only if g is a fixed point of f and there exists a reduction that reduces g to fg in which all of the steps in the reduction are with elements of P.

For an example of the difference between a fixed point that is *not* and a fixed point that *is* a *P*-reducible fixed point of a given combinator, let us consider the combinators *B* and *W* with

$$Bxyz = x(yz)$$

and

$$Wxy = xyy$$

and some arbitrarily given combinator *f*. Let $h = BWBf(W(Bf))$, and let $g = W(Bf)(W(Bf))$. Then *h* is a fixed point of *f*, but not a *P*-reducible fixed point of *f*, where *P* consists of the combinators *B* and *W* alone. One can prove that *h* is a fixed point of *f* by reducing *h* to *fg*—which can be done by applying *B*, *W*, and *B*—and then replacing the first occurrence of $W(Bf)$ in *fg* by *BWBf*, which is justified since $Bxyz = x(yz)$ for all combinators *x*, *y*, and *z*. For this half of the proof, we use the fact that if one expression reduces to another, then the two expressions are equal. To complete the proof, one shows that *h* cannot reduce to *fh* by applying elements of *P*, which can be done by simply exploring the possible reduction paths. In contrast, with a slight modification of the proof we have just completed, one can prove that *g* is a *P*-reducible fixed point of *f*. With a small amount of thought based on the proof just given, one can see that if one combinator is a *P*-reducible fixed point of a second combinator, then the first combinator is expressed purely in terms of the elements of *P* and the second combinator.

Definition. Where *P* is a given set of combinators, the *P-reducible weak fixed point property* holds for *P* if and only if for all combinators *f* there exists a *P*-reducible fixed point *g* of *f*.

By using the examples we have given earlier in this section, we can easily give three examples of a set that satisfies the *P*-reducible weak fixed point property. For our first example, we consider the set *P* consisting of *B* and *W* alone and the infinite set of combinators $W(Bf)(W(Bf))$ as *f* ranges over all possible combinators. To see that *P* has the desired properties, we borrow from an earlier example to show that $W(Bf)(W(Bf))$ reduces to $f(W(Bf)(W(Bf)))$, and note that the reduction requires the use of *B* and *W* only and is independent of *f*. For our second example, we consider the set *P* consisting of the combinator *U* alone, the infinite set of combinators *UUf* as *f* ranges over all possible combinators, and argue as we did in the first example. For our third example, we consider the set *P* consisting of the combinator *L* alone, the infinite set of combinators $Lf(Lf)$, and again argue as we did in the first example.

Given these three examples, one might naturally ask about an example of a set *P* that satisfies the weak fixed point property but fails to satisfy the *P*-reducible weak fixed point property. Succinctly stated, we know of no such set. In fact, as we conjecture in Section 5.5, we suspect that every set that satisfies the weak fixed point property also satisfies the *P*-reducible weak fixed point property.

For the next definition, we review our earlier discussion of position vectors presented in Section 2.3.2, but give the discussion in terms of combinatory logic. The *position vector* of a term is a *k*-tuple that gives the relative position of the term within the equation for a combinator or within some expression involving combinators. For example, given an equation for some combinator, the position vector [2,3,1] says that the corresponding term is the first subterm of the third subterm of the second argument (right side) of the equation containing the term. For a second example, if Θ is a fixed point combinator, then the position vector of Θ itself is the empty sequence of integers, since Θ has no relative position within itself; in such a case, we denote the position vector by []. On the other hand, if we write

FIXED(Θ)

to permit an automated theorem-proving program to study Θ, as we would if we were reducing Θ, then Θ has the position vector 1 within the given clause. When determining the position vector of a term within an expression, one might find it simpler to use the notation that explicitly employs the function a for "apply", as in Section 2.1, rather than using the notation of combinatory logic. However, we need not do so for our next illustration. Let us consider, for example, the expression

$W(B(Bf))(W(B(Bf)))(W(B(Bf)))$

and the corresponding three occurrences of W. Their respective position vectors are [1,1,1], [1,2,1], and [2,1].

By inspection, one can see what is meant by saying that, for example, the first occurrence of W is to the left of the second. For other cases, however, saying one term is to the left of another is not sufficiently precise for our purposes. Therefore, for precisely comparing two position vectors, we use the following two conditions. First, the position vector $V(A)$ of the term A is an initial segment of the position vector $V(B)$ of the term B if and only if $V(B)$ can be obtained from $V(A)$ by suffixing zero or more integers. Second, the position vector $V(A)$ of the term A is lexically less than or equal to the position vector $V(B)$ of the term B if and only if (1) $V(A_i)$ and $V(B_i)$ are the earliest pair of components of the two vectors that are unequal and $V(A_i)$ is less than $V(B_i)$, or (2) $V(A)$ is an initial segment of $V(B)$. In other words, we adopt the custom of lexical ordering that one finds in a dictionary.

For example, the position vector [1,1,1] is lexically less than or equal to the position vector [1,2,1], the position vector [1] is lexically less than or equal to the position vector [1,1], and the position vector [] is lexically less than or equal to the position vector [1]. The position vector []—the empty sequence of integers—says that the entire term is under consideration, as occurs when, say, we consider reducing Θf for some fixed point combinator Θ and some arbitrarily chosen combinator f.

For the following concept, we use only the second condition directly for comparing position vectors. We focus on the first of the two conditions only when we compare that concept to existing concepts in automated theorem proving.

Definition. A reduction $C = C_0, C_1, C_2, \ldots, C_k = D$ is a *standard reduction* of C to D if and only if, for $i = 1,2,\ldots,k\text{-}1$, the position vector of the term to which A_i applies is lexically less than or equal to that to which A_{i+1} applies.

We can immediately illustrate the use of the concepts of reduction and standard reduction. For example—as we learned earlier—the expression $W(Bf)(W(Bf))$ is reducible to the expression $f(W(Bf)(W(Bf)))$ relative to the set consisting of B and W. To see this, first note that, since $Wxy = xyy$, W applied to $W(Bf)(W(Bf))$ yields $Bf(W(Bf))(W(Bf))$. If B is then applied to the result, where $Bxyz = x(yz)$, one obtains the desired result. The given reduction is a standard reduction since, visually, W is applied to the left of B—or, precisely, since the respective position vectors of the terms corresponding to the two reduction steps are [] and [].

For a second example—illustrating that one can skip possible reductions—if one considers $\Omega\Omega\Omega$ with $\Omega = W(B(Bf))$, one can apply W and then B as reductions. One obtains $Bf(\Omega\Omega)\Omega$, and one says that this result is obtained from $\Omega\Omega\Omega$ with a standard reduction. If one continues to apply reductions, but

applies W to $\Omega\Omega$—that is, if one skips the application of B as a reduction—the result is still said to be obtained from $\Omega\Omega\Omega$ with a standard reduction. In other words, a standard reduction is permitted to skip reducible terms—in the example under discussion, to skip the reducible term whose position vector is [].

Again, we have a satisfying connection between combinatory logic and automated theorem proving. From the viewpoint of automated theorem proving, a reduction $C_0, C_1, \ldots, C_k$, of C to D is merely a sequence of paramodulations such that each paramodulation is from the left side of the equation for some A_i, where A_i are proper combinators. To be standard, the position vector of the *into term* for the paramodulation *from* A_i must be lexically less than or equal to that of the *into term* for the step *from* A_{i+1}.

On the other hand, although reductions in a sense share the objective that the demodulation procedure has—the objective of producing a canonical form—certain important differences exist between the two.

First, standard reductions, as pointed out, are allowed to skip reducible terms, but demodulation is not. For example—chosen merely to easily illustrate the point—demodulation is required to apply W to $Bf(WBB)F$, and then apply B to the result. A standard reduction could consist of either of the two steps alone, but would not be permitted to make this two-step sequence, for the position vector of the first step is [1,2] and of the second is [], and [1,2] is not lexically less than or equal to []. However, if the two steps were interchanged, a standard reduction would result. Also, if one first applied B, a standard reduction permits skipping the reduction that is then possible with W. In the given example, demodulation would be required to apply W and then B, rather than simply skipping the application of W.

For a second difference, in the usual algorithm, demodulators are applied inside out and, within a given level of function nesting, left to right. Also, when a term is successfully demodulated, demodulation is required to attempt to demodulate the subterms of the rewritten term; demodulation then uses an outside-in rule. In contrast—before any terms have been rewritten—for standard reductions, the inside-out requirement of the usual demodulation algorithm is replaced with an outside-in condition. If the usual algorithm is viewed in terms of position vectors, when the program applies two demodulators, the first must apply to a term whose position vector is lexically less than or equal to the second, or the second demodulator must apply to a term whose position vector is an initial segment of the position vector of the term to which the first demodulator applies.

Definition. The sequence $C_0, C_1, \ldots, C_k$ is an *expansion*, relative to the proper combinators $A_1, A_2, \ldots, A_k$, of the expression C to the expression D if and only if (1) $C = C_0$, (2) C_i is obtained from C_{i-1} by applying (in the spirit of equality substitution) the right side of A_i, (3) $C_k = D$, and (4) the terms that are replaced are each free of variables. If there exists an expansion of C to D, then C is *expandable* to D. The other remarks contained in the definition of reduction apply here but with the concept of reduction replaced with the concept of expansion.

In other words, the sequence $C_0, C_1, \ldots, C_k$ is an expansion if and only if the sequence $C_k, C_{k-1}, \ldots, C_0$ is a reduction. For example, the expression $f(W(Bf)(W(Bf)))$ is expandable to the expression $W(Bf)(W(Bf))$ relative to the set of combinators consisting of B and W, where f is an arbitrarily chosen combinator. In the obvious sense, the concept of expansion is the reverse of the concept of reduction. Therefore, from the viewpoint of automated theorem proving, an expansion, $C_0, C_1, \ldots, C_k$, of C to D is merely a sequence of paramodulations such that each paramodulation is from the right side of the

equation for some A_i, where each A_i is a proper combinator.

Except for the concepts in which P-reducibility plays a role, the definitions we have given for reduction and expansion and the related concepts are taken from the standard treatment of combinatory logic. For our purposes, we find it convenient to generalize certain concepts slightly. In particular, we define a *generalized reduction* and a *generalized expansion* by simply dropping the requirement that the terms being replaced each be free of variables and adding the requirement that all variables in the terms being replaced must remain unchanged. We take similar actions for the term *generalized standard reduction*. In other words, from the viewpoint of automated theorem proving, the terms generalized reduction and generalized expansion share the spirit of demodulation—in the *from clause* only can one apply a nontrivial variable replacement. In contrast, the terms reduction and expansion as used in the strict sense of combinatory logic go even further than demodulation by requiring that the *into terms* contain no variables.

Finally, when we turn in Section 4 to the discussion of various equations, kernels, and superkernels, we make an additional generalization. At that point, for reductions and expansions, we allow a nontrivial replacement for variables in both the *from clause* and the *into term*. Such usage includes reductions and expansions that violate the spirit of demodulation and, in fact, extends the corresponding concepts to the spirit of paramodulation in its most general form. Unless otherwise specified, when we use terms such as reduction and expansion, we make no distinction between the usual notion of the corresponding term and either of our generalizations. When the distinction is important, we shall be specific.

Theorem 1. If there exists a reduction of C to D, then there exists a standard reduction of C to D.

Theorem 2, **Church-Rosser Theorem of Combinatory Logic**. If the expression C equals the expression D, then there exists an expression E such that C and D are each reducible to E.

The Church-Rosser theorem makes combinatory logic behave substantially unlike many other areas of mathematics and logic. In particular, knowing that two expressions are equal, one cannot ordinarily guarantee that a third exists such that the two reduce to it. Equivalently, from the viewpoint of automated theorem proving, the equality of two expressions does not imply that a third exists such that the two demodulate to it. But, in combinatory logic, such a guarantee or implication does hold.

For example, if we consider the combinator M with

$$Mx = xx,$$

we see that $MB(MW) = BB(MW)$ and also that $MB(MW) = MB(WW)$. Therefore, $BB(MW) = MB(WW)$. By the Church-Rosser theorem, there must exist a third term to which the last two cited terms each reduce. The term $BB(WW)$ is such a term.

Of far greater interest is the example that focuses on some fixed point combinator Θ. Since Θ is a fixed point combinator, $\Theta x = x(\Theta x)$ for all combinators x. Therefore, if we select any arbitrary combinator f, we can apply the Church-Rosser theorem to the two expressions Θf and $f(\Theta f)$ to conclude that some third expression must exist to which the given two each reduce. Now if we choose for our combinator f some combinator that does not occur in Θ and such that the equation for f involves at least two variables, the common expression guaranteed by Church-Rosser to which Θf and $f(\Theta f)$ each reduce must have the form $f\Xi$ for some Ξ.

Even further, we can see by applying a little extra thought that there must exist an infinite set of combinators Ξ_f such that for each combinator f the infinite set contains exactly one combinator Ξ_f, and such that for any two distinct combinators f and g that do not occur in Θ Ξ_g can be obtained from Ξ_f by replacing all occurrences of f by g. As for the combinators h that occur in Θ, each of the corresponding Ξ_h is identical to the Ξ_f for f not in Θ except that f is replaced by h. In other words, the members of the infinite set of combinators Ξ_f for all possible combinators f are very tightly coupled. Indeed, except for the occurrence of the arbitrarily chosen combinator f, one single sequence of reduction steps is sufficient to reduce $f(\Theta f)$ to $f\Xi_f$ regardless of which combinator f is under consideration. The significance of this second example of the use of the Church-Rosser theorem will become clear only when we have fully discussed in Section 5.4.1.1 the kernel method and certain open questions concerning its properties.

Theorem 3. If the strong fixed point property holds, then the weak fixed point property holds.

One can easily see how a proof of Theorem 3 might proceed. After all, if the strong fixed point property holds, then $yx = x(yx)$ for some y and for all x. To establish that the weak fixed point property holds, one must find for each x some z dependent on x with $z = xz$. One can merely let $z = yx$, which exists by the strong fixed point property.

From the viewpoint of automated theorem proving, Theorem 3 can also be proved immediately. We simply state the hypothesis, and assume that the conclusion is false, as discussed in Section 2.1. To say that the strong fixed point property holds, we write the clause

EQUAL(a(Θ,x),a(x,a(Θ,x)))

for an appropriate Skolem constant Θ, and, to deny that the weak fixed point property holds, we write the clause

¬EQUAL(y,a(f,y))

for an appropriate Skolem constant f. The program immediately finds a proof by contradiction since the two given clauses contradict each other, which we can see by substituting f for x in the first and $a(\Theta,f)$ for y in the second.

We have come to the end of our introduction to combinatory logic, which completes the required background needed to present our in-depth discussion of the question that is central to this report. That question concerns the possible constructibility of fixed point combinators when given some chosen set P of combinators to be used for the construction. In other words, given a set P of combinators, the question we now investigate concerns the possible presence of the strong fixed point property for P.

4. The Problem of the Constructibility of Fixed Point Combinators

We now turn to the promised account of our research. The object of that research is the study, for various given sets of combinators, of the presence or absence of the strong fixed point property. In particular, when the strong fixed point property is absent, for certain specific sets P, we shall show that one cannot construct a Θ with

$$\Theta x = x(\Theta x)$$

for all combinators x. When the strong fixed point property is present, for certain specific sets P, we shall construct an appropriate combinator Θ, a fixed point combinator. What makes such constructions

especially significant is that each fixed point combinator Θ is found by successfully applying a general and systematic method, the *kernel method* on which this report focuses. The kernel method appears to be the first of its kind—no other method, as far as we can ascertain, exists for systematically constructing fixed point combinators when they exist. As we develop the kernel method—beginning with the discovery of the first kernel (see Section 4.2) and culminating with the appropriate formal definitions (see Section 4.5)—to have an accurate perspective, one must keep two important points in mind. First, although we strongly conjecture in Section 5.5.1 that the kernel method is complete, we have not yet proved that the method is guaranteed to find an appropriate fixed point combinator when one exists for the set P of combinators under study. Indeed, in Section 5.4.1, we offer the corresponding open question as a challenging research topic. Second, the kernel method is not an algorithm for constructing fixed point combinators, but is, instead, a global strategy for conducting a search for such combinators.

In addition to being the first method for systematically conducting an appropriate search, the significance of the kernel method rests in part with the fact that such searches can be computationally very arduous. In other words, the attempt to prove that the strong fixed point property holds for a given set P of combinators can require an exceedingly large effort. On the other hand, for those sets that do possess the property—and that we have studied—the kernel method always succeeds, and succeeds very quickly, in constructing an appropriate fixed point combinator. Even further, without using the kernel method, the attempt to construct the desired combinator can easily fail even when the set P does satisfy the strong fixed point property.

To put our research in perspective, to make this report self-contained, and to show precisely how valuable an automated theorem-proving program proved to be for this research, we shall begin by focusing on the specific problem that marked the beginning. Our discussion will follow closely the chronology of our study from its inception to its culmination. The specific problem that marked the beginning took the form of a request of relatively narrow scope. Specifically, we were asked by our colleague Ross Overbeek to search for an approach that would enable our automated theorem-proving program ITP to construct a fixed point combinator from the set P consisting of B and W alone. We were told of Statman's success in finding such a combinator, which we call Θ_4, and we were also given a proof of its constructibility. Implicit in the request was the requirement of showing that the program could solve this problem without using Statman's result. The approach—which is markedly different from the kernel method on which this report mainly focuses—used to fulfill that request is discussed in detail in our earlier paper [McCune87] and discussed in detail here in Section 4.1.1.

Since we wish to compare the general systematic kernel method for constructing (when they exist) fixed point combinators with our earlier approach, we shall review that earlier phase of our research. In addition, by discussing Θ_4 and four other fixed point combinators constructible from B and W—which we reported in our earlier paper—we can show precisely how we discovered the key concepts of *kernel* and *superkernel* on which the kernel method rests. We can then present and develop the basic method in terms of those concepts. We shall also show why certain decisions were made at various points in our research, for we also wish to show how research can be sharply aided by using an automated theorem-proving program. Even further, we wish to shed some light on how one generally pursues some research goal, especially for those who have little experience in that regard.

Although we shall later apply (in Section 4.3.1) the kernel method to the study of B and W—the two combinators that were our original interest—we shall first apply it (in Section 4.3) to other sets of combinators whose study is far simpler. When we do turn to the application of the kernel method to the study of B and W, rather than just those five fixed point combinators presented in our earlier paper, we shall succeed in constructing an infinite class of infinite sets of such combinators. As we proceed, we shall illustrate how complex the topic of the strong fixed point property is by giving a variety of examples that, in various ways, come close but fail to have that property. As stated, our account of the research will be essentially chronological.

4.1. The Original Problem

In this section, we focus on the specific problem that motivated our entrance into the field of combinatory logic. The account is essentially chronological—presenting the problem as it was given to us, and then turning to our various attacks on the problem.

The specific problem causing our research to commence asks for a proof, obtained with an automated theorem-proving program, that the strong fixed point property holds for the set consisting of B and W alone. Formally, if one is given

$$Bxyz = x(yz)$$

and

$$Wxy = xyy$$

as combinators, the object is to use an automated theorem-proving program to prove that there exists a y such that

$$yx = x(yx)$$

for all combinators x, where y is expressed purely in terms of B and W.

This problem was brought to our attention by our colleague Ross Overbeek. His interest in this problem resulted from his study of a more general problem that he found in the book *To Mock a Mockingbird* by the logician Smullyan [Smullyan84]. The more general problem asks whether there exists a regular combinator such that, when considered with the combinator B, the resulting two-element set of combinators satisfies the strong fixed point property.

The connection of the specific problem under discussion to the general problem posed by Smullyan rests with the first solution that was found for the general problem. In particular, the logician Statman found that the set consisting of B and W alone does in fact satisfy the strong fixed point property [Statman86]; Smullyan calls combinators that satisfy the equation for the strong fixed point property sages. Smullyan, in private conversation, had informed Overbeek of Statman's result. Statman proved that the combinator that we call Θ_4 with

$$\Theta_4 = B(WW)(BW(BBB))$$

has the property that

$$\Theta_4 x = x(\Theta_4 x)$$

for all combinators x. In other words, Statman had answered Smullyan's question in the affirmative—one

can construct a fixed point combinator from B and one other regular combinator, namely, W.

We are deeply indebted to all three people—Overbeek for bringing to our attention the specific problem under discussion, Smullyan for posing the more general problem that caught Overbeek's interest, and Statman for having provided the solution on which we first focused. Indeed, the study we present in this report has brought us immense satisfaction and, in addition, may have led us to discoveries that will prove very significant for combinatory logic.

When Overbeek suggested the problem, he told us of Statman's result. Overbeek also informed us that the automated theorem-proving program ITP had failed in its attempt to solve the problem in its abstract form—there exists a fixed point combinator constructible from B and W. Specifically, when paramodulation was used as the inference rule for attacking this problem, and when the program was given the clauses for B, W, reflexivity of equality, and the clause

¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))

which is equivalent to assuming that no y exists such that for all x

$yx = x(yx)$,

the program simply deduced a large number of conclusions without finding a proof by contradiction. This lack of success is understandable, for, as we shall see shortly, the theorem in the abstract form is far more difficult to prove than it might appear.

Overbeek also informed us that, by providing a fair amount of assistance, he had succeeded in getting ITP to prove the more concrete theorem, namely, Θ_4 is a fixed point combinator constructible from B and W. The denial of this easier-to-prove theorem is

¬EQUAL(a(Θ_4,f),a(f,a(Θ_4,f)))

where f is a Skolem constant arising from assuming the theorem false. To assist the program ITP, he named the various subexpressions of Θ_4, and then submitted the corresponding problem. This attempt succeeded; ITP used paramodulation and found a proof that Θ_4 is indeed a fixed point combinator.

We can see intuitively why an automated theorem-proving program might find it far easier to prove that Θ_4 has the desired property rather than finding Θ_4 or some other combinator of the appropriate type. After all, if we compare the clauses

¬EQUAL(a(Θ_4,f),a(f,a(Θ_4,f)))

and

¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))

that arise from denying the respective theorems, we see that the first clause is free of variables; of course, Θ_4 must be fully expanded. Therefore, in a loose sense, denying the two theorems reverses their respective generality—the denial of the theorem for Θ_4 is less general than the denial of the abstractly stated existence theorem. A theorem-proving program will ordinarily make fewer deductions when focusing on a clause free of variables than when focusing on a clause in which variables are present. In particular, one would expect the program to function more effectively when asked to prove the theorem about Θ_4 than it would when asked to prove the abstract existence theorem. In other words, although an existence theorem

might admit an infinite number of solutions, the program might well find it easier to prove the logically more general theorem about some specific solution. The same is often true of mathematicians and logicians; they often use the specific solution as a clue to decide how to proceed.

With these remarks, we have fulfilled our promise (made in Section 2.1) to explain why the logically weaker theorem might indeed be harder to prove. We also correct and clarify the discussion of generality found in our earlier paper on combinators. Specifically, in our earlier paper we mistakenly talked about one theorem being more general than another and implied that we meant in the logical sense; instead, we should have talked about harder and easier to prove, or talked about abstract and concrete. Consistent with this observation, we shall refer to theorems focusing on the existence of an appropriate combinator as more abstract than related theorems focusing on some specific combinator with the appropriate properties.

4.1.1. Our Original Approach

At this point, we focus directly on how we first attempted to satisfy Overbeek's request, that of having a theorem-proving program construct (without assistance) a fixed point combinator from B and W alone.

Despite acknowledging the greater ease of proving that the given combinator Θ_4 has the desired property when compared with finding such a combinator with no prior knowledge about its structure, we nevertheless began our study with the intention of having the program ITP prove the more abstract form of the theorem concerning B and W. At the general level, our objective was to find a means for the program to succeed where it had formerly failed—to find a way for the program to construct a fixed point combinator from B and W, rather than proving that some particular combinator supplied by the researcher has the desired property. Specifically, to increase the program's efficiency sharply, our task was to formulate a strategy that would curtail the myriad of irrelevant clauses that would otherwise be generated when ITP attempted to prove the more abstractly stated theorem.

We decided to begin with ITP's paramodulation proof of the more concrete theorem, and use it as a guide for finding a proof of the more abstract theorem. We made this choice because we prefer to aim at a specific target, such as a known proof, rather than aiming at a general target in the form of finding some acceptable proof. Using Overbeek's proof of the more concrete theorem as a guide, our first step was to obtain by hand a succession of proofs—each also using paramodulation—of the more abstract theorem asserting the existence of a fixed point combinator constructible from B and W. In general, we find it profitable to produce such proofs by hand, for the corresponding search frequently leads to the seeds of useful ideas.

Our goal was to find a much shorter proof than the one we had been given by Overbeek—to find a proof in which each step is obtained by paramodulating from the right side of B or W. In other words, we sought an input proof such that each deduction is an expansion from B or W. However, we allowed ourselves to consider expansions in the most general sense, that in which paramodulation might rely on a unification such that a nontrivial substitution occurs in both the *from clause* and the *into term*. Our decision to focus on input proofs relying on generalized expansions was based in part on an examination of ITP's proof that Θ_4 is a fixed point combinator, and in part on the desire to find a strategy to restrict ITP's

application of paramodulation.

We did in fact find the desired proof—a proof that is both an input proof and a forward proof. A forward proof is one in which each deduction step involves only axioms or the hypothesis of the theorem under study. In other words, the presence or absence of a ¬ sign, meaning **not**, is not the key; rather, we are interested in semantic criteria in preference to syntactic criteria. With one exception, a forward proof, therefore, avoids the use of all clauses that arise from assuming the conclusion of the theorem under study false; the exception occurs at the last step of a proof by contradiction, that which focuses on a pair of contradictory clauses. In contrast, a backward proof is a proof all of whose steps involve a clause arising from assuming the conclusion of the theorem under study false, or involve a descendant of such a clause. For historical considerations, we include the following proof since it played an important role in our research.

Forward Input Proof by Hand of the Constructibility of a Fixed Point Combinator

(1) EQUAL(x,x)

(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

(3) EQUAL(a(a(W,x),y),a(a(x,y),y))

(4) ¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))

from the second argument of (2) *into* a(a(B,x),y) of (2)
(5) EQUAL(a(a(a(a(B,a(B,x)),y),z),w),a(x,a(a(y,z),w)))

from the second argument of (3) *into* the first argument of (5)
(6) EQUAL(a(a(a(W,a(B,a(B,x))),y),z),a(x,a(a(y,y),z)))

from the second argument of (2) *into* a(B,a(B,x)) of (6)
(7) EQUAL(a(a(a(W,a(a(a(B,B),B),x)),y),z),a(x,a(a(y,y),z)))

from the second argument of (2) *into* a(W,a(a(a(B,B),B),x)) of (7)
(8) EQUAL(a(a(a(a(a(B,W),a(a(B,B),B)),x),y),z),a(x,a(a(y,y),z)))

from the second argument of (3) *into* the first argument of (8)
(9) EQUAL(a(a(W,a(a(a(B,W),a(a(B,B),B)),x)),y),a(x,a(a(y,y),y)))

from the second argument of (3) *into* the first argument of (9)
(10) EQUAL(a(a(W,W),a(a(a(B,W),a(a(B,B),B)),x)),
a(x,a(a(a(a(a(B,W),a(a(B,B),B)),x),a(a(a(B,W),a(a(B,B),B)),x)),a(a(a(B,W),a(a(B,B),B)),x))))

from the second argument of (2) *into* the first argument of (10)
(11) EQUAL(a(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),B))),x),
a(x,a(a(a(a(a(B,W),a(a(B,B),B)),x),a(a(a(B,W),a(a(B,B),B)),x)),a(a(a(B,W),a(a(B,B),B)),x))))

from the second argument of (3) *into*

a(a(a(a(a(B,W),a(a(B,B),B)),x),
a(a(a(B,W),a(a(B,B),B)),x)),a(a(a(B,W),a(a(B,B),B)),x))of (11)

(12) EQUAL(a(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),B))),x),
a(x,a(a(W,a(a(a(B,W),a(a(B,B),B)),x)),a(a(a(B,W),a(a(B,B),B)),x))))

from the second argument of (3) *into*

a(a(W,a(a(a(B,W),a(a(B,B),B)),x)),a(a(a(B,W),a(a(B,B),B)),x))of (12)

(13) EQUAL(a(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),B))),x),
a(x,a(a(W,W),a(a(a(B,W),a(a(B,B),B)),x))))

from the second argument of (2) *into*

a(a(W,W),a(a(a(B,W),a(a(B,B),B)),x)) of (13)

(14) EQUAL(a(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),B))),x),
a(x,a(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),B))),x)))

Clause (14) contradicts clause (4), and the proof is complete.

A glance at clause (14) shows that we indeed have succeeded in constructing the fixed point combinator Θ_4. In the notation of combinatory logic in which expressions are left associated unless otherwise indicated, the given proof constructs

$\Theta_4 = B(WW)(BW(BBB))$.

4.1.2. A Study of Our Proof by Hand

Let us pause to examine what one might learn from the given proof, why one might prefer such a proof, and how one might apply the knowledge to other problems in combinatory logic. Intuitively, the motivation for using expansions rests with the desire to construct a fixed point combinator from B and W, which in turn suggests that the object is to "build" an expression from B and W. Since expansions introduce additional occurrences of the combinators occurring on their left sides, one might conjecture that, rather than reductions, expansions play an important role in such constructions. In particular, expansions—and therefore, the corresponding paramodulation steps—attempt to juxtapose (more or less) one combinator with another, if the *into term* is appropriately chosen. We therefore have an intuitive explanation for preferring expansions to reductions—and even preferring them to using some of each—when reasoning forward from some set of combinators with the objective of constructing some combinator with a designated set of properties.

In fact, the given preference for expansions can be applied to other types of construction problems in combinatory logic. For example, let us consider the possible construction from B and W of a combinator that behaves as L does. In other words, in terms that are used in combinatory logic, let us attempt to define L in terms of B and W. As usual, we begin by assuming that such a construction is *not* possible using B and W alone, that no combinator exists that is expressible purely in terms of B and W and that behaves as L does with L satisfying

$Lxy = x(yy)$.

Such an assumption yields

$xf(x)g(x) \neq f(x)(g(x)g(x))$

for all x and for appropriate Skolem functions f and g.

Purely for convenience, let us abbreviate $f(x)$ and $g(x)$ with f and g, respectively, even though, technically, such an action corresponds to proving a weaker theorem; one can verify that the argument we now give applies equally without these abbreviations. Although for our purposes here we can equally apply reductions or expansions to complete the argument, later we shall see that situations exist for which one of the choices is not acceptable, especially when we discuss finding kernels. We choose to apply expansions. If we now attempt to expand the inequality with B and W—equivalently, paramodulate from the right sides of B and W—we discover that the only successful applications involve a term on the right side of the inequality.

The following quick proof is obtained, which we write as a mathematician might.

$xfg \neq f(gg) = Bfgg = W(Bf)g = BWBfg$

The proof is complete since x ranges over all combinators; in particular, one can set x to BWB to obtain a contradiction. From the viewpoint of automated theorem proving, the program would succeed in finding an equivalent contradiction by deducing the clause

¬EQUAL(a(a(x,f),g),a(a(a(a(B,W),B),f),g))

which contradicts the clause

EQUAL(x,x)

for reflexivity, a clause that is always included for problems involving equality when the chosen inference rule is paramodulation.

We complete our brief examination of the proof that, from B and W, one can construct a fixed point combinator by noting that, after the first seven steps, the combinator Θ_4 has been constructed, and noting that the last three steps apply to the second argument to transform it to $x(\Theta_4 x)$. Therefore, were it not for the possibility of deducing far too many conclusions, we might have immediately had the program ITP attempt to find this proof—or at least attempt to find one like it. In other words, even with the restriction of requiring that each step of the proof involve one of the input clauses, our experience with theorem-proving programs nevertheless suggested that a forward search for this type of proof might still generate a very large number of clauses. Therefore, we decided that additional action must be taken before we could have the program attempt to solve the given problem.

4.1.3. Extending Our Original Approach

Because of the concern for possible combinatoric explosion—later confirmed by an actual test with ITP—we chose to attempt to invert our forward proof, which brings us back to our chronological account of our attack on the given problem of proving the theorem that asserts that a fixed point combinator exists that is constructible from B and W. Again we chose to seek the inverted proof by hand, starting with the proof just discussed. We succeeded. We produced a proof that reasons backward from the denial of the theorem and that consists only of reduction steps from B and W. Even further, if one takes its steps in order and pairs them with the steps of the given forward proof, but taken in the reverse order starting with the next to the final step of the forward proof, one obtains a set of contradictory pairs of conclusions. One

must, however, pair the last step of the backward proof with one of the input clauses. In other words, the backward proof we produced by hand is indeed the inverse, in the obvious sense, of the cited forward proof.

For the following proof, one paramodulates from the left side of the appropriate clause. In contrast to the forward proof given earlier, here we abbreviate the proof by omitting the specific *into term* relevant for each step. Each deduced clause in the following proof conflicts with a clause in the forward proof. When we use notation of the form CONF 13, we mean that the clause under consideration conflicts with clause (13) of the preceding forward proof.

Backward Input Proof by Hand of the Constructibility of a Fixed Point Combinator

(1) EQUAL(x,x)

(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

(3) EQUAL(a(a(W,x),y),a(a(x,y),y))

(CONF 14)
(4) ¬EQUAL(a(x,f(x)),a(f(x),a(x,f(x))))

from (2) *into* (4), (CONF 13)
(5) ¬EQUAL(a(a(a(B,x),y),f(a(a(B,x),y))),
a(f(a(a(B,x),y)),a(x,a(y,f(a(a(B,x),y))))))

from (3) *into* (5), (CONF 12)
(6) ¬EQUAL(a(a(a(B,a(W,x)),y),f(a(a(B,a(W,x)),y))),
a(f(a(a(B,a(W,x)),y)),a(a(x,a(y,f(a(a(B,a(W,x)),y)))),
a(y,f(a(a(B,a(W,x)),y))))))

from (3) *into* (6), (CONF 11)
(7) ¬EQUAL(a(a(a(B,a(W,W)),x),f(a(a(B,a(W,W)),x))),
a(f(a(a(B,a(W,W)),x)),a(a(a(x,f(a(a(B,a(W,W)),x))),
a(x,f(a(a(B,a(W,W)),x)))),a(x,f(a(a(B,a(W,W)),x))))))

from (2) *into* (7), (CONF 10)
(8) ¬EQUAL(a(a(W,W),a(x,f(a(a(B,a(W,W)),x)))),
a(f(a(a(B,a(W,W)),x)),a(a(a(x,f(a(a(B,a(W,W)),x))),
a(x,f(a(a(B,a(W,W)),x)))),a(x,f(a(a(B,a(W,W)),x))))))

from (3) *into* (8), (CONF 9)
(9) ¬EQUAL(a(a(W,a(x,f(a(a(B,a(W,W)),x)))),a(x,f(a(a(B,a(W,W)),x)))),
a(f(a(a(B,a(W,W)),x)),a(a(a(x,f(a(a(B,a(W,W)),x))),
a(x,f(a(a(B,a(W,W)),x)))),a(x,f(a(a(B,a(W,W)),x))))))

from (3) *into* (9), (CONF 8)

(10) ¬EQUAL(a(a(a(x,f(a(a(B,a(W,W)),x))),a(x,f(a(a(B,a(W,W)),x)))),
a(x,f(a(a(B,a(W,W)),x)))),a(f(a(a(B,a(W,W)),x)),
a(a(a(x,f(a(a(B,a(W,W)),x))),a(x,f(a(a(B,a(W,W)),x)))),
a(x,f(a(a(B,a(W,W)),x))))))

from (2) *into* (10), (CONF 7)

(11) ¬EQUAL(a(a(a(x,a(y,f(a(a(B,a(W,W)),a(a(B,x),y))))),
a(a(a(B,x),y),f(a(a(B,a(W,W)),a(a(B,x),y))))),
a(a(a(B,x),y),f(a(a(B,a(W,W)),a(a(B,x),y))))),
a(f(a(a(B,a(W,W)),a(a(B,x),y))),a(a(a(a(a(B,x),y),f(a(a(B,a(W,W)),
a(a(B,x),y)))),a(a(a(B,x),y),f(a(a(B,a(W,W)),a(a(B,x),y))))),
a(a(a(B,x),y),f(a(a(B,a(W,W)),a(a(B,x),y)))))))

from (2) *into* (11), (CONF 6)

(12) ¬EQUAL(a(a(a(x,a(y,a(z,f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z)))))))),
a(a(a(B,x),a(a(B,y),z)),f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z))))),
a(a(a(B,x),a(a(B,y),z)),f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z))))),
a(f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z)))),
a(a(a(a(a(B,x),a(a(B,y),z)),f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z))))),
a(a(a(B,x),a(a(B,y),z)),f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z))))),
a(a(a(B,x),a(a(B,y),z)),f(a(a(B,a(W,W)),a(a(B,x),a(a(B,y),z)))))))

from (3) *into* (12), (CONF 5)

(13) ¬EQUAL(a(a(a(a(x,a(y,f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))))),
a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))),
a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))),
a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))),
a(f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y)))),
a(a(a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))),
a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y))))),
a(a(a(B,W),a(a(B,x),y)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,x),y)))))))

from (2) *into* (13), (CONF 2)

(14) ¬EQUAL(a(a(a(x,f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x))))),
a(a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x))))),
a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x)))))),
a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x))))),
a(f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x)))),
a(a(a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x))))),
a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x))))),
a(a(a(B,W),a(a(B,B),x)),f(a(a(B,a(W,W)),a(a(B,W),a(a(B,B),x)))))))

Clause (14) contradicts clause (2), and the proof is complete.

At this point in our study, we would have preferred to use the forward and the backward proofs (each obtained by hand) to guide our attack with ITP by having the program conduct two separate searches—a search for a forward input proof using expansions only, and a search for a backward proof using reductions only. Unfortunately, the first of the two searches was at the time out of reach, for we were not able to restrict ITP to the consideration of input deductions only, deductions such that at least one of the pair of clauses to which the inference rule is applied is an input clause; we did add that option to the program later. Therefore, at least with regard to a possible search for a forward proof, we decided to settle for less than our preference, and instructed ITP to search forward from the axioms, restricted only by the requirement that every attempt to draw a conclusion be based on the use of expansions.

Our decision to have the program search for a forward proof might seem odd in view of our reservation, expressed in the preceding section, concerning the possibly large set of deductions that might be drawn by ITP. No, we had not changed our minds—we had little hope of ITP succeeding, especially since we were settling for a less restricted search than desired. Rather, we pursued this course of action because the cost to us was minimal, because an incomplete attempt might nevertheless contain valuable results, and because our judgment concerning the likelihood of success might be in error—the program ITP might find a forward proof. As it turned out, the program did fail to find a proof while searching forward; far too many clauses were deduced.

We then had ITP attempt to find a proof by contradiction by reasoning backward from the denial of the desired result, subject to the restriction of applying reductions only. For this search, we were also able to have ITP restrict the application of paramodulation to considering input deductions only. We imposed this restriction by having the set of support consist solely of negative clauses, which in turn caused the program to deduce negative clauses only and prevented it from considering deductions that focus on a pair of clauses neither of which is an input clause. Again the program failed to find a proof for, although this search was less prolific than the forward search, too many clauses were deduced.

Because of the amount of CPU time required in each of the two unsuccessful attempts to find a proof, and because each of the searches had failed to duplicate a major portion of the corresponding proof by hand, it seemed obvious that far too much computer time would be required to force either branch to its completion. However, since both the forward and the backward searches had made substantial progress when compared to the proofs we were using as a guide, we conjectured that the two searches taken together might already contain a proof by contradiction. Therefore, based on criteria we give almost immediately, we extracted from each of the two searches a subset of conclusions; the consideration of the two subsets together turned out to be sufficient for reaching our original goal—the goal of having ITP prove the more abstract theorem under study.

In particular, as reported in our earlier paper, we had the program consider 2500 clauses obtained from the forward search and 4500 obtained from the backward search with the object of finding a clause from each set such that the two clauses contradicted each other. Noting that, in both searches, we had replaced clause (4) with a clause obtained by merely interchanging its arguments, we had the program focus on the clauses, deduced in the two searches, that might be of use. From the forward search, we included all 2500 clauses, all of whose first argument is of the form $a(x,t)$ for some term t not a variable; no other clauses could contradict any of the inequalities that were deduced by the backward search. (No

clause among the 2500 deduced by the forward search had an argument of the form $a(x,y)$.) The justification rests with the observation that each clause deduced by the backward search has the property that its first argument is an expression of the form $a(f(x),r)$ for some term r not a variable, and, among the various terms t and r, at least two might exist that permit the completion of a proof by contradiction.

4.1.4. Results of Our Original Approach

When the theorem-proving program ITP was assigned the task of finding a contradictory pair of clauses by considering all 2500 clauses selected from the forward search versus all 4500 selected from the backward search, the program was in effect required to make 11,000,000 comparisons—each equality versus each inequality. Actually, because of the use of a sophisticated indexing mechanism, the program did not make nearly as many as 11,000,000 comparisons. The completion of this assignment revealed far more than we had sought when trying to prove, in its abstract form, that the strong fixed point property holds for the set of combinators consisting of B and W alone.

In particular, rather than simply proving the existence theorem—which was our goal—and in the process constructing Statman's fixed point combinator Θ_4, we had instead proved the theorem and found five distinct fixed point combinators constructible from B and W alone. All five of the fixed point combinators, called Θ_1, Θ_2, Θ_3, Θ_4, and Θ_5, are given in the earlier paper; in that paper, they were called, respectively, N, M, L, J, and K. The precise structure of each is found by examining the instantiations required to establish the corresponding proofs by contradiction—the proofs consisting of an equality and an inequality that contradict each other. In other words, rather than directly constructing each of Θ_1 through Θ_5, we extracted these combinators from the corresponding proof by contradiction. In Section 4.3 and its subsections, we shall study Θ_1 through Θ_5, but from the viewpoint of constructing each by using the kernel method. That study will demonstrate the effectiveness and power offered by the new approach for such constructions.

The entire search that succeeded in finding the five distinct solutions (Θ_1 through Θ_5) to the problem of constructing fixed point combinators from B and W alone required approximately 35 CPU hours on an Encore Multimax (1 processor), which is the equivalent of 20 CPU hours on a Sun 3 workstation. A bidirectional search, restricted as discussed earlier, did in fact find Θ_4 in 1 hour on a Sun 3 workstation.

We conjectured that a far more efficient attack was possible, if appropriate strategies could be formulated. For example, could we use canonical forms to produce a sharp increase in efficiency? If so, demodulation would serve well. Of course, no problem exists with using B as a demodulator; in fact, we did take that action when searching for backward proofs. But the use of W presents great difficulty. If used as a demodulator, W has the potential of looping, non-termination. As a trivial example, W applied to WWW does not terminate. Nevertheless, perhaps we could use B as a demodulator, and apply W judiciously.

Instead, or even in addition, perhaps we could find a more powerful strategy for severely restricting the application of paramodulation even more than it was restricted in the proofs we produced by hand. If so, we could confine the program's attention to input proofs, and we could have the program use the more powerful strategy to further restrict both the use of reductions and the use of expansions.

In fact, a hierarchy of approaches might exist. For example, for forward proofs, one could simply paramodulate freely from all retained clauses. One could then refine this approach to require that only input clauses be used as *from clauses*. One could further refine the approach to require that paramodulation be applied only from the right side of the input clauses—require that only expansions be used. Finally, if the desired restriction strategy could be found—which it was, as we shall see in Sections 4.3 and 4.3.3—then one might, for example, require that the *into term* satisfy some property. Corresponding remarks hold for backward proofs.

What was clear, when we reviewed our progress and the results obtained with ITP, was that some new approach or strategy was needed. In particular, we concluded that starting with the denial of the abstract existence theorem, with the intention of completing a backward proof by contradiction, would require a substantial—and possibly an inordinate—amount of CPU time. On the other hand, an analysis of the forward search showed that even more CPU time would be required to complete a (forward) proof by contradiction starting with the axioms of *B* and *W*. Our conclusion was—and still is—that any of the usual approaches for using a theorem-proving program to solve the problem under study would indeed require too much CPU time. Therefore, only one choice remained; we must experiment further in search of something new.

4.2. The Discovery of the First Kernel

These next experiments proved most fruitful, eventually leading to the systematic approach—the kernel method on which this report focuses—for searching for fixed point combinators. The experiments were designed to see how valuable demodulation might be for increasing the efficiency of a search for fixed point combinators constructible from B and W.

Since the first useful observation produced by our experiments concerns certain as-yet-undefined properties shared by the five fixed point combinators Θ_1 through Θ_5, we need the definitions of these combinators and the definition of the concept of irreducible.

Θ_1 $((B((B((B(WW))W))B))B)$

Θ_2 $((B((B(WW))W))((BB)B))$

Θ_3 $((B((B(WW))((BW)B)))B)$

Θ_4 $(B(WW))((BW)((BB)B))$ /* Statman's fixed point combinator */

Θ_5 $(B(WW))((B((BW)B))B)$

Definition. An expression Λ is *irreducible* with respect to some given set P of combinators if and only if no reduction with an element from P applies successfully to Λ. More generally, we say that Λ is irreducible if and only if Λ is irreducible with respect to the set P consisting of the combinators that occur in Λ.

Our first useful observation was that, although the five fixed point combinators Θ_1 through Θ_5 are each irreducible with respect to the set P consisting of B and W, the corresponding expressions $\Theta_1 f$, $\Theta_2 f$, $\Theta_3 f$, $\Theta_4 f$, $\Theta_5 f$, which come from denying that each is a fixed point combinator, are not irreducible. One can easily see by inspection that Θ_1 through Θ_5 are each irreducible with respect to P. However, for far more complicated expressions, one might not find it so easy to use inspection for such a determination. In such an event, one can use an automated theorem-proving program for such tasks in the following manner.

With an automated theorem-proving program, one can immediately establish the irreducibility with respect to B and W of each of Θ_1 through Θ_5 by merely using demodulation on the expression under study. For example, if we employ the notation from automated theorem proving to study the combinator Θ_1, we see that

Θ_1 = a(a(B,a(a(B,a(a(B,a(W,W)),W)),B)),B)

cannot be demodulated with

B = EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

or with

W = EQUAL(a(a(W,x),y),a(a(x,y),y))

when the demodulator is applied left to right—applied as a reduction—regardless of the chosen *into term*. Of course, if applied as an expansion, then, for example, B does demodulate Θ_1, but expansions were not of interest at this point in our research.

To see that the second half of our observation holds—that concerning the reducibility, rather than the irreducibility, of various expressions related to Θ_1 through Θ_5—we can again rely on demodulation. To begin with, the expressions to be studied—$a(\Theta_1,f)$ through $a(\Theta_5,f)$—come from attempting to find a proof by contradiction establishing that each of Θ_1 through Θ_5 is a fixed point combinator. For example, the assumption that Θ_4 is not a fixed point combinator yields the clause

¬EQUAL(a(Θ_4,f),a(f,a(Θ_4,f)))

where, as expected, f is an appropriate Skolem constant. Next, if we write out fully each of $a(\Theta_1,f)$ through $a(\Theta_5,f)$ and then repeatedly apply B as a demodulator to each, the single common expression

a(a(W,W),a(W,a(B,a(Bf))))

is obtained. Although it is far from obvious, we shall shortly see (in Section 4.3 and its subsections) that this common expression contains the key to discovering our systematic kernel method for searching for fixed point combinators; indeed—as we shall also see—if the appropriate reductions are applied to the expression, one obtains a kernel.

However, before we elaborate on the role played by the common expression, and before we resume our chronological account, we pause to show how fortunate researchers can be. In particular, had we still been focusing directly on proving the abstract theorem that asserts that the strong fixed point property holds for the set P consisting of B and W alone—rather than studying the particular combinators Θ_1 through Θ_5—we would have begun by assuming the theorem false to obtain the clause

¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))

for the Skolem function f. In that case, we would not have discovered that $\Theta_1 f$ through $\Theta_5 f$ all reduce to one common expression when repeatedly reduced with B. Having missed this discovery, we would not have continued along the path that eventually led to our formulation of the concept of kernel. Finally, unless some other path had led to this concept, we would not have found the systematic method central to this report. In other words, were it not for the fortunate discovery of Θ_1 through Θ_5 which caused us to study them rather than continuing to directly study the abstract theorem concerning the possible presence of the strong fixed point property for P consisting of B and W, we might have failed to discover the vital concepts. Their discovery, of course, provides additional evidence of the value of using an automated theorem-proving program as an assistant for research.

Having completed our discussion of the fortunes of research in general, let us return to the chronological account of our study of B and W. We did not immediately grasp the full significance of our discovery that $\Theta_1 f$ through $\Theta_5 f$ all reduce to the common expression

$WW(W(B(Bf)))$,

which is the way one would write this expression in the notation of combinatory logic. We knew, of course, that W could not be used as a demodulator; too many terms exist for which the uncontrolled application of the corresponding reduction leads to non-termination. For a trivial example, the expression $a(a(W,W),W)$ will demodulate to itself—forever.

However, to see what would be obtained, we did study the results of applying (where appropriate) both W and B, as reductions, one step at a time to

$WW(W(B(Bf)))$.

One obtains the following sequence of equalities in which each expression is obtained from the preceding by the appropriate reduction—a reduction, from the viewpoint of automated theorem proving, such that the corresponding *into term* has a position vector of all 1's. Restricting reductions, as well as expansions, to *into terms* whose position vector consists of all 1's will turn out to be most useful (see Sections 4.3 and 4.3.3), and the corresponding restriction strategy is the new powerful strategy to which we alluded to earlier.

$WW(W(B(Bf))) =$
$W(W(B(Bf)))(W(B(Bf))) =$
$W(B(Bf))(W(B(Bf)))(W(B(Bf))) =$
$B(Bf)(W(B(Bf)))(W(B(Bf)))(W(B(Bf))) =$
$Bf(W(B(Bf)))(W(B(Bf)))(W(B(Bf))) =$
$f(W(B(Bf)))(W(B(Bf)))(W(B(Bf)))$

4.2.1. The First Kernel and the Strong Fixed Point Property for *B* and *W*

We can focus on the set P consisting of B and W alone, and immediately mine this set of equalities for results concerning the strong fixed point property, the weak fixed point property, and extensionality. In addition, we can revisit this set in the next section to extract results concerning kernels, their relation to the weak fixed point property, and the formulation of the systematic method for searching for fixed point combinators. At that time, we shall carefully study the expression $\Omega\Omega\Omega$ with

$\Omega = W(B(Bf))$,

and the fact that $\Omega\Omega\Omega$ not only equals $f(\Omega\Omega\Omega)$ but, far more significant, reduces to $f(\Omega\Omega\Omega)$.

Let us begin by using the set

$WW(W(B(Bf))) =$
$W(W(B(Bf)))(W(B(Bf))) =$
$W(B(Bf))(W(B(Bf)))(W(B(Bf))) =$
$B(Bf)(W(B(Bf)))(W(B(Bf)))(W(B(Bf))) =$
$Bf(W(B(Bf)))(W(B(Bf)))(W(B(Bf))) =$
$f(W(B(Bf)))(W(B(Bf)))(W(B(Bf)))$

of equalities—and the earlier result concerning the common expression to which $\Theta_1 f$ through $\Theta_5 f$ reduce—to show that the strong fixed point property holds for the set P consisting of B and W alone. Rather than focusing directly on the corresponding existence theorem stated in the abstract form, we shall show that the combinator Θ_4 has the property

$\Theta_4 x = x(\Theta_4 x)$,

and the proof will be complete.

As is so typical in mathematics and in logic—and as is almost always the case in automated theorem proving—we begin by assuming that Θ_4 fails to have the desired property. We obtain, in the notation of combinatory logic,

$$B(WW)(BW(BBB))f \neq f(B(WW)(BW(BBB))f)$$

for some Skolem constant f. To justify the existence of f as one would in mathematics, one could say that the assumed falseness of the theorem

$$\Theta_4 x = x(\Theta_4 x)$$

for all x implies that for some element f the strong fixed point property fails to hold. If we now reduce both sides of the inequality with three applications of B to get

$$WW(W(B(Bf))) \neq f(WW(W(B(Bf)))),$$

we encounter two occurrences of the common expression to which each of $\Theta_1 f$ through $\Theta_5 f$ reduces. But that expression is identical to the first element in the given sequence of equalities. We can, therefore, apply W twice—as we did in obtaining the sequence of equalities—to each of the two occurrences of the common expression. We obtain

$$W(B(Bf))(W(B(Bf)))(W(B(Bf))) \neq$$
$$f((W(B(Bf)))(W(B(Bf)))(W(B(Bf)))).$$

We then use the sequence of equalities to further reduce the left side of the resulting inequality to get

$$f(\Omega\Omega\Omega) \neq f(\Omega\Omega\Omega)$$

where

$$\Omega = W(B(Bf)),$$

which, as one says in mathematics, is a contradiction, and the proof is complete. Alternatively, from the viewpoint of automated theorem proving, the proof is also complete since the program must have reflexivity

EQUAL(x,x)

as an input clause for problems in which equality is present and paramodulation is the choice of inference rule.

By summarizing the proof we have just completed, we can provide a glimpse of where we are headed—in particular, we can touch on the importance of kernels for constructing fixed point combinators. For our proof, we began by assuming that the strong fixed point property does not hold. However, rather than assuming that no combinator Θ exists satisfying

$$\Theta x = x(\Theta x),$$

we instead assumed the theorem false for the specific combinator Θ_4. In contrast, when we present the kernel method, we shall see that one does not ordinarily have—nor does one need to have—the luxury of knowing of a specific combinator on which to focus. After all, when one begins a new study, in general no obvious candidate for a fixed point combinator is available. Typically, one must instead search with essentially no clues, which is one reason why the kernel method for constructing fixed point combinators is so attractive—the method does not depend on guidance from the researcher.

Continuing our summary, from the assumption that Θ_4 is not a fixed point combinator, we obtain the inequality

$\Theta_4 f \neq f(\Theta_4 f)$,

but with Θ_4 fully expressed in terms of B and W. We then reduce each side of this inequality by applying B three times to obtain a second inequality

$WW\Omega \neq f(WW\Omega)$

with

$\Omega = W(B(Bf))$,

and obtain a third inequality

$\Omega\Omega\Omega \neq f(\Omega\Omega\Omega)$

by reducing both sides of the second inequality with two applications of W. The left side of this third inequality is a *kernel*—an expression Γ such that Γ reduces to $f\Gamma$ subject to certain additional conditions, where f is the Skolem constant occurring in the denial that Θ_4 is a fixed point combinator. In other words, the third inequality has the form

$\Gamma \neq f\Gamma$

with

$\Gamma = \Omega\Omega\Omega$.

At this point, we cease applying reductions to the right side of the inequality, but continue to reduce the left side until a contradiction of the form

$f\Gamma \neq f\Gamma$

is obtained.

The kernel $\Omega\Omega\Omega$ with

$\Omega = W(B(Bf))$

turns out to be the key to our entire study of fixed point combinators constructible from B and W. In fact, rather than being relevant just to the construction of Θ_1 through Θ_5, the discovery of the kernel $\Omega\Omega\Omega$ led us to a most unexpected result. (We delay giving the definitions of *kernel* and of the related concept *superkernel* until Section 4.3; there we learn that, from a syntactic viewpoint, a kernel is an expression Γ that reduces to $x\Gamma$. In that section, we also present the appropriate theorems, various examples of kernels, and the general systematic method for using kernels for attempting to construct fixed point combinators when given some set P of combinators. In Section 4.5, we give the semantic and preferred definition of a kernel and of related concepts; there we learn that a kernel with respect to a set P of combinators is actually a set of tightly coupled P-reducible fixed points.)

We make one final observation before considering the weak fixed point property for the set P consisting of B and W. When studying the strong fixed point property for P, instead of following the course focusing on proving that Θ_4 has the appropriate properties, one could have searched for and found the kernel $\Omega\Omega\Omega$, noting that $\Omega\Omega\Omega = f(\Omega\Omega\Omega)$. Then, rather than reducing, one could have expanded $\Omega\Omega\Omega$ with B and W, eventually finding the five fixed point combinators Θ_1 through Θ_5. As we shall see in Section 4.3.2, such a two-stage search—attempting to find a kernel and, if successful, then attempting to find a corresponding fixed point combinator—is far more efficient than conducting the search for such an

object directly and in a single step.

4.2.2. Kernels Versus the Weak Fixed Point Property

To see directly—rather than relying on the appropriate theorem from combinatory logic—that the weak fixed point property holds for the set consisting of B and W alone, one merely replaces the constant f in $\Omega\Omega\Omega$ with x to obtain $\Delta\Delta\Delta$, and then examines the equalities obtained by applying to $\Delta\Delta\Delta$ the reductions corresponding to W, B, and B in order. Similar to the study of $\Omega\Omega\Omega$, one immediately discovers that $\Delta\Delta\Delta$ reduces to $x(\Delta\Delta\Delta)$, which of course implies that

$$\Delta\Delta\Delta = x(\Delta\Delta\Delta)$$

for all x. In other words, we have shown that for all x there exists a y such that

$$y = xy,$$

which is precisely the weak fixed point property. In particular, the y that must exist and depend on each chosen combinator E is obtained by replacing x uniformly in $\Delta\Delta\Delta$ by the chosen E.

The distinction between the concept of kernel (of a certain type) and the concept of some y that satisfies the weak fixed point property will become clearer in the next section. For now, one should not conclude that all y with

$$y = xy$$

are kernels or, equivalently, that the class of kernels is identical to the class of those elements that establish the presence of the weak fixed point property. The distinction does not rest with the constant f versus the variable x, for the syntactic definition of kernel allows for occurrences of x where one might have expected f to occur. Rather, the distinction rests with the difference between equal and reducible, as well as other properties. In particular, if A reduces to B, then $A = B$; the converse, however, is far from true. In that direction, the closest we get in most cases is the Church-Rosser theorem which says that if $A = B$, then there exists a C such that both A and B reduce to C.

With two examples, we can quickly sample the flavor of the difference between kernels and combinators that satisfy the weak fixed point property. First note that $x(\Delta\Delta\Delta)$ reduces to $x(x(\Delta\Delta\Delta))$, but $x(\Delta\Delta\Delta)$ is not the type of kernel on which we mainly focus in this report, for it lacks one of the required properties—the property that all reductions that map it to $x(x(\Delta\Delta\Delta))$ must satisfy the 1's rule. However, $x(\Delta\Delta\Delta)$ is a y satisfying the weak fixed point property for the set P consisting of B and W alone, and is in fact a semisimple kernel (see Section 4.5). For the second example, let

$$\Xi = W(BBBx),$$

which is obtainable from Δ by expanding with B. We then have

$$\Xi\Delta\Delta = \Delta\Delta\Delta = x(\Delta\Delta\Delta) = x(\Xi\Delta\Delta),$$

showing that $\Xi\Delta\Delta$ satisfies the weak fixed point property for the set P consisting of B and W alone. But $\Xi\Delta\Delta$ is not a kernel of any type since it does not reduce to $x(\Xi\Delta\Delta)$.

4.2.3. Extensionality and the Equality of Θ_1 through Θ_5

In this section, where we show that, when the law of extensionality is assumed, the five combinators Θ_1 through Θ_5 are all equal, we focus on the consequences that that discovery had for our research. Extensionality asserts that two combinators are equal if they have the same effect on other combinators.

Let us briefly focus on extensionality and on the information we extracted from the set of equalities given earlier. One might, at first glance, view right cancellation as a strengthening of extensionality —for all y for all z (if for all x, $yz = zx$), then $y = z$. After all, right cancellation—if, for all x, y, and z, $yx = zx$, then $y = z$—implies extensionality. Nevertheless, such a view is essentially wrong; the relation is syntactic rather than semantic. Such a view, therefore, can be misleading. Still, for some, familiarity with right cancellation may aid in understanding extensionality.

Extensionality is frequently assumed to be present when studying combinatory logic. In its presence, some of the results already given lead to interesting consequences. In particular, we showed earlier that each of $\Theta_1 f$ through $\Theta_5 f$ reduces to the common expression $WW(W(B(Bf)))$. If we now replace f by x, where x ranges over all combinators, the same argument shows that each of $\Theta_1 x$ through $\Theta_5 x$ reduces to $WW(W(B(Bx)))$—equivalently, from the viewpoint of theorem proving, repeated demodulation with B produces the same information from the five fixed point combinators under discussion. (That observation was one factor that eventually led to the discovery of our first kernel, the cube of $W(B(Bx))$, and that discovery caused us to pursue a research path that culminated much later in the semantic definition of kernels and related concepts that we give in Section 4.5. The other factor motivating this phase of our research was Smullyan's comments [Smullyan86] concerning the consequences of assuming extensionality.) Therefore, all five expressions—$\Theta_1 x$ through $\Theta_5 x$—are equal. We can immediately apply the definition of extensionality to deduce that, in its presence, all five of Θ_1 through Θ_5 are equal.

We were led to this result by a letter from Smullyan in which he proved that, in the presence of extensionality, $\Theta_4 = \Theta_5$ and $\Theta_1 = \Theta_2$. We used the program ITP to complete the proof of the equality of all five combinators—when extensionality is present—and published the result in our earlier paper on the subject. However, rather than producing for us a feeling of triumph, this result actually produced disappointment—disappointment because, in the obvious sense, the five fixed point combinators are not as distinctly different objects as we would prefer. Still, when extensionality is not present, no two of them are equal, which can be seen from our earlier result that all five of Θ_1 through Θ_5 are irreducible; we did derive some satisfaction from that fact. The irreducibility of Θ_1 through Θ_5 implies that no two are equal because, if two of them were, then, by Church-Rosser, a third combinator would exist to which the two reduce.

On the other hand, the kernels Γ_1 through Γ_5 defined and studied in Section 4.3.1, from which one can construct Θ_1 through Θ_5, are obviously equal—regardless of whether or not extensionality is present—for Γ_1 through Γ_5 each reduce to Γ_a, if one allows the empty set of reductions. As we shall learn in Section 4.4.2.1, our disappointment was eventually replaced with delight—delight at discovering a very large set of fixed point combinators constructible from B and W such that no two are equal even when extensionality *is* present.

4.2.4. Automated Theorem Proving and Research

We close this section with an observation, based on the material we have presented, concerning the role of automated theorem-proving programs. If one reviews the results discussed here and studies how they were obtained, one has a glimpse of the kinds of assistance that an automated theorem-proving program can provide. More important, one finds strong evidence for the claim that the use of an automated theorem-proving program does indeed provide valuable assistance for research.

In particular, the discovery of Θ_1 through Θ_5—which resulted from the use of the program ITP—triggered our study of the use of demodulation, which then led to our study of reductions in general. Because of our emphasis on automated theorem proving and the corresponding approach to proving theorems, we focused on $\Theta_1 f$ through $\Theta_5 f$. That focus resulted in the discovery of our first kernel and, somewhat later, in the formulation of the systematic kernel method for searching for fixed point combinators, which is the topic of the next section of this report.

4.3. The Syntactic Treatment and Introduction of Kernels

In this section, we come to the syntactic definition and introduction of kernels. We discuss various examples and, in general, prepare the way for presenting the kernel method which is central to this report and which is presented in detail in Section 4.5. In that section, we give the formal and semantic definition of a kernel, and then define various types of kernel.

We begin by resuming the chronology at the point where we in one sense discovered the concept of kernel and, in another, formulated that concept. To be precise, where $\Omega = W(B(Bf))$, we were already calling $\Omega\Omega\Omega$ a kernel although we had not yet decided exactly what definition to use. To give the definition on which we finally settled, we must first give the definition of the 1's rule; the corresponding concept in combinatory logic is that of *head reduction*.

Definition. The reduction $C_0, C_1, \ldots, C_k$ of the expression C to the expression D satisfies the *1's rule* if and only if the *into term* in each C_i has a position vector consisting of all 1's.

In other words, for a sequence of reductions to satisfy the 1's rule, each reduction must apply to the first symbol in the expression being reduced. If, for example, B is used to reduce $BW(BBB)f$ to $W(BBBf)$, then that application of B satisfies the 1's rule. On the other hand, if B is then used to reduce $W(BBBf)$ to $W(B(Bf))$, then that application of B fails to satisfy the 1's rule.

We can now give a syntactic definition of kernel—more precisely, a definition of what would eventually be called *simple kernel*—but, for pedagogical and historical reasons, we delay giving the most general definition. In the following definition, which we later generalize, we assume that no eliminator is included among the combinators under consideration. We recall that an eliminator is a combinator such that the list of variables on its right side is a proper subset of the list of variables on its left side. For example, the combinator K with

$$Kxy = x$$

is an eliminator. We temporarily exclude such combinators from consideration because their presence adds a degree of complexity that will, for now, simply obscure the picture. (To partially answer the obvious question that one might immediately ask, our desire to formulate a uniform approach, for searching

for fixed point combinators, that would be useful for all studies—including studies in which eliminators are present—led us to the classification of kernels that are simple, semisimple, and neither.)

Definition, syntactic. The expression Γ is a *simple kernel* with respect to the set P of combinators if and only if (1) there exists a sequence $A_1, A_2, \ldots, A_k$ with A_i in P such that Γ reduces to $x\Gamma$ by applying A_i in order as reductions, (2) the A_i satisfy the 1's rule, and (3) Γ is expressed purely in terms of elements from P and the variable x, where x implicitly ranges over all combinators. More casually, we also call Γ a *kernel* when all of the given conditions are satisfied except that all occurrences of the variable x are replaced by the constant f, where f is an arbitrarily chosen combinator.

To provide for the curious a hint of what is to come and before turning to specific examples of a simple kernel, let us pause for certain remarks—remarks whose significance and full meaning will become clear only after one gains experience with kernels. The most important point to keep in mind is that, until we turn in Section 4.5 to the semantic and formal treatment, we are discussing the concept of kernel from a syntactic viewpoint, discussing it as an expression.

In both the syntactic orientation and in the semantic treatment that we come to later, we often refer to a kernel by its name—using names such as $Lx(Lx)$ or $Lf(Lf)$. Rather than emphasizing the role of the variable x, we are far more likely to use names such as $Lf(Lf)$ and make references to a kernel Γ that reduces to $f\Gamma$ when we are focusing on how an automated theorem-proving program searches for and uses kernels. The main reason for this practice rests with the fact that the presence of variables permits a theorem-proving program to pursue paths that we generally wish it to avoid. In Section 4.5, in contrast to the syntactic definition, we give the semantic and preferred definition of kernel, and discover that a kernel is a set of combinators that are very strongly related to each other.

Next, one might naturally expect to see both forms of the definition of simple kernel—that focusing on f and that focusing on x—since f is an arbitrarily chosen combinator. After all, if Γ reduces to $f\Gamma$ for an arbitrarily chosen combinator f, then Γ reduces to $x\Gamma$ for all combinators x; the converse is obviously true. Of course, to be precise, the strict rules of combinatory logic do not permit one to reduce an expression Γ to $x\Gamma$ where x is a variable. Indeed, the strict use of the term reduction requires that the term being reduced contain no variables, and therefore no variables will occur after a reduction is applied.

Even further—again as one might expect from the discussion in the preceding sections—as we shall learn in Section 4.5, each simple kernel acts like a function of the variable x that occurs in it. (Technically, a kernel induces a one-to-one function that maps a combinator f to an element of the kernel.) For different choices of x, the kernel yields different combinator expressions. As we shall see when we visit different simple kernels, each of those we visit includes one or more occurrences of the variable x or, alternatively, of the arbitrarily chosen combinator f.

Summarizing, although we shall often say that Γ reduces to $f\Gamma$—since f is an arbitrarily chosen combinator—we shall frequently say that Γ reduces to $x\Gamma$. We discussed such compact and shorthand notation at the beginning of Section 3, justifying its use—in more detail than was needed by those who are familiar with mathematics and logic, but perhaps with the care that benefits the uninitiated. With the pause completed, we can now turn to specific examples of simple kernels, sometimes referred just as kernels.

The expression $\Omega\Omega\Omega$ is an example of a simple kernel with respect to the set P consisting of B and W alone, where $\Omega = W(B(Bf))$. Equally, the expression $\Delta\Delta\Delta$ is also a kernel with respect to the same set P, where $\Delta = W(B(Bx))$—in fact, in the spirit of the preceding remarks, the same kernel. Indeed, we showed in the preceding section that $\Omega\Omega\Omega$ is a kernel, and commented that the same argument shows that $\Delta\Delta\Delta$ is a kernel. By applying the appropriate reductions, one can also show that among the other simple kernels with respect to this set P are the left-associated cubes of the following four expressions: $W(BBBx), BWB(Bx), BW(BBB)x, B(BWB)Bx$.

For a simple kernel for which neither B nor W plays any role, let P consist of the combinator U alone, where

$$Uxy = y(xxy),$$

and let Γ be UUx; Γ is a kernel with respect to P. For a third example of a kernel, let P consist of S and L alone with

$$Sxyz = xz(yz)$$

and

$$Lxy = x(yy),$$

and let Γ be $Lx(Lx)$; Γ is a kernel with respect to P, even though S does not occur in Γ. We shall revisit these and other kernels later.

4.3.1. The Relation of Kernels to Fixed Point Combinators for B and W

In the section, we focus on the use of kernels for constructing fixed point combinators from B and W alone.

At this point in our research, we did not turn to other combinators to study. Rather, we continued to focus on the set P consisting of B and W alone, and also on the five kernels (with respect to P) we have just listed. These kernels—Γ_1, Γ_2, Γ_3, Γ_4, and Γ_5—are named to reflect a mapping whose discovery marked a high point in our research; the mapping pairs a kernel with the fixed point combinator from which the kernel derives its numerical subscript (see Section 4.4.3). Here are the ten expressions that we soon discovered naturally formed five pairs; each pair consists of a kernel and the corresponding fixed point combinator that maps to it.

$\Gamma_1 = W(B(Bx))(W(B(Bx)))(W(B(Bx)))$
$\Theta_1 = B(B(B(WW)W)B)B$

$\Gamma_2 = W(BBBx)(W(BBBx))(W(BBBx))$
$\Theta_2 = B(B(WW)W)(BBB)$

$\Gamma_3 = BWB(Bx)(BWB(Bx))(BWB(Bx))$
$\Theta_3 = B(B(WW)(BWB))B$

$\Gamma_4 = BW(BBB)x(BW(BBB)x)(BW(BBB)x)$
$\Theta_4 = B(WW)(BW(BBB))$

$\Gamma_5 = B(BWB)Bx(B(BWB)Bx)(B(BWB)Bx)$

$\Theta_5 = B(WW)(B(BWB)B)$

To participate in our research, one should pause at this point, examine the five given pairs, and attempt to discover the algorithm that pairs each kernel with its fixed point combinator. As one will eventually see, when we discuss various other fixed point combinators and their kernels, the algorithm we now give and the mapping it defines apply to far more than the five given pairs Θ_1 through Θ_5 and Γ_1 through Γ_5.

Algorithm for Mapping Fixed Point Combinators to Kernels

First, choose a combinator Θ from among Θ_1 through Θ_5. Second, write the expression

$\Theta f \neq f(\Theta f)$

with Θ fully expressed in terms of B and W; the inequality is derived from denying that the selected combinator Θ is a fixed point combinator. The constant f is present because we are assuming the falseness of the equation

$\Theta_4 x = x(\Theta_4 x)$

for all combinators x. However—and here we have a satisfying mnemonic—f will also play the role of the arbitrarily chosen combinator that occurs in the (alternative and syntactic) definition of simple kernel. Third, apply to Θf reductions that satisfy the 1's rule until a kernel is encountered. Fourth and final, for the image of Θ under the mapping, assign the kernel that is encountered.

Of course, since f is an arbitrarily chosen combinator, we are also entitled to replace f by the variable x in the kernel we obtain with the given algorithm. Observing the 1's rule is important, for if one does *not* observe it, one can arrive at the wrong kernel. For example, if one begins with Θf with

$\Theta_4 = B(WW)BW(BBB)$,

reduces with B, then reduces a second time with B but violating the 1's rule, and finally reduces a third time with B again violating the 1's rule, one eventually arrives at Γ_1, which is not the correspondent of Θ_4.

The given algorithm induces a one-to-one, onto, well-defined mapping from the set of five combinators to the set of five kernels. Under the mapping, the correspondent of Θ_1 is Γ_1, of Θ_2 is Γ_2, of Θ_3 is Γ_3, of Θ_4 is Γ_4, and of Θ_5 is Γ_5.

4.3.2. The Role of Kernels for Proofs of the Strong Fixed Point Property for *B* and *W*

At this point in the report, we switch our focus slightly, switch to focusing on the use of kernels for proving that the strong fixed point property holds for the set consisting of B and W alone.

To prove that the strong fixed point property holds for a set P of combinators, as we learned earlier, one can either deny that it does and seek a proof by contradiction, or one can focus on a specific combinator and prove that the combinator satisfies the equation for the strong fixed point property. We gave an example of the former in Section 4.1.1, and an example of the latter in Section 4.2.1. In this section, again focusing on the set P consisting of B and W alone, we examine a third possibility—we show how kernels can play a vital role in searching for a proof that the strong fixed point property holds.

If, from the preceding section, we identify the mapping with the sequence of reductions that induces it, we have the first stage of a two-stage proof by contradiction that Θ is a fixed point combinator, where Θ is any combinator from among Θ_1 through Θ_5. In that stage, the reductions are applied to both sides of the equation

$$\Theta f \neq f(\Theta f),$$

which comes from assuming that the combinator Θ is not a fixed point combinator, and one obtains

$$\Gamma \neq f\Gamma$$

where Γ is the kernel to which Θ is mapped. In the second stage of the proof by contradiction, the first occurrence of Γ is then reduced to $f\Gamma$ by the sequence of reductions that is known to exist because Γ is a kernel. If one considers the two stages in order, one deduces that

$$f\Gamma \neq f\Gamma,$$

which, from the viewpoint of mathematics, signals the completion of a proof by contradiction. Alternatively, from the viewpoint of automated theorem proving, if the two-stage proof is phrased in the clause notation and found by a theorem-proving program, the program also obtains a proof by contradiction, since the clause

EQUAL(x,x)

is always present in problems focusing on equality when paramodulation is used. As we shall see when we present (in Section 4.5) the kernel method for searching for fixed point combinators from some given set P of combinators, these two stages will be revisited, but they will be interchanged and will depend on expansions rather than on reductions.

We can extract from this proof by contradiction, consisting of two sets of reductions, a proof in the style of abstract algebra. At the same time, we can, by considering the specific application to Θ_4, in part again show why we allow the constant f—which we continue to use both as an arbitrarily chosen combinator and as a Skolem constant arising from denying that the strong fixed point property holds—to be replaced by a variable in the alternative definition of kernel. The first part of the proof proceeds by reducing with B, W, W, B, W, B, B, and B.

A Proof, in the Algebraic Style, That Θ_4 Is a Fixed Point Combinator

$$\Theta_4 x = B(WW)(BW(BBB))x = WW((BW(BBB))x) = W((BW(BBB))x)((BW(BBB))x) =$$
$$BW(BBB)x((BW(BBB))x)((BW(BBB))x) = W(BBBx)((BW(BBB))x)((BW(BBB))x) =$$
$$BBBx((BW(BBB))x)((BW(BBB))x)((BW(BBB))x) =$$
$$B(Bx)((BW(BBB))x)((BW(BBB))x)((BW(BBB))x) =$$
$$Bx(((BW(BBB))x)((BW(BBB))x))((BW(BBB))x) =$$
$$x(((BW(BBB))x)((BW(BBB))x)((BW(BBB))x))$$

We can complete the proof by extracting from the set of equalities

$$\Theta_4 x = BW(BBB)x((BW(BBB))x)((BW(BBB))x),$$

which permits us to substitute for $\Theta_4 x$ in $x(\Theta_4 x)$ to get

$x(\Theta_4 x) = x(((BW(BBB))x)((BW(BBB))x)((BW(BBB))x))$,

which implies that

$\Theta_4 x = x(\Theta_4 x)$

since

$\Theta_4 x = x(((BW(BBB))x)((BW(BBB))x)((BW(BBB))x))$

from the first set of equalities. We have proved, therefore, that Θ_4 is a fixed point combinator, and the proof is complete.

We can revisit the given algebraic proof, but view the proof in terms of the use of kernels. One first proves that $\Theta_4 x = \Gamma_4$. Then one proves that $\Gamma_4 = x\Gamma_4$. Next, since $\Theta_4 x = \Gamma_4$, $x(\Theta_4 x) = x\Gamma_4$. Therefore, $\Theta_4 x = x(\Theta_4 x)$, and the proof that Θ_4 is a fixed point combinator is complete.

In addition, a closer look at the algebraic proof shows that $\Theta_4 x$ reduces to Γ_4, which in turn reduces to $x\Gamma_4$. In other words, in addition to proving that Θ_4 is a fixed point combinator, we also have a proof that Γ_4 is a kernel. Even further—as a sign of what is forthcoming—we have a proof that the fixed point combinator under study reduces to a kernel.

4.3.3. Paramodulation and Strategy

The material contained in this section is intended mainly for those strongly interested in research in automated theorem proving, and especially for researchers concerned with paramodulation and strategies for controlling that inference rule. Certain aspects of the example we use are just a bit synthetic, but it serves well as an illustration.

If the proof given in the preceding section is expressed in clauses rather than in algebraic notation—except for the order of the arguments in the final equality—a corresponding paramodulation proof (of the type mentioned in the introduction) is obtained. The proof that results is restricted by a powerful strategy—every step but one satisfies the 1's rule. The one exception is the final step that yields $x(\Theta_4 x) = \Theta_4 x$. In particular, an automated theorem-proving program could quickly find almost this entire proof—starting with a clause that expresses that $\Theta_4 x = \Theta_4 x$—using the equations for B and W as *from clauses* and paramodulating from their respective left sides.

The search would be efficient since, with the exception of the final step, all paramodulation steps that do not satisfy the 1's rule would be avoided; in other words, if we ignore the last step, each paramodulation step would require an *into term* whose position vector is all 1's. Among the clauses found by in effect applying reductions with B and W, the program would deduce a so-called first clause asserting that

$\Gamma_4 = \Theta_4 x$

with

$\Gamma_4 = BW(BBB)x((BW(BBB))x)((BW(BBB))x)$,

and the program would later deduce a second clause asserting that

$x\Gamma_4 = \Theta_4 x$.

If one considers these two deduced equalities and uses the fact that all paramodulation steps are in effect

reductions, then one see that Γ_4 is a kernel. All steps in the deduction of these two clauses would satisfy the 1's rule.

By relaxing the 1's rule to permit paramodulation into terms s within expressions of the form $a(x,s)$, the program could then substitute from the first argument of the first clause into the term Γ_4 in the first argument of the second clause to obtain a clause asserting that

$$x(\Theta_4 x) = \Theta_4 x,$$

and the program could cease its attack—cease if the program had some condition to apply to indicate that the assignment had been completed.

One of the most natural conditions to use—if a way can be found to present it in clause form—focuses on a clause that captures the strong fixed point property, a clause that the program can use to recognize it has found an instance of the relevant equation. For example, if one included the clause

$$\neg\text{EQUAL}(a(f(y),a(y,f(y))),a(y,f(y)))$$

which can be obtained by first assuming that the strong fixed point property fails to hold and then interchanging the two arguments of the resulting inequality, and if one prevented the theorem-proving program from paramodulating into this inequality, then this clause could be used. The resulting proof would be a forward proof and—if one did not include the inequality—not the usual proof by contradiction.

Of course, as we commented in Section 2, seeking a proof by contradiction is the preferable approach for an automated theorem-proving program to take; testing pairs of clauses to see whether a contradictory pair has been found is a simple—although sometimes surprisingly expensive—test for the program to apply for determining that the assignment has been completed. Indeed, we know of experiments in which 90 percent of the computer's CPU time is spent in testing to see whether two unit clauses—clauses containing one literal—contradict each other. Research that produced a sharp decrease in the computer time required to make such a test would clearly merit interest.

In both the theorem-proving version under discussion and the preceding mathematical version of the proof that Θ_4 is a fixed point combinator, the simple kernel Γ_4 plays a vital role. Since simple kernels such as Γ_4 prove to be so valuable, and since such kernels rely so heavily on the use of the 1's rule, immediately—from the viewpoint of automated theorem proving in general—a natural question to ask concerns the possibility of using the 1's rule as a restriction strategy when attacking other classes of problem with an automated theorem-proving program.

If we wish to have an automated theorem-proving program seek to prove that Θ_4 is a fixed point combinator, but follow the usual approach of searching for a proof by contradiction, we would begin with

$$\neg\text{EQUAL}(a(\Theta_4,f),a(f,a(\Theta_4,f)))$$

with Θ_4 fully expressed in terms of B and W. We could then use the strategy that requires all paramodulation steps whose *into term* is contained in the first argument to satisfy the 1's rule. For the steps whose *into term* is contained in the second argument, we could extend the strategy to permit such terms to almost satisfy the 1's rule in the sense that the leading f is ignored—in other words, we could use a *pseudo 1's rule*. The program could find a backward proof that corresponds to the forward proof we just gave, a proof relying strongly on the use of the kernel Γ_4.

An examination of such a proof provides us with an interesting example illustrating why one must be cautious about cavalierly using one particular option for paramodulation that occasionally is suggested as the way one should implement paramodulation. The option in question has the program simultaneously replace all occurrences of a chosen *into term*, rather than replacing one occurrence at a time. Let us see what happens if this option is exercised when attempting to prove that Θ_4 is a fixed point combinator.

The proof by contradiction we have in mind would apply B and W as reductions until the clause

$\neg$EQUAL(Γ_4,a(f,Γ_4))

is obtained, where Γ_4 is fully expressed in terms of B, W, and the Skolem constant f that exists because of assuming the desired result false. The program would then continue to use B and W to reduce both sides of the inequality, applying where possible the reductions simultaneously to all occurrences of each chosen *into term*. However, rather than at some point obtaining

$\neg$EQUAL(a(f,Γ_4),a(f,Γ_4))

as desired, the program would instead eventually obtain

$\neg$EQUAL(a(f,Γ_4),a(f,a(f,Γ_4)))

since Γ_4 reduces to $f\Gamma_4$, which is one of the properties of being a kernel.

In other words, rather than deducing a clause that contradicts the clause for reflexivity, the program would deduce the first of many clauses that form an infinite loop. The program loops, of course, since both occurrences of Γ_4 will further reduce to $f\Gamma_4$; as is obvious, we are assuming that the 1's rule is not in use. On the other hand, if the 1's rule were in use in the modified form proposed to permit the needed substitution to occur on the right side of the inequality, then the program would simply terminate that path of inquiry with the deduction of a clause asserting that $f\Gamma_4 \neq f(f\Gamma_4)$. In either case, no contradiction would be discovered if all steps were restricted to being reductions.

We therefore have an example of why the option of simultaneous replacement of all like *into terms* should not be a requirement for paramodulation. Even further, we have an example of why the choice of this option would, for the purpose under discussion, be unwise; indeed, such a choice would lead to failure to reach the objective.

In the given forward proof, and in the corresponding backward proof by contradiction, we noted that the kernel Γ_4 plays an important role. However, rather than being used in the form in which a Skolem constant f is present, the kernel is used in the form in which the variable x replaces all occurrences of f, which explains in part why we have alternatives for the definition of kernel. The remainder of the explanation will be given when we again discuss kernels in the context of the weak fixed point property.

4.4. Superkernels

We now come to the point in our research at which we discovered—perhaps formulated is a better word to use—a concept closely related to that of simple kernel, namely, the concept of *simple superkernel*. Reliance on this new concept, as we shall see, gives one an attractive alternative (presented in Section 4.5.2) for applying the kernel method.

More important, the study of superkernels in the context of B and W unveiled an unexpected piece of the mystery concerning the construction of fixed point combinators. We say "unexpected" because we had no idea that more of the mystery remained hidden, especially after we had found the four combinators (different from Θ_4) that satisfy the equation for the strong fixed point property. Indeed, far greater than the surprise that we and Smullyan experienced at finding Θ_1 through Θ_3 and Θ_5 was our reaction to what we discovered once we began studying superkernels. We were—and still are—startled at what we found, startled to find how rich the B-and-W mine is if one is searching for fixed point combinators. The B-and-W mine is infinitely rich. Let us immediately turn to the relevant details.

4.4.1. The Discovery of Superkernels

At this point in the report, we focus on precisely how we found those objects that we would eventually call simple superkernels.

Even though we had extracted obviously useful information from the fixed point combinators Θ_1 through Θ_5, from the kernels Γ_1 through Γ_5, and from the mapping between the two sets, we found that far more information could be extracted. In particular, we come to the discovery or formulation of the concept of *simple superkernel*—a concept whose study led us to finding an infinite number of fixed point combinators constructible from B and W, and then to finding an infinite subset such that no two of its elements are equal even when extensionality is present. To set the stage for introducing this new concept, simple superkernel, let us again list Γ_1 through Γ_5, and then examine these five kernels in a different light—specifically, with regard to the possibility of reducing one to another by applying B and W.

$\Gamma_1 = W(B(Bx))(W(B(Bx)))(W(B(Bx)))$
$\Gamma_2 = W(BBBx)(W(BBBx))(W(BBBx))$
$\Gamma_3 = BWB(Bx)(BWB(Bx))(BWB(Bx))$
$\Gamma_4 = BW(BBB)x(BW(BBB)x)(BW(BBB)x)$
$\Gamma_5 = B(BWB)Bx(B(BWB)Bx)(B(BWB)Bx)$

Of course, we immediately note that all five of Γ_1 through Γ_5 are reducible, which we know because each is a kernel. However, if we consider which of the five has the property of admitting a reduction only if it satisfies the 1's rule, we discover that Γ_1 has that property, but the other four do not. In particular, except for Γ_1, each of the other four kernels can be reduced by an application of B that fails to satisfy the 1's rule. From a slightly different perspective, we can write each of the five kernels as the cube of some combinator or generator, respectively denoted as Ω_1 through Ω_5. Therefore—if we recall that we frequently make no distinction between writing a specific kernel in terms of the variable x or in terms of the constant f—in the notation focusing on generators, the kernel denoted earlier by $\Omega\Omega\Omega$ is now denoted by $\Omega_1\Omega_1\Omega_1$, where $\Omega_1 = BW(BBBx)$. With regard to reducibility with respect to B and W—and with the new perspective—we discover that Ω_1 is not reducible, but each of Ω_2 through Ω_5 is reducible. We can, therefore, reduce both the second and the third occurrences of the respective generator in Γ_2 through Γ_5; such reductions of Γ_2 through Γ_5 are clearly in violation of the 1's rule.

In our continued interest in showing how research did occur, we can hypothesize that one might respond to this observation that the 1's rule is too restrictive since, if viewed from the perspective of Ω_1 through Ω_5, only Ω_2 admits a reduction violating the 1's rule. Such a rejoinder could lead one to suggest

that a globalization of the 1's rule might merit consideration—might, in fact, be a better property on which to focus when compared to the given (strict) 1's rule. In other words, if paramodulation is the chosen rule for drawing conclusions—and despite the danger illustrated in the preceding section where we discussed one of the options for applying paramodulation—perhaps it would be profitable to consider simultaneous paramodulation steps *into* all like terms. Were one to play the game in this fashion, then only Γ_2 would admit a reduction violating the global 1's rule, and Γ_2 would then be the kernel that is singled out from among Γ_1 through Γ_5. Let us show why this alternative is of substantially smaller interest than adhering to the strict 1's rule, and, more important, let us show why Γ_1 rather than Γ_2 is the kernel to be given a special place. For pedagogical reasons, we first give a formal definition that is relevant to a globalization of the 1's rule; we define a global reduction.

Definition. The reduction $C_0, C_1, \ldots, C_k$ of the expression C to the expression D is termed a *global reduction* if and only if, whenever some term t is replaced by a term s in a reduction step, then all like terms are simultaneously replaced by s in that step.

Technically, a global reduction actually consists of applying identical reductions simultaneously to all occurrences of some chosen *into term.* We shall also have need of its counterpart, a *global expansion.* We can then talk about a global reduction that satisfies the 1's rule to mean that each of the global reduction steps involves a term whose position vector consists of all 1's. Similarly, we can talk about a global expansion that satisfies the 1's rule.

Immediately, we can study the difference between using ordinary reductions that satisfy the 1's rule and using global reductions that satisfy that rule. With that focus, we shall see that Γ_1 rather than Γ_2 is indeed the kernel of interest. The preference for—and the power of—Γ_1 rests with the fact that each of Γ_2 through Γ_5 is reducible with a global reduction to Γ_1—although, as we shall see immediately, the global reduction that is applied to Γ_2 to obtain Γ_1 does not satisfy the 1's rule or the global 1's rule. From that viewpoint, Γ_2 is not the kernel of interest; however, Γ_1 is since it is the only one of the five kernels to which all five reduce; Γ_1 reduces to itself with the empty set of reduction steps.

To prove that Γ_2 through Γ_5 each reduces to Γ_1, first we note that an application of B reduces $B(BWB)Bx$, which is Ω_5, to $BWB(Bx)$, which is Ω_3. A second application of B reduces Ω_3 to $W(B(Bx))$, which is Ω_1. We then note that an application of B reduces $BW(BBB)x$, which is Ω_4, to $W(BBBx)$, which is Ω_2. A second application of B reduces Ω_2 to $W(B(Bx))$, and Ω_1 is obtained again. Since Γ_2 through Γ_5 are, respectively, the cube of Ω_2 through Ω_5, Γ_2 through Γ_5 each reduces to Γ_1—although, as commented earlier, a global reduction that violates the 1's rule is required for Γ_2. In contrast to Γ_1, the kernel Γ_2 offers only one point of interest, namely, Ω_2 is the only generator from among the five generators for which a reduction exists that does *not* satisfy the 1's rule.

By reviewing the preceding discussion, we see why the 1's rule, in the strict form, is preferred to its global counterpart and, similarly, why ordinary reductions are preferred to global reductions. First, we note that, if we were to focus on the globalization of the 1's rule, then—except for Γ_2—each of the kernels under discussion admits exactly one reduction, and that reduction satisfies the global 1's rule. One might mistakenly conclude from this observation that the kernel Γ_2 is the most interesting kernel from among Γ_1 through Γ_5. On the other hand, were we to focus on the strict 1's rule, then Γ_1 is the exception—is the only kernel admitting a single reduction, and further, a reduction satisfying the strict 1's

rule. We are not implying, however, that reductions satisfying the global 1's rule are of no use—in fact, quite the contrary. With one global reduction satisfying the 1's rule, Γ_5 reduces to Γ_3, and with a second global reduction satisfying that rule Γ_3 reduces to Γ_1. Therefore, both Γ_5 and Γ_3 reduce with the global 1's rule to Γ_1. As for Γ_4, use of the global 1's rule yields Γ_2.

But the picture now becomes slightly inhomogeneous. In particular, were this aspect of the study to complete in a totally satisfactory manner, we would now find that Γ_2 reduces to Γ_1 also with the global 1's rule. Unfortunately, as already shown, such is not the case; rather, we must apply a reduction that, although it has the property of simultaneously replacing all occurrences of each chosen *into term*, does not satisfy the global 1's rule. To reduce Γ_2 to Γ_1 in a single step, we apply B simultaneously to all occurrences of $BBBx$; in other words, we apply a global reduction that violates the 1's rule.

If we combine the current discussion with the earlier concern for the option in which paramodulation is required to attempt to replace all like terms at once rather than replacing a single occurrence, we conclude that focusing on so-called ordinary as opposed to global paramodulation more readily points to the kernel of interest. After all, the single occurrence option leads to choosing Γ_1 as the kernel of interest, which is the better choice from what we have said so far. Nevertheless, we also see that global reductions are indeed the type of reduction to use when studying a set of related kernels with the goal of finding one to which they all reduce. The argument for preferring Γ_1, however, is far from complete, as we shall soon see. Therefore, to complete that argument, we now define the concept of simple superkernel.

Definition, syntactic. The expression Γ is a *simple superkernel* with respect to the set P of combinators if and only if (1) Γ is a simple kernel with respect to P and (2) exactly one reduction applies to Γ and that reduction satisfies the 1's rule.

The connection between the two given properties in the definition of simple superkernel rests with the fact that, since a simple superkernel Γ is a simple kernel, Γ is reducible and, therefore, at least one reduction can be applied to Γ. A simple superkernel is distinguished from an ordinary simple kernel in that the latter always admits a reduction that violates the (strict) 1's rule. On the other hand, if one applies the sequence of reductions to a simple superkernel Γ that reduces Γ to $x\Gamma$, no guarantee exists that each intermediate expression—each descendant—will share the unique reduction property that Γ has. For now, we consider that the line of descendants terminates when $x\Gamma$ is obtained, where Γ is some simple kernel under study. When we come to the concept of kernel itself—of which simple kernel is a subclass—we shall be interested in lines of descendants that do not terminate with $x\Gamma$, where Γ is a kernel that is not simple.

As an example of what can occur with a superkernel's descendants, let us consider the superkernel Γ_1. If one reduces Γ_1 with W and then with B, one obtains

$$Bx((W(B(Bx)))(W(B(Bx))))(W(B(Bx))),$$

which admits a reduction with W that violates the 1's rule. One can summarize this situation by saying that, although superkernels admit a single reduction such that the reduction satisfies the 1's rule, this property is not necessarily inherited by its descendants under reduction. We are forced to include the word "necessarily" in our summarizing sentence because superkernels exist for which all descendants behave essentially as desired. The superkernel $Lf(Lf)$ is such an example in that all descendants admit a

single reduction, and that reduction satisfies the pseudo 1's rule.

We chose the term superkernel to suggest that such an object behaves like a kernel, but more so—more so in the sense that it offers great power for constructing fixed point combinators, just as ordinary kernels do, but offers even more power. The added power of almost all superkernels comes in part from the fact that other kernels can be obtained from them. For example, each of Γ_2 through Γ_5 can be obtained from Γ_1. Therefore, since Θ_2 through Θ_5 are obtainable, by respectively expanding Γ_2 through Γ_5, we see that the five combinators Θ_1 through Θ_5 can all be obtained by expansions if we start with Γ_1. (Of course, in contrast to having all expansions observe the strict 1's rule in the process of obtaining Θ_2 through Θ_5, respectively, from Γ_2 through Γ_5, obtaining Θ_1 through Θ_5 from Γ_1 requires the use of some expansions that violate the 1's rule.) However, if one is not concerned with this restriction rule, then Γ_1 is indeed powerful, for one can pursue various expansion paths and obtain all of Θ_1 through Θ_5 from Γ_1. In fact, as we shall see when we discuss a variant of the kernel method in Section 4.5.2, one could approach the construction of fixed point combinators from *B* and *W* purely from the viewpoint of the superkernel Γ_1. We shall give examples of other superkernels later in this report.

The process for obtaining kernels from a superkernel consists, as expected, of applying global expansions that satisfy the 1's rule and also applying global expansions that violate the 1's rule—that avoid terms whose position vector consists of all 1's. Indeed, as an example, a glance at the earlier discussion of the reducibility of Γ_2 through Γ_5 to Γ_1 shows that focusing on Γ_1, replacing the reductions by the corresponding expansions, and inverting their order is just what is needed to obtain the set of kernels under consideration. In particular, one can first obtain Γ_3 from Γ_1 and then Γ_5 from Γ_3, each by applying a global expansion satisfying the 1's rule. To obtain Γ_2 from Γ_1, one simply applies a global expansion simultaneously to a set of identical terms none of which satisfies the 1's rule constraint. Finally, one completes this aspect of the study by applying to Γ_2 a global expansion satisfying the 1's rule to obtain Γ_4.

In other words, if one were to have discovered Γ_1 first, one might then have been led quickly to the discovery of each of Γ_2 through Γ_5. Then, from each of the five kernels, one could have discovered Θ_1 through Θ_5, respectively, and, at that point, one would have succeeded in constructing five distinct fixed point combinators from *B* and *W* alone.

This entire process leading to the discovery of Θ_1 through Θ_5 would have been—and actually is—far more efficient than the brute-force approach we were required to use when we were confronted with the original problem. Indeed, compared to the standard approach in automated theorem proving—or in mathematics or logic, for that matter—of denying that a fixed point combinator exists with the object of finding a proof by contradiction and, if successful, extracting from the proof the desired object, the kernel method is far superior. As evidence, we note that our discovery of Θ_1 through Θ_5 did in fact require more effort on our part, and far more CPU time for the computer. Specifically, as we shall learn in Section 4.5.3, when we asked our program ITP to study *B* and *W* in the context of fixed point combinators, the equivalent of 20 CPU hours of computer time on a Sun 3 workstation was required to find Θ_1 through Θ_5. In contrast, when the problem was attacked with the kernel method applied by ITP, the same information was found in 8 CPU minutes on a Sun 3 workstation.

Still later—as we learn in Section 5.3.1—after we had fully developed the kernel method and after the automated theorem-proving program OTTER (Other Theorem-proving Techniques for Effective Research) had been designed and implemented, OTTER applied the kernel method to find in 2 CPU seconds a proof that the set P consisting of B and W alone satisfies the strong fixed point property. We shall, as promised, give in Section 4.5.2 the precise method for searching for fixed point combinators—a method that closely follows the outline we have just given—and then apply the method to the study of B and W.

Of course—and this fact was disappointing when pointed out by Smullyan in private correspondence [Smullyan86]—Θ_1 through Θ_5 are not distinct when extensionality is present; as commented earlier, he supplied proofs that $\Theta_4 = \Theta_5$ and $\Theta_1 = \Theta_2$. That disappointment lingered until we made the startling discovery alluded to earlier, the discovery that establishes that Γ_1 is far more useful, as we shall see in the next section, than we had originally thought—even more useful than evidenced by the existence of Θ_1 through Θ_5. This additional use for Γ_1, and therefore the additional power offered by it, also contributes to the choice of the term superkernel. However, rather than attempting to show how the discovery was made, let us present specific results, and then focus on various properties of the results.

4.4.2. Other Superkernels for B and W

In this section, we discover that Γ_1 is not the only superkernel that exists with respect to the set consisting of B and W alone.

As discussed in Section 4.1, during our original exploration of the B-and-W mine, we stumbled onto a vein that yielded more than the fixed point combinator Θ_4 that initiated our study—we found Θ_1 through Θ_5. That particular vein, as we prove later, quickly runs out—Θ_1 through Θ_5 exhausts it. However, as we learn in this section, further exploration of the B-and-W led to the discovery of a far richer vein—a vein located not far from the first vein, and one that runs forever. Let us set the stage for extracting the riches from that second vein.

We begin by considering the expression

$$\Omega_6 = W(B(B(Bx)))$$

obtained from Ω_1 by inserting an additional B immediately to the right of W and fully right associating the result. (The notation involving, for example, Θ_6 will be completely explained in Section 4.4.3 where we impose an enumeration on a set of combinators.) Next, let us consider the left-associated fourth power of Ω_6, and call the result Γ_6. First we note that Γ_6 is reducible, but only one reduction is possible, a reduction with W satisfying the 1's rule. Second, we note that, if one applies to Γ_6 in order the reductions W, B, B, and B, one obtains

$$\Gamma_6 = x(\Gamma_6),$$

which says that Γ_6 reduces to $x(\Gamma_6)$. Third, we note that the given reductions all satisfy the 1's rule. We thus see that the expression Γ_6 is a simple superkernel with respect to I' consisting of B and W alone.

Of course—for now—we are mainly interested in the construction of fixed point combinators, not just the discovery of superkernels. However, at least for Γ_1, we already observed that its discovery could have led to the discovery of four additional kernels Γ_2 through Γ_5, and each of these five kernels in turn

could have led to the construction of the five fixed point combinators Θ_1 through Θ_5. The natural questions to ask concern the possibility of expanding Γ_6 to a fixed point combinator, and the possibility of expanding Γ_6 to the construction of additional kernels that might in turn expand to yet more fixed point combinators. In other words, in contrast to extracting Θ_1 through Θ_5 from various proofs of the abstract form of a theorem that asserts that combinators exist that satisfy the equation for the strong fixed point property, we might instead construct fixed point combinators directly by expanding various kernels.

If, for example, we begin with Γ_6

$W(B(B(Bx)))(W(B(B(Bx))))(W(B(B(Bx))))(W(B(B(Bx))))$

and expand three times with

$Wxy = xyy$

subject to the constraint of the 1's rule and the constraint of always choosing the shortest term that expands, we obtain

$W(WW)(W(B(B(Bx))))$

as the result. Then, if we apply four expansions with B, each to the entire expression under consideration, and therefore each satisfying the 1's rule, we obtain in order

$B(W(WW))W(B(B(Bx)))$
$B(B(W(WW))W)B(B(Bx))$
$B(B(B(W(WW))W)B)B(Bx)$
$B(B(B(B(W(WW))W)B)BB)x$

as the result, which we rewrite as $\Theta_6 x$.

At this point, the construction of the desired fixed point combinator related to Γ_6 is virtually complete. In fact, we need only recall that

$\Gamma_6 = x(\Gamma_6)$,

which follows from the fact that Γ_6 reduces to $x(\Gamma_6)$, and then apply the expansions just presented to the right side of this equality as well. We obtain

$\Theta_6 x = x(\Theta_6 x)$,

which is obviously an instance of the equation for the strong fixed point property. Summarizing, we have succeeded in first finding a new superkernel, and then using it to find a corresponding fixed point combinator constructible from B and W alone. When we present the kernel method for constructing (when they exist) fixed point combinators from some arbitrarily given set P of combinators, we shall see that what we have just witnessed very closely resembles an application of that method.

Let us pause to supply certain relevant history. To begin with, we focused on the form of the superkernel that contains the Skolem constant f rather than that containing the variable x to ensure that our program does not apply any unwanted instantiations during its search for kernels or for fixed point combinators. (The search for kernels derived from superkernels is described in Section 4.5.2 as the first stage of the three-stage kernel method; the search for fixed point combinators from kernels is described as the second stage of the two-stage version presented in Section 4.5.1, or the third stage of the three-stage

version of the kernel method.) In particular, were we to focus on the form of the superkernel containing the variable x, then various paths corresponding to various instantiations of x would be considered by the program, and we wish to avoid those paths.

Next, we chose to study Γ_6 in the context of possible superkernels in part because earlier in our research we had discovered a superkernel smaller than Γ_1, nam ly, $W(Bx)(W(Bx))$. We reasoned that if the square of $W(Bx)$ is a superkernel, which it is, and since the cube of $W(B(Bx))$ is a superkernel, then the fourth power of $W(B(B(Bx)))$ deserves study. The other force for studying Γ_6 rests with the success we had with Γ_1. In short, because of the two given reasons, we were encouraged to pursue the study of Γ_6, encouraged despite having already discovered that the use of the smaller superkernel $W(Bx)(W(Bx))$ does not lead to the construction of a fixed point combinator from B and W alone. We shall see, in Section 4.5.5, that this smaller superkernel is in fact useful in another context.

Noting the strong connection between Γ_6 and Γ_1 and the fact that one could parallel the construction of Θ_1 from Γ_1 to obtain Θ_6 from Γ_6, the next natural step was to explore the possible constructibility of other kernels from Γ_6 to parallel the construction of Γ_2 through Γ_5 from Γ_1. We found 13 kernels constructible from Γ_6. As in our discussion of how one could have found Γ_2 through Γ_5 from Γ_1, to find the additional kernels, we used global expansions satisfying the 1's rule and global expansions violating that rule. In particular, we used our program to search for new kernels, starting with the superkernel $\Gamma_6 = \Omega_6\Omega_6\Omega_6\Omega_6$, where $\Omega_6 = W(B(B(Bf)))$. Again, we used the form of the generator in which the Skolem constant f occurs, rather than that in which the variable x occurs, to avoid certain search paths.

The program's search was one of level saturation; for each *into term* of potential interest in Ω_6, the program applied, where possible, an identical expansion to all three copies of the chosen term—applied in effect, but not actually, a global expansion. Once all such *into terms* were considered, the resulting kernels (each of level 1) were collected and any duplicates removed; the program then turned to applying the same approach to a newly found kernel of level 1. The approach terminates when an expression of the form pf is found, where p contains no occurrences of f.

If, instead of terminating the search, the corresponding branch of the search were pursued, no additional information of value would be found; further expansions merely yield expressions that offer no more power than one of those kernels already found. We are, of course, interested only in the set of kernels no two of which offer the same power for the construction of fixed point combinators. Summarizing, we do, therefore, have a use for paramodulation where a single *into term* is the focus of attention—even though other like terms exist—and also for paramodulation where the unification focuses on an *into term* simultaneously with all like terms.

For the following 14 kernels, including the superkernel Γ_6, which result from the program's efforts, we simply give the generator of each and note that each kernel is the fourth power of its generator. For each generator, we are permitted the replacement of the constant f by the variable x because the corresponding global expansion path applies.

$W(B(B(Bx)))$
$W(B(BBBx))$
$W(BBB(Bx))$
$BWB(B(Bx))$

$W(BB(BBB)x)$
$BWB(BBBx)$
$W(B(BBB)Bx)$
$BW(BBB)(Bx)$
$B(BWB)B(Bx)$
$BW(BB(BBB))x$
$B(BWB)(BBB)x$
$BW(B(BBB)B)x$
$B(BW(BBB))Bx$
$B(B(BWB)B)Bx$

Because we now understood how we could have found Θ_1 through Θ_5 by using kernels—which is the second stage of the two-stage method we shall present for attempting such constructions—we decided to consider the 14 kernels and attempt to construct from each a fixed point combinator. Only later would we discover that the mapping from kernels to fixed point combinators is not necessarily one-to-one—kernels exist from which one can construct more than one fixed point combinator. Our program found the following 14 fixed point combinators constructible from B and W; each is obtained from the kernel whose generator is given in the corresponding position of the preceding list.

The procedure for obtaining a fixed point combinator from the corresponding kernel adheres to the following rules: expand with B or W such that the *into term* satisfies the 1's rule; consider an *into term* only if it contains the symbol f; always choose the shorter of two *into terms* to expand when a choice exists.

$B(B(B(B(W(WW))W)B)B)B$
$B(B(B(W(WW))W)B)(BBB)$
$B(B(B(W(WW))W)(BBB))B$
$B(B(B(W(WW))(BWB))B)B$
$B(B(W(WW))W)(BB(BBB))$
$B(B(W(WW))(BWB))(BBB)$
$B(B(W(WW))W)(B(BBB)B)$
$B(B(W(WW))(BW(BBB)))B$
$B(B(W(WW))(B(BWB)B))B$
$B(W(WW))(BW(BB(BBB)))$
$B(W(WW))(B(BWB)(BBB))$
$B(W(WW))(BW(B(BBB)B))$
$B(W(WW))(B(BW(BBB))B)$
$B(W(WW))(B(B(BWB)B)B)$

(In Section 5.3.4.2, we learn that one can construct more than one fixed point combinator from each of the 14 kernels, if one relaxes the rule of always choosing the shorter of the two possible *into terms*.)

At this point, we were about to make a discovery that we would find remarkable and that would simply produce elation. To make this discovery and complete this phase of our research, all we had to do was compare, summarize, and generalize our results and observations in the following way. First, we

note that the square of $W(Bx)$ is a superkernel, but one that fails to expand to a fixed point combinator constructible from B and W alone; as commented earlier, this superkernel will, however, be useful in another study (see Section 4.5.5). Next, we observe that the cube of $W(B(Bx))$ is a superkernel from which one can obtain four other kernels; each of the five kernels leads to the construction of a fixed point combinator with respect to B and W. Then, we note that the fourth power of $W(B(B(Bx)))$ is a superkernel from which one can obtain 13 additional kernels; each of the 14 leads to the construction of a fixed point combinator from B and W. Therefore, by applying an argument almost identical to that used earlier, we should be able to conclude that the fifth power of $W(B(B(B(Bx))))$ will work—will prove to be a superkernel from which many kernels can be obtained, each of which leads to the construction of a corresponding fixed point combinator.

Further, since we know that the first two superkernels lead, respectively, to the construction of 5 and 14 fixed point combinators, we should also be able to predict the number at the next stage. A review of the preceding discussion of Γ_6 is sufficient to show that, in fact, all is as it should be with regard to the kernel expressed as the fifth power of its generator—we can prove that we have found the next in the sequence of superkernels, and can predict the number of fixed point combinators that can be traced to the new superkernel. With regard to this fifth power superkernel, 42 kernels can be derived from it, and 42 fixed point combinators can be constructed from B and W, one from each kernel. Finally, we could draw the desired general conclusions, and here they are.

4.4.2.1. Intermediate Summary Concerning *B*-and-*W* Fixed Point Combinators

At this point we give a partial summary of the results pertinent to the construction of fixed point combinators from B and W alone. As we shall learn in Section 4.5.4, to make the summary complete would require too much additional information at this time—information that we only obtained later in our research.

One obtains a superkernel—that will lead to the successful construction of fixed point combinators from B and W alone—by taking the right-associated expression, which will turn out to be the generator of the superkernel, consisting of W, followed by k-1 occurrences of B, followed by a variable x, and then taking the left-associated k-th power of the resulting generator with $k \geq 3$. From this superkernel, if one includes it in the count, one obtains p kernels where p is the number of ways to associate k+1 letters. Each of the kernels leads to the construction from B and W of a fixed point combinator; the correspondence is one-to-one. In Section 4.7, we shall see that, if we do not adhere to all three rules given earlier for expanding a kernel in the attempt to construct a fixed point combinator, in general the mapping from kernels—derivable from a single superkernel—to fixed point combinators is many-to-one.

We can finally put this entire discussion in perspective and present the discovery that did indeed astound us. From B and W alone, one can construct an infinite number of fixed point combinators. This infinite set can be partitioned into classes, each class consisting of a finite set of combinators all of which contain $2k$-1 occurrences of B and k occurrences of W, where the superkernel from which the class is obtained is the k-th power of the generator of the superkernel.

In the context of the preceding remarks, the classes consist of 5 combinators of length 8, 14 of length 11, 42 of length 14, ... Each class is generated by a single superkernel, a fully left-associated

expression of its fully right-associated generator. In other words, the number of elements in a class is equal to the number of ways one can associate k+1 letters, where the superkernel from which the class is obtained is the k-th power of an expression or generator consisting of k+1 letters, counting multiple occurrences.

Within a class, ordinarily no two combinators are equal; however, if one assumes extensionality, then all elements of a given class are equal. We shall supply proofs of both assertions near the end of this subsection. If one chooses two distinct classes and selects an element from each, those two elements are provably not equal; this statement holds even when extensionality is present, which provides a most satisfactory contrast to the fact that, in the presence of extensionality, Θ_1 through Θ_5 are all equal.

We therefore have constructed from B and W an infinite set $\Gamma(BW)$ of superkernels each of which leads to the construction of a finite class of fixed point combinators such that, when extensionality is absent, one can prove that no two elements within a class are equal. Since we have already discussed how one obtains the appropriate kernels from the superkernels in $\Gamma(BW)$, for convenience we sometimes refer to the resulting set of kernels also as $\Gamma(BW)$. On the other hand, even when extensionality is present, we can still construct an infinite set of fixed point combinators no two of which are equal. We simply form a subset of the infinite set of combinators constructible from $\Gamma(BW)$ by selecting one element of each class (defined by the length of its elements), where the union of the classes is the set of combinators generated by $\Gamma(BW)$. Let us call that union $\Theta(BW)$, and keep in mind that its elements are combinators that are constructed directly and indirectly from the superkernels in $\Gamma(BW)$.

We were thus able to assuage the disappointment produced by our earlier discovery—made because of certain proofs supplied to us by Smullyan—that showed that Θ_1 through Θ_5 are all equal in the presence of extensionality. In fact, the universe had rotated 180 degrees; in contrast to the five combinators reported in our earlier paper, we now had access to an infinite set $\Theta(BW)$ of fixed point combinators containing an infinite subset such that no two elements of the subset are equal even when extensionality is present. But far more riches lay ahead. Indeed, we were to discover later, as we shall relate, that even at this point we were naive about what could be extracted from the B-and-W mine.

4.4.2.2. Two Proofs Concerning the Possible Equality of B-and-W Fixed Point Combinators

We now fulfill an earlier promise by giving two proofs concerning the possible equality of fixed point combinators constructible from B and W alone.

Let us complete the account of this phase of our research by proving the assertions concerning the possible equality of various pairs of fixed point combinators constructible from B and W when extensionality is absent and when it is present. The entire argument rests on the Church-Rosser theorem for applicative systems. That theorem states that if two combinators are equal, then a third combinator exists such that the first two are reducible to the third. The converse, of course, is clearly true; if two combinators reduce to a third—whether or not the third is in fact one of the two—then the two are equal. The proof that, in the absence of extensionality, no two combinators in a class (defined by the length of its elements) are equal follows from the fact that all of the combinators contained in the infinite set we constructed are irreducible. No reduction with B or W applies, for example, to Θ_1 through Θ_5 or to any of the 14 combinators constructed by starting with Γ_6. By Church-Rosser, if two elements of a class were equal,

then a third combinator must exist such that the two reduce to it, which cannot be since all combinators under consideration are irreducible. These observations in fact prove that no two combinators in the entire set constructed from $\Gamma(BW)$ are equal when extensionality is absent.

In contrast, when extensionality is present, all elements in each class are provably equal. For a proof of this fact, merely recall that all elements within a given class reduce to the superkernel used to construct that class. However, since no reduction exists that reduces two different superkernels to a third expression, we can again apply the Church-Rosser theorem to deduce that the infinite set of superkernels has the property that no two of its elements are equal, whether extensionality is present or not. We therefore conclude that, if one selects two different classes from those whose union is the infinite set constructed from $\Gamma(BW)$, and then selects one element from each, the two elements can be proved to be unequal even in the presence of extensionality. How far we have traveled from the discovery of five fixed point combinators that, in the presence of extensionality, are provably equal!—we have proved that we can extract from the infinite set $\Theta(BW)$ of combinators an infinite subset such that no two elements of that infinite subset are equal even in the presence of extensionality.

If one reviews what we have learned to this point concerning the construction of fixed point combinators from B and W alone, one finds that these two combinators, when considered together, offer astounding power. Even further, one sees how complex this aspect of combinatory logic is, and what beauty is present if viewed from an algebraic outlook.

Along that same line of thought, we can draw an analogy that might serve as a preview of what we discuss in Section 4.7 when we revisit B and W. We have at this point succeeded in finding a set of fixed point combinators that we can enumerate (see Section 4.4.3), starting with Θ_1 through Θ_5, then numbering those obtained from Γ_6, and proceeding in an orderly fashion. If we take this action, we obtain a one-to-one mapping of this set of combinators onto the positive integers. One can then take the integers, construct from it the rationals, and extract from the rationals an infinite class of infinite sets. Consistent with this analogue, one might wonder what can be done with the infinite set of combinators so far constructed. When we revisit B and W, let us see how much of this analogy applies and, depending on how far we get, what we learn about the power of the set P of combinators consisting of B and W alone.

4.4.3. Enumerating an Infinite Set of Fixed Point Combinators

We now consider the infinite set of fixed point combinators constructed in the preceding section, and then well-order them in such a way that the set naturally maps to the positive integers. In other words, we show how one can take the 5 combinators that reduce to Γ_1, the 14 that reduce to Γ_6, the 42 that reduce to Γ_{20}, ..., and arrange them in such a way that one can identify the first, the second, and, in general, the n-th for any positive integer n. Of course, when we say, for example, that the five combinators (Θ_1 through Θ_5) reduce to Γ_1, we are allowing one to use reductions that violate the 1's rule. By focusing on the reducibility of combinators to superkernels rather than to kernels, we are setting the stage for showing that the superkernels, even more than the kernels derived from them, play the key role in our enumeration.

4.4.3.1. Three Properties of Patterns Based on the Ways to Associate Letters

So that our procedure for enumerating the fixed point combinators constructible from B and W alone will be easy to follow, we pause to study three properties possessed by the patterns for associating three or more letters. The first useful property concerns the numbers 5, 14, and 42—which happen to be the number of combinators in the classes that are, respectively, reducible to Γ_1, Γ_6, and Γ_{20}. If one were asked about the significance of a sequence of numbers beginning with these three numbers, one might respond with the observation that these numbers give, respectively, the ways one can associate k letters for $K = 4$, $K = 5$, and $K = 6$.

The second useful property concerns a procedure for assigning numbers to the various patterns for associating letters, when one has a choice about the association. For example, the first choice to be made occurs when associating three letters; one can left associate the expression fully, or one can right associate it fully. As we shall see—because the first superkernel of interest has a generator consisting of four letters—we shall ignore the association patterns for three letters and, instead, begin with those for four letters. For the enumeration of the various association patterns, we shall choose to assign 1 to the fully right-associated pattern rather than to the fully left-associated, first, because we shall in general observe this preference for our enumeration of the infinite set of combinators, and, second, because the superkernels from which the combinators are constructed can each be expressed in terms of a fully right-associated generator.

To amplify the second factor, the right-associated pattern of the generator of each superkernel will be crucial for assigning the appropriate positive integer to the combinator that directly corresponds to that superkernel. Even further—as we shall see in this section—the assignment of the appropriate integer to the respective combinators that directly correspond to each of the kernels derived from a given superkernel depends on the association of the letters in the generator of the respective kernels; that association is in turn strongly related to the pattern of the letters in the generator of the superkernel from which the kernels are derived.

To continue our illustration of the scheme for numbering the various patterns for associating letters, we note that five ways exist to associate four letters. Now, since the number of ways to associate, for example, four letters does not depend on the presence or absence of duplicate letters—which is the third property of interest concerning association patterns—we can focus on the number of elements in each subgrouping of letters and ignore the letters themselves. With that observation, and with the preference for right association over left association, the five associations are in order of preference 1,(1,2), 1,(2,1), 2,2, (1,2),1, and (2,1),1. These five patterns would ordinarily be assigned, respectively, the numbers 3 through 7 had we assigned the numbers 1 and 2 to the two ways to associate three letters. However—because of an oddity or property of the set P consisting of B and W alone—when we enumerate the infinite set of combinators, we shall ignore the two patterns for associating three letters and begin numbering with the patterns for associating four letters. The justification for this decision will be obvious when we focus directly on the various superkernels, including the superkernel $W(Bx(W(Bx)))$.

We could then extend the enumeration of patterns of association by next considering the 14 patterns that arise from associating 5 letters. Again, we would simply consider the patterns based on the subgroupings of symbols and the number of symbols in each subgrouping, rather than being concerned with

the actual letters themselves. For example, since 5 ways exist to associate four letters—were it not for our intention of ignoring the two ways to associate three letters—the so-called patterns of the form 1,4 would be ordinarily be assigned the numbers 8 through 12, and then the patterns 2,(1,2) and 2,(2,1) would be assigned the numbers 13 and 14, respectively. However, since we intend to ignore the two ways of associating three letters, these patterns will receive, respectively, the numbers 6 through 10 and 11 and 12. The given procedure could be extended in the obvious fashion, assigning the patterns for associating letters to positive integers, resulting in a one-to-one onto mapping, which completes our detour into the study of association patterns in general.

4.4.3.2. The Enumeration Procedure

We are now prepared to enumerate the infinite set of combinators constructed in the preceding section; obviously, we apply a procedure closely related to that just illustrated. The focal point for the enumeration is the set of superkernels consisting of $\Gamma_1 = \Gamma_1^8$, $\Gamma_6 = \Gamma_6^{11}$, $\Gamma_{20} = \Gamma_{20}^{14}$, and, in general Γ^{3k+2} for $k \geq 2$, where Γ^8, for example, is used as a shorthand notation for referring to the superkernel that generates the combinators of length 8. The connection between these superkernels and the patterns of association of letters rests with the symbols that occur in the generator of each superkernel. In particular, we begin with Γ_1, recall that Γ_1 is the cube of $W(B(Bx)) = \Omega_1$, and note that four symbols occur in the generator Ω_1.

Next—and here we come to yet another illustration of the delight offered by mathematics in general and by combinatory logic specifically—if one looks at Γ_1 and the four kernels Γ_2 through Γ_5 that one can obtain from Γ_1, an interesting mapping begins to emerge, if one takes note of the ways to associate four letters. The mapping yields a one-to-one onto correspondence between the set consisting of the kernels Γ_1 through Γ_5 and the five patterns for associating four letters—the letters W B B and x of which the generator Ω_1 consists. Now if one wishes to attempt to find the precise mapping and determine, without our assistance, which kernel is mapped to which association pattern, we recommend pausing immediately. If, instead, one prefers simply to see how it all works, then one can continue on.

The mapping is obtained by starting with the letters W, B, B, and x of the generator Ω_1, marking them so that one can follow their movements as one applies global expansions to obtain the various kernels, and then recognizing the association pattern (of four letters) that results when one ignores all other combinator occurrences. For example, Γ_1—which is obtained from Γ_1 by the empty set of global expansions—is mapped to the pattern 1,(1,2) since $\Omega_1 = W(B(Bx))$. Next, the kernel Γ_2, which is obtained from Γ_1 by a global expansion with B that violates the 1's rule, is mapped to the pattern 1,(2,1).

To see why this is true—and here and for the next three kernels, so that all will be clear, we explicitly give parentheses that do not ordinarily appear in combinatory logic—note that the generator Ω_2 is $W(((BB)B)x)$; but the first occurrence of B is ignored because it arises from the global expansion that yields Γ_2 from Γ_1 and, therefore, is not descended from one of the original four letters in Γ_1. In other words, one ignores the first occurrence of B in the generator Ω_2 when one is seeking to correctly identify the corresponding association pattern. If one continues along this line of reasoning, one sees that Γ_3, generated by $\Omega_3 = ((BW)B)(Bx)$, is mapped to the pattern 2,2; the first occurrence of B is introduced by a global expansion. Then, we find that Γ_4, generated by $((BW)((BB)B))x$, is mapped to the pattern (1,2),1; the

first two occurrences of B are introduced by global expansions. Finally, we find that Γ_5, generated by $((B(((BW)B)))B)x$, maps to the pattern (2,1),1.

However, as indicated when we discussed mapping association patterns to positive integers, we wish to begin with the association patterns for four letters, and skip those for three. Let us see why. We immediately encounter the property (of B and W) that causes us to introduce the oddity in our enumeration—the oddity that results in an assignment of positive integers different from that dictated by consideration of all patterns of association for three or more letters. The property of note asserts that, when B and W are the only combinators under study, the superkernel $W(Bx)(W(Bx))$ is not relevant. Its lack of relevance rests with the fact that one cannot construct fixed point combinators from B and W alone that correspond to or emanate from $W(Bx)(W(Bx))$; we show why this is so in Section 4.5.5. Were $W(Bx)(W(Bx))$ relevant, then we would begin assigning positive integers based on the association patterns for three letters since this superkernel is generated by an expression consisting of three letters. Therefore, since the superkernel $W(Bx)(W(Bx))$ plays no role for our infinite set of combinators, we assign the positive integers based on association patterns for four or more letters, and 1 through 5 are, respectively, assigned to Γ_1, Γ_2, Γ_3, Γ_4, and Γ_5. The fixed point combinators that are, respectively, constructed from these five kernels inherit the corresponding numbering, which explains how we chose the designation for the set of combinators Θ_1 through Θ_5.

One can then consider the 14 kernels that are derived from Γ_6—including, of course, Γ_6 itself—and map the corresponding fixed point combinators to the positive integers 6 through 19. For example, the superkernel Γ_6 has its corresponding combinator mapped to 6, and the kernel obtained from Γ_6 by a global expansion with B that focuses on the deepest term possible has its corresponding combinator mapped to 7. The fixed point combinators that one constructs from considering the superkernel Γ_{20} come next, and are similarly mapped to the positive integers 20 through 61. Since one can continue to obtain superkernels by inserting a B to the right of W in the generator of the preceding superkernel and then fully right associating the result, one can find a fixed point combinator constructible from B and W alone that maps to any chosen positive integer n. We thus have a mapping from the infinite section $\Theta(BW)$ of combinators constructed in the preceding set to the positive integers.

The mapping is one-to-one because of the procedure we have used to produce it. (Of course, if the mapping from fixed point combinators to kernels were not one-to-one, we would be forced to add a rule to address that problem. But, for the infinite set under discussion, the mapping from combinators to kernels is one-to-one.) The mapping from the infinite set of combinators to the positive integers is onto, and one can fairly easily find the combinator that is mapped to a given integer in the following way.

One first computes the first few values for the number of ways one can associate four letters, five, six, and such. One considers these values in order, and computes a set of numbers by taking the first number (5), adding the first two (5 + 14), adding the first three, and, in general, adding the first k. The object is to find two numbers during the additive part of the computation such that the first is a number less than or equal to the n that one is seeking to find the inverse image of, and the second is greater than or equal to n. Of course, n may be so large that one must return to the first part of the computation, that focusing on computing the number of ways to associate some number of letters, in order to find a sum that is greater than or equal to the n under consideration.

The number of summands that is required to compute the smaller of the two numbers that traps n dictates the number of superkernels that must be skipped to arrive at the superkernel and corresponding class of combinators in which the inverse image of n resides. In particular, if the smaller of the two numbers is strictly less than n and the number of summands required to obtain that smaller number is k, then one skips the first k superkernels; if the smaller number equals n, then one skips the first k-1 superkernels.

For example—since we shall learn that the classes whose union is $\Theta(BW)$ are, respectively, composed of combinators of length 8, 11, 14, 17, and the like—if $n = 65$, then one enters the class whose superkernel is Γ_{62}^{17} because 5 + 14 + 42 = 61, forcing one to skip $\Gamma_1 = \Gamma_1^8$ and Γ_6 and Γ_{20}. In that case, obviously, one also knows that the kernel of the desired fixed point combinator is located close to Γ_{62} because 62 is close to 65 and 62 is the inverse image of the combinator whose kernel is Γ_{62}. Similarly, if n is 59, then Γ_{20} is the superkernel of interest, and the kernel of the desired combinator is located far from Γ_{20} but still in the set derived from the superkernel Γ_{20}.

Of course, one need not process the kernels within a class one at a time, but one can, instead refine the search. The refinement we have in mind focuses on computing the number of ways to associate k letters by using the values for numbers less than k. Then one uses these subcomputations to suggest where to look in the class of combinators that one has determined to contain the inverse image of the n under consideration. Thus we see that indeed the infinite set of combinators under study does map onto the positive integers.

Reviewing the material in this section now shows even more clearly why we number association patterns with a preference for the more right associated over the more left associated. This action gives Γ_1 the first place in the enumeration, which it deserves for its role in producing Θ_1 through Θ_5 and producing, by continued insertion of the combinator B, the infinite sequence of superkernels. The preference for right-associated patterns also matches well the implied preference for kernels that are not far from the superkernel from which they are obtained, not far in the sense that fewer rather than more global expansions are required to obtain them. Of course, we do encounter one drawback with our preference and the corresponding enumeration; global expansions that violate the 1's rule lead to kernels with lower numbers in the enumeration when compared to those yielded by global expansions that satisfy the 1's rule. Since the enumeration is driven by the letter patterns in the superkernels, which in turn determine the letter patterns in the kernels derived from them, we also see why superkernels play an even more important role in our study than do kernels.

4.4.3.3. Comparing Combinators within a Class

We close this section (Section 4.4.3.3) with a hint about the relation that the various combinators within a given class bear to each other, putting that hint in the context of information we already know. We mostly confine our attention to Θ_1 through Θ_5, but note that the observation applies to the other classes as well.

As we noted earlier, each of Θ_1 through Θ_5 is irreducible, contains the same number (3) of occurrences of W, and the same number (5) of occurrences of B. Also as we noted, the number of occurrences of W is equal to the power (3) of the generator of the superkernel Γ_1, and the number of

occurrences of B is equal to $2k$-1, where k is the power of the generator in the formula for Γ_1 (or the appropriate superkernel). The structure—and therefore the precise pattern of the occurrences of B and W within a combinator—of the combinators within a class varies according to when, which type of, and how many global expansions are applied to the superkernel to obtain the corresponding kernel, which in turn determines how many expansions must be then applied to free the fixed point combinator from the Skolem constant f or, equivalently, from the variable x.

We thus see how the properties of each combinator are essentially completely determined by those of its corresponding kernel, which are in turn determined by those of the superkernel from which the kernel is obtained and those of the path pursued to obtain it. This observation produces added interest when one takes into account the fact that Θ_1 through Θ_5 are provably pairwise unequal—if extensionality is not present—but the kernels Γ_1 through Γ_5 are all equal even when extensionality is absent; Γ_1 through Γ_5 each reduces to Γ_1. Again, we find that much of the mystery for constructing fixed point combinators—at least, for B and W—is unraveled by studying superkernels and the kernels derived from them. We also add to one's understanding of why we are so fond of and so intently interested in superkernels. As we study sets of combinators other than that consisting of B and W alone, the reasons for our preoccupation with superkernels will become even clearer.

Now, the appropriate action to take is to—finally!—present the systematic kernel method for the construction (when they exist) of fixed point combinators. As we shall see, the kernel method applies regardless of which set P of combinators one is asked to study—or at least it has to all sets we have studied. The waiver we have added to the preceding sentence does not arise from the fact that sets exist for which one cannot construct a fixed point combinator; we do study such sets later in this report, especially in Section 5.3.6. Rather, the waiver arises from the fact that we have not yet succeeded in proving that the kernel method is *complete*, that is, always succeeds when it should; we address that question in Sections 5.4.1.1 and 5.5.1.

4.5. The Kernel Method for Searching for Fixed Point Combinators

We now come to the heart of this report, the presentation of the systematic method for searching for fixed point combinators. Since the method relies heavily on the use of kernels, we naturally call the method the *kernel method*. The kernel method, as we commented in the introduction, appears to be the first systematic means for searching for fixed point combinators. Fixed point combinators were known as paradoxical combinators in the early days of combinatory logic, because the Russell Paradox and other paradoxes can be formulated in terms of fixed point combinators. As commented in the introduction, the Russell Paradox asks one to consider the set S whose members are all sets that are not members of themselves; if S is a member of itself, then S is not a member of itself, and if S is not a member of itself, then S is a member of itself. Fixed point combinators are also of interest because Gödel's self-referential sentence and Kleene's recursion theorem can be interpreted as applications of such combinators [Barendregt81].

As evidence of the power of the new method which relies heavily on the use of kernels, we shall apply it to the successful construction of the infinite set of combinators obtained from the study of Γ_1, Γ_6, Γ_{20}, and the rest of that sequence of superkernels. We thus again see why we are so fond of and so

intently interested in superkernels. For additional evidence, we shall show how use of the kernel method enabled us to extract far more from the *B*-and-*W* mine—enabled us to make the startling discovery of an infinite class of infinite sets of fixed point combinators constructible from *B* and *W* alone. That infinite class contains as one of its members the infinite set $\Theta(BW)$ of fixed point combinators. That infinite set of fixed point combinators contains, among others, Θ_1 through Θ_5. In Section 4.4.3.2, we showed how one can enumerate $\Theta(BW)$.

For evidence that establishes the generality of the kernel method, we again study Smullyan's question [Smullyan84]: Can one construct a fixed point combinator from *B* and one other regular combinator? However, in contrast to relying on the use of the combinator *W*, we answer that question in the affirmative by studying other combinators to take the place of *W*, among which is *N* with

$$Nxyz = xzyz$$

for all combinators *x*, *y*, and *z*, where, by a convention in combinatory logic that we observe here, all expressions are assumed to be left associated unless otherwise indicated. We also observe the convention that *x*, *y*, and *z* and such always refer to variables that range over all combinators. The combinator *N* is a new combinator, at least in the sense that neither Smullyan nor Barendregt are familiar with it—which we learned, respectively, from a phone conversation and a letter—and in the sense that we have not as yet found any references to it in the literature. In addition to the answer relying on the use of the combinator *N*, we also present alternative answers to the Smullyan question, among which is that focusing on *B* and *H* alone with

$$Hxyz = xyzy$$

(see Section 5.3.5).

Since the application of the kernel method to the study of certain sets, such as that consisting, respectively, of *B* and *W* and *B* and *N*, is complex—as we shall see when we turn to specific applications in Section 4.5.4—we first apply the method to two very simple cases. Fortunately, as we shall see, the complexity arises from the properties of the combinators themselves—which is how it should be—and not from the properties of the kernel method. The two simple cases demonstrate the versatility and ease of use—for a researcher or an automated theorem-proving program—offered by the kernel method for constructing fixed point combinators. A study of those two cases will amply prepare one for the thorough exploration of the *B*-and-*W* mine and the *B*-and-*N* mine that follows.

Definitions and Conventions

For the person who has decided to begin with this section rather than reading our chronological account of how we formulated the kernel method, we shall present in this section certain material we have already covered. By taking this action, we can make this section virtually self-contained and, for those who have read the earlier sections, provide a review and summary of that earlier material. In addition, we shall present new material to enable us to fulfill certain promises concerning the generalization of various concepts and concerning the fine details required to fully understand the kernel method. The treatment of this material will be brisk.

We begin with the definitions of a kernel and of a superkernel. However, in contrast to the earlier definitions given, respectively, in Sections 4.3 and 4.4.1, here we define kernel and superkernel in their full generality and discuss the earlier-presented concepts of simple kernel and simple superkernel as special cases. In addition, we treat the concepts of kernel and superkernel semantically rather than syntactically. (For those who enjoy the historical aspects of research we point out that we formulated the semantic definitions we give in this section almost eighteen months after we had discovered the concepts as syntactic objects, as expressions satisfying certain requirements.) We also include other definitions that are relevant, and various examples to aid in one's understanding of the different concepts.

Definition. The sequence $C_0, C_1, \ldots, C_k$ is a *reduction*, relative to the proper combinators $A_1, A_2, \ldots, A_k$, of the expression C to the expression D if and only if (1) $C = C_0$, (2) C_i is obtained from C_{i-1} by applying (in the spirit of equality substitution) the left side of A_i, (3) $C_k = D$, and (4) the terms that are replaced are each free of variables.

For our next definition, that of *kernel*, we have two choices, syntactic and semantic. Although the syntactic form of the definition—which we give first—is clearly the simpler, the semantic form captures the precise mathematical nature of the concept of kernel as a set rather than as an expression.

Definition, syntactic. Where P is a set of combinators, the expression Γ is a *kernel* with respect to P if and only if there exists a sequence $A_1, A_2, \ldots, A_k$ with A_i in P such that Γ reduces to $x\Gamma$ by applying A_i in order as reductions, and Γ is expressed purely in terms of elements from P and x, where x ranges over all combinators.

For the semantic definition of kernel, we need the definitions of *fixed point of a combinator*—not to be confused with fixed point combinator—and *P-reducible fixed point of a combinator*. In addition, so that the precise nature of the concept of kernel will be clear, we shall follow its definition with a detailed discussion focusing on the need for the added precision present in the semantic definition.

Definition. Where f and g are given combinators, the combinator g is *a fixed point of* f if and only if $g = fg$.

Definition. Where f and g are given combinators, and where P is a given set of combinators, the combinator g is a *P-reducible fixed point of* f if and only if (1) g is a fixed point of f and (2) there exists a reduction that reduces g to fg in which all of the steps in the reduction are with elements of P.

Definition, semantic. Where C is the set of all combinators, and where P is any given subset of c, the set Γ of P-reducible fixed points is a *kernel with respect to P* if and only if (1) for all f in C there exists exactly one P-reducible fixed point Γ_f of f in Γ and (2) there exists a unique sequence of reduction steps with elements of P with the property that this sequence reduces Γ_f to $f\Gamma_f$ for any f in C. Where Γ is a kernel with respect to some given set P, and where x implicitly ranges over all combinators, we say that Γ_x reduces to $x\Gamma_x$ or, simply, that Γ reduces to $x\Gamma$. Finally, we frequently refer to a kernel by its name by saying, for example, the kernel $W(Bx)(W(Bx))$, or by the pattern of symbols that occur in one of its elements by saying, for example, the kernel $W(Bf)(W(Bf))$.

If Γ is a kernel with respect to some given set P of combinators, then Γ induces a function that maps every combinator f to a P-reducible fixed point of f. The image of each combinator f under this induced function is Γ_f. The range of the induced function is the kernel Γ itself. Even further, the P-reducible

fixed points of the set Γ are very tightly coupled. In particular, if one considers two combinators $f \neq g$, then Γ_f is identical to Γ_g except that certain occurrences of f have been replaced with occurrences of g.

Let us study the various aspects of the concept of kernel, especially since one could easily conclude that the given definition is needlessly complicated. First of all, to see why we cannot say "all occurrences of f", let us consider the set P consisting of the single combinator L. Only one kernel exists with respect to P, as far as we know. That kernel is $Lx(Lx)$. One of its elements is $LL(LL)$, and one of its elements is $LB(LB)$. Clearly, although these two given elements of the kernel $Lx(Lx)$ are very tightly coupled, one cannot replace *all* occurrences of L in the first and obtain the second. Instead, one must replace only the second and fourth occurrences of L by B.

As for the second important aspect of our definition, defining a kernel with respect to some given set P to be a set of P-reducible fixed points is not sufficient, for such a definition does not adequately couple the elements of the set. To see why this is so, let us focus on the set P consisting of the combinators B and W. From P, as we learned in Section 4.4.2, one can construct an infinite set of distinct kernels (kernels in the sense of the formal and semantic definition we have just given). Among those kernels are G_1, G_2, G_3, and the like. Were it not for the second condition of our definition of kernel—the condition that tightly couples the elements of a kernel—one could construct a set Δ of P-reducible fixed points, one for each element of the infinite set C of combinators, by replacing x in Γ_1 by f_1, x in Γ_2 by f_2, ..., for the elements f_i of C. Where $f \neq g$, there do not exist two elements Δ_f and Δ_g of the resulting set such that Δ_g can be obtained from Δ_f by simply replacing f by g. In contrast, when we informally define a kernel as a Γ that reduces to $x\Gamma$, implicit in this informal definition is the existence of a unique reduction that appropriately couples all of the elements of Γ.

Since we have completed the detailed discussion that addresses the need for the added precision in the semantic definition of the concept of kernel—when compared to the syntactic—we now turn to the definitions of kernels of various types.

Definition. Where P is a set of combinators, the set Γ is a *superkernel* with respect to P if and only if (1) Γ is a kernel with respect to P and (2) exactly one reduction applies to each of the elements of Γ and that reduction involves the first symbol in Γ. From the syntactic viewpoint, a *superkernel* is an expression that is a kernel that admits exactly one reduction, a reduction that involves the first symbol of the expression. In other words, the only reduction step that applies to a superkernel—from the semantic or syntactic viewpoint—is a reduction that satisfies the 1's rule, to be defined almost immediately.

As one can show with the appropriate review, the kernels and superkernels encountered in the preceding material all have one property in addition to those given in the more general definitions. Specifically, but informally, for each of the kernels and superkernels on which we have previously focused, all of the corresponding set of reduction steps that establish each to be, respectively, a kernel or a superkernel must satisfy the 1's rule, which we define here again. Indeed, that additional property explains why we referred to those kernels and superkernels, respectively, as *simple kernels* and *simple superkernels*. The corresponding formal definitions rely on the following concepts.

Definition. The *position vector* of a term is a k-tuple that gives the relative position of the term within the equation for a combinator or within some expression involving combinators, see Section 2.3.2 for examples and discussion.

Definition. The reduction $C_0, C_1, \ldots, C_k$ of the expression C to the expression D satisfies the *1's rule* if and only if the *into term* in each C_i has a position vector consisting of all 1's.

Definition. The set Γ is a *simple kernel* with respect to the set P of combinators if and only if (1) Γ is a kernel with respect to P and (2) the reductions that reduce each element Γ_f to $f\Gamma_f$ all satisfy the 1's rule. From the syntactic viewpoint, a *simple kernel* is a kernel Γ such that all reduction steps that reduce Γ to $x\Gamma$ satisfy the 1's rule.

Definition. The set—syntactically, expression—Γ is a *simple superkernel* with respect to the set P of combinators if and only if Γ is a superkernel with respect to P and, as a kernel, is also simple.

Between simple kernels and (so to speak) ordinary kernels, we also consider *semisimple kernels*; similarly, we consider *semisimple superkernels*. For those two concepts, we need the following definition.

Definition. The reduction $C_0, C_1, \ldots, C_k$ of the expression C to the expression D satisfies the *pseudo 1's rule* if and only if the *into term* in each C_i has a position vector consisting of all 1's, where the computation of the position vector is permitted to ignore any and all occurrences of leading f's with f an arbitrarily chosen combinator.

To see what we mean by *leading f*, in the expression

$$f(f(ff(W(Bf)))),$$

only the first two occurrences of f are considered leading f's.

Definition. The set Γ is a *semisimple kernel* with respect to the set P of combinators if and only if (1) Γ is a kernel with respect to P and (2) the reductions (if any) that reduce each element Γ_f to $f\Gamma_f$, after f isolates, all satisfy the pseudo 1's rule. From the syntactic viewpoint, an expression Γ is a *semisimple kernel* if and only if Γ is a kernel such that all reductions (if any) that reduce Γ to $x\Gamma$ after x isolates satisfy the pseudo 1's rule.

Definition. The set—syntactically, expression—Γ is a *semisimple superkernel* with respect to the set P of combinators if and only if (1) Γ is a superkernel with respect to P and (2), as a kernel, Γ is also semisimple.

Since the set of reduction steps that are needed to reduce an element Γ_f of a kernel to $f\Gamma_f$ after f isolates may be empty, every simple kernel is a semisimple kernel. (It follows that every simple superkernel is a semisimple superkernel.)

Examples and Relationships among Kernels of Various Types

Before we turn to specific examples, let us make certain observations concerning kernels in general. For that purpose, we let P be an arbitrarily chosen set of combinators for which kernels exist with respect to P, and let Γ be such a kernel.

To begin with, technically speaking, Γ is a function of the arbitrarily chosen combinator f, where Γ reduces to $x\Gamma$. Since Γ reduces to $x\Gamma$—which follows from the fact that Γ is a kernel—$\Gamma = f\Gamma$ for any arbitrarily chosen combinator f. Since f is an arbitrarily chosen combinator, $\Gamma = x\Gamma$ for all combinators x, which implies that P satisfies the weak fixed point property—for all x, there exists a y, such that $y = xy$.

The proof of this assertion is immediate; one simply lets y be Γ. In other words, any expression that is a kernel with respect to a set P of combinators will always serve as the y needed to show that that set satisfies the weak fixed point property. Just as the y, in the equation for the weak fixed point property, depends on x, so also does the kernel depend on x. However—and here we have a significantly stronger result—if the y that satisfies the equation for the weak fixed point property is actually a kernel, then y reduces to xy, rather than simply being equal to xy.

Let us now turn to various examples. For the set P consisting of L alone with $Lxy = x(yy)$, the expression $Lx(Lx)$ is a simple superkernel, and, therefore, a semisimple superkernel, with respect to P. If we add the combinator S to P with $Sxyz = xz(yz)$, then $Lx(Lx)$ is still a simple superkernel with respect to the expanded set of combinators. For a superkernel that is semisimple but not simple, we consider the set P consisting of B and M with $Bxyz = x(yz)$ and with $Mx = xx$, and the expression $BxM(BxM)$. To present a superkernel that is not even semisimple, we consider the set P consisting of B, S, and I with $Ix = x$, and the expression $S(Bx)I(S(Bx)I)$. As for a simple kernel that is not a superkernel, we can focus on the set P consisting of B and W with $Wxy = xyy$, and the expression $BWBx(BWBx)$. If we next focus on the expression $W(Bx)(BWBx)$, we have a semisimple kernel that is not a superkernel. Finally, if we again consider the set P consisting of B, S, and I, the expression $SII(I(Bx(SII)))$ is a kernel with respect to P, but not a superkernel and not even semisimple.

With these definitions and examples in hand, we can now turn to the kernel method itself.

4.5.1. Description of the Two-Stage Kernel Method

The kernel method for searching for fixed point combinators can be applied as a two-stage method or a three-stage method. We first focus on the two-stage version and, after completing that discussion, then turn to the three-stage version, a discussion of the properties of the kernel method and various conjectures, and a summary of how the kernel method is applied. Consistent with our custom, we present the two-stage version of the kernel method by giving ever more detailed pictures of it.

4.5.1.1. Top Level of Detail

At the top level, the kernel method consists of two stages. In the first stage, the object is to find kernels with respect to the set P of combinators under consideration. The object of the second stage of the kernel method is to search for fixed point combinators constructible from the elements of that set P. The connection between the two stages rests with the use by the second stage of the kernels found in the first stage. Of course, depending on the properties of the set P of combinators, failure may occur at either stage—as it should.

Motivation for Interest in Kernels

Our interest in kernels comes in part from the typical desire in mathematics or in logic of finding necessary and sufficient conditions for some property to hold. The property in focus here, of course, is the strong fixed point property. We are seeking necessary and sufficient conditions for a set P of combinators to satisfy that property. The following condition is sufficient. If there exists a Γ such that Θf reduces to Γ which in turn reduces to $f\Gamma$ for an arbitrarily chosen combinator f, then

$\Theta f = \Gamma = f\Gamma,$

which implies that one can substitute Θf for Γ in $f\Gamma$ to obtain

$\Theta f = f(\Theta f),$

which implies that

$\Theta x = x(\Theta x)$

since f is an arbitrarily chosen combinator. If the condition holds, then of course Γ is a kernel with respect to P—in the most general sense of the term kernel—which explains much of our interest in kernels.

To find a necessary condition for a set P of combinators to satisfy the strong fixed point property—and the connection of such a condition to kernels—we can apply the Church-Rosser theorem. In particular, let P be a set of combinators that satisfies the strong fixed point property. Then

$\Theta x = x(\Theta x)$

for all combinators x, and therefore

$\Theta f = f(\Theta f)$

for an arbitrarily chosen combinator f; in other words, we can focus on a combinator f that is not in P. Next, in the spirit of combinators that are *isolating* (see Section 3.1), let any term s be called *isolated* if the term is not grouped with any terms to its right and no terms occur to its left. For example, in the expression $f(\Theta f)$, the first occurrence of f is isolated. From the Church-Rosser theorem, Θf and $f(\Theta f)$ must both reduce to some common expression E. Since f is not in P, the isolated occurrence of f in $f(\Theta f)$ must remain isolated, which implies that E has the form $f\Delta$ for some Δ. Therefore, a necessary condition for the set P to satisfy the strong fixed point property is that there exists an isolator Ξ, and there exists a Δ such that Θf reduces to $f\Delta$. In Section 3.1, we defined a combinator to be an isolator if the combinator is proper and the first variable on the right side of its equation is isolated from the rest of the right side.

If we could force the issue a little further, then indeed we would have the perfect connection of the strong fixed point property to kernels. Specifically, if we could also prove that, for some Δ of the type just discussed, Θf must first reduce to Δ, then we would have Θf reduces to Δ which in turn reduces to $f\Delta$. Such a Δ would be a kernel, and we would then have a theorem of the form one often seeks: A set P of combinators satisfies the strong fixed point property if and only if there exists a kernel Γ with respect to P such that there exists a fixed point combinator Θ with Θf reducing to Γ_f in Γ, where f is an arbitrarily chosen combinator. Currently, we have not yet succeeded in forcing the issue to this point, which is one of the completeness aspects of the kernel method we address in Section 4.5.3.

If we switch our focus of attention by replacing the strong fixed point property by the weak fixed point property, we can again apply the type of analysis we have just given, and we have another reason for our interest in kernels. In particular, the existence of a kernel Γ with respect to the set P of combinators is sufficient for P to satisfy the weak fixed point property. To amplify this comment, since Γ is a function of the arbitrarily chosen combinator f—for example, the kernel $\Omega\Omega\Omega$ with $\Omega = W(B(Bf))$ depends on the choice of f—the existence of a kernel Γ immediately establishes that the weak fixed point property holds. After all, if Γ reduces to $f\Gamma$, then $\Gamma = f\Gamma$. Since f is an arbitrarily chosen combinator, we

can choose for y in the equation for the weak fixed point property the kernel Γ and have

$$y = xy$$

for all x, with y clearly depending on x.

On the other hand, for a necessary condition for P to satisfy the weak fixed point property, we first note that for all x there must exist a y such that

$$y = xy,$$

which implies that y depends on x. Then, by a slightly more complicated argument than the one we gave earlier that invoked the Church-Rosser theorem, we see that both y and fy must reduce to $f\Delta$ for some Δ and for an arbitrarily chosen combinator f. In other words, we again have arrived at essentially the same necessary condition, a Δ must exist such that y reduces to $f*D$. (Here we have an excellent example of a situation that calls for the use of the definition of kernel in which the Skolem constant f takes the place of the variable x; we choose this form because we wish to prevent the theorem-proving program from following paths that might occur if the program is allowed to instantiate x.) However, again we note that we have not yet proved that the presence of the weak fixed point property for a set P implies that a kernel Γ with respect to P must exist. In particular, we cannot as yet assert that P satisfies the weak fixed point property if and only if a kernel with respect to P exists.

Since we continually focus on both the strong and the weak fixed point properties throughout this report, the preceding remarks show why we are indeed interested in—actually, fascinated with—the study of kernels and, of course, with the study of superkernels.

4.5.1.2. Second Level of Detail

At the next level of detail, the first stage of the kernel method searches for kernels by beginning with the assumption that none exists—by assuming that no Γ exists with Γ reducing to $f\Gamma$—and searching for a proof by contradiction. However, rather than coping with this assumption directly, the first stage of the method focuses on the equation

$$y \neq fy$$

obtained by denying that the weak fixed point property holds for the set P of combinators that is under study. This action is taken in part because no obvious way exists to deny that kernels exist; in particular, the property of one expression reducing to another cannot be stated in first-order predicate calculus. Of course, unless some additional restriction is made, a proof by contradiction obtained with the given inequality as the focus would prove only that the weak fixed point property holds for P. Therefore, to obtain a proof that a kernel exists—more precisely, to actually produce a kernel—the first stage of the kernel method is required to use expansions with the elements of P.

To comment further on the distinction between proving that the weak fixed point property holds and proving that a kernel exists, we emphasize that we are not interested in simply finding a y such that $y = xy$; rather, our goal is to find a y such that y reduces to xy. This distinction is important, as we saw in Section 4.2.2 where we briefly examined the case in which there exists a y that satisfies the equation for the weak fixed point property but fails to be a kernel. For a second example of such a y—one somewhat pertinent to our study of B and W—if one considers $BWBx(W(Bx))$, one finds that indeed $y = xy$, but y does not

reduce to xy. In other words, this *y* can be used to prove that the weak fixed point property holds for the set *P* consisting of *B* and *W* alone, but the direct consideration of the expression itself does not prove that a kernel exists—although, its use, if the appropriate actions are taken, sheds some light on the possibility of constructing a kernel with respect to the set *P*.

Because of this distinction between one expression being equal to another and—the stronger property—one expression reducing to another, we use expansions in the search for a kernel. The first stage of the kernel method is successful in finding a kernel when an inequality is obtained that contradicts one of the equations that captures the actions of an element of *P* or contradicts the axiom of reflexivity.

Second Stage at the Second Level of Detail

In contrast to the first stage where the weak fixed point property is the focus of attention, the second stage of the kernel method begins with the assumption that the strong fixed point property fails to hold for the set *P* being studied. The search is conducted by applying the elements of *P* as expansions to the various kernels that are found in the first stage of the method, where the kernels that are found contain occurrences of the Skolem constant *f*. If we replaced *f* by the variable *x*, we would be violating the strict rules of combinatory logic—one cannot, strictly speaking, reduce or expand expressions containing variables—but, more important, the automated theorem-proving program might follow paths that would then exist because variables unify with terms that do not unify with constants.

In this stage, the program "knows" it has succeeded when it finds an expression of the form Θf. Of course, for the program to have actually constructed a fixed point combinator Θ, Θ must be free of any occurrences of *f*; the program does not test for this freeness property, but instead leaves that task to the researcher. The task can be made trivial if the researcher uses the ANSWER literal and uses the clause

¬EQUAL(a(y,f(y)),a(f(y),a(y,f(y))))| ANSWER(y)

whose first literal arises from assuming that the strong fixed point property does not hold.

4.5.1.3. Third Level of Detail

For the third level of detail concerning the kernel method, we focus on the actions one can take if the method is to be applied by an automated theorem-proving program. In particular, we come to the answers to questions about problem representation, choice of inference rule, and the type of strategy that one might employ.

Representation for Stage 1 of the Two-Stage Kernel Method

For the first stage of the kernel method, one can present the assignment to a theorem-proving program by choosing clauses that capture the actions of the combinators in *P*, a clause for reflexivity of equality, and a clause that corresponds to assuming that the weak fixed point property fails to hold. For example, if the combinator *B* is an element of *P*, then the clause

EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))

can be used. The clause

EQUAL(x,x)

suffices for reflexivity. However, rather than simply requiring the use of a clause denying that the weak fixed point property holds, the kernel method requires using the clause

¬EQUAL(y,a(f,y)) | ANSWER(kernel(y))

instead.

We require appending the ANSWER literal to the clause one would ordinarily use so that, when the program succeeds in applying the first stage of the kernel method, the kernel will be found as the argument of the ANSWER literal (see Section 2.6). Use of the ANSWER literal saves the researcher the trouble of solving for the kernel the program finds; in particular, one is not required to analyze the various unifications that were used to complete the corresponding proof. We recommend the use of the function *kernel* within the ANSWER literal as a mnemonic to permit the researcher to easily follow the program's progress. As we discussed in Section 2.6, when the ANSWER literal is used, the program ignores this literal when testing to see whether a proof has been completed.

Inference Rule for Stage 1 of the Two-Stage Kernel Method

Having discussed the representation of the problem, we can quickly answer the question concerning the choice of inference rule. By definition, paramodulation is the choice for the kernel method.

Strategy for Stage 1 of the Two-Stage Kernel Method

For the following discussion of the strategy used for stages 1 and 2, one might find it instructive to consult An Example of Applying the Kernel Method, which we give in the unnumbered section whose title is "Strategy for Stage 2 of the Two-Stage Kernel Method". One can choose the first illustration, if one wishes to take the viewpoint of automated theorem proving, or one can choose the second, if one wishes to take the viewpoint of combinatory logic. If one wishes to understand how we chose the type of strategy to use, we give that discussion when we come to the fourth level of detail.

The first aspect of our strategy for stage 1 requires that the expansions always be applied to the right side of the inequalities that can be traced to the assumption that the weak fixed point property fails to hold. After all, we are seeking a y that reduces to xy, rather than merely trying to prove that there exists a y with $y = xy$. We next require that all expansions satisfy the 1's rule; later in this report, we shall discover that this requirement must be relaxed when, for example, one is searching for semisimple kernels, kernels such as $W(Bf)(BWBf)$. Finally, we require that every expansion be applied to a term containing an occurrence of f. Again, we shall discover that this requirement also must be relaxed; otherwise one would fail to find kernels such as $LO(LO)f$ with $Lxy = x(yy)$ and $Oxy = y(xy)$.

Contrary to the typical use of an automated theorem-proving program in which finding a single proof ends the search, the first stage of the kernel method is ordinarily permitted to continue finding one proof after another; each proof corresponds to finding a new kernel. In fact, the program sometimes finds a second kernel by extending the path of inquiry that has just succeeded in finding a smaller kernel. For example, if the program continues along the path that yields the kernel $W(Bf)(W(Bf))$ and further expands the clause that leads to this kernel, it finds the kernel $BWBf(BWBf)$.

Representation for Stage 2 of the Two-Stage Kernel Method

With regard to the second stage of the kernel method—keeping our comments at the third level of detail—the program continues to use almost all of the input clauses that are used in the first stage, those clauses that capture the actions of the combinators in P and the clause for reflexivity. In addition, the program uses some of the clauses that are deduced by the first stage—specifically, those that give the information about which kernels were found—but we require replacement of the predicate ANSWER by the predicate FPF to mean fixed point of f (see Section 3.4), and removing the occurrence of the function *kernel*. These clauses take the form

FPF(Γ)

where Γ ranges over the various kernels that are found in the first stage of the kernel method. This notational change meshes well with the clause

¬FPF(a(x,f)) | ANSWER(fixed(x))

which can informally be thought of as meaning "if a kernel Γ can be expanded to the point at which an expression of the form Θf is obtained, then a fixed point combinator Θ has been constructed". As already noted, the program "knows" it has succeeded when it finds an expression of the form Θf. If Θ is free of any occurrences of f—which the researcher can determine by inspecting the argument of the ANSWER literal—then a fixed point combinator has been constructed.

From the viewpoint of combinatory logic, the reason that finding such a Θ is sufficient for knowing that a fixed point combinator has been constructed is that, if one starts with a kernel $\Gamma = f\Gamma$ and succeeds in expanding Γ to Θf, one can then apply the same expansion to expand $f\Gamma$ to $f(\Theta f)$. In other words, since two expressions are known to be equal if one can be obtained by expansion from the other, the program has found a Θ with $\Theta f = f(\Theta f)$. Such a result asserts that a fixed point combinator Θ has been constructed from the elements of P since, from one viewpoint, we can then simply replace f by x without destroying the underlying argument. We thus again see why both definitions of a kernel, that in which a constant f occurs and that in which a variable x occurs in place of f, are useful. From a different viewpoint, the program is implicitly focusing on expressions of the form

$$\Theta f \neq f(\Theta f)$$

for some Θ expressed in terms of the elements of P alone, where the inequality comes from denying that Θ is a fixed point combinator.

Indeed, a contradiction will be found by the program exactly when an expression of the form Θf is deduced. Unfortunately, Θ may still contain occurrences of f, which is contrary to the goal; however, when a fixed point combinator has in fact been constructed, Θ will be free of such occurrences when a contradiction is obtained.

Inference Rule for Stage 2 of the Two-Stage Kernel Method

As expected, the kernel method requires the use of paramodulation as the inference rule.

Strategy for Stage 2 of the Two-Stage Kernel Method

With regard to strategy, we recommend that the program be given the kernels one at a time for its search for fixed point combinators. Next, we usually recommend treating the generator of the kernel under consideration as a constant—for example, $W(Bf)$ is the generator of the kernel $W(Bf)(W(Bf))$—and recommend using expansions that satisfy the 1's rule until exactly one occurrence of the generator remains. Our experience so far suggests that to ignore the use of the 1's rule does not lead to useful information. However, as we shall see when we study the set P consisting of L and S alone, there exist cases where one must ignore the recommendation of temporarily treating the generator of the kernel as a constant.

When the program has progressed to the point at which the generator of the kernel under consideration occurs exactly once, the generator is then replaced by its definition to permit further expansion with the elements of P. At that point, we use a strategy similar to that used in the first stage of the kernel method. We then impose the strategy of requiring that all expansions involve a term containing the constant f, and that all expansions observe the 1's rule. However, in contrast to the search for kernels, we know of no cases in which useful information is missed if one observes the given restriction strategy. In those cases in which one cannot obtain an expression in which the generator occurs exactly once—such as that focusing on S and L—the program is permitted the immediate replacement of the generator with its definition followed by expansion with the elements of P, subject to the given restrictions.

An Example of Applying the Kernel Method from the Viewpoint of Automated Theorem Proving

(1) EQUAL(x,x)
(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))
(3) EQUAL(a(a(W,x),y),a(a(x,y),y))
(4) EQUAL(a(M,x),a(x,x))
(5) ¬EQUAL(y,a(f,y)) | ANSWER(kernel(y))

(6) ¬EQUAL(a(y,z),a(a(a(B,f),y),z)) | ANSWER(kernel(a(y,z)))
(7) ¬EQUAL(a(y,y),a(a(W,a(B,f)),y)) | ANSWER(kernel(a(y,y)))
(8) ANSWER(kernel(a(a(W,a(B,f)),a(W,a(B,f)))))

Clause (7) contradicts clause (1), establishing that a kernel, $W(Bf)(W(Bf))$, has been found.

(1) EQUAL(x,x)
(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))
(3) EQUAL(a(a(W,x),y),a(a(x,y),y))
(4) EQUAL(a(M,x),a(x,x))
(5) EQUAL(Ω,a(W,a(B,f)))
(6) FPF(a(Ω,Ω))
(7) ¬FPF(a(x,f)) | ANSWER(fixed(x))

(8) FPF(a(M,r))
(9) FPF(a(M,a(W,a(B,f))))
(10) FPF(a(a(a(B,M),W),a(B,f)))
(11) FPF(a(a(a(B,a(a(B,M),W)),B)),f)
(12) ANSWER(fixed(a(a(B,a(a(B,M),W)),B)))

In clause (11), we have an expression of the form Θf with f free. Since clauses (11) and (7) contradict each other, and since Θ contains no occurrences of f, $\Theta = B(BMW)B$ is a fixed point combinator constructed from B, W, and M.

An Example of Applying the Kernel Method from the Viewpoint of Combinatory Logic

(1) $x = x$
(2) $Bxyz = x(yz)$
(3) $Wxy = xyy$
(4) $Mx = xx$
(5) $y \neq fy$

(6) $yz \neq Bfyz$
(7) $yy \neq W(Bf)y$

Statement (7) contradicts statement (1), establishing that a kernel, $W(Bf)(W(Bf))$, has been found.

(1) $x = x$
(2) $Bxyz = x(yz)$
(3) $Wxy = xyy$
(4) $Mx = xx$
(5) $\Omega = W(Bf)$
(6) $\Omega\Omega = f(\Omega\Omega)$

(7) $M\Omega = M(W(Bf)) = f(\Omega\Omega)$
(8) $BMW(Bf) = f(\Omega\Omega)$
(9) $B(BMW)Bf = f(\Omega\Omega)$

From statement (9), we have an expression of the form Θf with f free. Since Θ contains no occurrences of f, $\Theta = B(BMW)B$ is a fixed point combinator constructed from B, W, and M.

4.5.1.4. Fourth Level of Detail

Since we now have a somewhat detailed description of how the program can apply the two-stage kernel method, let us turn to the fourth level of detail, that giving an analysis of the reasons for some of our decisions with regard to strategy and related matters. Since various subtleties and complexities arise in the following discussion, one might prefer to delay reading this material until we have applied the

kernel method to various sets P. In that event, one might wish to move directly to Section 4.5.2, where we present the kernel method as a three-stage approach for searching for fixed point combinators. However, since that treatment and the material that follows it touch on points that may be somewhat obscure until one has experienced various applications of the kernel method, the wise course might be simply to avoid that material for now. On the other hand, for those who wish a glimpse of the entire picture, we note that, after we discuss the three-stage view of the kernel method, we turn in Section 4.5.3 to an examination of properties of the method, including various conjectures.

An Analysis of Our Choice of Strategy for Stage 1 of the Two-Stage Kernel Method

We begin our analysis by focusing on our decisions regarding the choice of strategy to use in stage 1 of the two-stage kernel method. We prefer to use expansions applied to the right side of inequalities that can be traced to the inequality

$$y \neq fy$$

rather than using reductions—even though we are seeking a Γ that reduces to $f\Gamma$—because there exist kernels that will not be found with reductions but that will be found with expansions. For example, the kernel $\Omega\Omega\Omega$ with $\Omega = W(B(Bf))$—a kernel whose derivation we give in Section 4.7—is found in stage 1 with the approach we have given. But this kernel will not be found if reductions are used in place of expansions.

The second aspect of our strategy—the requirement that all expansions satisfy the 1's rule—is present because we are typically seeking simple kernels. In the definition of a simple kernel all reductions are required to satisfy the 1's rule, and no reduction is applied after f isolates. On the other hand, since there exist sets P of combinators that satisfy the strong fixed point property such that the kernels corresponding to the fixed point combinators are semisimple, for those cases one must relax the strategy. In particular, when a kernel is semisimple—which means, if the kernel is not simple, that a reduction must be applied even after f isolates—one is sometimes forced to paramodulate *into* a variable, *into* the variable y of fy. Equivalently, but from the viewpoint of combinatory logic, one may be required to expand the variable y of fy to find the desired semisimple kernel. Since little knowledge in this regard exists now, we immediately encounter an interesting area for research: Under which conditions is one required to consider *into terms* that are variables?

Even though we eventually focus on kernels in general, for the examples on which we focus in the next few sections, simple kernels are all that is needed, which means we can observe the given restriction strategy of applying an expansion only if it satisfies the 1's rule. Eventually, since we wish to provide a universal method for searching for fixed point combinators, we must address the problem of searching for semisimple kernels and also for searching for kernels that are not even semisimple.

An Example Concerning Kernels That Are Not Semisimple

For an example of the case requiring the definition of kernel to be extended to include semisimple kernels and those that behave even more poorly, we consider the combinators B, C, S, and I, whose respective actions are given with the following equations.

$$Bxyz = x(yz)$$

$Cxyz = xzy$

$Sxyz = xz(yz)$

$Ix = x$

This set of combinators is of particular interest in that one can construct any noneliminating combinator from these four combinators—formally, this set is complete for noneliminating combinatory logic. (In Section 3.3, we defined a combinator to be eliminating if it is proper and if there exists at least one variable appearing on the left side of its equation that does not appear on the right side.)

Next, we consider the fixed point combinator $B(BM(CSI))B$ and its (more general, and not even semisimple) kernel Γ equal to the square of $CSI(Bx)$. To reduce this kernel Γ to $x\Gamma$, one must apply a reduction with I after x isolates, and a reduction that does not even satisfy the pseudo 1's rule. As expected, we wish this Γ to be captured by some definition of kernel since the spirit of that concept focuses on an expression for which reductions cycle Γ to $x\Gamma$, to $x(x\Gamma)$, to $x(x(x\Gamma))$, forever, and the given Γ certainly has that property. Even more important, we wish to present a method that will easily find this fixed point combinator, and the kernel method will do so, with the more general definition of kernel given at the beginning of this section. In other words, we are showing why our original concept of a kernel proves to be inadequate—showing why the study of kernels that we now classify as simple, although it is clearly of interest, is not adequate for a full study of the strong fixed point property. Nevertheless, we admit openly to preferring, for certain aesthetic reasons, simple kernels to those that are not simple.

Now, if one considers this Γ—the square of $CSI(Bx)$—and applies reductions that satisfy the 1's rule until x isolates, one sees that additional reductions are required to obtain $x\Gamma$. Specifically, a reduction with I is needed, a reduction that violates the 1's rule and, in fact, violates the pseudo 1's rule—I must be applied to a term whose position vector does not consist of all 1's, even if x is ignored. In other words, we see that Γ is not even semisimple; a reduction with I is required that does not satisfy the pseudo 1's rule.

An Interesting Question

Obviously, we can and did eventually extend the definition of kernel appropriately, but, for now, let us act as if such cases do not exist. Put another way, let us confine our attention to those cases for which we can demonstrate that our original definition of kernel is sufficient—that focusing on simple kernels only. With these actions, we can revisit, but from a slightly different viewpoint, an interesting research question. What properties of a fixed point combinator are sufficient and what properties are necessary for its kernel Γ to reduce to $f\Gamma$ without requiring any reductions after f isolates, where f is an arbitrarily chosen combinator? The examples we discuss will amply illustrate how the version under discussion of the kernel method works—the version that rests on the definition of simple kernel given in Section 4.3—and will establish that its use gives a person or a computer a powerful means for searching for fixed point combinators.

Additional Analysis of Our Choice of Strategy for Stage 1 of the Two-Stage Kernel Method

As for the final aspect of our strategy for controlling stage 1 of the kernel method—the requirement that every expansion apply to a term containing an occurrence of f—we impose this requirement to

prevent the program from finding kernels whose use can lead to the construction of uninteresting fixed point combinators. For example, in stage 1, the program can find the kernel Γ_4, which is equal to the cube of $BW(BBB)f$. If during its search for kernels, the program were to continue along the path that yields Γ_4 and expand with B, then it could find a kernel whose use leads to the combinator $\Theta_4{}'$ with

$\Theta_4{}' = B(WW)(B(BW)(BB)B)$.

The combinator $\Theta_4{}'$ is trivially equal to Θ_4, even in the absence of extensionality, and, in fact, reduces to Θ_4, which makes the combinator $\Theta_4{}'$ quite uninteresting.

Despite the value of avoiding such uninteresting results—which is why we typically require that every expansion be applied to a term containing at least one occurrence of f when searching for kernels—we again note this practice is not free of a price. As indicated earlier, the observance of this practice can cause one to fail to find the desired kernel, a kernel such as $LO(LO)f$. Our only solution now is that of suggesting more research, research that focuses on a strategy to find kernels of all types—simple, semi-simple, and other—but without producing trivial and uninteresting kernels, where a kernel is termed trivial when its use leads to a combinator that reduces to one that will be discovered without the use of the trivial kernel.

An Analysis of Our Choice of Strategy for Stage 2 of the Two-Stage Kernel Method

As for our choice of strategy governing the second stage of the two-stage kernel method, we require that all expansions satisfy the 1's rule because our experience shows that those paths not observing this restriction merely lead to duplicating results. However, we have not proved that this must be the case. We require that every expansion involve a term containing the constant f in order to avoid constructing combinators that are equal to ones that we construct by observing the requirement. For example, if an expansion with B is applied to the term $B(BWB)Bf$ on the path that leads from the kernel Γ_5 to the fixed point combinator Θ_5—applied before the point at which the generator of Γ_5 occurs exactly once—then one can succeed in finding a combinator that reduces to Θ_5. The combinator that is obtained is, therefore, equal to Θ_5, and this equality holds even in the absence of extensionality. Finally, we recommend the strategy of attempting to remove all but one occurrence of the generator of a kernel before replacing the generator by its definition—at least, in many cases—to avoid duplicating results.

If we focus jointly on the requirements that expansions must satisfy the 1's rule and must involve a term containing an occurrence of f, we can see why one might terminate the stage 2 search under the following condition. When the expansions are applied to the point at which f frees, then the search ceases; by freeing, we mean that we have an expression of the form Θf. Of course, as pointed out earlier, Θ is not necessarily a fixed point combinator, for, to be so, it must itself be free of any occurrences of f.

A Duality between Isolating and Freeing

We thus encounter a duality between isolating and freeing, a duality that might merit exploration. The two concepts are dual in the sense that to isolate a term such as f means to produce ft for some t—to separate f from the rest of the expression with f on the left. To free a term such as f means to produce tf for some t—to separate f from the rest of the expression, but with f on the right. The second aspect of the duality between isolating and freeing rests with the fact that one isolates a term, such as f, by applying

reductions, but one frees a term by applying expansions.

As we have seen, both concepts—isolating and freeing—play a vital role for the search for fixed point combinators, even more so if the search is conducted with the kernel method relying on the definition of simple kernel given in Section 3.3. In particular, in stage 1—if the object is to find simple kernels—we search for a Γ that reduces to $f\Gamma$ such that no reductions are applied after f isolates. Even when one searches for kernels that are not simple, one seeks a Γ that reduces to $f\Gamma$, where the isolation of f signals that the goal may be near at hand. One might, upon reading the preceding remarks, comment that we use expansions and not reductions, which adds some confusion. Our response to such a justifiable comment is that we are forced to use expansions because reductions are not always adequate for the task, and, at least for simple kernels, one must not confuse the goal with the method for reaching that goal. Then, in stage 2, we use the kernels found in stage 1 to attempt to construct an expression Θf with f free of Θ. We ordinarily terminate the stage 2 search precisely when f frees—of course, with the intention of having a Θ that contains no occurrences of f.

The concept of isolating f is also of value for the other side of the question concerning the constructibility of fixed point combinators. In particular, when we prove that the strong fixed point property fails to hold for a given set P of combinators—as in Theorem 6 of Section 4.6 and in Corollary 3 of Section 5.3.6.2, when we study the set P consisting of B and L—we rely heavily on showing that one cannot isolate f. In addition, this lack of a path that isolates f is of use when we prove that a set P of combinators fails to possess the weak fixed point property (see Theorem 7 in Section 4.6).

4.5.2. Description of the Three-Stage Kernel Method

Having completed our analysis focusing on why we have chosen the various strategies for controlling both stage 1 and stage 2 of the two-stage kernel method, let us revisit the method, but now present it as a three-stage method and show how superkernels can play an important role. In the first of the three stages, the object is to find superkernels rather than kernels, again using the combinators in P to expand the right side of inequalities that can be traced to the inequality

$$y \neq fy$$

which arises from assuming that the weak fixed point property fails to hold for P. The object of the second stage is to find kernels by focusing on the superkernels found in the first stage. Finally, in the third stage, the object is to construct fixed point combinators from the kernels found in the second stage.

Stage 1 of the Three-Stage Kernel Method

In the first stage of the three-stage version of the kernel method, the program uses the same clauses it would use in the first stage of the two-stage version. Since it is not clear that there exists a simple and straightforward means for preventing the program from pursuing paths that lead to finding kernels in addition to finding superkernels, perhaps the wisest course of action is to search for superkernels as a team consisting of the researcher and the program. For example, in view of the fact that superkernels must not admit a reduction other than that satisfying the 1's rule, the program could be instructed to produce a small set of results, which the researcher could then examine, discarding the ones that, if completed, will admit a reduction that does not satisfy the 1's rule.

We can immediately touch on how one might decide that a partial result will lead to finding a kernel rather than finding a superkernel, giving a fuller treatment in a later section. Earlier in this section, we mentioned the kernels $W(Bx)(W(Bx))$ and $BWBx(BWBx)$. The first of these two is a superkernel, and the second—which is not a superkernel—is obtainable from the first by a global expansion with B. The expansion path that leads to finding the superkernel produces the inequality

$$yy \neq W(Bf)y,$$

which contradicts the axiom of reflexivity. If the ANSWER literal were used, then the clause corresponding to the given inequality would contain a literal whose argument is the superkernel $W(Bf)(W(Bf))$. If the program continues along the same expansion path and expands with B, then the inequality

$$yy \neq BWBfy$$

is obtained, leading to the kernel $BWBf(BWBf)$, which will be found in the corresponding ANSWER literal. To obtain this last inequality—and here we have, perhaps, a strategy that can be used by a person or by a program for avoiding the pursuit of kernels that are not superkernels—the *into term* that is expanded in the preceding inequality is $W(Bf)$. From this example, we can extract—and then attempt to determine under which conditions it works—the following rule for preventing a person or a program from pursuing paths that lead to kernels that are not superkernels. The only terms—from the viewpoint of using paramodulation, the only *into terms*—that are considered for expansion are those that contain the constant f, but that do not end in f. This rule could be used with the two-stage version of the kernel method to have the program pursue only those paths that might lead to superkernels.

The inference rule one must use is, as expected, paramodulation. Unless one is seeking superkernels that require a reduction after the constant f isolates, the search is restricted to applying expansions all of which satisfy the 1's rule.

Stage 2 of the Three-Stage Kernel Method

For the second stage of the three-stage kernel method, the program considers those clauses from the first stage whose argument is a superkernel. The object is to successfully apply global expansions to these superkernels and then, recursively, to any expressions that are obtained. The search along any specific path is terminated when the constant f is freed; further expansions merely produce kernels that lead to uninteresting fixed point combinators, if they lead anywhere.

For example, the kernel Γ_4 that leads to the construction of Θ_4 is equal to the cube of $BW(BBB)f$. One could globally expand Γ_4 with B to get a kernel equal to the cube of $B(BW)(BB)Bx$. From this kernel, one obtains Θ_4', which is equal to Θ_4 even in the absence of extensionality and is, therefore, decidedly uninteresting. Each successful global expansion produces a kernel.

Inference Rule for Stage 2 of the Three-Stage Kernel Method

In contrast to the two-stage approach, paramodulation in the usual sense is not the inference rule to use. Instead, one uses a rule called *hyperparamodulation* [Wos80], a rule that can treat terms many at a time rather than one at a time. Specifically, when an *into term* has been chosen for possible expansion, all like terms are considered simultaneously. In other words, the program applies a global expansion, rather than an ordinary expansion. Like terms are expanded, if possible, simultaneously and in the same

manner.

Strategy for Stage 2 of the Three-Stage Kernel Method

The strategy for finding kernels from superkernels permits using global expansions that violate the 1's rule as well as those that satisfy it. This extra latitude is required to avoid missing useful kernels. For example,

$$\Gamma_2 = W(BBBx)(W(BBBx))(W(BBBx))$$

is obtained from the superkernel

$$\Gamma_1 = W(B(Bx))(W(B(Bx)))(W(B(Bx)))$$

by applying a global expansion with B that violates the 1's rule—equivalently, by applying an expansion that violates the global 1's rule.

Strategy for Stage 3 of the Three-Stage Kernel Method

The third stage of the three-stage method is treated as the second stage of the two-stage method is—relying on the same notation, using paramodulation, and restricted with the same strategy. The search in the third stage is terminated as soon as the constant f is freed. However, as commented earlier, we are not guaranteed that a fixed point combinator Θ has been found; such a Θ must be free of any occurrences of f.

Comparison of the Two-Stage and Three-Stage Versions of the Kernel Method

At least conceptually—and perhaps with regard to efficiency—the three-stage version of the kernel method has somewhat greater appeal than the two-stage version does. In particular, the three-stage version has a more mathematical flavor than does the two-stage version, and its application seems to provide the researcher with more insight concerning the nature of the set P of combinators under investigation. A person might find it more convenient to apply the three-stage version of the kernel method rather than the two-stage, of course substituting algebraic notation for the clause notation that a theorem-proving program would use.

4.5.3. Properties of the Kernel Method

Before we turn to specific applications of the kernel method, we pause to examine certain properties of the method and to present certain relevant conjectures. The kernel method is general; it can be applied to any set P of combinators. The kernel method is easy to apply for a person or a program, at least in principle, and obviously far more effective than the brute-force approach we used to find Θ_1 through Θ_5 in our original attack (see Section 4.1.1) on fixed point combinators constructible from B and W alone. In particular, the kernel method is far more effective than the standard approach that one would take in automated theorem proving or in mathematics or logic. As we noted, the brute-force approach required approximately 20 CPU hours on a Sun 3 workstation to find Θ_1 through Θ_5, and the kernel method required (with ITP) 8 CPU minutes to complete that search. Of course, there exist cases that admit the construction of a fixed point combinator but that require a substantial effort—even if the kernel method is used.

An Example of a More Complicated Fixed Point Combinator

For the person who wishes to immediately see an example of a more complicated fixed point combinator, let us focus on the combinators S and K with

$$Sxyz = xz(yz)$$

and

$$Kxy = x$$

and the shortest (in symbol count) fixed point combinator we know of that is constructible from S and K alone, namely,

$$S(K(SS(SKK)(SS(SK))))(S(KS)K).$$

As far as we know, the given fixed point combinator is in fact shorter than any previously known, if the construction is limited to the use of S and K alone.

Two Questions Concerning Completeness

Although we shall provide strong evidence of the usefulness and power of the kernel method, there remain two unanswered questions concerning the completeness of the method. First, is the kernel method complete, complete in the sense that the method will *always* construct a fixed point combinator for the set P under study when such a construction is possible—when the set P satisfies the strong fixed point property? Equivalently, the first question asks whether the method establishes that the question focusing on the presence of the fixed point property is semidecidable—asks whether the method will always succeed in proving, for those sets P for which it is true, the theorem that asserts that the strong fixed point property holds.

From a slightly different viewpoint, the first question asks whether the following conjecture is true: to every fixed point combinator Θ, there corresponds a kernel Γ such that Θx reduces to Γ which in turn reduces to $x\Gamma$. For this conjecture, as shown earlier when we examined the set P consisting of B, C, S, and I, we of course have in mind the more general definition of kernel given at the beginning of this section (Section 4.5) rather than that given in Section 4.3. We believe that this conjecture is true, and believe that the kernel method—with the more general definition of kernel—is a method that establishes that the question focusing on the presence of the fixed point property is semidecidable.

This conjecture seems most reasonable in that one can easily and quickly prove that if the strong fixed point property holds for a set P of combinators, then the weak fixed point property holds for P also (see Section 3.4, Theorem 3). Therefore, in the obvious sense, the conjecture amounts to asserting that if the strong fixed point property holds for a set P, then the reducible weak fixed point property holds—for all x there exists a y such that y reduces to xy.

To prove the preceding assertion must require more ingenuity than proving that the strong implies the weak since, to prove that strong implies weak, one is showing that if there exists a y such that for all x

$$yx = x(yx),$$

then for all x there exists a y such that

$$y = xy.$$

In other words, in a sense, one is proving that the quantifiers can be interchanged, so to speak, with of course a corresponding change in the relevant equation. To prove that strong implies reducibly weak, on the other hand, one must show that, from an equality, one can obtain a result about reducibility. In the simplest sense, since in general Θx does not reduce to $x(\Theta x)$, one must expect to encounter far more difficulty in proving the more general result.

Implicit in our position regarding the kernel method is the belief that one cannot embark on a search that is infinitely long without finding a kernel or finding a fixed point combinator when either of the two is in fact constructible. We also conjecture that, when one is restricted to the consideration of regular combinators, the fixed point combinators obtained with the kernel method are irreducible—in combinatory logic, have a normal form—whether one uses the two-stage or the three-stage version. However, the restriction to regular combinators may not be enough, for there may exist fixed point combinators, involving regular combinators only, that cannot be reduced to an irreducible form; if so, we indeed have another puzzle to solve.

In addition, we conjecture that restricting the study to the use of regular combinators implies that one need only search for simple kernels. In other words, we conjecture that, to every fixed point combinator with the property that all combinators occurring in it are regular, there corresponds a simple kernel such that the fixed point combinator can be obtained from the kernel by expansion—indeed, can be obtained by using the kernel method restricted to the search for simple kernels.

We can immediately show why the admission of combinators that are not regular presents a problem. The fixed point combinator $LO(LO)$ with $Lxy = x(yy)$ and $Oxy = y(xy)$ does not have a normal form. Its kernel is $LO(LO)f$. Despite the unnerving behavior we encounter here, both this fixed point combinator and its kernel can be found with the kernel method. Unfortunately, to find this kernel requires an expansion that does not involve a term containing f, which means—as we commented earlier—that cases exist for which our strategy does not apply universally. Incidentally, with this example, we witness an unusual event—Θx reduces to $x(\Theta x)$, where Θ is $LO(LO)$.

The second completeness question focuses on a different aspect of the kernel method. Will the kernel method always find a kernel when one exists for the set P of combinators under study? Of course, the method as given is not complete in this sense, for it does not address the case in which the kernel requires a reduction after the constant f isolates. However, it appears that, if we make certain obvious extensions to the method, the result is complete in this sense. The corresponding conjecture, as well as those just mentioned, might prove to be fascinating research topics.

Using the Kernel Method to Show That the Strong Fixed Point Property Fails to Hold

Next we come to the use of the kernel method for the other side of the question concerning the constructibility of fixed point combinators, the side that asks one to prove that, for the set P under consideration, the strong fixed point property does *not* hold. (We discuss this aspect of combinatory logic more fully in Section 5.3.6.)

For the so-called negative side of the question—although the kernel method is designed to search for kernels and then attempt to use those it finds to construct fixed point combinators—the method can still provide important clues. Specifically, if one studies a set P of combinators with the kernel method,

and if a substantial amount of effort by the researcher or a substantial amount of CPU time by the computer is expended without constructing a fixed point combinator, then the conjecture that P fails to possess the strong fixed point property is definitely in order. Unfortunately, as is so often true in science, we cannot give a quantitative value for "substantial" for a person or for a computer.

To prove such a conjecture, one must, of course, supply a formal proof. However, an examination of the partial results obtained with the kernel method—whether obtained by person or by computer—may indeed provide an important clue about how to proceed to complete the required formal proof.

The Kernel Method for Studying the Weak Fixed Point Property

Just as one can use the kernel method to study the strong fixed point property for some arbitrarily chosen set P of combinators, one can also use the method to study the weak fixed point property. If the weak fixed point property does in fact hold for the set P, then we conjecture that the kernel method will always succeed in finding a kernel. On the other hand, if a kernel is found, then the weak fixed point property does indeed hold—in fact, it holds in a very strong way. After all, every kernel can serve as a y such that

$$y = xy,$$

which says that, if one finds a kernel, one has proved that the weak fixed point property holds for P.

But, even more, every kernel can serve as a y such that y reduces to xy, the reducible weak fixed point property; the converse, as we pointed out earlier in this section, is not necessarily true of a y for which one has merely shown that $y = xy$. Indeed, the expression $BWBx(W(Bx))$ can serve as a y that satisfies the equation for the weak fixed point property, a y that establishes that the set P consisting of B and W alone satisfies the weak fixed point property. However, one can see by merely applying the obvious reductions that $BWBx(W(Bx))$ is not a kernel—if we call the expression Δ, Δ does *not* reduce to $x\Delta$.

Again, as with the strong fixed point property, if the first stage of the kernel method—in either the two-stage version, or the three-stage version—fails to yield results after a substantial amount of effort or CPU time, then one can strongly conjecture that the weak fixed point property does not hold for the set P under study. If one wishes to prove such a claim true, one must, of course, supply an appropriate formal proof.

A Look Ahead

Let us now turn to various applications of the kernel method—to the consideration of various sets P of combinators—keeping in mind that we shall focus exclusively for a while on simple kernels. Only later, after one has gained familiarity with simple kernels, will we turn to kernels that even lack that property. We have chosen to delay facing the added complexity inherent in the concept of kernel in its full generality until we have focused on enough examples that the researcher has a good grasp of how it all works. One does not pay any significant price for this delay because the study of simple kernels suffices to illustrate most of what is needed for the search for fixed point combinators. We shall show by example how one finds various simple kernels, and then how one considers these kernels in the attempt to construct a fixed point combinator.

Although we shall often rely on the notation of combinatory logic—for those who are interested in automated theorem proving or concerned about how the program functions—one can keep in mind that the inference rule to be used is paramodulation. We shall use expansions that satisfy the 1's rule, applying each to the right side of inequalities that can be traced to the inequality that originates with the assumption that the weak fixed point property fails to hold. We shall also use other aspects of the strategy we have discussed in this section, such as always expanding a term that contains the constant *f*.

As a sample of what is to come, and to illustrate some of the intriguing behavior of combinatory logic, we shall find kernels, such as $W(Bx)(W(Bx))$, that in fact do not lead to the construction of fixed point combinators—at least not for the set *P* that we are studying at the time. Failures of this type are not detected until stage 2, and accurately predicting such failures by merely glancing at the kernel under consideration is not always easy.

There do, however, exist kernels that are obviously useless for the purpose at hand. For example, the kernel $W(B(Bx)W)(B(Bx)W)$ is—with a little insight—of no use for constructing a fixed point combinator from *B* and *W* alone. Its uselessness in this context follows from the fact that, regardless of the number of expansions with *B* and *W*, the final occurrence of *W* in this kernel will maintain its position, which means one can never obtain an expression of the form Θf. A quick explanation of why the last occurrence of *W* stays put rests with the fact that the equations for both *B* and W—$Bxyz = x(yz)$ and $Wxy = xyy$—have the property that the last variable on the right side of the equation is the same as the last variable on the left.

Of course, the two examples we have just examined are little more than nuggets. Therefore, to see how rich the various mines actually are, let us begin by applying the kernel method to the simple examples referred to earlier, and then extensively explore the *B*-and-*W* mine to extract its treasure, and later turn to exploring the *B*-and-*N* mine which, from what we know, has never been explored before by anyone. In particular, the combinator *N* with

$Nxyz = xzyz$

has not been studied by Smullyan or Barendregt in any context.

4.5.4. Applying the Kernel Method

As promised at the beginning of Section 4.5, we begin our study of applying the kernel method with two very simple cases. We then turn to the application concerned with the study of *B* and *W*. For these examples, although we apply the two-stage version rather than the three-stage version, we shall make certain observations concerning the use of the three-stage version, especially for the application to *B* and *W*.

Example 1 of Applying the Kernel Method

For our first application—although example 2 is even simpler—we focus on the set *P* of combinators consisting of *S* and *L* alone, where these two combinators have the following properties.

$Lxy = x(yy)$
$Sxyz = xz(yz)$

As expected, we also have as part of our axioms

$x = x$,

the axiom of reflexivity of equality.

As dictated by the first stage of the kernel method, we also consider the weak fixed point property which states that, for all x, there exists a y such that

$y = xy$.

Of course, we assume that the property does not hold. We obtain

$y \neq fy$

where y is a universally quantified variable that implicitly ranges over all combinators, and, from the viewpoint of automated theorem proving, f is a Skolem constant arising from our assumption that the weak fixed point property fails to hold. From the viewpoint of mathematics, f is a combinator that, by assumption, exists from the assertion that the weak fixed point property fails to hold.

The given inequality is included because, in the first stage of the method, we are searching for kernels, and a very close connection exists between the existence of kernels and the presence of the weak fixed point property, as we noted in various ways. In particular, when one succeeds in constructing a kernel Γ with respect to some set P of combinators, then one has also succeeded in proving that the weak fixed point property holds for P. Although we can only conjecture that the converse is true—that the presence of the weak fixed point property is equivalent to the constructibility of a kernel—we nevertheless act as if it is, which accounts for our use of the given inequality.

To review again why finding a kernel implies the presence of the weak fixed point property, we first note that, when one constructs a kernel, one has proved that Γ reduces to $f\Gamma$ for an arbitrarily chosen combinator f. But that proof also shows that Γ, with all occurrences of f replaced by x, reduces to $x\Gamma$. In other words, showing that a kernel exists with respect to P implies that $\Gamma = x\Gamma$, where Γ is expressed in terms of elements from P and the variable x. This equality is an instance of the equation defining the weak fixed point property. When a specific value of the variable x is chosen—when a specific combinator is under consideration—then the y that must exist and depend on x, required by the weak fixed point property, is obtained by substituting the chosen value for all occurrences of x in Γ and in $x\Gamma$.

The first stage of the kernel method immediately considers applying expansions with L and S to the given inequality and to its descendants, continually searching for a deduced inequality that contradicts one of the three given equalities—the first two that, respectively, give the actions of L and S, and the third, which is the axiom of reflexivity. In contrast to the approach for constructing kernels from a superkernel—the approach that applies simultaneous expansions to like terms—here we apply each expansion to a single occurrence of a term.

Although, ordinarily, the first stage requires at least some expansion, in this case the program immediately finds a contradiction without such steps. The contradiction is provided by the inequality

$y \neq fy$

and the equality

$Lxz = x(zz)$,

which is simply an alphabetic variant of the axiom for L, written in this form to permit us to easily give the substitution that establishes the contradiction. The substitution—corresponding to the successful unification of the two given expressions—establishing the contradiction consists of replacing x by f, z by Lf, and y by $Lf(Lf)$.

We thus see that the first stage of applying the kernel method to the set P consisting of L and S terminates and yields the expression $Lf(Lf)$, which is a kernel since it reduces to $f(Lf(Lf))$ with one application of L. This kernel is obtained either by applying the given substitution and seeing how the variable y is correspondingly instantiated, or by using the ANSWER literal. Immediately, one notes that the distinction between two expressions being equal and the stronger condition that the first reduces to the second comes into play here. In particular, proving that

$$Lf(Lf) = f(Lf(Lf))$$

would not have sufficed, for the definition of kernel requires a Γ that reduces to $f\Gamma$, or, equivalently, a Γ that reduces to $x\Gamma$ where the variable x appears wherever the Skolem constant f appears.

Having found a kernel at stage 1—and without using any expansions—we can turn to the second stage and attempt to construct a fixed point combinator from this kernel, using L and S as possible expansions. If we apply our strategy of writing the kernel in terms of its generator Lf, no expansions apply. We therefore skip that step and instead consider $Lf(Lf)$ directly even though its generator occurs more than once. The only expansion that applies to $Lf(Lf)$ is S, which yields $SLLf$ as the result. Since we now have an expression of the form Θf with Θ containing no occurrences of f—in other words, we have succeeded in freeing f—our work is essentially finished.

Indeed, to complete a proof showing that SLL is the desired object, we merely apply the same expansion to $f(Lf(Lf))$ and obtain

$$SLLf = f(SLLf),$$

proving that SLL is a fixed point combinator constructible from S and L alone. Actually, in the most technical sense, we must either begin with

$$Lx(Lx) = x(Lx(Lx))$$

as the equation for our kernel—which is justified by the given argument establishing that $Lf(Lf)$ is a kernel—or we must add the obvious step of replacing f by x in the equality that completed our proof. In either case, the kernel method for searching for fixed point combinators succeeds in proving that the set consisting of S and L alone satisfies the strong fixed point property.

Some Observations Concerning Example 1

Before turning to the second example, which is even simpler, we make the following observations. The kernel $Lx(Lx)$ is a simple superkernel, since the requirements of the corresponding definition are satisfied. Specifically, no reduction applies other than one satisfying the 1's rule. Next, the attempt to apply global expansions with L and S—with the object of finding kernels from superkernels—yields nothing from this superkernel. Nevertheless, we still have established that, for the set P consisting of L alone, the weak fixed point property holds. In particular, we have

$$Lx(Lx) = x(Lx(Lx)),$$

which is an instance of the equation characterizing the weak fixed point property—for all x there exists a y such that

$$y = xy.$$

We conjecture that no other superkernels, and no other kernels, exist that are constructible from S and L alone. In addition, we conjecture that no other fixed point combinators—other than SLL—are constructible from S and L alone. The second part of our conjecture concerning S and L follows from the more general conjecture concerning the completeness of the kernel method, discussed in the preceding section; in the section on highlights, we return to this completeness aspect in the context of the corresponding open question that asks about the accuracy of the conjecture. Specifically, for every fixed point combinator Θ, does there exist a (not necessarily unique) kernel Γ that expands to Θ?

We immediately have two open questions for possible research. Is $Lx(Lx)$ the only kernel that exists with respect to the set of combinators consisting of L alone? Is SLL the only fixed point combinator that can be constructed from S and L alone?

Example 2 of Applying the Kernel Method

Let us now focus on an even simpler application of the kernel method. The case concerns the set P consisting of U alone.

$$Uxy = y(xxy)$$

To give the relevant substitution easily, let us proceed as we did in Example 1 and consider

$$Uxz = z(xxz)$$

and

$$y \neq fy$$

and—as in the study of L and S—the empty set of expansions needed to complete a proof by contradiction at stage 1 of the kernel method. The substitution that consists of f for z, U for x, and UUf for y completes such a proof. Equivalently, from the viewpoint of automated theorem proving, the corresponding two clauses unify, and the given substitution is obtained from the unification of the two.

The kernel UUf, which is obtained by applying the given substitution and seeing how y is correspondingly instantiated, could be considered by stage 2 for possible expansion with U with the intention of constructing a fixed point combinator. However, since the constant f is already free, such consideration is not necessary. By an argument that is almost identical to that we just used to obtain the kernel UUf, we immediately see that

$$UUx = x(UUx),$$

which is just what is needed for stage 2. Therefore, UU is a fixed point combinator constructible from U, and its kernel is UUx.

Although we shall give other examples of this type later, nevertheless this case is very unusual, for seldom does one encounter the case in which Θx reduces to $x(\Theta x)$; the same equation

$$UUx = x(UUx),$$

which we can obtain almost by mere inspection of the equality and inequality that characterize the problem, establishes that both the weak and the strong fixed point property hold for *P* consisting of *U* alone.

If we continue as we did in the preceding example, we note that no other fixed point combinators and no other kernels appear to be constructible from *U* alone, which again suggests two open questions for research. In other words, if we were viewing the kernel method from the three-stage perspective, the second stage would yield no additional kernels, and the first and third stages would require no expansions.

Example 3 of Applying the Kernel Method

For our third application of the kernel method, let us fulfill our promise of revisiting *P* consisting of *B* and *W* alone—let us begin exploring the *B*-and-*W* mine with the intention of extracting its riches. As with any mine, some veins are far richer than others, some are barely worth working, and others run out almost immediately.

To start our exploration, we rely on four statements. But, in contrast to our earlier approach in which we chose the variables so that no two statements share a variable—a decision motivated by the desire to introduce the material gradually—here we simply write the statements in their natural form. Therefore, most variables are shared by all statements, which is fine since variables are relevant only to the statement in which they occur—equivalently, from the viewpoint of automated theorem proving, relevant only to the clause in which they occur. Since we are studying *B* and *W* in the context of the two-stage approach for searching for fixed point combinators, we have the following statements (or their equivalent in clauses) to consider.

(1) $x = x$
(2) $Bxyz = x(yz)$
(3) $Wxy = xyy$
(4) $y \neq fy$

Among others, the following two inequalities are deduced in stage 1.

(5) $yz \neq Bfyz$
(6) $yy \neq W(Bf)y$

The first of the two inequalities is obtained by expanding the input inequality (4) with *B*, and the second by expanding the result with *W*. The automated theorem-proving program first detects a contradiction between statements (5) and (2) and then, if allowed to proceed, detects a contradiction between statements (6) and (1). Although an analysis of either contradiction will produce the same information, we prefer to focus on the second because it is easier to see what the program has found.

An examination of the substitution (unification) on which the second proof by contradiction rests shows that a kernel, $W(Bf)(W(Bf))$, has been constructed. The substitution consists of replacing *y* in statement (6) and *x* in statement (1) by $W(Bf)$. Of course, if one were using the program in the manner we recommend, this analysis would be unnecessary, for the ANSWER literal would contain as its argument the kernel $W(Bf)(W(Bf))$. Actually, the argument of the ANSWER literal would be

kernel(a(a(W,a(B,f)),a(W,a(B,f))));

nevertheless, for convenience, we shall often omit the use of the functions *a* and *kernel*. If one wishes to

verify that stage 1 has indeed found a kernel—that the given Γ does reduce to $f\Gamma$—one simply applies to Γ the reductions in reverse order that correspond to the expansions that led to finding that kernel. In particular, the application, as reductions, of W followed by B suffices.

Since stage 1 has succeeded in finding a kernel, and since we recommend that stage 2 consider such kernels one at a time, the kernel method now attempts to complete the construction of a fixed point combinator. As it turns out, the program does not succeed, for the kernel $W(Bf)(W(Bf))$ cannot lead to the construction of a fixed point combinator from B and W alone. However, as we shall see in Section 4.5.5, this kernel is of use when the set P is extended to consist of B, W, and M; in fact, for this extended set of combinators, $W(Bf)(W(Bf))$ is a useful superkernel. We shall also give in Section 4.5.5 an intuitive explanation of why this kernel fails us in our quest for fixed point combinators constructible from B and W alone.

The program's effort even at this point has not been wasted, despite the inability to use $W(Bf(W(Bf)))$ in the context of the strong fixed point property. In particular, by finding the kernel $W(Bf)(W(Bf))$, the program has succeeded in showing that the weak fixed point property holds for the set P under study. By merely replacing f by x in this kernel, one has a y that obviously depends on x and that has the required property that

$$y = xy$$

for all combinators x.

If this visit to the B-and-W mine were our first exposure to what it contains—in particular, if we did not already know about Θ_1 through Θ_5—we would still be encouraged to note that the kernel method had succeeded in proving that the weak fixed point property holds for the set P consisting of B and W alone. After all, since the strong fixed point property implies the weak fixed point property, we could be close—metaphorically, since we have found silver, perhaps further exploration will lead to the discovery of gold in the B-and-W mine.

Indeed, if we continue along the path that yields

(5) $yz \neq Bfyz$,

in addition to the inequality (6) leading to $W(Bf)(W(Bf))$, we deduce the conclusion

$$xyz \neq B(Bf)xyzz,$$

which in turn leads to

$$yyz \neq W(B(Bf))yz$$

which contradicts statement (1). An analysis of this proof by contradiction yields a second kernel, a kernel reminiscent of, but not identical to, the kernel Γ_1. In particular, the analysis yields $\Omega\Omega z$ rather than $\Omega\Omega\Omega$ as the kernel, where $\Omega = W(B(Bf))$. (Strictly speaking, we call expressions such as $\Omega\Omega z$ *kernel schemata*; in Section 4.6, we shall extensively study the expression $\Omega\Omega z$.)

If the second stage of the kernel method attempts to expand $\Omega\Omega z$, it succeeds. Even further, if the program imposes the strategy of treating the generator of a kernel as a constant until expansions produce an expression containing a single occurrence of the generator—for this case, Ω—one of the expansion paths leads to the expression $WW\Omega$. An analysis of this expansion path shows that the variable z is instantiated to $\Omega = W(B(Bf))$, and we find ourselves in effect again studying the kernel Γ_1.

As we saw earlier, the use of Γ_1 by the second stage of the kernel method does indeed lead to the construction of a fixed point combinator—the combinator Θ_1. In other words, the second stage finds Θ_1 as a fixed point combinator constructible from *B* and *W* alone. The approach for finding Θ_1 in the second stage observes the strategy of considering $WW\Omega$ and replacing Ω by its definition, and then relying on the strategy of requiring all expansions to satisfy the 1's rule and applying them only to terms containing an occurrence of *f*. Summarizing, the two-stage kernel method succeeds in its search, and proves that the strong fixed point property holds for the set *P* consisting of *B* and *W* alone.

If the kernel method is viewed as a three-stage procedure, the program could use Γ_1 and find Γ_2 through Γ_5, each of which leads, as we know, to its own fixed point combinator. Alternatively, if used as a two-stage procedure, the first stage would also find—in addition to Γ_1—Γ_2 through Γ_5. Therefore, we have the choice of letting the first stage of the two-stage version of the kernel method find kernels, some of which may be superkernels, or testing each kernel as it is found to see whether it is in fact a superkernel, which means using the three-stage version. In the latter case, the superkernel could be used to find kernels, rather than finding them with the first stage; this approach appears to offer more efficiency, which explains our preference for the three-stage version of the method.

The first stage of either version would also construct various other superkernels, including $\Gamma_6 = W(B(B(Bf)))(W(B(B(Bf))))(W(B(B(Bf))))$. However, when compared to relying on the kernel method entirely, the approach discussed earlier for constructing additional superkernels from Γ_1—the approach of inserting the combinator *B* to the right of *W* in all occurrences of the generator of the preceding superkernel—is simpler, more direct, and apparently more efficient. As one might suspect, combinatory logic is sufficiently complex that such a mechanistic approach to generating superkernels from earlier superkernels seldom succeeds. In Section 4.7, we shall see that, in addition to $\Gamma_1 = \Omega\Omega\Omega$, we can profitably study other instances of the kernel schema $\Omega\Omega z$ and use those instances for constructing fixed point combinators different from Θ_1, Θ_2, and the like.

Comparing the Power of the Kernel Method to a Standard Approach

With regard to constructing the *B* and *W* fixed point combinators Θ_1 through Θ_5, we obtain very satisfying results from a comparison of the kernel method with that given in our earlier paper, where we used the straightforward (and one might say brute-force) method—equivalently, the standard approach in automated theorem proving or the approach in mathematics or logic.

With the theorem-proving program ITP, the straightforward method required the equivalent of 20 CPU hours on a Sun 3 workstation; the kernel method required 8 CPU minutes on that machine. Even more satisfying, with our new program OTTER applying the refined version of the kernel method, 2 CPU seconds suffice to prove that from *B* and *W* alone one can construct fixed point combinators. The straightforward method requires extracting the fixed point combinators from the proofs by contradiction—we shall give an example—but the kernel method requires virtually no such extraction. Finally, and most important, the brute-force method sheds little or no light on related constructions other than Θ_1 through Θ_5; in contrast, the kernel method finds many interesting kernels.

For example, one could have found the infinite set $\Gamma(BW)$ through the use of the kernel method. In addition, the discussion just completed shows that use of this systematic method finds the kernel schema

$W(B(Bf))(W(B(Bf)))z$. From this schema, we can produce an infinite class of kernels, each obtained by an appropriate instantiation of z. One rule for obtaining a member of this class is that of instantiating z by a term expressed purely in terms of some association of Ω's, where $\Omega = W(B(Bf))$. In Section 4.7, we explore some of these instantiations. In the promised book, we explore other instantiations of z that lead to the construction of fixed point combinators.

Although later we shall discuss in detail some of these instantiations and the corresponding fixed point combinators they lead to—instantiations such as the replacement of z by $\Omega\Omega$, by $(\Omega\Omega)\Omega$, and by $\Omega(\Omega\Omega)$—we can make certain observations to show how interesting this infinite class of kernels is. If one excludes the kernel $\Omega\Omega\Omega$ already studied, this class of kernels generates an infinite class of infinite sets of fixed point combinators—constructible from B and W alone—whose properties are quite different from those of the fixed point combinators obtained from $\Gamma(BW)$. For example, the one-to-one mapping of kernels to fixed point combinators is replaced with a many-to-one mapping. The ability to construct the fixed point combinators, corresponding to a kernel, by focusing exclusively on expansions satisfying the 1's rule appears to be preserved. However, we cannot say for certain that this is true, for there do exist paths from a kernel of the type under discussion that violate the 1's rule. Our experiments with $\Omega\Omega(\Omega\Omega\Omega)$ suggest that these additional paths, if followed, merely produce fixed point combinators that duplicate those that can be constructed by observing the 1's rule.

Similar to an earlier comment, we note that—in contrast to constructing fixed point combinators directly from kernels—if we instead generate kernels from a superkernel, we must then use global expansions that satisfy and global expansions that violate the 1's rule. On the other hand, by letting stage 1 find all kernels for the assigned study—such as that leading to the construction of Θ_1 through Θ_5—the required expansions all satisfy the 1's rule because the respective kernels are constructed right to left, thus avoiding the need to apply an expansion violating the 1's rule as is needed when generating Γ_2 through Γ_5 from Γ_1.

In short, with regard to power, scope, and efficiency, the kernel method dwarfs the straightforward or brute-force approach for constructing fixed point combinators from some given set P. In other words, the kernel method is far superior to the standard approach taken in automated theorem proving or in mathematics and logic. As far as we can ascertain, the kernel method is the only systematic method for constructing—when they exist—fixed point combinators, a property that makes this method indeed appealing.

As additional evidence of the usefulness and importance of the concept of kernel—before we continue our exploration of the B-and-W mine—we pause to discuss in Section 4.5.5 how one can use kernels that appear to have failed to fulfill their intended use. We shall follow this discussion with a study in Section 4.6 of how one can attempt to prove that the strong fixed point property fails to hold, when that is the case. Only then—after we cover these two topics—do we allow ourselves in Section 4.7 to again enter the B-and-W, searching for, and then extracting, its hidden riches.

4.5.5. Kernels That Mistakenly Appear to Be Useless

In this section, to avoid discarding as useless kernels that fail to be useful in the study in which they are found, we focus on using such kernels for other studies.

By a useless kernel, we mean a kernel that cannot be directly used—regardless of the choice of the set P of combinators—to construct a fixed point combinator. Of course, were it not for the fact that such abundance would essentially destroy the interest and beauty, we would prefer that each kernel found in stage 1 of the two-stage kernel method be used by stage 2 to complete the desired construction for the given set P under consideration. Fulfillment of such a preference is obviously an impossibility; there exist sets P of combinators that satisfy the weak fixed point property, but that fail to satisfy the strong fixed point property. For example, in Section 4.6 we prove Theorem 6 which establishes that the set P consisting of B and L alone fails to satisfy the strong fixed point property; that set does, however, satisfy the weak fixed point property, which can be proved by considering the kernel $Lx(Lx)$. Nevertheless, since we would not have it all our way if we could, we prefer not to waste any kernels that the first stage of the two-stage kernel method finds.

In particular, when presented with a kernel, we would like to find a set P of combinators for which the given kernel leads to the construction of a fixed point combinator. What we strongly suggest in this section is that no kernels are useless—given a set P of combinators and a kernel with respect to that set, we show how one can extend P to a set P' so that the given kernel can be used to construct a fixed point combinator for P'. Put another way, we show how one can take a set P, known to satisfy the weak fixed point property because of the existence of a kernel Γ, and extend P to a set P' such that P' satisfies the strong fixed point property.

Even further, the proof that P' fulfills the claim rests on the use of Γ to construct a fixed point combinator from P'. Finally, we show how P can be extended to P' in a nearly minimal way—in other words, we show how to embed a set satisfying the reducible weak fixed point property in a set that is not much larger and that satisfies the strong fixed point property. Of course, the nearly minimal property is the interesting property, for we could simply add S and K to achieve everything since S and K form a complete set of combinators. Such an extension is not very satisfying, especially because K is an eliminator and researchers often wish to avoid the use of eliminators.

One way to attempt to avoid wasting any kernels—or, for that matter, wasting any superkernels—is to pause and consider what must occur for a kernel to be successfully used for constructing fixed point combinators. At the same time, we can examine the differences between kernels and fixed point combinators both from a general viewpoint and from the viewpoint of the kernel method. A natural choice for a starting point is the consideration of the two simple examples discussed in the preceding section, that focusing on S and L, and that focusing on U alone.

Putting the Kernel UUx to Use

Since the second example is the simpler, we begin by seeing what we can learn from its consideration. Earlier, we found that the fixed point combinator UU can be constructed from U alone, and can be obtained at stage 2 from the corresponding kernel UUf found at stage 1. Stage 2 is not required to take any action because f is already free. Since the expression UU from which f is free contains no occurrences

of f, the kernel method is in fact successful in its search for a fixed point combinator when presented with the set P consisting of U alone. In other words, to construct the desired combinator, one takes the kernel UUf found in stage 1 of the kernel method, and simply discards f.

The simplicity of this application of the kernel method might be misleading; one might conclude that, since stage 2 took no actions, we cannot learn anything from this example. Such is not the case. Indeed, to see what we learn, let us be very precise and write the kernel as UUx rather than as UUf. Let us continue in this precise mode and note that $UUx = x(UUx)$, which we in effect learned at stage 1. Our choice to be precise—perhaps pedantic is a better word—immediately makes it easy to see an important difference between kernels and the fixed point combinators constructed from them, when such construction is possible. The kernel UUx explicitly depends on x in contrast to the fixed point combinator UU which is independent of all combinators not in the set P, which consists of U alone. The kernel UUx can serve as a y satisfying for all x there exists a y with

$$y = xy,$$

which is, of course, the equation for the weak fixed point property. In contrast, the fixed point combinator UU can serve as a y satisfying there exists a y such that for all x

$$yx = x(yx),$$

which is, of course, the equation for the strong fixed point property.

Summarizing, kernels are each a function of the particular value chosen for x, a function of the chosen combinator; fixed point combinators are each independent of the value chosen for x, independent of the choice of combinator. From the viewpoint of stage 2 of the two-stage kernel method, we can rephrase this remark and say that stage 2 constructs (if successful) a fixed point combinator from a kernel by constructing an expression that is independent of x from one that is dependent on x.

Obviously, for the kernel UUx, nothing is directly required of stage 2; one merely drops the variable x. Finding the kernel UUx establishes that the set P consisting of U alone satisfies the weak fixed point property. Noting that $UUx = x(UUx)$ and that UU is obviously independent of x establishes that P also satisfies the strong fixed point property. Therefore, to extend this set P to a set P' from which one can construct a fixed point combinator requires no action. In that regard, the next example is more interesting.

Putting the Kernel $Lx(Lx)$ to Use

If we now focus on the results of applying the two-stage kernel method to the set P of combinators consisting of S and L alone, we recall that, in stage 1, the kernel $Lf(Lf)$ is found. Then we note that stage 2 applies S as an expansion to this kernel to construct the fixed point combinator SLL. If we consider this example in the terms used to analyze the preceding example, we see that stage 2 obtains an expression SLL which is independent of x from an expression $Lx(Lx)$ which is dependent on x. To do so, stage 2 must remove the duplicate occurrences of x in the kernel, and must also free x from what is intended to be a fixed point combinator.

To remove duplicate occurrences of some term by expansion, one must apply a combinator that is itself a replicator. In the case under discussion, the combinator S with

$$Sxyz = xz(yz)$$

is used. (No such action is required in the preceding example because no undesirable duplication exists in the kernel UUx.) To free a term, as stage 2 frees x when it considers $Lx(Lx)$, one must expand with a combinator such that the last variable on the left side of its equation is a variable that is matched with the term to be freed. Again, the combinator S is used. Success is obtained because its last variable, z, is matched with x in $Lx(Lx)$.

We thus learn that, for examples like $Lf(Lf)$—which we have actually studied at this point as $Lx(Lx)$—a replicator must be used. In addition, we learn that the replicator or replicators must free x, or some other combinator or combinators must be used. Finally, from the viewpoint of stage 2 and its use of kernels, we recognize that what results is a fixed point combinator if the expression from which x is freed is independent of x—contains no occurrences of x.

Before returning to our main interest, exploration of the B-and-W mine, we can invert some of the preceding discussion and extract from this second example a little more. In particular, we can imagine that we have been presented with the kernel $Lf(Lf)$. We can also imagine that we have been requested to use this kernel to construct a fixed point combinator, and to do so in a way that supports the suspicion that no kernel need be wasted or discarded. Even if we had not experienced the success of stage 2 and did not know much about the kernel method in general, we might still understand enough about the desired construction to realize that one or more replicators must be used to remove the duplicate occurrences of f in $Lf(Lf)$. In addition, if we knew of the strong fixed point property, we would certainly know that we must find a way to free f. Therefore, our response to the hypothetical request might be to look at a list of replicators to see whether any looked promising. If we happened upon S, we might indeed immediately see that this combinator could be used to remove unwanted duplication, possibly to free f, and, therefore, to construct SLL from $Lf(Lf)$.

On the other hand, if we first happened on the combinator M with

$$Mx = xx,$$

we might instead—noting that M is a replicator—apply M as an expansion and obtain $M(Lf)$. If we took this action, we would be encouraged since the result contains no duplicate occurrences of f. We might then search for a way to conveniently free f. Our experience with B, gleaned from the study of the fixed point combinators Θ_1 through Θ_5, might immediately have led us to consider using B as an expansion to free f. By doing so, we would obtain BML as a fixed point combinator, constructible from B, M, and L, whose corresponding kernel is indeed $Lf(Lf)$. In other words, we would have succeeded in fulfilling the hypothetical request of showing how one can avoid wasting the discovery of the kernel $Lf(Lf)$—avoid in a (so to speak) minimal way wasting this kernel—and we would be better prepared for returning to the discussion of B and W and making use of the next kernel we consider.

In fulfilling the request, we would have shown how to extend the set P consisting of L alone—which we can prove satisfies the weak fixed point property but fails (see Theorem 5 of Section 4.6) to satisfy the strong fixed point property—to either of two sets that each do satisfy the strong fixed point property. For the first extension of P, we formed the set P' by adjoining S only; for the second set, we formed the set P'' by adjoining M and B to P. The set P' shares with the set P the property that it contains only regular combinators; the set P'' does not have this property because M is one of its elements. Both extension sets do not rely on the use of eliminators, such as K—a feature that often pleases

researchers. Finally, the kernel $Lx(Lx)$ maintains its property of being a superkernel, for both P' and P''.

Putting the Kernel $W(Bx)(W(Bx))$ to Use

Let us now return to the first kernel that stage 1 of the two-stage kernel method finds when the kernel method is applied to the set P of combinators consisting of B and W alone. That kernel, $W(Bf)(W(Bf))$, did not prove useful for constructing fixed point combinators from P; in Section 4.5.5, we shall show intuitively why this kernel fails in that regard. What we show here is how this kernel—which inspection shows to be a superkernel with respect to the set P—can be used, by choosing the appropriate context, for the construction of fixed point combinators. This kernel merits study because it precedes, in a sense we now discuss, all elements in the sequence of superkernels that begins with Γ_1, the superkernel that led to finding the set $\Gamma(BW)$ of superkernels and the corresponding infinite set $\Theta(BW)$ of fixed point combinators.

In particular, we noted when studying $\Gamma(BW)$ that one obtains from Γ_1 the next superkernel Γ_6 by inserting a B to the right of W and then fully right associating the result. Obviously, one can obtain Γ_1 from $W(Bx)(W(Bx))$ by the same type of move. In other words, the superkernel $W(Bx)(W(Bx))$, rather than Γ_1, could be used to generate $\Gamma(BW)$. In addition, no smaller superkernel, constructible from B and W alone, exists. Therefore, one might say that $W(Bx)(W(Bx))$, rather than Γ_1, is the actual origin of the infinite set of fixed point combinators we studied earlier—the set that can be traced to $\Gamma(BW)$—even though this superkernel has no fixed point combinator as its correspondent.

In view of the position that $W(Bx)(W(Bx))$ holds, it—more than many other kernels—deserves serious attention. Therefore let us show how this kernel can be used for establishing the presence of the strong fixed point property for other sets P of combinators—even though it fails in this regard for the set consisting of B and W alone—and further suggest in turn that all kernels found by the two-stage kernel method are useful (with minimal effort) for constructing fixed point combinators, even if other combinators must be adjoined. Of course, by doing so, we show that superkernels found by the three-stage kernel method are also always of use in some context. For our attack, we can imitate the actions we could have taken to cope with the hypothetical request to make use of the kernel $Lx(Lx)$.

We can begin our attack by searching for a combinator to remove the duplicate occurrences of x in $W(Bx)(W(Bx))$. Among the various combinators mentioned in this report to this point, one exists that is clearly promising if used to expand this superkernel. That combinator,

$Mx = xx$,

obviously unifies with the superkernel—its right side has the superkernel as an instance, in fact. Access to an automated theorem-proving program certainly is not needed for observations of this type to come immediately to mind. Indeed, one of the pleasing properties of the kernel method is that both stages are often very easy to implement by hand, and, even better, one can often immediately see what actions are required to achieve one's objective.

For example, by hand, one can instantly produce from the superkernel the following sequence of expressions, each obtained by applying an expansion that satisfies the 1's rule.

$W(Bx)(W(Bx))$,

$M(W(Bx))$,
$BMW(Bx)$,
$B(BMW)Bx$

For this sequence, one has simply looked for appropriate combinators to apply as expansions, and then one has applied them. In particular, after choosing and then using the replicator M to remove the duplicate occurrences of x, one turns to choosing and using some combinator—in the case under discussion, the combinator B—that can eventually free x.

Simply stated, that is the key—choose one or more replicators to use as expansions to remove duplicate occurrences of the undesired symbol(s), and then choose combinators to use as expansions to free the undesirable symbol(s). Both choices are dictated by the pattern of the symbols in the expression under consideration. If one has had experience with various combinators, the replicator M is a natural choice since the superkernel $W(Bx)(W(Bx))$ is an instance of xx, the right side of the equation for M. Since kernels often take the form $\Omega\Omega$, $\Omega\Omega\Omega$, and the like, where Ω is a generator, one can usually use the replicator M profitably when the goal is to extend a set P to a set P' of combinators for which the given kernel can be used for constructing fixed point combinators. Having chosen and applied M to obtain the fully right-associated expression $M(W(Bx))$, a natural choice to use as an expansion to free x is B because repeated use of B as an expansion produces ever more left-associated expressions from right-associated expressions. Since the generators of kernels frequently are right associated, and since expansion of them with a replicator frequently produces a right-associated expression, one often finds the combinator B very useful.

Restricting the application of expansions by requiring that each application satisfy the 1's rule meshes well with the kernel method and the desire to avoid duplicating results. For example, if we relax this restriction, we can obtain the fixed point combinator $BM(BWB)$ from the superkernel $W(Bx)(W(Bx))$. However, as we shall shortly see, the same result can be obtained from a kernel that in turn is obtained by the three-stage kernel method from the given superkernel. As noted in Section 4.5.1.3 with regard to the two-stage method, and noted in Section 4.5.2 with regard to the three-stage method, in general we recommend using the strategy of observing the 1's rule when applying expansions to obtain the various fixed point combinators for their (so to speak) natural corresponding kernel.

Summarizing, the superkernel $W(Bx)(W(Bx))$ serves well for proving that the strong fixed point property holds for the set of combinators consisting of W, B, and M. Although the discovery of this superkernel was made while studying the set P consisting of B and W alone, and although we soon found that this superkernel was of no use in that context, we were nevertheless able to achieve the objective of using the superkernel for constructing fixed point combinators by merely adjoining the single combinator M. Unfortunately, the addition of M results in a set P' that does not consist of regular combinators only. Since P already satisfies the strong fixed point property, our main gain by extending P to P' is the demonstration that we can make the superkernel $W(Bx)(W(Bx))$ useful for the construction of fixed point combinators. The property of being a superkernel is not lost by extending P in this manner.

If we wish to use the superkernel under discussion for constructing fixed point combinators, but also wish to extend P in a way that preserves the property of containing regular combinators only, then the replicator M cannot be considered. Instead, another combinator immediately comes to mind, the

combinator S with

$Sxyz = xz(yz)$.

The choice of the replicator S is almost as natural as the choice of M, for the right side of S obviously matches the symbol pattern of $W(Bx)(W(Bx))$. Since this pattern occurs rather frequently in kernels, one can often profitably use S to find a set P' for which a given kernel leads to the construction of a fixed point combinator.

The expansion with S yields $SWW(Bx)$, which says that we need only find a combinator to free x, since the unwanted duplicate terms have been removed. Again the combinator B is an obvious choice. Its application yields $B(SWW)Bx$, which means we have constructed the fixed point combinator $B(SWW)B$. The addition of S to P yields a set of combinators that contains only regular combinators and a set for which $W(Bx)(W(Bx))$ is still a superkernel, but with the added property that its use leads to the desired construction. The expansions that stage 2 could use to find $B(SWW)B$ from $W(Bx)(W(Bx))$ all satisfy the 1's rule.

Proceeding in the fashion just illustrated gives one a bonus. Specifically, if one wishes to present a proof in the algebraic style that some given Θ constructed by the two-stage kernel method is a fixed point combinator, one can proceed in the following manner. First, one considers a sequence of reductions that corresponds to the expansions used by stage 2 to obtain Θ from a kernel Γ, but in reverse order. Then, second, one applies this reduction sequence to $x(\Theta x)$ to obtain $x\Gamma$, and also applies the sequence to Θx to obtain Γ. Third and finally, one considers the sequence of reductions that corresponds to the expansions used by stage 1 to find Γ, but in reverse order, and applies the sequence to Γ by itself to obtain $x\Gamma$. And here we have a justification for having stage 1 continue to expand, in its search for a contradiction, to the point at which the axiom of reflexivity

$x = x$

comes into play. In particular, if the program ceases its stage 1 search by finding a kernel, but finding it before reflexivity comes into play, then one must look for the expansions that would have been used, if one wishes to have the needed reduction sequence that reduces Γ to $x\Gamma$. If one follows the given three steps, one then knows that $\Theta x = \Gamma = x\Gamma$, and that $x(\Theta x) = x\Gamma$, so $\Theta x = x(\Theta x)$, which completes the sought-after proof.

Since $W(Bx)(W(Bx))$ is a superkernel for the set P' consisting of B, W, and M and for the set P'' consisting of B, W, and S, we can see what kernels stage 1 of the three-stage kernel method can find from this superkernel. Recalling that the rule we use for obtaining kernels from superkernels is that of applying global expansions until x is free in all occurrences of the generator of the superkernel, we find that $BWBx(BWBx)$ is the only new kernel that is found, for P' or for P''.

If we focus on this new kernel and restrict our study to P' to see what (if any) fixed point combinators we can construct, we obtain the following expressions by using expansions satisfying the 1's rule.

$BWBx(BWBx)$
$M(BWBx)$
$BM(BWB)x$

We immediately see that the expression $BM(BWB)$ is a fixed point combinator constructible from the set

P' consisting of B, W, and M. Consistent with what we learned earlier, we recognize that continued expansion of either $B(BMW)B$ or $BM(BWB)$ produces no additional fixed point combinators of interest. Such actions merely produce expressions reducible to—and therefore equal to—one of the two fixed point combinators we have constructed from B, W, and M.

Just as one could have found each of Θ_1 through Θ_5 by focusing exclusively on the superkernel Γ_1, one could also find both $B(BMW)B$ and $BM(BWB)$ by expanding the superkernel $W(Bx)(W(Bx))$. Of course, to do so, one must use expansions that violate the 1's rule. As we noted earlier, we prefer to observe the strategy that requires all expansions to satisfy the 1's rule—when one is seeking to expand a kernel to a fixed point combinator—to avoid duplicating the results one obtains from other kernels. For example, the combinator $BM(BWB)$ whose corresponding kernel is $BWBx(BWBx)$ can be constructed from the superkernel $W(Bx)(W(Bx))$ if one uses expansions that violate the 1's rule. Therefore, for reasons of simplicity and efficiency, we prefer adhering to the restriction of requiring all expansions to satisfy the 1's rule. Still, regardless of the choice of the set P, again we see that more than one way exists to search for fixed point combinators starting with a given superkernel.

If we shift our attention from P' to P''—in other words, focus on B, W, and S—we can still use the new kernel $BWBx(BWBx)$ profitably. In particular, we can construct the fixed point combinator $S(BWB)(BWB)$ by following our usual approach. We are immediately confronted by two fixed point combinators, $BM(BWB)$ and $S(BWB)(BWB)$, that are provably unequal—they are each irreducible and obviously unalike—in the absence of extensionality. But the two combinators have a common kernel, $BWBx(BWBx)$. This phenomenon is different from Θ_1 through Θ_5, for each of them corresponds to a different kernel, Γ_1 through Γ_5, respectively. The similarity rests with the fact that, as is true for Θ_1 through Θ_5, both $S(BWB)(BWB)$ and $BM(BWB)$ share the same superkernel.

The set P'' behaves with respect to the kernels $BWBx(BWBx)$ and $W(Bx)(W(Bx))$ as the set P' does—the replicator that is adjoined to P to obtain P'', for example, does not appear in either of the kernels that are used to construct fixed point combinators. This remark might lead one to ask about finding a kernel for the subset of P'' that consists of B and S alone, or lead one to ask about the presence or absence of the weak fixed point property for that subset. We lean in the direction that says that this subset does not satisfy the weak fixed point property, and therefore, no kernel can be found with respect to this subset; but, we have no proof of either claim. For the subset consisting of S and W alone, we can prove that the weak fixed point property is absent and, therefore, no kernel exists with respect to this subset. The proof rests on the realization that, for a subset to satisfy the weak fixed point property, the subset must contain an isolator; neither S nor W is an isolator (see the remark that follows Theorem 4 of Section 4.6).

If we return to the consideration of P', but in the context of the preceding discussion, we find the situation far cloudier. Although we shall eventually discuss the possible presence of the strong fixed point property for the subset consisting of B and M alone, if we wish to avoid adding any complications at this point, we can only make the following observations. The set consisting of B and M alone does satisfy the weak fixed point property, which we can prove by noting that $M(BxM)$ is a kernel. (Incidentally, this kernel is discussed by Smullyan on page 135 in his book *To Mock a Mockingbird* [Smullyan84]; however, in private conversation with him, we were informed that he had no idea that a method like the kernel method existed.) The kernel $M(BxM)$ cannot be expanded to a fixed point combinator when B and M are

the only admissible combinators to use. Its inadequacy rests with the fact that reductions, if applied to Θx and its descendants, with B and with M each must produce an expression with the property that any subexpression containing x must end in x.

As for the other two-element subset of P', that consisting of W and M, we can prove that this subset fails to satisfy the strong fixed point property. The proof rests on the consideration of a possible reduction of some hypothetical Θ that is assumed to be a fixed point combinator for this subset. If one appends x to such a Θ and applies reductions with W and M, one can show that all expressions that are obtained are of the form CD where D must consist only of occurrences of x. From this conclusion, by using the Church-Rosser property, we can show that one cannot reduce Θx to $x\Delta$ for an appropriate Δ; we give the formal details of this proof in Section 5.3.6. We also can prove that the subset of P' consisting of W and M in fact fails to satisfy the weak fixed point property (see Corollary 2 of Section 5.3.6.1).

The Uselessness of the Kernel $W(Bx)(W(Bx))$ for B and W Alone

Let us close our discussion of the superkernel $W(Bx)(W(Bx))$ by focusing on why it is useless for constructing fixed point combinators from B and W alone. If one restricts expansions to those applications of B and W satisfying the 1's rule, then, as an example of what can occur, along one path one obtains

$W(Bx)(W(Bx))$
$B(W(Bx))W(Bx)$
$B(B(W(Bx))W)Bx$,

which admits no additional expansions. For a sample of what can happen if we permit expansions violating the 1's rule, one can obtain

$W(Bx)(W(Bx))$
$W(Bx)(BWBx)$
$BWBx(BWBx)$
$B(BWBx)(BWB)x$
$BB(BWB)x(BWB)x$
$B(BB(BWB)x)(BW)Bx$
$B(B(BB(BWB)x))BWBx$

and continue until no additional expansions apply.

For these two examples of paths that one might pursue, as well as the others that can be considered, inspection shows that no way exists for the two occurrences of x to be reduced to one. However, if one is to succeed in constructing a fixed point combinator, one must show that an expression can be obtained containing a single occurrence of x. Therefore, we have the suggestion, at least intuitively, of a proof that the superkernel $W(Bx)(W(Bx))$ cannot be expanded to a fixed point combinator constructible from B and W alone.

The Uselessness of the Kernel $Lx(Lx)$ for B and L alone

Having given the type of argument one might use to show that no fixed point combinator can be constructed from a specific given kernel, we can attempt to apply the argument to the study of the set P consisting of B and L alone. Smullyan asked us to consider this set, for he did not know if it had

sufficient power to construct a fixed point combinator. Earlier, we saw that $Lx(Lx)$ is a superkernel with respect to the set of combinators consisting of L alone; therefore it is a superkernel with respect to the set P consisting of B and L since B does not occur in the superkernel. The preceding argument, if applied to possible expansions with L and B, would yield

$Lx(Lx)$
$B(Lx)Lx$
$BBLxLx$,

suggesting that no fixed point combinator can be constructed from P, starting with the given superkernel. This observation strongly suggests that no fixed point combinators can be constructed from P in any way—even if one does not focus on the given superkernel—since $Lx(Lx)$ yields no additional kernels.

Of course, we would prefer to say *proves* rather than *suggests*, but we have not yet succeeded in proving that, for every fixed point combinator, a kernel corresponding to that combinator must exist. A proof of that theorem, which would establish the completeness of the kernel method, is the object of one of the open questions we pose for research in Section 5.4.1. In addition, there exist other superkernels, for example, $L(Bx)(L(Bx)(L(Bx))$ and $L(B(Bx))(L(B(Bx)))(L(B(Bx)))(L(B(Bx)))$ and the like. Indeed, the expression $L(Bx)(L(Bx))z$ is a kernel schema.

If one were attempting to show that the consideration of B and L alone cannot suffice to construct a fixed point combinator, and if one attempted to employ an argument similar to that used regarding $W(Bx)(W(Bx))$, then the obstacle of these additional kernels would be encountered. Instead, we can approach—and often succeed in answering—questions focusing on establishing the lack of fixed point combinators constructible from certain given sets of combinators in another way, which is the topic of the next section, Section 4.6. Before we turn to that topic, however, let us see what we have learned from the material in this section.

Twists and Turns for Kernels That Mistakenly Appear to Be Useless

The following material directly and indirectly focuses on a number of questions that include those with incomplete answers and those on which we can shed little light. Common to the questions are various observations gleaned from comparisons concerning combinators that behave the same, concerning kernels that are closely related, and concerning reducibility versus equality. Since following this particular trail through our studies of combinatory logic requires one occasionally to double back, cross narrow but raging streams, navigate hairpin turns, and leap deep crevasses, one might choose to avoid this trail entirely and instead turn to the next section, Section 4.6.

The Beginning of the Trail

When one is presented with a kernel, one immediately knows that any set containing the combinators occurring in that kernel satisfies the weak fixed point property. For example, if given the kernel $Lf(Lf)$, one merely replaces f by x to obtain

$Lx(Lx) = x(Lx(Lx))$,

which is an instance of the equation

$y = xy$

for the weak fixed point property. Since $Lx(Lx)$ can serve for y in the preceding equation, we know that any set containing L must satisfy the weak fixed point property.

What one might not immediately see is that our use of the kernel $Lf(Lf)$ is not restricted to sets containing L. In particular, we noted in Section 3.2 that one could speak informally and say that $BWB = L$, or speak carefully and say that BWB is in the same class as L. Depending on the context, either mode of speech is allowed because BWB behaves as L does,

$BWBxy = x(yy)$.

Therefore,

$BWBx(BWBx) = x(BWBx(BWBx))$.

Especially if one has never seen the expression $\Gamma = BWBx(BWBx)$ before but has had some experience with kernels, one might instantly consider the possibility that a stronger condition exists than $\Gamma = x\Gamma$—perhaps Γ reduces to $x\Gamma$. If so, and if no reductions are required after x isolates, then Γ is in fact a simple kernel. Such is the case. So, we have indeed found another use for the kernel $Lf(Lf)$; in particular, we have found an alternative proof establishing that any set containing B and W also must satisfy the weak fixed point property.

A Rapid Stream

If the preceding discussion causes one to spontaneously conclude that any combinator in L's class behaves exactly as BWB does, then a small error is made. For a glimpse of what can go wrong, we consider the combination QM with

$Qxyz = y(xz)$

and

$Mx = xx$.

Although one can quickly see that QM is an L, the attempt to reduce $QMx(QMx)$ to $x(QMx(QMx))$ meets with one small difficulty. When one applies a reduction with Q, one obtains $x(M(QMx))$ with x isolated, and one is therefore tempted to cease applying reductions.

If one resists this temptation to stop and instead applies M as a reduction even though the application does not satisfy the 1's rule, then one shows that $QMx(QMx)$ reduces to $x(QMx(QMx))$. At that point, one has clearly proved that any set containing Q and M must satisfy the weak fixed point property whether L is present or not. Of course, in the obvious sense, L is present, at least the L that is QM. However, $Lx(Lx)$ reduces to $x(Lx(Lx))$ without any reductions being used after x isolates, but this is not true when L is replaced with QM.

Since, to put it mildly, we would find it awkward to have $Lx(Lx)$ be a kernel but $QMx(QMx)$ not be one, the obvious solution—had we not already presented it—is to extend the definition of kernel to include both expressions. In particular, we see directly why we consider kernels of various types—why, for example, we also study semisimple kernels. Indeed, $QMx(QMx)$ is a semisimple kernel, for the proof that it is a kernel requires a reduction that satisfies the pseudo 1's rule but does not satisfy the (strict) 1's

rule.

When one discovers the fixed point combinator $S(QM)(QM)$, additional pressure exists for the consideration of kernels that are not simple. Indeed, if one calls this combinator Θ and writes out the type of proof we prefer for showing that $\Theta x = x(\Theta x)$, one finds that the expression that acts like the kernel that naturally corresponds to Θ is in fact $QMx(QMx)$. Therefore, if one extends the kernel method to the consideration of kernels like $QMx(QMx)$, the extended method can be used to prove that the set P consisting of Q, M, and S satisfies the strong fixed point property.

Occurrences like the one under discussion demonstrate the complexity of combinatory logic, and of mathematics and logic in general. So-called puzzles that focus on how one should define a concept that is apparently understood—or understood intuitively—are part of the intrigue and charm of these fields.

Doubling Back on the Trail

As for the other branch concerning the possible temptation—the branch that assumes that one succumbs to rather than resists the temptation to continue to reduce $x(M(QMx))$—we can profitably explore it. In particular, one might conjecture that the beginning of the reduction sequence is missing and, therefore, that $M(QMx)$ is actually the kernel. The conjecture is a good one, for $M(QMx)$ does reduce to $x(M(QMx))$ and does not require any reductions after x isolates, which means that $M(QMx)$ is a simple kernel. Consideration by the kernel method of this kernel leads to the construction of the fixed point combinator $BM(QM)$.

In other words, if one succumbs to the temptation to avoid reductions after x isolates, one backtracks and finds a kernel that proves that the set P consisting of the combinators Q, M, and B satisfies the strong fixed point property. In contrast, if one resists the temptation to avoid using reductions after x isolates, then one finds that an extension of the kernel method leads to proving that the strong fixed point property holds for the set consisting of Q, M, and S—the combinator B is replaced by the combinator S, if one shifts from one kernel to a second to which the first kernel reduces.

A Crevasse to Cross

If we carefully select from the various discoveries we have made, we find that the spectrum continues to widen. For the first discovery to consider, the fixed point combinators Θ_1 through Θ_5 (which are pairwise unequal in the absence of extensionality) each reduce to a distinct kernel, all of which in turn reduce to a common superkernel Γ_1 with respect to the set consisting of B and W alone.

For the second—the consideration of which already leads to sharp differences from the first—the fixed point combinators $S(BWB)(BWB)$ and $BM(BWB)$ (which are unequal) both reduce to a common kernel $BWBx(BWBx)$, which reduces to $W(Bx)(W(Bx))$, a superkernel for both P' and P'' with P' consisting of B, W, and M and with P'' consisting of B, W, and S.

For the third—where we find further differences—the unequal fixed point combinators $S(QM)(QM)$ and $BM(QM)$ reduce, respectively, to distinct kernels; but the kernel $M(QMx)$ corresponding to the second fixed point combinator reduces to the kernel $QMx(QMx)$ corresponding to the first. Furthermore, $M(QMx)$ is a superkernel for the set consisting of Q and M; but—with the definition of kernel in its fullest sense—so is $QMx(QMx)$. After all, neither $M(QMx)$ nor $QMx(QMx)$ admits a reduction other than the obvious

one that satisfies the 1's rule. Taking the preceding observations into account, we see that the first of the two is a simple superkernel, and the second is a semisimple superkernel, both with respect to the set P of combinators consisting of Q and M alone. To add to the complexity, each of BWB and QM is in the same class as L is.

For a second set of differences and similarities, we can begin by noting that $Lx(Lx)$, which is a superkernel with respect to the set of combinators that occur in it, reduces to $f(Lx(Lx))$ without requiring reductions after x isolates. In contrast, if we consider $Lx(Lx)$ and replace L with BWB—a decision motivated by the fact that BWB is an L—we see that $BWBx(BWBx)$, although it reduces to $x(BWBx(BWBx))$ without requiring any reduction after x isolates, is not a superkernel with respect to the set of combinators that occur in it. Then, with the same motivation, if we consider $QMx(QMx)$, we find that the sequence of reductions that shows it to be a kernel requires a reduction after x isolates, which is unexpected since QM is an L and $Lx(Lx)$ is therefore obviously closely related to $QMx(QMx)$. Finally, with a similar motivation, we see that $M(QMx)$ is a kernel, and one that can be proved to be a kernel without requiring reductions after x isolates. But, if we note that QM is an L, we might expect $M(Lx)$ to be a kernel; but $M(Lx)$ is not.

We even have a third set of comparisons. The set of combinators that contains only L satisfies the weak fixed point property, but fails to satisfy the strong fixed point property (see Theorem 5 of Section 4.6). Therefore, the presence of L in some set of combinators does not guarantee that the set satisfies the strong fixed point property. In contrast, even though BWB is an L and QM is an L—and one might, therefore, expect that the presence of either pair in a set brings to that set no more power than L does—the presence in a set of combinators either of B and W or of Q and M is sufficient for that set to satisfy the strong fixed point property and, of course, the weak fixed point property.

Of course, we know that the presence of B and W is sufficient, for we have studied that pair and found it to be strikingly productive for constructing fixed point combinators. To prove that the presence of Q and M in a set P of combinators is sufficient to establish that P satisfies the strong fixed point property, one need only consider the combinator $Q(QM)M$. Its natural kernel correspondent is $M(QMx)$, which happens to reduce to the kernel $QMx(QMx)$, and also happens to be the natural kernel correspondent for the fixed point combinator $BM(QM)$.

Of course, as we discused earlier in this section, the superkernel equal to the square of $W(Bx)$ is not useful for such constructions. For constructing fixed point combinators from B and W alone, one must instead move to the next superkernel in the chain, the superkernel equal to the cube of $W(B(Bx))$. As for the set consisting of Q and M alone, one can find the kernel $M(QMf)$ in stage 1 of the two-stage kernel method by applying one expansion with Q that satisfies the 1's rule to the inequality

$$y \neq fy,$$

to obtain

$$xz \neq Qxfz,$$

which contradicts the equation for M

$$Mx = xx.$$

Stage 2 can then use this kernel to construct $Q(QM)M$ by applying one expansion with Q that satisfies the

1's rule. Therefore, although the presence of L in a set of combinators does not guarantee that the set satisfies the strong fixed point property, the presence of either the pair B and W or the pair Q and M does.

The differences in the behavior of L, B and W with BWB an L, and Q and M with QM an L—and such differences in general—may be completely explained in terms of the kernels that one can find with the kernel method. Obviously, we would be more than pleased if this were the case, for it would be one more plaudit for the kernel method. For example, if the two-stage or three-stage version of the kernel method considers L alone, the only superkernel it finds is $Lf(Lf)$. If either version considers Q and M alone, then the method finds the superkernel $M(QMf)$ among others. Although $M(QMf)$ is a kernel, MLf is not, even though QM is an L.

If either version of the kernel method considers B and W alone, the first superkernel it finds is $W(Bf)(W(Bf))$, from which it finds the kernel $BWBf(BWBf)$. In contrast to $M(QMf)$, a replacement of BWB by L yields a kernel. However, the superkernel $W(Bf)(W(Bf))$ is not useful for establishing that the presence of B and W guarantees the presence of the strong fixed point property. As noted, one must move to the next superkernel in the chain, the superkernel equal to the cube of $W(B(Bf))$.

Hairpin Turn

We can take a slightly different approach to a study of the differences between L, B and W with BWB an L, and Q and M with QM an L, which will naturally lead to the next topic of reducibility versus equality. Since $Q(QM)M$ is a fixed point combinator, we know that any set of combinators containing Q and M must satisfy the strong fixed point property. We also know that the strong fixed point property implies the weak fixed point property. Given these two facts, one can immediately ask the following question. How can we use the knowledge of a specific y that satisfies the statement, there exists a y such that for all x

$$yx = x(yx),$$

to obtain a (different) y that satisfies the statement, for all x there exists a y with

$$y = xy?$$

The path we recommend is to append a variable to the fixed point combinator under study, and reduce the result in search of a kernel; of course, since in some cases focusing on simple kernels is not sufficient, the broader definition of kernel must be used, that which allows reductions after x isolates. However, in the case under discussion, if one reduces $Q(QM)Mx$, one obtains the kernel $M(QM)x$, and no reductions are required after x isolates—the kernel that naturally corresponds to $Q(QM)M$ is simple.

Speaking generally but informally, we can say from a strong y, one attempts to obtain its corresponding reducible weak y—not the same y—by finding the kernel that corresponds to the fixed point combinator, which is the strong y. In other words, one can put into the strong/weak framework our comments about a fixed point combinator and its corresponding kernel; the kernel for a given fixed point combinator is the reducible weak y that corresponds to a given strong y.

Of course, we do not know whether our conjecture asserting that the kernel method is complete in this regard is true—we have not yet proved that to every fixed point combinator there corresponds a kernel. This conjecture, put into the context of a weak correspondent of a strong y, asserts that to every

strong y there corresponds a reducible weak y—of course, not the same y. As already noted, we do know that we must rely on the broader definition of kernel, which one can see from studying the fixed point combinator $S(QM)(QM)$ whose kernel is $QMx(QMx)$. To reduce that kernel Γ to $x\Gamma$ requires a reduction after x isolates. In contrast, the reducible weak y that corresponds to the strong y SLL is $Lx(Lx)$, which requires no reductions after x isolates to prove that $Lx(Lx)$ is a kernel. Nor does the reducible weak correspondent of $S(BWB)(BWB)$, $BWBx(BWBx)$, require any reductions after x isolates.

The completion of this part of the picture, and the connection to the preceding remarks about which L's guarantee the presence of the strong fixed point property, rests with the different behavior of QM and BWB, each of which is an L. In particular, a study of Q and M alone by the kernel method leads to the construction of $Q(QM)M$ whose reducible weak correspondent is $M(QMx)$, which reduces to $QMx(QMx)$, and which one can informally write as $Lx(Lx)$. In contrast, a study by the kernel method of B and W alone first leads to the superkernel $W(Bx)(W(Bx))$, from which one obtains $BWBx(BWBx)$, which also can be informally written as $Lf(Lf)$. But $BWBx(BWBx)$ is not the reducible weak correspondent of any fixed point combinator expressed purely in terms of B and W.

In fact, even the approach that produces $QMx(QMx)$ from $M(QMx)$ fails; in other words, one cannot even obtain the kernel $BWBx(BWBx)$ by reducing the weak correspondent of some fixed point combinator expressed purely in terms of B and W. Indeed, if one wishes to focus on a reducible weak correspondent of some strong y with respect to B and W alone, one must consider an expression such as $\Gamma_1 = \Omega\Omega\Omega$, where $\Omega = W(B(Bx))$. But such expressions do not reduce to any expression that can be informally written in terms of L.

Finally, the kernel $Lx(Lx)$ is not the weak correspondent of any fixed point combinator expressed purely in terms of L—the set of combinators consisting of L alone does not satisfy the strong fixed point property. One might say that some L's are more powerful than other L's, just as some groups, in the mathematical sense, have far more interesting properties than others. Other than this last tiny observation, precisely what we should make of these similarities and differences is far from clear at this time. So the picture continues to alternately become clearer and cloudier. Perhaps one key to the mystery rests with the study of reducibility versus equality—the former obviously implies the latter, but not conversely.

A Switchback

From the perspective of equality and reducibility, the knowledge that SLL is a fixed point combinator—that

$$SLLx = x(SLLx)$$

for all combinators x—permits one to immediately deduce that both $S(BWB)(BWB)$ and $S(QM)(QM)$ are fixed point combinators. In particular, in the style of proof we prefer, one proves that SLL is a fixed point combinator by reducing both $SLLx$ and $x(SLLx)$ with S to obtain, respectively, $Lx(Lx)$ and $x(Lx(Lx))$. Then one takes advantage of the knowledge that the first of the two expressions is a kernel and reduces it with L to obtain $x(Lx(Lx))$. Since both $SLLx$ and $x(SLLx)$ reduce to a common expression, they must be equal, and the proof is complete.

Next, if we focus on $S(BWB)(BWB)$, we can take the preceding proof and produce a second proof by merely applying the combination BWB where we had applied L. After all, BWB is an L, so it behaves as L

does. However, to make such a minute transformation of the first proof to the second, we must rely on our knowledge that *BWB* is in fact an *L*. On the other hand, if we did not know this fact, then we would simply pursue our usual course in such proofs, that of applying as reductions *B*, *W*, and *B*. In other words, we would consider the combinators separately rather than as the combination *BWB*. Although the proof would have extra steps in it, it would still resemble the first proof—that focusing on *L* directly—and, in particular, no reductions would be needed after *x* isolates.

For the third case, that focusing on the expression *Q(QM)M*, we could simply transform the first proof by the minute action of replacing the reductions with *L* by ones with *QM*, used as a combination. However, if one were not aware that *QM* is an *L*—or, for that matter, if we wished to produce a proof precisely in our usual style of applying combinators one at a time—then one would encounter a possibly interesting difference. In particular, although the kernel *QMx(QMx)*—which is *not* the natural kernel of the fixed point combinator *Q(QM)M*—appears on the branch that proceeds from *Q(QM)Mx* and also (farther out) on the branch that proceeds from *x(Q(QM)Mx)*, if one wishes to use this kernel in a proof, then one is forced to apply a reduction on the first branch even after *x* isolates. This difference results directly from the consideration of *Q* and *M* as separate combinators rather than as the combination *QM*.

A Gentle Downhill Slope

From here on, the path is downhill, mostly concerned with reviewing material presented earlier in this section. To begin with, we learned that one need not discard a kernel as useless simply because it fails to play the role it was intended for. For example, if one were unaware of Theorem 5 of Section 4.6 which says that one cannot construct a fixed point combinator from a single regular combinator, then one might have attempted to apply the kernel method to the consideration of the combinator *L* by itself for such a construction. Stage 1 of the two-stage method or of the three-stage method would return *Lf(Lf)* as the only superkernel. Since no expansion with *L* applies to *Lf(Lf)*, even if we relax all restrictions, stage 2 returns nothing.

Nevertheless, even though *Lf(Lf)* is discovered in the doomed attempt to construct a fixed point combinator from *L* alone, one could put this kernel to good use. By giving the kernel method access to the combinator *S*, it finds *SLL* as a fixed point combinator; if instead, one gives the method access to *B* and *M*, it finds *BML* as a desired object. In both cases, no other solutions are found if the method focuses exclusively on the kernel *Lf(Lf)*.

If one has the kernel method begin from the beginning, then the consideration of *S* and *L* together appears to produce nothing new. However, for the second case, the kernel method finds the superkernel *M(BxM)*, but discovers correctly that no fixed point combinator can be constructed with *B*, *M*, and *L*, starting with this superkernel. The clue explaining why *M(BxM)* is useless in this regard is that each of *B*, *M*, and *L* has the property that the same variable appears at the end of the left side and the right side of the equation that characterizes the combinator. Therefore, a reduction of Θx and of $x(\Theta x)$, where Θ denotes a fixed point combinator expressed in terms of *B*, *M*, and *L*, must force *x* to be the last symbol at each stage.

We are not certain whether other interesting results are obtainable from the consideration of the various subsets of the set consisting of *B*, *M*, and *L*. We have studied the possible presence or absence of the strong fixed point property for the subset consisting of *B* and *M* alone, and our study has led us to

conjecture that this subset cannot satisfy the property—which is also Smullyan's conjecture. Eventually, we shall present our incomplete effort in that regard.

Since we have not yet completed our study of B and M—and, therefore, cannot satisfy the curiosity of logicians or of ourselves on the topic—we shall instead supply a diversion in lieu of the desired information. This diversion does fit with some of our discussion, for it makes use of the kernel $Lx(Lx)$. Specifically, if one considers the combinator N with

$$Nxyz = xzyz,$$

then one can use $Lx(Lx)$ to construct from B, L, and N the fixed point combinator $N(BBL)L$. The fixed point combinator $N(BBL)L$ is a reduction with B of the fixed point combinator SLL, which can be seen by noting that $BN(BB)$ is an S. Both fixed point combinators have as their corresponding kernel $Lx(Lx)$.

This combinator N, which may not have been studied by many researchers—or may not have been studied by any before us—is unusually useful for constructing fixed point combinators from kernels. When used in conjunction with B, its power is in part derived from the fact that B as an expansion left associates expressions, and N as an expansion prefers them that way in contrast to S which does not. The other major contributing factor to the power of N rests with the fact that kernels are frequently expressible as a left-associated power greater than 1 of some generator, and the generator is frequently right associated. Although we could continue to focus on this diversion concerning the combinator N, we instead return to comments about the kernel $Lx(Lx)$ and results obtained with it.

In addition to the use of $Lx(Lx)$ as a kernel, we can make use of the fixed point combinators we obtain by combinator adjunction. From SLL, in view of part of our earlier discussion, we can immediately produce the fixed point combinators $S(QM)(QM)$ and $S(BWB)(BWB)$ since QM and BWB are each an L. From BML, we can similarly produce $BM(QM)$ and $BM(BWB)$. Of course, as already noted, one cannot predict by inspection what the kernel correspondent of each will be, predict the precise kernel from which each is obtainable by the kernel method. To make such a determination, one appends x to the combinator and applies a sequence of reductions, the first part of which consists of reductions satisfying the 1's rule, and the last part of which is either empty or applied after x isolates.

As expected, one cannot escape redundancy—redundancy in the form of constructing a given fixed point combinator more than once, or, for that matter, redundancy in the material we present about fixed point combinators. For example, if one is studying B and W as the only combinators of interest, and if the kernel method is applied to this study, it returns $W(Bf)(W(Bf))$ as one of the superkernels it finds. Again, as with $Lf(Lf)$, the method cannot construct any fixed point combinators from this superkernel if it is restricted to the use of B and W alone.

The similarity to $Lf(Lf)$ continues, for the method can profitably use S or use the pair B and M. In the first case, the kernel method constructs $B(SWW)B$ as a fixed point combinator, and in the second case it constructs $B(BMW)B$ as such. However, the superkernel $W(Bf)(W(Bf))$ does offer more than $Lf(Lf)$ in the sense that, from it, one obtains the kernel $BWBf(BWBf)$. This kernel can be obtained by global expansions applied to the superkernel to which it belongs, or directly by stage 1 of the two-stage kernel method. If one continues to search with B and W for kernels constructible from the superkernel under consideration, nothing else is discovered.

A quick way to know that the search for kernels from superkernels should be discontinued along a given path is to stop when the Skolem constant f goes free, meaning that every occurrence of the generator of the current expression has the form Ef for some E containing no occurrences of f. This stop condition is similar to that used when searching for a fixed point combinator along a path that begins with a kernel. Specifically, such searches along a given path should be terminated when one obtains Θf for some Θ, which can be rephrased to the condition that says to stop when f goes free. Stopping at such a point does not guarantee that a fixed point combinator has been constructed, for Θ may contain occurrences of f. Nevertheless, we do rely on this condition, and simply inspect what we find.

If, after f goes free, one does continue to search for fixed point combinators by applying additional expansions (not necessarily satisfying the 1's rule), and if Θ is indeed free of occurrences of f, one simply finds other expressions that are obviously equal to that already in hand. For example, if one obtains $\Theta_4 f$ with

$$\Theta_4 = B(WW)(BW(BBB)),$$

one can expand further along one path to get (with f appended)

$$J_1 = B(WW)(BW(WBB))$$
$$J_2 = B(WW)(BW(WWB))$$
$$J_3 = B(WW)(B(BW)(WW)B)$$
$$J_4 = B(WW)(BBBW(WW)B)$$

and along another path to get (with f appended)

$$J_5 = BBWW(BW(BBB))$$

among other expressions. Although each of (J_1) through (J_5) is provably a fixed point combinator, each is also reducible to Θ_4—regardless of the presence or absence of extensionality—and, therefore, the additional expansions merely produce other versions of the fixed point combinator Θ_4.

We can summarize this discussion in the following way. Since, in most cases, we are interested in a fixed point combinator only if it is irreducible, and since (J_1) through (J_5) are each reducible to Θ_4, they offer little interest. By avoiding expansion paths of the type just discussed, we can reduce, but not eliminate, the redundancy that one encounters. Incidentally, as commented earlier, the fixed point combinators Θ_1 through Θ_5 are each irreducible.

For the type of redundancy for which we have no simple rule for avoiding, we focus on the kernel $BWBx(BWBx)$. Although this kernel is of no use for constructions from B and W alone, one would immediately suspect it is of use for others if either S is adjoined or M is adjoined. After all, BWB is an L and, therefore, the kernel resembles $Lx(Lx)$ very closely. If one does in fact attempt to make use of this kernel, in the first case one again constructs $S(BWB)(BWB)$, and in the second one constructs $BM(BWB)$. Since we have seen these combinators before, we find little pleasure in finding them again along a slightly different path of inquiry. Of course, one could have anticipated the outcome if one took the various facts into account and remained aware of the relationships such as BWE is an L.

4.6. Where Fixed Point Combinators Fear to Tread

In the preceding section, we presented a detailed treatment of a systematic method—the *kernel method*—for searching for fixed point combinators when given some set P of combinators to use for such a construction. In that same section, we then successfully applied the kernel method to the study of various sets. In other words, we used the kernel method to prove that the strong fixed point property holds for a variety of sets P of combinators, each proof resting on the actual construction of an appropriate fixed point combinator. In this section, in contrast, for certain sets P we address the other side of the question—that of showing that the strong fixed point property is *not* satisfied. We show, in particular, what actions one might take when the kernel method fails to construct a fixed point combinator after what the researcher decides is a substantial amount of computer time or—if applied by hand—a substantial effort. Specifically, we show how one can prove, for two different sets of combinators, that the strong fixed point property cannot be satisfied. In Section 5.3.6, we generalize the two results and also give related results.

Of course, the long-term goal is to know when one can and when one cannot construct a fixed point combinator for a given specific set P of combinators. By studying various sets that do satisfy the strong fixed point property, one can, theoretically, be on the way to finding out which conditions are sufficient for the desired construction. On the other hand, by studying sets that fail to satisfy the property, one might discover the necessary conditions. Of course, either study might meet with severe obstacles. For example, the second study requires knowing that a given set P does indeed fail to satisfy the strong fixed point property, but such knowledge is often lacking, which brings us back to the main topic of this section. Specifically, how might one proceed in the attempt to prove that no fixed point combinators can be constructed from a given set P of combinators? Since this question is so closely related to both the question focusing on necessary conditions and the question focusing on sufficient conditions for such constructions, our approach will be to attack all three questions more or less simultaneously. A natural course to take is to consider what we have learned from our successes and our failures.

As we learned, the program or person applies the first stage of the two-stage method by beginning with the denial of the weak fixed point property, and uses expansions to search for kernels; in the second stage, if any kernels are found in the first stage, the program or person attempts to expand each to one or more fixed point combinators. Alternatively, one can apply the three-stage method whose first stage searches for superkernels rather than searching for kernels; if any superkernels are found, then the second stage searches for kernels by expanding each superkernel subject to various expansion rules, and the third stage attempts to construct from each kernel that is found one or more fixed point combinators.

We also learned that, although application of the kernel method can strongly suggest what may be true, the method itself does not directly address the question of proving that no fixed point combinator can be constructed from a given set P of combinators. Of course, when no kernels are found in the first stage, then we predict with confidence that no fixed point combinators are constructible from P; unfortunately, as discussed in the preceding section, we cannot take this position with certainity since we do not yet know whether the kernel method is in fact complete. Therefore, since establishing that a property does not hold is often as interesting as establishing that it does, we focus in this section on methods for showing that fixed point combinators are *not* constructible for various sets P.

As we proceed, we suggest that one recall that, for the example of P consisting of B and L, we did show how the kernel method attempted to address the lack of such constructions, but we could not be definitive because of certain missing pieces of the theory. Specifically, we do not know for certain that, for every fixed point combinator, a kernel must exist from which the combinator can be constructed, which is what we mean when we say we do not know if the kernel method is complete with respect to constructing fixed point combinators. In addition, we have not yet proved that our method finds all kernels that exist—the method that focuses on finding and then expanding superkernels, or the method whose first stage searches directly for kernels. One could refer to this aspect as completeness with respect to constructing kernels, in contrast to completeness with respect to constructing fixed point combinators.

Let us begin our discussion of the cases where fixed point constructions fail with a discovery made in the preceding section. There we learned that one can find a superkernel with respect to some given set P of combinators such that, apparently, neither the superkernel nor the kernels obtained by expanding it are sufficient for constructing a fixed point combinator from the elements of P alone. We examined two interesting cases in that regard. In the first, we studied the set P consisting of B and W alone, the superkernel $W(Bx)(W(Bx))$, and the kernel $BWBx(BWBx)$ obtained from that superkernel. In the second, we studied the set P consisting of L alone and the superkernel $Lx(Lx)$ from which we obtained no additional kernels.

In one sense, when the desired construction fails, the fault lies with the superkernel or kernels themselves. In another sense, however, the fault lies with P itself; P may not contain precisely what is needed to complete the desired construction. Although we can make some observations about kernels, our main success concerns sets P and their properties. Specifically, we can identify certain necessary properties of P for it to satisfy the strong fixed point property, and—perhaps because we have nothing to contribute on the related topic—we suggest that the question concerning sufficient properties is worthy of future research.

A Preliminary Search for Some Necessary Conditions

To begin our study of what properties of P are necessary for constructing fixed point combinators, and to show how subtle this topic is, we review certain properties of $W(Bx)(W(Bx))$. Although a study of examples may only indirectly address the question of why some set P fails to satisfy some desired property, we can justify such an indirect approach because it so often succeeds, at least for us, and because it illustrates one approach to research.

First of all, $W(Bx)(W(Bx))$ is a superkernel with respect to the set P, where the only combinators in P are B and W. Second, all indications suggest that this superkernel is useless for directly constructing a fixed point combinator from the elements of P. All is not lost, however, for, if P is extended to the set P' consisting of B, W, and M, then this superkernel proves to be very useful. Indeed, if we expand with the elements of the larger set P', we construct the fixed point combinator $B(BMW)B$. Summarizing, we have a kernel that lacks at least one crucial property for constructing a fixed point combinator from B and W alone. Although we cannot say what that property is, we did discuss intuitively why this kernel fails to lead to such a construction for the set P even though the desired combinator—for example, any of Θ_1 through Θ_5—does exist. In particular, we are still unable to provide a precise statement about what is

sufficient for some set P of combinators to possess the strong fixed point property.

In addition, at least so far, our demonstration of how one might conduct research has not yet led to an adequate identification of many of the properties P must have to at least give it a substantial chance of satisfying the strong fixed point property. In particular, our example of results obtained from stage 1 of the kernel method has not yet enabled us to find enough interesting properties that P must have for P to satisfy the strong fixed point property. Therefore, perhaps we can profitably apply one of our recommended approaches to research—that of attempting to isolate the diverse aspects of a problem with the intention of uncoupling various properties.

In particular, to see what we can learn, we can focus directly on an example of results obtained from stage 2 by itself. We can then look at the two examples together and perhaps gain some insight into what properties are indeed needed for the strong fixed point property to hold. We take actions of this type—focusing on the separate stages of a procedure and studying examples pertinent to each—because, occasionally, such actions lead to finding a missing piece of the underlying theory. By following this approach, we can show how complicated it all is, and also continue to show how research can be conducted. And of course, we might finally succeed in identifying some of the necessary properties that a set must have for it to satisfy the strong fixed point property. Then we can turn to some theorems concerning the absence of the strong fixed point property and apply those theorems to various sets P.

With regard to an example pertinent to the second stage of the two-stage kernel method—in order to isolate stage 1 from stage 2—let us examine an expression Δ that works well for that stage, but that does not work well for stage 1. By working well for stage 2, we mean that one can expand the expression Δ to Ξf for some Ξ and some Skolem constant f, suggesting that Ξ *is* in fact a fixed point combinator. By not working well for stage 1, we mean that the expression Δ fails to satisfy the requirements for being a kernel—in particular, Δ does not reduce to $f\Delta$—suggesting that Ξ is in fact *not* a fixed point combinator. We use Δ and Ξ rather than Γ and Θ because the latter two symbols are reserved, respectively, for kernels and fixed point combinators, and we are not actually focusing on either. The example in question focuses on using B and W to expand an expression that contains four occurrences of $W(Bf)$. To make it easy to follow the various expansions, we frequently denote $W(Bf)$ by Ω, which is a slight extension of our use of this Greek letter for the generator of a kernel.

The following sequence of equalities shows why, if the expression we now give were chosen for study, one might indeed have been encouraged.

$$W(Bf)(W(Bf))((W(Bf))(W(Bf))) = \Omega\Omega(\Omega\Omega) =$$
$$B(\Omega\Omega)\Omega\Omega = BB\Omega\Omega\Omega\Omega = W(BB)\Omega\Omega\Omega = W(W(BB))\Omega\Omega =$$
$$W(W(W(BB)))\Omega = B(W(W(W(BB))))W(Bf) = B(B(W(W(W(BB))))W)Bf$$

We thus see that, if by some means one had discovered the expression $\Omega\Omega(\Omega\Omega)$, one might have suspected that a fixed point combinator was nearby. After all, the corresponding expansions just given would have been rewarded—rewarded in the sense that a crucial part of the proof of constructibility of a fixed point combinator would have been obtained; in particular, one obtains an expression of the form Ξf with Ξ free of any occurrences of f. In addition, if one then took into account the fact that $W(Bf)(W(Bf))$ is a superkernel that fails to lead to a fixed point combinator constructible from B and W alone and also considered the apparent reason for the failure, one might have been further encouraged.

The apparent reason for failure of the superkernel $W(Bf)(W(Bf))$ to lead to the construction of a fixed point combinator rests with the inability of the combinator W to perform its magic, that of removing the duplicate occurrences of f. In contrast, the new expression $\Omega\Omega(\Omega\Omega)$ does expand sufficiently; W can work its magic, yielding an expression with one occurrence of f, and an expression of the form Ξf with Ξ obviously containing no occurrences of f and f free.

Unfortunately, despite these encouraging observations, a piece would still be missing—an aspect of the potential proof must still be addressed. Specifically, when one attempts to prove that the starting expression $\Omega\Omega(\Omega\Omega)$ is a kernel, one fails. In particular, if one applies reductions to this expression, one obtains the sequence

$$W(Bf)(W(Bf))(W(Bf)(W(Bf))) =$$
$$\Omega\Omega(\Omega\Omega) =$$
$$Bf\Omega\Omega(\Omega\Omega) =$$
$$f(\Omega\Omega)(\Omega\Omega),$$

which is not acceptable since f is not isolated. Instead, one is required to obtain

$$f(\Omega\Omega(\Omega\Omega)),$$

and thus the promise is not fulfilled, for the expression $\Omega\Omega(\Omega\Omega)$ is in fact not a kernel. Nevertheless, the example does illustrate a viable approach to research.

Completion of the Search for Some Necessary Conditions

Having tried an indirect approach that does in fact not produce the precise information about why a set P fails to possess the strong fixed point property, we turn to a second approach that does lead to success. In particular, we consider various sets P that do satisfy the strong fixed point property with the intention of discovering what is common to them. For example, in the preceding sections, we learned that the set P consisting of B and W alone has this property in abundance—an infinite set of fixed point combinators can be constructed from B and W alone. For a second example, the set consisting of S and L alone has the property, but barely; we were able to find only one appropriate combinator, SLL. Similarly and next, when P consists of U alone, UU appears to be the only fixed point combinator that can be constructed. Finally, when we constructed $B(BMW)B$, we proved that the set P consisting of B, W, and M also has the strong fixed point property. Given these four sets, we can immediately focus on what properties are common to them.

Rather than attempting to enumerate by inspection various common properties of the four sets with the intention of solving the problem of knowing when one can and cannot construct fixed point combinators, we instead focus on the most obvious property shared by the four sets—from each, we can construct a fixed point combinator Θ. We can, therefore, shift our attention to how we usually prove that some given combinator Θ is a fixed point combinator. We typically begin with the equation

$$\Theta x = x(\Theta x)$$

with the object of showing that the equation holds for all combinators x.

Our preferred style of proof proceeds by invoking the Church-Rosser theorem—when two expressions are equal, then a third expression exists such that each of the two original expressions reduces to the

third. Therefore, if the given equation holds, a third expression must exist—possibly equal to one of the two—such that both Θx and $x(\Theta x)$ reduce to it. Since the claim is that Θ is constructible from the elements of the given set P, one is permitted the use of a specific reduction step only if the corresponding combinator is an element of P.

We need not focus on a specific P, but let us assume that no variables occur in Θ; if any did, we could simply replace them with any combinator of our choice since $\Theta x = x(\Theta x)$. Let us also assume no eliminators are among the elements of P, to enable us to state the argument more simply. Finally, let us assume that we are focusing on an instance of the equation for the fixed point combinator in which the variable x has been replaced by an arbitrarily chosen combinator f. Since the type of proof we are pursuing relies on reducing both sides of the equation for the strong fixed point property to a common expression, the first important point to note is that the only possible expressions that Θf and $f(\Theta f)$ can both reduce to must have a leading f. Since this point is crucial, let us pause and examine this observation in some detail.

First of all, the equation characterizing the strong fixed point property must hold for any combinator x. Therefore, as indicated, we can let f be such a combinator, and, since the equation must hold for all combinators x, the equation must hold for f not an element of P. Equivalently, we can choose f as we would if we were starting with the denial that Θ is a fixed point combinator and using an automated theorem-proving program to attempt to find a proof of that claim. It follows that no reduction with an element of P can apply to a term whose first symbol is f—from the viewpoint of automated theorem proving, no paramodulation will succeed involving the left side of an element of P and an *into term* beginning with f. Therefore, if one succeeds in reducing both Θf and $f(\Theta f)$ to some third expression, that third expression must begin with the f of $f(\Theta f)$; that occurrence of f will not disappear. In particular, the supposed pair of reductions that reduce Θf and $f(\Theta f)$, respectively, to a third expression must ignore the leading f in $f(\Theta f)$. The common expression to which both Θf and $f(\Theta f)$ reduce must, therefore, have the form $f\Delta$ for some Δ.

Next, we observe that, since Θ contains no occurrences of f—Θ is expressed purely in terms of elements from P—some single reduction step must replicate f as Θf is being reduced to $f\Delta$. After all, Θf and $f(\Theta f)$ are each being successfully reduced to $f\Delta$, $f(\Theta f)$ contains one more occurrence of f than does Θf, and none of the reductions being used—and here is a case in which we rely on the lack of any eliminators in the set P—can remove an occurrence of f. The set P, therefore, must contain at least one replicator—some combinator such that a variable on its left occurs more than once on its right. To be transparent, the f of Θf is the only occurrence of f, and a second occurrence must be produced because Θf and $f(\Theta f)$ are both being reduced to a third expression. Some reduction, therefore, must unify a term containing this single f with a variable on its left side, and then produce at least two occurrences of f in the result, implying that the variable in question must occur more than once on the right side. In other words, we have found one of the necessary conditions that P must satisfy for P to possess the strong fixed point property—P must contain at least one replicator.

We can complete our analysis concerning necessary conditions by observing that the first occurrence of f in $f(\Theta f)$ remains isolated throughout the reduction process—the third expression to which Θf and $f(\Theta f)$ both reduce must have the form $f\Delta$ for some Δ. Therefore, a combinator must exist in P that isolates one of the occurrences of f that result from reducing Θf to $f\Delta$, and we have found another necessary condition.

Some Useful Theorems

We have thus proved the following useful theorem.

Theorem 4. If P is a set of combinators such that the strong fixed point property holds for P, then P must contain a combinator that is a replicator and a (not necessarily different) combinator that is an isolator. Any fixed point combinator Θ constructed from the elements of P must contain an occurrence of a combinator such that, when it is applied to reduce Θf to the common expression to which $f(\Theta f)$ reduces, the combinator replicates f, where f is an arbitrarily chosen combinator. Finally, any fixed point combinator Θ constructed from the elements of P must contain an occurrence of a combinator such that, when applied as a reduction to Θf to reduce Θf to the common expression to which $f(\Theta f)$ reduces, the combinator isolates f, where again f is an arbitrarily chosen combinator.

Theorem 4 gives necessary conditions for a set P of combinators to satisfy the strong fixed point property. If we replace the strong fixed point property in Theorem 4 by the weak fixed point property, then the corresponding theorem is also true, which gives us necessary conditions for a set P of combinators to satisfy the weak fixed point property.

We can quickly apply Theorem 4 to one of the four cases cited in the introduction of this report. There we claimed that the set P consisting of S and W alone fails to satisfy the strong fixed point property. We can easily prove this claim now by applying Theorem 4 and examining the two equations

$$Sxy = xz(yz)$$

and

$$Wxy = xyy$$

for S and W. We immediately see that P does indeed contain a replicator; both S and W have that property. However, neither S nor W is an isolator, as one can see by inspecting the right side of each; in particular, neither S nor W can isolate an occurrence of f. Therefore, the set P consisting of S and W alone cannot satisfy the strong fixed point property.

We can use Theorem 4 to prove another useful theorem (Theorem 5) concerning sets P that cannot satisfy the strong fixed point property. This theorem answers, in the negative, the following previously open question posed by Smullyan in his book *To Mock a Mockingbird* [Smullyan84]; Statman also answered this question [Statman86], but his proof is quite different from ours. Does there exist a single regular combinator such that one can construct from it alone a fixed point combinator?

Theorem 5. If P consists of a single regular combinator, then P fails to satisfy the strong fixed point property.

Proof. Let Ψ be the only element in P, and let Ψ be a regular combinator. Assume by way of contradiction that P in fact satisfies the strong fixed point property. Then, by Theorem 4, Ψ must be a replicator. Since Ψ is regular, the first variable on its right side must occur exactly once. Therefore, since Ψ replicates some term, Ψ must contain more than one variable. In addition, the replicating variable must not be the first variable on the right side of its equation, which is also the first variable on the left side of that equation.

Now if Ψ is applied as a reduction to an expression of the form AB with A free of any occurrences of some constant f and A and B are not empty, then—with the following detailed argument—we can prove that the expression CD that is obtained is such that C is not empty, D may be empty, and C is free of any occurrences of f. To see that this conclusion is valid, first note that B must unify with the last variable on the left side of the equation for Ψ. Next, since the equation for Ψ must contain more than one variable on its left—a replicating variable must exist on the left that is different from the first variable that occurs there—and since all of the variables on its left are distinct because Ψ is regular and therefore proper, D must not unify with the first variable on the left side of the equation for Ψ. Then we note that, since by Theorem 4 Ψ must isolate f in some application, the first variable on its right must be isolated from the rest of the expression on its right. In particular, the right side must be of the form xS for some nonempty expression S of variables, where S contains no occurrences of x. Next note that, since the variable x occurs on both sides of Ψ, C must not be empty. Even further, C is the expression substituted for x in the reduction with Ψ. Finally and most important, C must be free of occurrences of f because x is isolated from S.

To complete the proof, we now apply Ψ as a reduction to Θf, where Θ is an assumed fixed point combinator constructible from Ψ alone. By earlier remarks, repeated application must eventually reduce Θf to $f\Delta$ for some expression Δ. But Θ is free of any occurrences of f. Therefore, each reduction with Ψ produces some CD with C not empty and free of f. We can then conclude that, regardless of the number of applications of Ψ, at no point will an expression of the form $f\Delta$ for some Δ be produced, which contradicts the original assumption that P satisfies the strong fixed point property, and the proof is complete.

For the curious reader, we note that, in the proof of Theorem 5, we did not assume that the regular combinator is noneliminating. Because we did not make such an assumption, D could in fact be empty. For example, the right side of the equation for Ψ could have the form $x(yy)$ with the left side of Ψ containing three or more variables. Our proof of Theorem 5 appears to be simpler than that credited to Statman.

From the proof of Theorem 5, we can extract a useful lemma, Lemma 1.

Lemma 1. If Ψ is a combinator that is regular, that contains more than one variable on its left side, and that is isolating, then reducing the expression AB with Ψ when A is free of occurrences of some symbol, say f, produces an expression CD with C not empty and free of occurrences of f.

Proof. The proof of this lemma is essentially the second paragraph of the proof of Theorem 5.

Examples of isolating combinators include B, L, and U, which have the following equations, respectively.

$$Bxyz = x(yz)$$
$$Lxy = x(yy)$$
$$Uxy = y(xxy)$$

Of these three combinators, L and U are replicating, and B and L are regular.

A review of the preceding theorems and observations suggests we might now be able to complete our analysis of the four cases cited in Section 1. When P consists of S and W, we have already given the answer: Theorem 4 shows that P cannot satisfy the strong fixed point property. When P consists of B

and W alone, an abundance of fixed point combinators can be constructed, which we have done, so the strong fixed point property holds for this P. When P consists of S and L alone, again we know that the strong fixed point property holds because of SLL. We are left with the case in which P consists of B and L alone. This case is covered by Theorem 6, which we now easily prove—indeed, we give two proofs of that theorem.

Theorem 6. A fixed point combinator cannot be constructed from B and L alone.

Proof. The combinators B and L are each isolating. By Lemma 1, if either is applied to an expression of the form AB with A free of occurrences of f with both A and B nonempty, then the expression CD will be obtained with both C and D nonempty and with C still free of occurrences of f. For Θ to be a fixed point combinator, a reduction of Θf to $f\Delta$ must exist. But, if Θf is repeatedly reduced with B and L, one can never obtain an expression of the form $f\Delta$ for some Δ, and the proof is complete.

We give the following alternative proof for Theorem 6 because it may provide one with additional techniques that can be used elsewhere.

Proof. Assume, by way of contradiction, that the strong fixed point property holds for the set P consisting of B and L alone. Then there exists a combinator Θ, which is constructed from B and L alone, such that, for an arbitrary combinator f, $\Theta f = f(\Theta f)$. (We use the constant f rather than F to be consistent with our notational convention when denying some theorem is true.) By the Church-Rosser property for applicative systems, there exists a combinator Δ such that Θf reduces to Δ and $f(\Theta f)$ also reduces to Δ. (The reductions that are used are paramodulations from the left sides of B and L.) Since $f(\Theta f)$ reduces to Δ, and since the first occurrence of f cannot be affected by any reduction with B or L, Δ must be of the form $f\Xi$ for some combinator Ξ. Therefore, Θf reduces to $f\Xi$ for that same Ξ. The combination Θf obviously has the form CD, where C contains no occurrences of f. Let us consider a one-step reduction of CD and show by case analysis that the result $C'D'$ is such that C' contains no occurrences of f.

Case 1. The reduction involves C only or D only. Obvious.

Case 2. The reduction is with B and involves both C and D. Then C must unify with $(Bx)y$, D must unify with z, C' must be the image of x, and D' must be the image of yz. Therefore, C' must be a subterm of C, which implies that C' contains no occurrences of f.

Case 3. The reduction is with L and involves both C and D. Then C must unify with Lx, D must unify with y, C' must be the image of x, and D' must be the image of yy. Therefore, C' must be a subterm of C, which implies that C' contains no occurrences of f.

We can conclude that, regardless of the number of reductions we apply starting with $CD = \Theta f$, we can never obtain an expression of the form C^*D^* with C^* containing an occurrence of f. In particular, we can never reduce Θf to $f\Xi$, and we have arrived at a contradiction. In other words, the proof is complete; the strong fixed point property fails to hold for the set P consisting of B and L alone.

With the following theorem, Theorem 7, we can strengthen the result that asserts that the set P consisting of S and W alone cannot satisfy the strong fixed point property. In particular, the set P fails to satisfy the weak fixed point property.

Theorem 7. If the set P of combinators satisfies the weak fixed point property, then P must contain an isolator.

Interesting Comparisons

We can now make some interesting comparisons that again illustrate the complexity and beauty of combinatory logic and, especially, of the construction of fixed point combinators. From Theorem 5, we know that the set P consisting of L alone fails to satisfy the strong fixed point property. However, since $Lx(Lx)$ is a kernel, we know that this set P does satisfy the weak fixed point property; we recall the every kernel serves well as a y for which

$$y = xy$$

holds. From Theorem 6, if we extend P by including B, one still cannot construct a fixed point combinator from this extended set, even though the first stage of the two-stage kernel method is successful. However, if we extend P further to consist of the combinators L, B, and M, then the strong fixed point property does hold for the resulting set. In particular, the sequence

$Lx(Lx)$
$M(Lx)$
$BMLx$

is precisely what is needed to complete a proof that BML is a fixed point combinator constructible from P.

Although one could not be certain, one might have easily guessed that such would be true since we showed earlier that $BM(BWB)$ is a fixed point combinator, and BWB is an L. In Section 3.2, we noted that saying that BWB is an L means that BWB behaves the same as L does, satisfies the same equation. Of course, noting that BWB is an L and part of some fixed point combinator does not obviously guarantee that one can simply replace BWB by L since the actions of the individual combinators that combine into another combinator may permit one to complete a proof in contrast to using the entire combination at once. However, as it turns out, a careful analysis will enable one to prove that such a replacement will in fact always work. The proof rests on the fact that the equations for proper combinators have the property that their respective left sides are fully left associated, which implies that, for example, QM or BWB or L must unify with a single variable or be part of a term that unifies with a single variable when a reduction is applied. Of course, one might be forced to apply a reduction after x isolates for the type of proof we prefer.

For example, in the proof that we prefer showing that SLL is a fixed point combinator, one first reduces $SLLx$ to its natural kernel correspondent $Lx(Lx)$, and then reduces $Lx(Lx)$ to $x(Lx(Lx))$ with a single application of L. No reductions are required after x isolates; the kernel $Lx(Lx)$ is a simple kernel. On the other hand—keeping in mind that QM is an L—when we prove that $S(QM)(QM)$ is a fixed point combinator, we first reduce $S(QM)(QM)x$ to its kernel $QMx(QMx)$, but we then must apply two reductions to this kernel to obtain $x(QMx(QMx))$, the second of which is a reduction with M after x isolates. In other words, the kernel $QMx(QMx)$ that corresponds to the fixed point combinator $S(QM)(QM)$ is a semisimple kernel even though its (so to speak) equivalent kernel $Lx(Lx)$ is simple. If we instead study $S(BWB)(BWB)$, we find that it behaves precisely the way SLL behaves; its kernel is $BWBx(BWBx)$, which is simple.

If we pursue this line of inquiry somewhat further, we find another striking difference in the behavior of the various L's. Specifically, from the superkernel $Lx(Lx)$, we cannot construct a fixed point

combinator from the combinators that occur in that kernel—namely, L—which follows from Theorem 5 of Section 4.6. On the other hand, from the kernel $QMx(QMx)$, the second stage of the two-stage kernel method will succeed in constructing the fixed point combinator $Q(QM)M$. The reason for this success, when compared to the failure with using L to construct a fixed point combinator from the kernel $Lx(Lx)$, rests with the fact that each of Q and M can be used separately to expand the kernel $QMx(QMx)$. In other words, even though QM is an L, we have an example in which one kernel is useless for the construction of fixed point combinators in contrast to a (so to speak) equivalent kernel that is useful.

Of course, if one closely examines the fixed point combinator $Q(QM)M$, one discovers that its natural kernel correspondent is $M(QMx)$, and not $QMx(QMx)$. From what we know, the fact that one could have forced the proof, that $Q(QM)M$ is a fixed point combinator, to continue past the kernel $M(QMx)$ to rely instead on the kernel $QMx(QMx)$ has essentially no significance. Each of the two kernels is a superkernel, and, as it turns out, the first simply reduces to the second; such occurrences are encountered rather frequently.

To add further to the possible confusion—actually, we prefer to say, to add further to the intrigue and challenge of combinatory logic—we note that a study of the combinator QLM shows it to be, as expected, a fixed point combinator, but one whose corresponding kernel is $Lx(Lx)$. Indeed, although the reduction of $QLMx$ does, at one point, arrive at $M(Lx)$—which one would expect since the reduction of $Q(QM)Mx$ arrives at $M(QMx)$—we cannot think of any significant conclusions to draw from this occurrence. In particular, $M(Lx)$ is just not a kernel.

If we turn our attention back to the kernel $WBx(WBx)$ with the object of emulating our study of the kernel $QMx(QMx)$, we find disappointment. Specifically—as we have learned more than once—we cannot construct a fixed point combinator from this superkernel, even if we are allowed to expand with both B and W. Indeed, instead of paralleling the successful construction that produces the fixed point combinator $Q(QM)M$, our attempt with $W(Bx)(W(Bx))$ more or less mirrors our attempt with the kernel $Lx(Lx)$. Of course, we have the small difference that we can succeed in expanding $W(Bx)(W(Bx))$ with B in contrast to totally failing with L applied to $Lx(Lx)$, but the corresponding expansion path, as we saw earlier, leads nowhere.

Only if we take a global view do we find the study of $W(Bx)(W(Bx))$ profitable for constructing fixed point combinators from the combinators mentioned in it. In particular, we could view this superkernel as the real origin of the sequence that we have studied, the sequence beginning with Γ_1 equal to the cube of $W(B(Bx))$. We could have obtained Γ_1 from $W(Bx)(W(Bx))$ by the procedure that we discussed in Section 4.4.2, the procedure that relies on the insertion of B to the right of W in the generator of the preceding superkernel. In fact, to contribute to the mystery of combinatory logic, we know that the set consisting of B and W satisfies the strong fixed point property, but, although BWB is an L, the set consisting of L alone does not. Our earlier comments also show that this mystery is reinforced; the set consisting of Q and M satisfies the strong fixed point property, but, although QM is an L, we cannot simply replace the combination of Q and M with L.

Of course, some of the mystery is explained by the obvious fact that the association of the combinators in an expression gives or takes away power. For example, if we let E denote a combinator such that $Exyz = xyz(xyz)$, then one can show that EQM is a fixed point combinator whose kernel is $QMx(QMx)$. In

contrast, EL is not a fixed point combinator—one cannot reduce ELx, for the combinator E involves three variables. This example clearly does not contradict our earlier observations concerning the replacement of a combination of combinators by its equivalent to obtain a second fixed point combinator from a first. In particular, no justification exists for even considering L when studying EQM since Q and M are not present in the form QM. Therefore, when applying E as a reduction, Q and M are unified with separate variables, rather than being a term or part of a term that is unified with a single variable.

A little more of the mystery is explained by considering an example showing that one can apply (so to speak) part of a combinator without being able to apply the entire combinator. For example, $S(KS)K$ is a combination that acts like B, where $Sxyz = xz(yz)$, $Kxy = x$, and $Bxyz = x(yz)$. If one reduces, as much as possible, $S(KS)Kxy$, one obtains $S(Kx)y$. In contrast, one cannot reduce Bxy at all. These little asides add, at least for us, to the understanding of how combinatory logic works. For yet another illustration, we note that one can often find the equational properties of a proper combinator (or a proper combination of combinators) by simply appending a sequence of variables and then performing the corresponding reductions. For example, the sequence

$$BWBxy = W(Bx)y = Bxyy = xyy$$

shows, as we have already seen, that BWB is an L. If some of the appended variables do not participate in the reduction, one simply discards them.

Let us return to the set P consisting of L alone and recall that we learned that extending this P to contain S is sufficient to have the strong fixed point property hold with SLL a fixed point combinator. For a similar analysis, we also discussed the superkernel $W(Bx)(W(Bx))$ and the fact that it cannot be expanded to a fixed point combinator from B and W alone, but adjoining M is sufficient, for one can then obtain $B(BMW)B$ as the corresponding desired combinator by expansion.

Finally, by maintaining our focus on the combinator L, we can make one more interesting comparison of the strikingly different behavior and properties of individual combinators. In particular, we note the sharply weaker power offered by the use of a regular combinator when compared to that of a combinator that is not regular. For example, as proved, the combinator L by itself offers insufficient power for constructing a fixed point combinator even though it has enough power to prove that the weak fixed point property holds. In contrast, the combinator U with

$$Uxy = y(xxy)$$

has the needed power, which one can see by recalling that UU is a fixed point combinator.

For a second example—and here we can apply the procedure of appending variables to see how some combination behaves—the combinator $CB(BWT)$ with

$$CB(BWT)xyz = x(zyz)$$

is regular and therefore, by Theorem 5 of Section 4.6, lacks the power for constructing a fixed point combinator if used by itself; it does, however, have enough power to satisfy the weak fixed point property, as one can see from studying the kernel schema $CB(BWT)xy(CB(BWT)x)$. (If one wishes to simplify this case by focusing on a specific kernel, one can set the variable y to any chosen combinator.) To determine for oneself how the combination $CB(BWT)$ behaves, one can use the following equations.

$Txy = yx$

$Bxyz = x(yz)$

$Wxy = xyy$

$Cxyz = xzy$

In contrast, if we replace CB by BQ_3, note that the combinator $BQ_3(BWT)$ satisfies

$BQ_3(BWT)xyz = z(yxy)$,

and call that combination E, then E does have sufficient power for constructing fixed point combinators. Specifically, $E(EE)E$ is such that $E(EE)Ex = x(E(EE)Ex)$. To study this combinator, one can use the preceding equations and that for Q_3.

$Q_3xyz = z(xy)$

We thus have arrived at an interesting parallel to certain earlier results, specifically, to the results concerning I and U. We note that $CB(BWT)$ acts like L, for it satisfies the weak but not the strong fixed point property. We also note that $BQ_3(BWT)$ acts like U; it alone is sufficient for proving that the strong fixed point property holds for a set.

A natural question to ask at this point concerns sufficient conditions for guaranteeing that some set P of combinators satisfies the strong fixed point property. This area might well be an interesting research area; in particular, one might try to prove a result closely related to the converse of Theorem 4 of Section 4.6. We can offer only a few unsatisfactory observations, all of which focus on trivial sets of sufficient conditions. For example, if P contains B and W, then the strong fixed point property holds. Similarly, if P contains U, or contains S and L, everything works well. Perhaps a study of these various sets P, as well as others that satisfy the strong fixed point property, will shed a little light.

We close this section by fulfilling our promise of citing other open questions deserving attention. For example, the combinator

$Mx = xx$

is obviously a replicator, and the combinator

$Bxyz = x(yz)$

is obviously an isolator. In view of Theorem 4 of Section 4.6 and the results concerning B and W, perhaps the set P consisting of B and M alone satisfies the strong fixed point property. This question was asked of us by Smullyan in a private conversation, and at one time we were convinced we had an answer; but now, we are not certain. Because of our respect for Smullyan, we devote an entire section in the forthcoming book to this question and to our incomplete attack on it.

Finally, there is the question—perhaps questions is a better term—surrounding the possible completeness of the kernel method. These questions, as one might predict, currently occupy our attention more than any others, and we therefore feature them in yet another section, Section 5.4.1. Specifically—for the first of the two vital questions—is it the case that to every fixed point combinator Θ their corresponds a kernel Γ such that Θx reduces to Γ which in turn reduces to $x\Gamma$? Second, if one applies either the two-stage or the three-stage version of the kernel method without the various restrictions concerning variables and the 1's rule, is it the case that the method will find all kernels Γ?

4.7. Another Family of Fixed Point Combinators for B and W

In this section, we return to our study of the set P of combinators consisting of B and W alone. Our goal is to extend the results of Section 4.4.1. Having found in that section that B and W are sufficient for constructing fixed point combinators of lengths 8, 11, 14, 17, and, in general, $3k+2$ for all $k \geq 2$, one might naturally ask about the existence of fixed point combinators for the other possible lengths. To answer this question, let us review some of the results presented earlier.

First, recall that we focused in Section 4.4.1 on various classes of combinators, grouped according to their length. For example, those of length 8 were traceable to the superkernel $\Gamma_1 = \Omega_1\Omega_1\Omega_1$ with $\Omega_1 = W(B(Bx))$, those of length 11 were traceable to $\Gamma_6 = \Omega_6\Omega_6\Omega_6\Omega_6$ with $\Omega_6 = W(B(B(Bx)))$, and those of length 14 were traceable to $\Gamma_{20} = \Omega_{20}\Omega_{20}\Omega_{20}\Omega_{20}\Omega_{20}$ with $\Omega_{20} = W(B(B(B(Bx))))$. Next recall that the number of fixed point combinators for the first three classes is, respectively, 5, 14, and 42, and, in general, the number in a class of this type is equal to the number of ways to associate n letters, where n is the number of symbols (not counting parentheses) in the generator of the corresponding superkernel. Each superkernel is obtained from the preceding superkernel by inserting a B to the right of W in its generator, observing the rule that the generator of each superkernel of the type under discussion is right associated.

As for fixed point combinators of length less than 8 constructible from B and W alone, none exists. The proof of this assertion will be given later in this section. As for fixed point combinators of length 8, exactly five that are irreducible—Θ_1 through Θ_5 studied earlier—can be constructed. We also delay giving a proof of this assertion until after we have answered the question we are studying, that concerned with combinators of lengths other than $3k+2$ for $k \geq 2$. Finally, we note that we are interested only in fixed point combinators that are irreducible, combinators such that any attempt to reduce them with B or W fails. For example, although the combinator

$$\Theta_4' = B(WW)(B(BW)(BB)B)$$

has length 9 and, if substituted for Θ, satisfies the equation

$$\Theta x = x(\Theta x)$$

for the strong fixed point property, Θ_4' is not of interest for our search for fixed point combinators. We have no interest in Θ_4' because Θ_4' reduces with B to Θ_4, and therefore Θ_4' is trivially equal to Θ_4 regardless of the presence or absence of extensionality.

With the brief review in hand, we can now turn to our attack on the question regarding the existence of other classes of fixed point combinators constructible from B and W alone. In particular, we can search for combinators of the lengths not yet found. For this phase of our research, we shall as usual focus heavily on expansions with B and W—paramodulations from the right sides, respectively, of

$$Bxyz = x(yz)$$

and

$$Wxy = xyy.$$

Of course, as expected, we prefer to conduct our search for new classes of combinators by relying heavily on the kernel method. In the first stage of that method, in addition to finding the superkernel $W(B(Bf))$ cubed, which we have studied extensively, one also immediately finds the superkernel

$BW(B(Bf))$ cubed. The following expansions with B and W, starting with the negation of the equation for the weak fixed point property, lead to its discovery.

$y \neq fy$
$yz \neq Bfyz$
$uvz \neq B(Bf)uvz$
$uvv \neq W(B(Bf)u)v$
$uvv \neq BW(B(Bf))uv$

Since the last of these inequalities contradicts the reflexivity property of equality, $x = x$—which can be seen by the appropriate use of unification—we know that a kernel has been found.

The kernel is the cube of $BW(B(Bf))$—alternatively, the cube of $BW(B(Bx))$—which is obtained by analyzing the contradiction that is found, or by examining the argument of the ANSWER literal when one uses such a literal. We have in fact found a superkernel since no reduction applies to this kernel other than that involving its leading symbol. By applying B and W as reductions, one can check that this superkernel—which we shall call Γ_1^9 to reflect the length of the combinator that can be constructed from it and also to reflect that that combinator is the first in some ordering—does indeed reduce to $f(\Gamma_1^9)$. As we shall soon see, this new superkernel opens another path to discovering additional classes of fixed point combinators constructible from B and W alone. However, before we can explore this path, we make various observations in keeping with our objective of showing how research can be conducted, and we correctly record history.

The Actual Discovery of Γ_1^9

We did not discover Γ_1^9 in the manner just discussed, that of examining what one finds in the first stage of the kernel method; in fact, if we can trust memory, the kernel method had not yet been formulated. Instead, we made the discovery directly as a result of attempting to modify the superkernel $\Gamma_1 = \Omega\Omega\Omega$, with $\Omega = W(B(Bf))$, from which one can construct Θ_1 through Θ_5. Our goal in attempting to modify Γ_1 was to find irreducible fixed point combinators of length 9. We thought some modification of interest might be feasible because we had already studied the relation of Γ_1 to Γ_6 and then to Γ_{20}—in particular, we had discovered that an insertion of an extra B to the right of W proved profitable. Put another way, if inserting a B to the right of W yields success, perhaps one can in some fashion insert a B to the left of W.

A glance at Γ_1^9 and at Γ_1 immediately shows that indeed a uniform replacement of W by BW produces the new superkernel. If one then expands Γ_1^9 with B and with W, one discovers the fixed point combinator

$B(B(B(WW)BW)B)B$,

which is just the combinator of length 9 we were seeking. As a preview of what is to come, note that the fixed point combinator of length 9 we have just given closely resembles Θ_1; one can obtain it by replacing the third occurrence of W in Θ_1 by BW. We thus have a nice example of how pursuing a research path based on insight motivated by imitation of earlier success led to a desired result that would later be corroborated by pursuing a more general research path based on extending a uniform methodology.

If one attempts to apply the same technique and proceed further in this direction to find a fixed point combinator of length 10, the attempt fails. In particular, a uniform replacement of BW in Γ_1^9 by BBW does not yield a kernel, a fact that one can check by applying the appropriate reductions.

Resuming the Search for Other Classes of Fixed Point Combinators

Having updated our historical account of our research, we can resume the search for interesting classes of fixed point combinators by focusing on Γ_1^9 and its relation to Γ_1, the superkernel from which Θ_1 through Θ_5 can be constructed and from which other superkernels such as Γ_6 and Γ_{20} can be found. We noted earlier that the application of various global expansion rules to Γ_1 yields the four kernels Γ_2 through Γ_5, and that the application of expansions subject to the 1's rule to each of Γ_1 through Γ_5 leads to the construction of Θ_1 through Θ_5, respectively. Mathematics and logic again exhibit orderliness, symmetry, and beauty—in particular, we can apply the same approach to Γ_1^9 as we did to Γ_1, obtaining four additional kernels and four additional fixed point combinators.

The five combinators one obtains closely resemble Θ_1 through Θ_5, respectively, except that the third occurrence of W (counting from the left) is replaced by BW. Here are the five.

Θ_1^9 $(B((B((B(WW))(BW)))B))B$

Θ_2^9 $(B((B(WW))(BW)))((BB)B)$

Θ_3^9 $(B((B(WW))((B(BW))B)))B$

Θ_4^9 $(B(WW))((B(BW))((BB)B))$

Θ_5^9 $(B(WW))((B((B(BW))B))B)$

The parallel extends even further. First, just as one can obtain additional superkernels from Γ_1 by repeated insertion of a B to the right of W and then fully right associating the result, one can also obtain additional superkernels from Γ_1^9 in the same way. In this fashion, one obtains Γ_6^{12}, Γ_{20}^{15}, Γ_{62}^{18}, and the like. Second, each of the superkernels in this sequence of superkernels is obtainable from a superkernel in the sequence Γ_1, Γ_6, Γ_{20}, and the like by simply replacing W by BW in all occurrences of the corresponding generator. Third, from each of the new superkernels, one can obtain additional kernels by applying global expansion rules in a manner similar to the way we obtained Γ_2 through Γ_5 from Γ_1. In other words, we have a set of sets of kernels with each set a companion to one of the sets generated, respectively, by Γ_1, Γ_6, Γ_{20}, and the like. Also—as with the sets of combinators derivable from Γ_1, Γ_6, and the like—the number of combinators in each of these new sets is, in ascending order, 5, 14, 42, and, in general, the number of ways in which n letters can be associated. However, in contrast to the originally studied sets of kernels, for the new sets, n is one less than the number of elements in the generator of the corresponding new superkernel—equivalently, n is the number of symbols, exclusive of grouping symbols, in the generator of the old superkernel. Fourth, from each kernel in this infinite set of sets of kernels, one can obtain precisely one fixed point combinator by applying expansions with B and W subject to the 1's rule. The fixed point combinators of each new set, considered in ascending order, have respective lengths 9, 12, 15, and, in general $3k$ for all $k \geq 3$.

Summarizing, we have in fact succeeded in constructing another infinite set of combinators that parallels our construction presented in Section 4.4.1. As before, we have a one-to-one mapping from kernels to fixed point combinators. In short, we have succeeded in reaching our goal of constructing new classes of combinators of lengths different from those given in Section 4.4.1. So far, in our study of the set P consisting of B and W alone, we have constructed 5 combinators of length 8 and from these as companions 5 of length 9, 14 of length 11 and from these as companions 14 of length 12, 42 of length 14 and from these as companions 42 of length 15, ... However, as we shall see immediately, our study eventually revealed an even richer source of fixed point combinators constructible from the elements of P—we uncovered a far richer vein in the B-and-W mine.

A Far Richer Vein for Constructing Fixed Point Combinators from B and W

To discuss the new source of combinators, we now turn to where the parallelism between Γ_1^8 and Γ_1^9 breaks down. Pursuing this new path of inquiry will finally lead to the examination of the overwhelming abundance of fixed point combinators mentioned earlier when we talked about an infinite class of infinite sets. In particular, instead of constructing a third infinite set of combinators that are companions of the infinite set studied in Section 4.4.1, we shall show how one can construct an infinite number of infinite sets of combinators.

Our first action in that regard is to return to applying the kernel method and to a closer examination of the extraction of superkernels. Indeed, let us return to the sequence of expansions given at the beginning of this section, the five expansion steps that could have been used to discover Γ_1^9.

$y \neq fy$
$yz \neq Bfyz$
$uvz \neq B(Bf)uvz$
$uvv \neq W(B(Bf)u)v$
$uvv \neq BW(B(Bf))uv$

For the extraction of Γ_1^9 from the fifth and last inequality—which is accomplished by analyzing the unification that shows that this inequality contradicts reflexivity—we are forced to set $u = v = BW(B(Bf))$. We note that, for the contradiction, we have no choice about how to replace the variables of that final inequality. In contrast, however, note that a close examination of the following sequence of inequalities, which one could have used to discover Γ_1^8, shows that the situation is markedly different.

$y \neq fy$
$yz \neq Bfyz$
$uvz \neq B(Bf)uvz$
$uuz \neq W(B(Bf))uz$

The most important difference one finds when extracting Γ_1^8 from this sequence of inequalities, when compared to extracting Γ_1^9 from the preceding sequence, is that one has a degree of freedom—not all of the variables in the last inequality must be replaced with variable-free expressions to obtain a contradiction with reflexivity. Indeed, rather than finding the superkernel $\Gamma_1^8 = \Omega\Omega\Omega$—which more closely reflects the way we have treated this phase of our research—a careful analysis of the unification that establishes the contradiction with reflexivity shows that the more general expression $\Omega\Omega z$ is found.

(For finding the various kernels and superkernels, one might prefer to rely on the ANSWER literal rather than being forced to analyze the unification establishing that a contradiction has been found; the ANSWER literal can be used to present the sought-after construct explicitly.)

The expression $\Omega\Omega z$ is more general than $\Omega\Omega\Omega$ in the sense that $\Omega\Omega\Omega$ is an instance of it. In fact, as we noted in Section 4.5.4, we call such an expression a *kernel schema* because all instances of it are kernels. One can easily check that all instances of $\Omega\Omega z$ are kernels by choosing an arbitrary instance e of z, replacing Ω by its definition $W(B(Bf))$, reducing with W and B constrained by the 1's rule, and finding that one indeed obtains $f(\Omega\Omega E)$. Even further, all instances of $\Omega\Omega z$ are simple kernels.

Because of its generality, the discovery of this kernel schema $\Omega\Omega z$ proved to be most significant, for its use, as we shall immediately see, leads to the construction of an infinite class of infinite sets of fixed point combinators, where the construction depends on B and W alone. A comparison of the paths that lead to this discovery and that lead to Γ_1^9 provides an example of how the smallest differences at one end of a path can result in immense differences at the other end. Specifically, the small difference between the two expansion sequences rests with the term in the third inequality to which W is applied. In the sequence that could have been used to discover Γ_1^9, the right side of W is unified with the term $B(Bf)uvz$. In the sequence that could have been used to discover Γ_1^8—more precisely, to discover $\Omega\Omega z$—the right side of W is unified with the term $B(Bf)uv$. From this small difference—from the viewpoint of automated theorem proving, unifying with a term versus unifying with one of its subterms—we encounter the immense difference of finding Γ_1^9 and the means for finding five fixed point combinators of length 9 versus finding $\Omega\Omega z$ and, as we shall see, the means for constructing an infinite class of infinite sets of combinators, some of which fulfill our desire for finding combinators of some of the lengths not yet encountered.

To see precisely how the use of $\Omega\Omega z$ does indeed lead to the construction of an infinite class of infinite sets of fixed point combinators—and to complete our study of such constructions from the set P consisting of B and W alone—we again point out that $\Omega\Omega z$ is a kernel schema with $\Omega = W(B(Bf))$. The replacement of the variable z by any expression, even if that expression contains combinators other than B and W, always yields a kernel—in fact, all instances of $\Omega\Omega z$ are simple kernels. We restrict our attention to instances that are free of variables and free of all combinators other than B and W, for we are focusing exclusively on B and W and on the variant of kernel in which a Skolem constant f occurs, rather than that in which a variable occurs in place of f.

Of course, from the viewpoint of combinatory logic, the constant f denotes an arbitrarily chosen combinator. For example, our favorite $\Omega\Omega\Omega$ is a kernel—and, in fact, a superkernel—obtainable from $\Omega\Omega z$ by replacing z by Ω. More generally, since the only reduction that applies to $\Omega\Omega$ is one that satisfies the 1's rule, all instances of $\Omega\Omega z$ such that the expression substituted for z is irreducible are superkernels. In contrast, the instance $\Omega\Omega(\Omega\Omega)$ of $\Omega\Omega z$, although a kernel, is not a superkernel—the second occurrence of $\Omega\Omega$ is also reducible with W. We shall call objects such as $\Omega\Omega(\Omega\Omega)$ pseudo-superkernels; they admit a reduction that does not satisfy the 1's rule, but they do not reduce globally to another kernel. In addition, as with a superkernel, one can derive a set of kernels from them by globally expanding, where some of the global expansions satisfy the 1's rule, and some violate it.

The contrast between $\Omega\Omega(\Omega\Omega)$ and $\Omega\Omega\Omega$, or any of the kernels we studied to derive $\Gamma(BW)$, is far greater than the distinction between being a pseudo-superkernel and a superkernel. For, although one sees that $\Omega\Omega(\Omega\Omega)$ is a kernel by applying reductions to $\Omega\Omega(\Omega\Omega)$ that satisfy the 1's rule and continuing until the first occurrence of f isolates to obtain $f(\Omega\Omega(\Omega\Omega))$, one finds that $\Omega\Omega(\Omega\Omega)$ does not even globally reduce to a superkernel. Therefore, $\Omega\Omega(\Omega\Omega)$ is unlike the kernels in any of the sets we studied earlier—for example, Γ_2 through Γ_5 which reduce to Γ_1. If one searches for a global reduction with the object of eventually finding a suitable superkernel to which $\Omega\Omega(\Omega\Omega)$ reduces, one first finds that each occurrence of $\Omega\Omega$ reduces to $Bf(\Omega\Omega)$ and no further. One is then faced with the choice of again reducing each occurrence of $\Omega\Omega$, or of applying B in a manner consistent with the 1's rule. In either event, one never finds the sought-after superkernel. Nevertheless, we shall see that, not only is $\Omega\Omega(\Omega\Omega)$ useful for constructing fixed point combinators from B and W alone, but that $\Omega\Omega(\Omega\Omega)$ is in fact rather prolific in that regard; in other words, in our exploration of the B-and-W mine, we have come upon a very rich vein indeed. We shall also see that $\Omega\Omega(\Omega\Omega)$ is useful for deriving other kernels.

Let us proceed in part as if the study of $\Omega\Omega(\Omega\Omega)$ were taking place at this very moment. By taking this approach, we can in effect practice research—can apply some of the techniques for conducting research that we have been illustrating throughout. We note immediately that, when trying to construct fixed point combinators from $\Omega\Omega(\Omega\Omega)$, one should expect behavior somewhat or radically different from that encountered with Γ_1 and with its related superkernels such as Γ_6. After all, we just saw that, unlike $\Omega\Omega\Omega$, $\Omega\Omega(\Omega\Omega)$ does not have a superkernel to which it globally reduces. Despite this expectation of odd behavior, the appropriate action is simply to rush forward, using what might be relevant from the study of $\Omega\Omega\Omega$, and see what occurs.

Naturally, we attempt to apply expansions to $\Omega\Omega(\Omega\Omega)$, but restrict our attention to those satisfying the 1's rule; as it turns out, applying expansions that violate the 1's rule leads to the construction of fixed point combinators that duplicate those found by observing the 1's rule restriction. We choose this expansion approach because, although $\Omega\Omega(\Omega\Omega)$ is not a superkernel, it is still a kernel and, therefore, offers the potential for constructing fixed point combinators just as all kernels offer that potential. In addition, that approach worked with the other kernels constructed from B and W alone.

Two possible expansions exist for the first step. We can apply B to the entire expression $\Omega\Omega(\Omega\Omega)$, or we can replace Ω by its definition $W(B(Bf))$ and apply B to the definition of Ω. Our view is that one should, if possible, always delay applying an expansion to such a definition until all occurrences but one of the generator of the kernel—in this case, Ω—have been removed. Of course, as we learned earlier, studying kernels such as $Lf(Lf)$ whose generator is Lf can lead to the construction of a fixed point combinator, such as SLL. Such a construction succeeds without observing our preference of waiting until one occurrence of the generator remains. In other words, we avoid considering any expansion applied to Ω until exactly one occurrence of Ω remains—we treat Ω as a constant, expanding its definition only when Ω occurs once. Therefore, the first step we apply is that of expanding all of $\Omega\Omega(\Omega\Omega)$ with B to obtain $B(\Omega\Omega)\Omega\Omega$.

We immediately encounter some of the expected different behavior of $\Omega\Omega(\Omega\Omega)$ when compared with, say, $\Omega\Omega\Omega$. In particular, there exist two applicable expansions that satisfy the 1's rule and that apply to $B(\Omega\Omega)\Omega\Omega$—in sharp contrast with the fact that, at each point on the path from $\Omega\Omega\Omega$ to the place

where one occurrence of Ω remains, only one applicable expansion exists satisfying the 1's rule. We can either expand with W—as we did with ΩΩΩ—to get $W(B(\Omega\Omega))\Omega$, or we can expand with B to get $BB\Omega\Omega\Omega\Omega$. In the first case, for further expansion, we have no choice; we must expand $W(B(\Omega\Omega))\Omega$ with B, since we are restricted to expansions that satisfy the 1's rule. The second case, however, is quite different from the first; as with $B(\Omega\Omega)\Omega\Omega$, we again have more than one choice. In particular, we can expand $BB\Omega\Omega\Omega\Omega$ in three ways with W, and all of the three satisfy the 1's rule. Using the equation

$$Wxy = xyy$$

for W, we can expand $BB\Omega\Omega$, $BB\Omega\Omega\Omega$, or $BB\Omega\Omega\Omega\Omega$ by unifying the right side of the equation for W with a term in $BB\Omega\Omega\Omega\Omega$. In other words, we continue to encounter the expected odd behavior of the kernel $\Omega\Omega(\Omega\Omega)$ when compared to the behavior of any of Γ_1 through Γ_5, for example.

At this point in the research, we thus have four promising paths of expansion to explore. If one pursues this line of attack to its logical conclusion—to the point at which one occurrence of Ω remains—one can traverse not just the four paths mentioned so far, but instead one can traverse a larger set of paths, nine of which produce the following results of interest.

$W(W(W(BB)))\Omega$
$W(W(BW(BB)))\Omega$
$W(W(B(BWB)))\Omega$
$W(BW(W(BB)))\Omega$
$W(W(B(BW)(BB)))\Omega$
$W(BW(BW(BB)))\Omega$
$W(BW(B(BWB)))\Omega$
$W(BW(B(BW)(BB)))\Omega$
$W(B(BW(BW))(BB))\Omega$

The other paths one might traverse in search of a possibly useful expression for constructing fixed point combinators—an expression that contains Ω exactly once—clearly seem to be fruitless. For example, by pursuing one of the so-called fruitless paths to its conclusion, one can obtain

$$W(W(BWBB))\Omega,$$

which is obviously very closely related to the first of the nine listed expressions. In fact, the expression in question is reducible to the first of the given nine by applying B to it, which means that the candidate expression is trivially equal to one of the nine we have already designated as interesting.

To avoid the corresponding wasted effort, we can tighten our rules for exploring the paths emanating from $\Omega\Omega(\Omega\Omega)$. We simply add the restriction that every expansion that is applied on the path from $\Omega\Omega(\Omega\Omega)$ to that point at which exactly one Ω remains must involve at least one occurrence of Ω. As a consequence of this rule, one is not allowed, for example, to expand the term $W(BB)$ with B—in other words, one is prevented from applying the expansion that leads to the fruitless result we have just been examining. Summarizing, a violation of the newest restriction rule will merely lead to an expression that is reducible to one of the nine listed expressions, a conjecture that seems to be obviously true. The expansion paths that involve an Ω but that violate the 1's rule, as commented earlier, duplicate one of the nine results; we verified this fact with one of our automated theorem-proving programs by exploring all such

paths.

The next task in our research is to consider each of the nine listed expressions in the attempt to produce a fixed point combinator. Rather than continuing to "practice" the application of various research techniques, we shall instead quickly present the results. We imitate the process that yields Θ_1 from Γ_1, Θ_2 from Γ_2, and the like. With the object of obtaining an expression of the form Θf with Θ expressed solely in terms of B and W, we expand (subject to the 1's rule) each of the nine listed expressions, after replacing Ω by $W(B(Bf))$, and indeed obtain nine fixed point combinators constructible from B and W alone. For each fixed point combinator, the required expansions are, respectively, with B, B, and B—the first to free W from $W(B(Bf))$, and the last two to free the two occurrences of B. Here are the nine fixed point combinators that one constructs.

$B(B(B(W(W(W(BB))))W)B)B$
$B(B(B(W(W(W(BW(BB))))W)B)B$
$B(B(B(W(W(W(B(BWB))))W)B)B$
$B(B(B(W(W(BW(W(BB))))W)B)B$
$B(B(B(W(W(W(B(BW)(BB))))W)B)B$
$B(B(B(W(BW(BW(BB))))W)B)B$
$B(B(B(W(BW(W(B(BWB))))W)B)B$
$B(B(B(W(BW(B(BW)(BB))))W)B)B$
$B(B(B(W(B(BW(BW))(BB)))W)B)B$

If we now collect and condense the results obtained by our study of $\Omega\Omega(\Omega\Omega)$, we find that our strategy for directing our search for fixed point combinators is indeed effective. Starting with $\Omega\Omega(\Omega\Omega)$, we have succeeded in constructing—rather than the single fixed point combinator that one might have expected if one extrapolated from Γ_1 through Γ_5—one fixed point combinator of length 11, three of length 12, three of length 13, and two of length 14. Even further, we have succeeded within our standard constraints, those we used to construct Θ_1 through Θ_5, and those that could have been used to construct Θ_1^9 through Θ_5^9.

In particular, our approach consists of beginning with a kernel, such as $\Omega\Omega(\Omega\Omega)$, and focusing on its (so to speak) generator, in this case Ω. We then apply expansions, all of which satisfy the 1's rule and involve a term containing at least one occurrence of the generator—Ω is treated in effect as a constant during this phase. We continue expanding until the generator occurs precisely once. At that point, we replace the generator by its definition and apply additional expansions, all of which satisfy the 1's rule, and all of which involve a term containing the Skolem constant f—from the viewpoint of combinatory logic, containing the arbitrarily chosen combinator f. If we do not insist that every expansion applied at this point satisfy the 1's rule, as with Γ_1 and Θ_2 through Θ_5, we obtain fixed point combinators that are obtainable from kernels derivable from $\Omega\Omega(\Omega\Omega)$. If we do not insist that every expansion involve a term containing f, then we obtain fixed point combinators that are trivially reducible to ones we obtain within our restrictions; extensionality is not relevant to this remark. We stop expanding when we obtain Θf, where Θ contains only combinators from the set P that we are studying. For our current discussion—and perhaps in the vast majority of cases—this approach produces an irreducible fixed point combinator.

We must exercise some caution when we say the majority of cases, because of the existence of the fixed point combinator $LO(LO)$ which can be found with an extension of the kernel method that allows expansions into terms that do not contain an occurrence of f. This combinator is not irreducible, in fact, does not have a normal form. In particular, if one attempts to reduce $LO(LO)$—in the terms of automated theorem proving, if one attempts to demodulate this expression—one obtains $O(LO(LO))$ after one reduction with L. Obviously, an attempt to reach a termination point, a point at which no reductions apply, must fail. The fault does not lie with the presence of a replicator—in this case, L—for the fixed point combinators we succeeded in constructing from B and W each contain an occurrence of a replicator, and all are irreducible.

More important than the preceding remarks—at least with regard to the goal of constructing so-called missing combinators—our approach leads to filling in one of the gaps regarding combinators of various lengths; in particular, we have succeeded in constructing combinators of length 13. Of course, we also found additional combinators of length 11, 12, and 14. If we then return to our earlier study of $\Omega\Omega\Omega$ and continue to imitate that study, we are left with two interesting questions with regard to $\Omega\Omega(\Omega\Omega)$. First, can we produce other kernels from $\Omega\Omega(\Omega\Omega)$ by global expansion as we produced Γ_2 through Γ_5 from Γ_1, even though $\Omega\Omega(\Omega\Omega)$ is itself not a superkernel? Second, what are the properties of the mapping from the sets of fixed point combinators to the new kernels (if any) from which they evolve? In addition, one might wonder what would occur if we had allowed Ω to be replaced by its definition before that point at which one Ω remains, and then applied one or more expansions to the result. Finally, one might wonder about possible invariants for the various classes of combinator in which only B and W appear.

As for the first question, one might guess its answer by quickly considering the relation that each of Γ_2 through Γ_5 has to Γ_1, and then noting that the generator of Γ_1 is Ω, which is also the generator of $\Omega\Omega(\Omega\Omega)$. Making such guesses is one of the research techniques that we sometimes employ, and we suggest this action because of our continued objective of showing how research can be conducted. The most obvious guess is correct; if one globally expands $\Omega\Omega(\Omega\Omega)$ with B and W along all possible paths—expanding all four occurrences of Ω simultaneously in the same fashion—one obtains four useful kernels. Of course—just as with the global expansion of $\Gamma_1 = \Omega\Omega\Omega$—in addition to the four useful kernels, other kernels can be obtained. For example, $\Omega = W(B(Bf))$ can be expanded to $W(BBBf)$—which generates a useful kernel—and then further expanded to $W(WBBf)$, which generates a kernel that is not useful. The reason that the latter result is not useful is that it trivially reduces to the former, and leads to a fixed point combinator that trivially reduces to one that is obtainable with the kernel generated by $W(BBBf)$.

We thus find that one can profitably imitate the approach that succeeded in the earlier study of $\Omega\Omega\Omega$, the difference being that one expands four occurrences of Ω at this point, in contrast to expanding three in the earlier study. To summarize the results, one again obtains five expressions that, respectively, generate five kernels. Here are the five useful generators—including that which prompted pursuit of this path—that one obtains.

$\Omega_1 = W(B(Bf))$
$\Omega_2 = W(BBBf)$
$\Omega_3 = BWB(Bf)$

$\Omega_4 = BW(BBB)f$

$\Omega_5 = B(BWB)Bf$

The five kernels are simply of the form $\Delta\Delta(\Delta\Delta)$, where Δ ranges over the five generators Ω_1 through Ω_5. Of equal interest, we can obtain these kernels even though $\Omega\Omega(\Omega\Omega)$ is not a superkernel.

With regard to the second question, that concerned with the properties of the mapping from sets of combinators to their respective kernels, we can apply to the four new kernels extracted from $\Omega\Omega(\Omega\Omega)$ the approach that we applied to $\Omega\Omega(\Omega\Omega)$ itself. The remarks we made earlier concerning the paths and restrictions regarding the construction of the nine combinators from $\Omega\Omega(\Omega\Omega)$ hold here as well. From each of the four kernels, we obtain nine fixed point combinators. Therefore, in contrast to the one-to-one mapping from Θ_1 through Θ_5 to their respective kernels Γ_1 through Γ_5, here we have a nine-to-one mapping from sets of combinators to their respective kernels. For each of the five kernels related to and including $\Omega\Omega(\Omega\Omega)$, the corresponding nine fixed point combinators are all obtained by applying expansions with B and W that satisfy the 1's rule. In other words, if one wishes to find the kernel that corresponds to one of the 45 fixed point combinators obtained from $\Omega\Omega(\Omega\Omega)$ and the four kernels derived from it, one merely affixes f and applies reductions satisfying the 1's rule. Paths exist within this restriction that lead from nine of the 45 combinators to $\Omega\Omega(\Omega\Omega)$, in contrast to the single path from Θ_1 to Γ_1.

If one does not impose the 1's rule restriction and instead affixes f and reduces along an arbitrary path starting with one of the nine combinators in a set, one can arrive at a kernel that does not correspond to the combinator under consideration. For example, as with

$\Theta_4 = B(WW)(BW(BBB))$,

one could start with one of the nine combinators obtained from and corresponding to $\Delta\Delta(\Delta\Delta)$ with $\Delta = BW(BBBf)$, reduce with B satisfying the 1's rule, and then reduce with B again but violating the 1's rule. In this way, one would arrive at $\Omega\Omega(\Omega\Omega)$—just as one could begin with Θ_4 and arrive at Γ_1, which is not the kernel corresponding to Θ_4 under the mapping we have discussed—and would fail to arrive at the appropriate kernel.

Next, within a given set of nine combinators, each of the members (if one ignores the association of symbols) has the same tail or end—specifically, all of the symbols of the corresponding kernel in order—with the constant f, of course, excluded. The beginning of each combinator, within a set of nine, consists of the number of B's needed to free f from the rest of the generator of the corresponding kernel. The order of the symbols in the middle of the members of a set of nine combinators reflects the choice of expansion path traversed on the way from $\Delta\Delta(\Delta\Delta)$ to the point at which only one occurrence of Δ remains, where Δ is, respectively, Ω_1 through Ω_5.

Somewhat hidden in the last remark is the question about what would have occurred had we replaced Ω—perhaps all occurrences of Ω—by its definition, and then expanded the result. On the one hand, had we replaced all occurrences of Ω by its definition $W(B(Bf))$ and then globally expanded the result, the interesting results we would have obtained would merely duplicate what we constructed from the other four kernels obtained from $\Omega\Omega(\Omega\Omega)$. To see that this is true, one might re-examine the derivation of Γ_2 through Γ_5 from Γ_1 and consider the combinators Θ_1 through Θ_5. On the other hand, had we replaced one or more Ω's by its definition but not globally expanded the result—not treated like terms in

the identical manner—we would simply be unable to complete the construction of a fixed point combinator.

Finally, regarding invariants, first note that Θ_1 through Θ_5 each contain three occurrences of W. One of the occurrences results from the W that appears in the corresponding kernel, and the other two are present because of the need to remove two of the three occurrences of Ω in $\Omega\Omega\Omega$. Then note that W occurs four times in each of the 45 combinators constructed from $\Omega\Omega(\Omega\Omega)$ and the four kernels derived from $\Omega\Omega(\Omega\Omega)$. One occurrence of W results from its existence in the corresponding kernel, and three result from the need to remove three of the four Ω's, or its equivalent, in the corresponding kernel. As for the number of occurrences of B, $\Omega\Omega\Omega$ and its four derived kernels behave differently from $\Omega\Omega(\Omega\Omega)$ and its four derived kernels. For Θ_1 through Θ_5, B occurs five times in each. For the fixed point combinators constructed from $\Omega\Omega(\Omega\Omega)$ and its derived kernels, the number of B's varies depending on the expansion path that was taken to reach the point at which one Ω, or its equivalent, remains.

If one wishes to continue along the obvious line of inquiry and desires to find a set of fixed point combinators constructible from B and W alone such that all (or most) members of the set contain five occurrences of W, then six, and more, a way exists to satisfy that desire. Let us immediately show how one can find such a set, omitting many of the details that we have supplied for our earlier studies. However, as we proceed, one must be prepared for a change. The number of occurrences of W may not be the same for all members in a set of fixed point combinators constructed from a given kernel.

Finding the Desired Sets of Combinators

To find the desired sets of combinators, we simply return to the study of $\Omega\Omega z$. If one instantiates z to $\Omega\Omega\Omega$, one obtains the kernel $\Omega\Omega(\Omega\Omega\Omega)$. One can then apply to $\Omega\Omega(\Omega\Omega\Omega)$ the procedure we used when studying $\Omega\Omega(\Omega\Omega)$, and construct a set of fixed point combinators most of which contain exactly five occurrences of W. Those that are constructed with this approach but that do not contain five occurrences of W contain four occurrences. For example, there exists an expansion path focusing on $\Omega\Omega(\Omega\Omega\Omega)$ such that a single expansion with W removes two occurrences of Ω. Here is the initial segment of such a path.

$\Omega\Omega(\Omega\Omega\Omega)$
$B(\Omega\Omega)(\Omega\Omega)\Omega$
$WB(\Omega\Omega)\Omega$
$B(WB)\Omega\Omega\Omega$

One can expand this last expression with W in two ways, obtaining the following two expressions.

$W(W(B(WB)))\Omega$
$W(BW(B(WB)))\Omega$

To complete the construction, one simply expands each of the two expressions with B three times.

To construct additional fixed point combinators containing exactly four occurrences of W such that the kernel to which they correspond contains five occurrences of W, one globally expands $\Omega\Omega(\Omega\Omega\Omega)$ to derive four new kernels. These new kernels are related to $\Omega\Omega(\Omega\Omega\Omega)$ as the four kernels derived from $\Omega\Omega(\Omega\Omega)$ are related to $\Omega\Omega(\Omega\Omega)$. If one explores the other expansion paths that begin with $\Omega\Omega(\Omega\Omega\Omega)$, it

appears that all of the fixed point combinators that result contain exactly five occurrences of W.

One encounters additional contrasting results if one replaces the study of $\Omega\Omega(\Omega\Omega\Omega)$ with a study of $\Omega\Omega(\Omega(\Omega\Omega))$. Although as expected one can again construct fixed point combinators, where $\Omega\Omega(\Omega\Omega\Omega)$ leads to combinators not all of which contain the same number of occurrences of W, it appears that all combinators obtained from $\Omega\Omega(\Omega(\Omega\Omega))$ contain exactly five occurrences of W.

To construct, from B and W alone, fixed point combinators containing six occurrences of W—as one might immediately guess—one studies $\Omega\Omega(\Omega\Omega(\Omega\Omega))$, obtained by instantiating z of $\Omega\Omega z$ to $\Omega\Omega(\Omega\Omega)$. The picture continues to change. Where most of the combinators one obtains from $\Omega\Omega(\Omega\Omega(\Omega\Omega))$ do indeed contain six occurrences of W, some contain five and some four occurrences. The number depends on how many times—zero, one, or two—a single expansion with W removes two occurrences of Ω. Similarly, for certain instances of $\Omega\Omega z$ that contain seven occurrences of Ω, a single expansion with W can remove three occurrences of Ω, resulting in a corresponding deficit in the number of occurrences of W in the fixed point combinator(s) one constructs from such a kernel. The generalization of this remark holds for the appropriate instances of $\Omega\Omega z$ containing eight, nine, and more occurrences of Ω.

A glance at the full set of variable-free instances of $\Omega\Omega z$ in which each instance contains exactly six occurrences of Ω suggests that the association of the Ω-expression that replaces z plays a vital role in determining which properties the corresponding sets of fixed point combinators will have. Since we have not made a study of this set of instances, we can only conjecture that the cardinality of the combinator set may vary as the focus shifts from one instance to another, and that, with regard to the lengths of the combinators in a set, the results will also vary.

As for fixed point combinators containing seven, eight, and more occurrences of W—although we have not attempted to supply a formal proof—it seems clear that they can be constructed from a kernel that is obtained by choosing an appropriate, variable-free instance of $\Omega\Omega z$. Specifically, all that is required is that the instance contain seven, eight, and, in general, the number of occurrences of Ω that equals the number of occurrences of W one wishes.

Invariants of Sets of Combinators

Even further, z can be instantiated to any association of Ω's to yield a kernel from which one can derive various other kernels. From each of the kernels, one can construct a set of fixed point combinators. For most members of a given set, the number of occurrences of W is equal to the number of occurrences of Ω in the chosen instance of $\Omega\Omega z$. In other words, if we ignore the exceptional combinators, the number of W's is an invariant of the set of sets constructed from the corresponding instance of $\Omega\Omega z$ and its derived kernels.

For a set of such sets, the number of occurrences of B is not a constant, for it depends on the expansion path that is taken to produce an expression containing exactly one generator of the kernel. For each variable-free instance of $\Omega\Omega z$, it may be that all sets within the corresponding set of sets of fixed point combinators contain the same number of elements; if so, each of the kernels in that part of the space leads to the construction of the same number of combinators, and we have another invariant.

Finally, we have the invariant pertaining to the tail or end of each combinator, if we partition the combinators based on the generator of the corresponding variable-free instance of $\Omega\Omega z$. In particular, if

we call the generator Δ, then the tail of all fixed point combinators obtained from any kernel generated by Δ consists of the symbols in the order that they occur in Δ, excluding the constant f. For example, all fixed point combinators constructed from $\Omega\Omega\Omega$, from $\Omega\Omega(\Omega\Omega)$, from $\Omega\Omega(\Omega\Omega\Omega)$, and from $\Omega\Omega(\Omega\Omega(\Omega\Omega))$ end with W, B, B since $\Omega = W(B(Bf))$. On the other hand, if we consider a variable-free instance of $\Omega\Omega z$ such that the term replacing z is not an expression purely in terms of Ω, then, from B and W alone, one may fail to construct a fixed point combinator.

Completing Our Search for Fixed Point Combinators Constructible from *B* and *W*

We can complete our search for combinators constructible from B and W alone by noting that, if we take actions corresponding to those we took when studying $\Omega\Omega\Omega$ and apply them to other instances of $\Omega\Omega z$, we can obtain two other sets of sets of fixed point combinators. For the first set, we proceed as we did when constructing combinators of length 9 that are companions to the five (Θ_1 through Θ_5) of length 8. We merely choose an instance of $\Omega\Omega z$ such that the replacement term is an expression purely in terms of Ω and replace W by BW in all occurrences of the generator of the corresponding pseudo-superkernel, derive the related kernels with the global expansion rule, and then construct the corresponding set of fixed point combinators by traversing the various expansion paths.

For the second set of sets, we proceed as we did when obtaining combinators of length 11, 14, 17, and the like from the superkernel $\Gamma_1 = \Omega\Omega\Omega$. For the first set in the set we are constructing, we merely choose an instance of $\Omega\Omega z$ such that the replacement term for z is an expression purely in terms of Ω and insert a B to the right of W in all occurrences of the generator, requiring that the resulting generator be fully right associated. The result is a pseudo-superkernel from which we can derive related kernels and then construct appropriate combinators. By repeating the process—inserting yet another B to the right of each W and right associating—we obtain the next set in this set of sets. The remaining sets are obtained similarly. Summarizing, we can find combinators of any length greater than or equal to 8, with the exception of 10.

Two Promised Proofs

We are left with the task of supplying two promised proofs: first, no irreducible fixed point combinators exist that contain 7 or fewer symbols and that are constructible from B and W alone; second, there exist precisely five irreducible combinators of length 8 that are constructible from B and W alone. For both proofs, we relied on the use of one of our automated theorem-proving programs and a brute-force approach. Specifically, we had the program examine all possible irreducible combinators Θ through length 9, where each Θ consists of some association of the combinators B and W. Our approach consisted of affixing the constant f to a candidate combinator Θ, and then reducing with B and W to see whether we could obtain an expression with a leading f that is isolated.

The justification for this approach rests with the fact that every irreducible fixed point combinator Θ with f appended must reduce to $f\Delta$ for some Δ. After all, if Θ is a fixed point combinator, then, by definition,

$$\Theta x = x(\Theta x)$$

for all combinators x and, therefore,

$\Theta f = f(\Theta f)$

for an arbitrary combinator f that does not necessarily involve B or W. By Church-Rosser, there must exist a Δ' such that both Θf and $f(\Theta f)$ reduce to Δ'. But, since no reduction with B or W can involve the leading f in $f(\Theta f)$, Δ' must be of the form $f\Delta$ for some Δ.

Of all such expressions Θf, none with Θ of length less than 8 proved suitable, five (Θ_1 through Θ_5) with Θ of length 8 were accepted, and five (Θ_1^9 through Θ_5^9) with Θ of length 9 also were accepted. The consideration of the irreducible Θ of length 10, by affixing f to Θ, also showed that no irreducible fixed point combinators of that length exist. Although this brute-force approach is far from elegant, it does demonstrate the value of access to an appropriate theorem-proving program. Of course, we would be delighted to find—or, for that matter, to be given—an elegant proof showing that no irreducible fixed point combinators of length less than 8 can be constructed, exactly five of lengths 8 and 9 exist, and none of length 10 exist.

We can restrict our attention to irreducible combinators Θ since, if Θ' is a fixed point combinator constructible from B and W alone, then Θ' must have a normal form—Θ' must reduce to some irreducible Θ. The justification for this assertion rests with the fact that $\Theta' f$ must reduce to $f\Delta$ for some Δ, and there must exist, therefore, a standard reduction of $\Theta' f$ to $f\Delta$. But, by the nature of B and W, such a standard reduction must systematically consider every symbol of Θ' in order until f is isolated—must contain a step reducing a term that begins with a symbol of Θ' for every such symbol. If Θ' did not reduce to some irreducible Θ, then the standard reduction under discussion would never terminate with the isolation of f, which it must.

A Review of the Exploration of the *B*-and-*W* Mine

We close our study of the set P consisting of B and W by briefly reviewing the entire picture, and by making a few observations. The most important point to note is that the set consisting of the combinators B and W does indeed possess the strong fixed point property—one is able to construct from B and W alone a wide variety of fixed point combinators. When we were told of Statman's discovery of the combinator that we call Θ_4—a discovery that answered a question posed by Smullyan—and began our study of B and W, we had no idea of how powerful the two combinators are when considered together. In fact, from what we know now, no one would have guessed the true nature of things.

As we commented earlier, Smullyan seemed surprised and pleased to discover that—because of our successful construction of Θ_1 through Θ_5—Θ_4 is not the only solution to the problem of constructing, from B and W alone, a fixed point combinator. However, his pleasure—and, even more, our elation—was more than tempered by the fact that Θ_1 through Θ_5 are provably equal in the presence of extensionality. As it turned out, our disappointment was dwarfed by our reaction to the results obtained when we completed our study. Rather than merely finding Θ_1 through Θ_5, or finding an infinite set of combinators of the desired type, as we have shown, we in fact found an infinite class of infinite sets of the combinators we were seeking. We also found, within the infinite set $\Theta(BW)$, an infinite subset such that no two of its members are equal even in the presence of extensionality.

Perhaps more important, we found a uniform method for constructing, from a given set of combinators, fixed point combinators whenever such a construction is possible. Or, at least, that is how it appears

at this time—we have not as yet succeeded in proving that our method is complete. Our method, called the kernel method, is as far as we know the first systematic method for searching for fixed point combinators—equivalently, the first systematic method for showing, when it is true, that a given set P of combinators possesses the strong fixed point property. In addition, the kernel method appears to be the first systematic method for proving, when it is true, that the weak fixed point property holds for some given set P.

The kernel method, as we have shown, meshes extremely well with our long-term goal of automating various aspects of research in mathematics and logic—our goal of sharply expanding the uses of an automated theorem-proving program. However, even without access to a computer program, one can easily apply the kernel method in many cases. We plan to apply the method to the study of various sets P to show which have the strong fixed point property and to find, for sets that do possess the property, shorter fixed point combinators than those already known.

For sets P for which the strong fixed point property does not hold, we can use the kernel method, with high probability, for their identification even though it is not designed for that purpose. In particular, if a search for kernels constructible from the elements of some given set P yields none after a few seconds of use by a computer or a few minutes by a researcher, one can reasonably conjecture that P lacks the property. Of course, one must then supply a proof, but, as is well known in research, being disposed to follow a single path of inquiry is often what is needed to enable one to find a sought-after answer. In other words, if one can drop the consideration of attempting to construct a fixed point combinator and, instead, concentrate on trying to find a proof that no such construction is possible, then the likelihood of success is usually increased sharply. We have a similar but cloudier situation when the kernel method finds some kernels, further effort leads to the discovery of no additional kernels, and the program fails after a substantial effort to construct from any of the kernels it has found a fixed point combinator. Again, one can reasonably conjecture that the strong fixed point property does not hold for the set P under study, and one can concentrate on finding a proof to that effect.

The kernel method is also of value for investigating the presence of the weak fixed point property when given some set P of combinators. When one finds a kernel for P, one has succeeded in proving that the weak fixed point property holds for P. When a serious attempt with the method fails, then one can strongly conjecture that the weak fixed point property is lacking. Of course, again, one must then supply a proof.

In short, the kernel method may prove to be an important aid for attacking many problems of diverse types. By quickly glancing at the four sets P visited in the introduction—B and W, B and L, S and W, and S and L—we have a good illustration of how effective this new method is in the context of the weak and the strong fixed point properties. When the kernel method is applied to the set P consisting of B and W, the first kernel discovered in stage 1 of the two-stage version—the kernel equal to the square of $W(Bf)$—immediately settles the issue for the weak fixed point property. To be precise, the form $W(Bx)(W(Bx))$ of the kernel can easily be shown to be a y satisfying

$$y = xy$$

for all x; one merely reduces this form of the kernel with W, and then with B. With regard to the question of the strong fixed property holding for P, $W(Bx)(W(Bx))$ provides no clues. Instead, one must focus on

the next kernel, the cube of $W(B(Bf))$—equivalently, the cube of $W(B(Bx))$—to settle the question. With this kernel, one finds Θ_1 by applying expansions satisfying the 1's rule, showing that the strong fixed point property holds for P. Even further, the first stage of the two-stage version of the kernel method actually produces the kernel schema $\Omega\Omega z$ with $\Omega = W(B(Bf))$—equivalently, $\Omega\Omega z$ with $\Omega = W(B(Bx))$. Focusing on this schema and its variable-free instances leads to the infinite class of infinite sets of fixed point combinators constructible from B and W alone, the class that establishes that B and W are indeed a powerful combination of combinators.

If the two-stage version of the kernel method is then applied to the second set P, that consisting of B and L, its first stage yields the kernel $Lf(Lf)$. By setting y equal to $Lx(Lx)$ and applying a one-step reduction with L, one sees that

$$y = xy$$

for all x, and, as expected, y does depend on x. In other words, one has a quick proof that the weak fixed point property holds for this second set P. The attempt by the second stage of the two-stage version of the kernel method to construct a fixed point combinator from B and L by using the kernel $Lf(Lf)$ fails. The attempt to find other kernels at the first stage or to apply global expansions to $Lf(Lf)$—which turns out to be a superkernel—to find other kernels also fails. This discovery naturally leads to the conjecture that the strong fixed point property does not hold for the set P consisting of B and L alone. Of course, since we do not know for certain that our approach to finding kernels is complete nor do we know that to every fixed point combinator there must correspond a kernel, we would still have more work to do. The situation is further complicated by the fact that B is an isolator and L is a replicator, and we know that both types of combinator—although not sufficient for the desired construction—are needed for constructing a fixed point combinator. Nevertheless, because we are so fond of the kernel method and do in fact believe it to be complete, we would in a situation like this recommend attempting to prove that P lacks the strong fixed point property, which we succeeded in doing with Theorem 6 of Section 4.6.

When the kernel method is applied to the third set P, that consisting of S and W, no kernels are found at the first stage, suggesting that neither the weak nor the strong fixed point property holds. A glance at S and W—even if the kernel method is ignored—is in fact sufficient to show that the two properties fail to hold; after all, neither combinator is an isolator, and—for one of the two properties to hold—an isolator must be available for the required construction.

Finally, we come to the fourth set P, that consisting of S and L. Since consideration by the kernel method of the set consisting of B and L shows that the presence of L alone is sufficient for proving that the weak fixed point property holds for any set of combinators for which L is an element, we know immediately that this fourth set P possesses the weak fixed point property. In contrast to the failure when attempting to use the kernel $Lf(Lf)$ to construct fixed point combinators from B and W alone, the attempt to complete such a construction with S and L succeeds. The result is the discovery of SLL, which was known to be a fixed point combinator long before the kernel method existed.

However, in addition to verifying known results, application of the kernel method to the study of this fourth set P shows how poor the combination of S and L is when compared to the richness of the combination of B and W. In particular, from what we have found so far—for which a formal proof is lacking, which suggests a possibly interesting area of research—SLL is the only fixed point combinator

constructible from S and L alone, in contrast to the infinite class of infinite sets of fixed point combinators constructible from B and W alone. Especially for those who are new to combinatory logic—as we certainly are—this result is not a result that one would necessarily expect. After all, both S and L are replicators, where B is not. In addition, and perhaps more important, S is so powerful that, if K is adjoined to S, one has a complete set of combinators for combinatory logic. One might, therefore, expect that if S and L together are sufficient for the strong fixed point property to hold, and if the set consisting of B and W also possesses that property, then the richness would rest with S and L rather than with B and W. Indeed, B and W are so powerful that, among the infinite number of infinite sets of fixed point combinators that one can construct from B and W alone, one can find an infinite set such that no two are equal even when extensionality is present.

As an aside, for those who are curious about the possible uniqueness of the set consisting of B and W with respect to richness, a second infinite set of fixed point combinators can be constructed if one studies the set P consisting of S, C, and B. We even have a third example of such richness, an example that focuses on the combinator N, a combinator that Smullyan has not studied. The combinator N with

$$Nxyz = xzyz$$

when considered with B leads to an infinite set of fixed point combinators, and a set that is somewhat reminiscent of $\Theta(BW)$. Because of other discoveries concerning the power of N—the use of the combinator N seems to sharply increase the likelihood of finding gold when exploring a randomly selected mine—we intend to stake out our claim in earnest by studying N intensely. For a fourth example of a rich mine, one can study the set P of combinators consisting of B and H. We present, in Section 5.3.5, some of the results obtained by our exploration of these various mines.

If we take the various observations made in this review into account, perhaps the simplest point we can make is that combinatory logic is most intriguing and, because of its deep nature, quite unpredictable. The kernel method may prove to be an important aid in unraveling the mysteries of this important field of logic and mathematics.

5. Highlights

In this section, we cover the highlights of our research, dividing the material into four major subsections (5.2 through 5.5) focusing, respectively, on the impetus for our research, on what we discovered, on questions that remain unanswered, and on conjectures that we suspect are valid and that we intend to study. To permit one to briskly visit the various high points, we insert Section 5.1 where we simply summarize the most significant aspects that are more fully developed in the material that follows. Therefore, almost of necessity, the material in Section 5.1 will take the form of a series of somewhat disconnected headlines, but will include references to the appropriate subsection(s).

At the beginning of each of Sections 5.3 through 5.5—whose titles reflect their respective content—we shall again briefly highlight the more significant aspects covered there. Finally, so that one can decide whether or not to read the material covered in a particular subsection of one of the major sections, we include at the beginning of each a summary of the significant points covered in that subsection. In view of the organization of this entire section (Section 5), various topics will be visited more than once and, for convenience, certain observations will be repeated.

5.1. Summary of Highlights

The most satisfying result of our research concerns our introduction of the concepts of *kernel* and *superkernel*, and our formulation of the systematic and uniform method, called the *kernel method*, which relies on those two concepts to search for fixed point combinators. Where f is an arbitrarily chosen combinator—from a syntactic perspective—a kernel with respect to a given set P of combinators is an expression Γ such that Γ reduces to $f\Gamma$ by applying elements from the set P. A superkernel is a kernel Γ such that exactly one reduction can be applied to Γ, and that reduction satisfies the *1's rule*—the term to which the reduction applies contains the first symbol of Γ. For a fuller discussion of the concepts of kernel and superkernel, see Section 5.3.2; for examples of each, see Sections 5.3.3 through 5.3.5; for the preferred and semantic treatment of these concepts, see Section 4.5.

The power of the kernel method is demonstrated by its successful use in answering various interesting questions from combinatory logic (see Sections 5.3.3 through 5.3.5). The questions focus on the strong and on the weak fixed point properties. What makes the kernel method especially appealing is that each of the questions we have considered can be answered by our automated theorem-proving program OTTER in less than one CPU minute of computer time. We say "can be answered" to avoid any confusion with our earlier remarks about how we originally obtained some of our answers, before we had formulated the kernel method and before we had access to our new automated theorem-proving program. In contrast, if one were to avoid the use of the kernel method, to answer many of these questions might require orders of magnitude more effort.

With the kernel method, where B and W satisfy the equations

$$Bxyz = x(yz)$$

and

$$Wxy = xyy,$$

one can prove that the set P of combinators consisting of B and W alone offers far more richness than was evident even as late as 1986 (see Section 5.3.3). Specifically, one can prove that an infinite set $\Theta(BW)$ of irreducible fixed point combinators can be constructed from P such that $\Theta(BW)$ contains an infinite subset with the interesting property that no two of its elements are equal even in the presence of extensionality. Even further, by continuing to apply the kernel method to P, one can construct an infinite class of infinite sets of irreducible fixed point combinators such that each set in the infinite class shares the cited interesting property with $\Theta(BW)$. The combinators in $\Theta(BW)$ can be enumerated in such a way that the enumeration is based on the ways to associate the symbols in the respective generators of each of the superkernels from which $\Theta(BW)$ is directly and indirectly obtained. No fixed point combinators can be constructed from B and W alone whose length is less than 8. From B and W alone, one can construct exactly 5 irreducible fixed point combinators of length 8, 5 of length 9, and none of length 10; for any length greater than or equal to 11, one can construct more than 14. Section 5.3.3 contains more detail on the material cited in this paragraph.

The combinator N with

$$Nxyz = xzyz,$$

to which we have so far found no reference in the literature, offers interesting and powerful properties (see

Section 5.3.4). In particular, since

$BN(BB)xyz = xz(yz)$,

which means that $BN(BB)$ acts like the combinator S, the system consisting of N, B, and K is a complete system for combinatory logic. From B and N alone, one can construct an infinite set $\Theta(BN)$ of irreducible fixed point combinators which contains an infinite subset with the property that all of the elements of the subset are distinct even in the presence of extensionality. Therefore, the sets $\Theta(BN)$ and $\Theta(BW)$ share that important property mentioned earlier focusing on richness; of course, these two sets exhibit marked differences also.

The set P consisting of the combinators H and B alone with

$Hxyz = xyzy$

satisfies the strong fixed point property. With one exception, also satisfying that property are the sets obtained from adjoining to the combinator B any one of the regular and noneliminating combinators whose left side of its equation contains exactly three variables, whose right side is fully left associated, and whose right side has the property that exactly one of its variables appears twice. The exception is the combinator N_1 with

$N_1xyz = xyyz$,

which, by Theorem 9 of Section 5.3.6.2, lacks the needed power. Section 5.3.5 contains more detail on the material cited in this paragraph.

To complement the use of the kernel method and to give additional evidence of the value of automated theorem proving for solving problems from combinatory logic, we have adapted some of the techniques of automated theorem proving to prove that certain sets of combinators fail to satisfy various properties that are frequently studied. In particular, if the set P consists of a single regular combinator, or if P consists of the combinators B and L alone, or if P consists of the combinators L and Q alone, then P cannot satisfy the strong fixed point property (see Corollary 3 of Section 5.3.6.2). If the set P consists of S and W alone, then P fails to satisfy the weak fixed point property. If the set P consists of L and N alone, then P satisfies the weak fixed point property—since L alone satisfies the weak fixed point property—but P fails to satisfy the strong fixed point property. Section 5.3.6 contains more detail relevant to the topics of this paragraph.

One can prove that if P satisfies the strong fixed point property, then P must contain a *replicating* and an *isolating* combinator, not necessarily distinct from each other (see Section 5.3.6). A combinator is a replicator if at least one of its variables appears more than once on the right side of its equation. Examples of replicators include

$Wxy = xyy$,
$Lxy = x(yy)$,
$Nxyz = xzyz$,

and the like. A combinator is an isolator if the first variable on the right side of its equation is isolated from the rest of the right side, which may be empty. Examples of isolators include

$Lxy = x(yy)$,

$Qxyz = y(xz)$,
$Mx = xx$,
$Ix = x$,
$Kxy = x$,

and the like. A similar theorem holds when a set satisfies the weak fixed point property even though it fails to satisfy the strong fixed point property.

The kernel method—which appears to be the first systematic method for searching for fixed point combinators—attacks the problem of proving that the strong fixed point property holds for some given set P of combinators by first attempting to prove that the weak fixed point property holds for P. However, if the method does succeed in its first objective, it more than succeeds; in particular, the method achieves its objective by finding a kernel—in other words, by finding a y such that y reduces to xy, which is a stronger result than finding a y that satisfies the equation

$y = xy$,

which is the equation for the weak fixed point property. If the kernel method succeeds in finding a kernel, then it turns to attempting to reach its second objective, that of expanding the kernel, by using the elements of the set P under study, to a fixed point combinator. For a fuller description of the kernel method, its various aspects, and both the two-stage and the three-stage version of the method, see Section 5.3.2.

The kernel method thus attacks the problem of proving that the strong fixed point property holds for a given set P of combinators in a fashion that is opposite from what one might expect. In particular, since the presence of the strong fixed point property for a set P of combinators implies the presence of the weak fixed point property and not conversely, at least intuitively one might expect that first proving the weaker of the two properties would not be the preferred order. After all, many situations exist in mathematics and in logic, for example, where proving the weaker of two properties is of no value in proving the stronger of the two. As it turns out, such an intuitive judgment is in error. Indeed, the kernel method derives much of its power precisely because of (so to speak) inverting the importance of the order of the two properties. This point is developed more fully in Section 5.3.2.

We have selected from our successful attempts to construct a fixed point combinator by relying heavily on the kernel method the following fixed point combinators of interest. For each, we shall comment on its significance.

From S and K with

$Sxyz = xz(yz)$

and

$Kxy = x$,

one can construct

$S(K(SS(SKK)(SS(SK))))(S(KS)K)$,

which is of interest because its length, from what we have found, is strictly less than that of any other fixed point combinator constructible from S and K alone.

From B and W^1 with

$Bxyz = x(yz)$

and

$W^1xy = yxx,$

one can construct

$B(B(B(W^1W^1)(BW^1))B)B$
$B(B(W^1W^1)(BW^1))(BBB)$
$B(B(W^1W^1)(B(BW^1)B))B$
$B(W^1W^1)(B(BW^1)(BBB))$
$B(W^1W^1)(B(B(BW^1)B)B)$

as fixed point combinators, which are of interest on their own, but are also interesting because one can obtain from them an important set of five fixed point combinators constructible from B and W alone. Indeed, the five combinators one obtains by uniformly replacing W^1 by W happen to be the only irreducible fixed point combinators constructible from B and W alone that have length 9.

From B and N alone, one can construct the fixed point combinator

$B(B(N(BB(N(BBN)N))N)B)B,$

which is of interest because it is the first we constructed when we turned to the study of these two combinators. We find this combinator interesting also because our program used the kernel method to find it in less than one CPU minute of computer time, and we suspect that its construction would have required far more time if a person were unacquainted with the kernel method. Perhaps more interesting is the fixed point combinator, more precisely, fixed point combinator schema,

$B(B(B(N(BNB)z)N)B)B$

constructible from B and N alone. This expression is of interest because neither B nor N is eliminating, and we are surprised that one can construct, without using eliminators, a fixed point combinator schema in which a variable occurs.

For our final example, we return to a theorem that, if the kernel method is ignored, might require a substantial effort to prove, but is easy to prove with the kernel method. The theorem asserts that the strong fixed point property holds for the set consisting of B and H alone. As usual, our proof rests entirely on supplying an appropriate fixed point combinator—in this case, the combinator

$B(B(H(B(BH)(BB))(H(BH)(BB))H)B)B$

suffices. This fixed point combinator is of interest because one can construct it by starting with the superkernel that is obtained from Γ_1 by simply replacing B by H; Γ_1 is the superkernel that leads directly and indirectly to the construction of the fixed point combinators of length 8 in the study of B and W.

Of the various questions that remain unanswered, one of the more pressing focuses on the completeness of the kernel method with respect to the construction of fixed point combinators. Is it the case that, for every set P of combinators and for every fixed point combinator Θ with respect to P, there exists a kernel Γ with respect to P such that Θf reduces to Γ which in turn reduces to $f\Gamma$, where f is an arbitrarily

chosen combinator? For a related question, if a set P of combinators satisfies the weak fixed point property, does there exist a kernel Γ with respect to P? Then, we have a question raised by Smullyan in private conversation [Smullyan87]: Does the set P consisting of the combinators B and M alone satisfy the strong fixed point property? Finally, a question we raise because of our research: Does the set consisting of B and S alone satisfy the weak fixed point property?

With regard to conjectures, we conjecture that indeed every fixed point combinator has a natural kernel that corresponds to it, a kernel such that the fixed point combinator can be obtained from its kernel by applying expansions with elements of P. We next conjecture that every kernel with respect to a given set P can be found by applying expansions to the right side of the equation

$$y \neq fy,$$

which in turn is obtained by assuming that the weak fixed point property fails to hold for P. To be more precise, the given procedure for finding all kernels must be augmented with the use of instantiation, for we know of kernel schemata that the kernel method finds, schemata such that one is required to replace their variables by appropriate terms to obtain kernels. Next, because of certain research that is still incomplete, we conjecture that the set P of combinators consisting of B and M alone does indeed fail to satisfy the strong fixed point property even though P satisfies the weak fixed point property. To see that P satisfies the weak fixed point property, one need only consider the kernel $M(BfM)$, where f is an arbitrarily chosen combinator. The consideration of this kernel, however, sheds no light on the possible presence or absence of the strong fixed point property for the set consisting of B and M alone, for it cannot be expanded to a fixed point combinator by using B and M (see Section 5.3.6 for an explanation). Finally, we conjecture that the set consisting of B and S alone fails to satisfy the weak fixed point property.

Research, from Zero Almost to Infinity

The title of this section reflects our subjective reaction to 21 months of increasingly rewarding research, research whose results we present throughout this report. For an overview, we began by trying to crush a specific problem with mere power, turned to the use of what we eventually termed simple kernels, were forced to use for broader studies what would be called semisimple kernels, and—to reach our goal of providing a uniform method for constructing (when they exist) fixed point combinators—were required to occasionally employ kernels that are not even semisimple. Then, both to address the problem of proving that no such combinators can be constructed—when such is the case—and to identify those cases for which the method cannot succeed without even submitting such to the kernel method, we borrowed techniques from the theory of automated theorem proving and applied them to combinatory logic. In other words, we have an example of how one's fortune can be made—of how research can begin with virtually no information, progress through various stages of increasing knowledge and access to ever more effective techniques, and culminate with a systematic method, the kernel method, for searching for various desired objects and with a set of theorems that assert, for certain cases, that no search can succeed. To paint a more complete picture of how research can progress and of what actually occurred—especially for those who choose to read this section only—we close this section with a few paragraphs that give the appropriate details.

With regard to research in general, our incomplete study of combinatory logic provides an excellent illustration of how one can use an automated theorem-proving program most profitably. This study also graphically illustrates the course that research can take. In particular, when we were presented with the single problem of constructing from B and W a fixed point combinator—the problem that prompted our entrance into combinatory logic (see Section 5.3.2)—we simply attempted to crush the problem with the power of one of our theorem-proving programs and the might of a large computer. Fortunately, we failed; had we succeeded, we might not have begun the formulation of a more complex attack on the given problem (see Sections 4.1.1 through 4.1.4). As one discovers in this report, our reaction to failure was to formulate additional strategies to augment the various techniques offered by our theorem-proving programs, techniques that include types of strategy, various inference rules, and a procedure for canonicalization. The resulting approach—although primitive when compared to how we now attack problems of searching for fixed point combinators—succeeded in rediscovering Statman's combinator, which we call Θ_4, and four new fixed point combinators, Θ_1 through Θ_3 and Θ_5.

As one might predict, we experienced great delight at making a contribution, admittedly small, to a field that has captured the interest of some of the most famous logicians. In addition, we were pleased to find another impressive example establishing the value of the use of an automated theorem-proving program for research. If one wonders how much we contributed to the discovery, we can say that, although we are accorded by our colleagues the attribute of expert in automated theorem proving, in combinatory logic we are clearly novices—an evaluation that one can accept as accurate, rather than as modest. No doubt more important, at least from the global perspective, than our reaction of delight was the fact that we now had momentum—the drive to attempt to demonstrate more clearly how useful automated theorem proving is.

We therefore chose the imprecise objective of sharply reducing the time required by our theorem-proving program to search for and construct the combinators Θ_1 through Θ_5, the time equivalent to 20 CPU hours on a Sun 3 workstation. After pursuing this objective for three months—focusing almost exclusively on the combinators B and W—we succeeded; we reduced the required CPU time from 20 CPU hours to 8 minutes on a Sun 3 workstation. The credit for this sharp reduction in CPU time goes entirely to the kernel method; in fact, this was our first use of the method. However, we restricted the method to the use of what we would eventually call simple kernels. When we then turned our attention to using the kernel method to the study of combinators other than B and W, we encountered cases for which we discovered we must use what we would eventually call semisimple kernels. Still later, we discovered that we must occasionally use kernels that are not even semisimple, since our goal is to provide a uniform means for addressing the question of constructing fixed point combinators for an arbitrarily chosen set P of combinators.

Even with the kernel method fully developed, a severe obstacle still remains. In part and from a general perspective—although we conjecture that all is in order—the obstacle is the lack of a completeness proof for the kernel method if used in its full generality. Also in part and from a general perspective—since we fear that all is *not* in order—the obstacle is the lack of a decision procedure for determining the presence or absence of the weak and strong fixed point properties.

From a concrete perspective, the obstacle is encountered when the kernel method either fails to find any kernels, or finds one or more kernels but cannot find a way to use one of them to construct a fixed point combinator. Of course, in both cases, we refer to failure after a reasonable amount of time—we consider 1 CPU minute more than reasonable. Although we consider the discovery of no kernels within the first CPU minute of computer time to strongly suggest that the weak fixed point property fails to hold for the set P of combinators under study, we cannot be certain that, if given more CPU time, a kernel might not emerge. The only exception occurs when the theorem-proving program exhausts all paths of expansion, signified by a message that in effect no conclusions are left to be drawn. In that case, because of certain results in automated theorem proving concerning semidecidability with respect to the consideration of an unsatisfiable set of statements, we can say for certain that the weak fixed point property fails to hold. Of course, when the weak fails to hold for the set P, then the strong cannot hold either—the strong fixed point property implies the weak fixed point property.

When the kernel method succeeds in finding one or more kernels, but cannot after a CPU minute of computer time complete the construction to a fixed point combinator, then again we are confident, confident that the strong fixed point property does not hold, and certain that the weak does. Again, we cannot be certain; in particular, there might exist a fixed point combinator that does not reduce to a kernel. Or, there might exist a kernel that corresponds to the sought-after but not constructed fixed point combinator such that the method has not found that kernel. Because of the cited considerations, we are left with the problem of what to do when the kernel method does not succeed in one of its two objectives—at least, we are left with this problem for some, but not all, cases.

As implied in the preceding sentence, we can in fact cope, for some cases, with the situation when the kernel method fails in one of its two objectives. Indeed, in Section 5.3.6, we give a number of theorems that can be used to prove, for sets P of combinators with appropriate but diverse sets of properties, that P cannot satisfy the weak or the strong fixed point property, respectively. Some of those theorems enable us to entirely avoid using the kernel method, substituting instead an immediate proof showing that the method must fail. Of course, as is most likely obvious, the source of the failure rests with the set P of combinators under study, and not with the kernel method itself. For those cases for which one of the theorems applies, we can save our effort and the computer's time and, far more important, can cope with the uncertainty that would ordinarily be present if the kernel method either fails to find a kernel, or fails to use a found kernel to construct a fixed point combinator.

In other words, for most of the cases we have studied, we can win the game—and more important, from the viewpoint of mathematics and logic, prove that we have won—regardless of which side we must take. When, for example, the strong fixed point property is not possessed by a given set P of combinators, we can frequently use one of the theorems of Section 5.3.6 to supply a formal proof to that effect. On the other hand, when, for example, the strong fixed point property is possessed by a set P under study, we can use the kernel method to find an appropriate fixed point combinator for the set P. With the expected warnings about completeness that we have given, we can say that our study has enabled us to realize, at least to a fair extent, the researcher's vision—the vision of penetrating some area to the extent that one can cope often with the positive or the negative side of the corresponding questions.

5.2. The Impetus for Our Research

The sole impetus for our entrance into the field of combinatory logic was no more—but no less—than a straightforward request by our colleague Ross Overbeek. Overbeek asked us to search for a mechanism that an automated theorem-proving program could use to find the same answer that the logician Statman had found when he succeeded in answering a question concerning the strong fixed point property. The (formerly) open question, posed by Raymond Smullyan in his book *To Mock a Mockingbird* [Smullyan84], asks: Can one find a single regular combinator that, when considered with the combinator B, permits one to construct a fixed point combinator? Equivalently, but phrased in the terms we have used throughout this report, does there exist a set P of combinators satisfying the strong fixed point property such that P consists of B and one other regular combinator?

Statman had answered the question in the affirmative [Statman86]; from

$$Bxyz = x(yz)$$

and

$$Wxy = xyy,$$

he succeeded in constructing the combinator

$$B(WW)(BW(BBB)),$$

which we call Θ_4 to reflect its place in our enumeration of such combinators. To make the picture totally clear, Overbeek was not asking us to find a way for a theorem-proving program to verify that Θ_4 is a fixed point combinator—that task, as it turns out, is not much of a challenge for any of our programs. Rather, he asked us to find a way for one of our programs to find Θ_4, starting only with the equations that give the actions of B and W. Little did Overbeek know what a torrent of interest he would set loose, and what would result from that interest—which is in part the topic of this entire section.

What made our colleague's request especially challenging—perhaps piquant, or even amusing, is a better word—is the combination of our extensive knowledge of automated theorem proving and our almost nonexistent knowledge of combinatory logic. The trepidation we might have felt because of our lack of knowledge about this area of logic was dwarfed by our ever-present fascination with attempting to find new techniques, or to adapt existing ones, for attacking problems from a field that had not previously been considered as a target for automated theorem proving. As additional incentive, Overbeek informed us that Smullyan's book, from which the original question was taken, contains a number of potentially interesting problems to consider. The prospect of adding to the type and number of open questions that could be answered by relying heavily on an automated theorem-proving program was irresistible.

To compensate for our lack of knowledge concerning combinatory logic, we were given what turned out to be one crucial item of information. Specifically, we were given Overbeek's proof, obtained with the theorem-proving program ITP, verifying that Θ_4 is indeed a fixed point combinator. The notion was that we might use that proof as somewhere between a catalyst and a guide to formulating a new approach or to extending an existing approach. However—and this point cannot be overstated—we were not to cheat in any way; for example, if we presented an approach that one could justly say was tuned to the specific question by focusing, say, on Θ_4, then we would have failed to fulfill Overbeek's request.

To emphasize and further clarify this point, the temptation is great to judge a piece of research of this type a success merely because of obtaining the desired answer to a given question. For example, if one obtains with a computer program a proof of a known theorem, one should carefully study the technique that was used, in order to estimate how much of the success should be attributed to knowledge of how the known proof proceeds. In sharp contrast is the case in which an open question is answered with the assistance of—perhaps, solely by—a computer program. For that case, the term "cheating" is inappropriate—in effect, has no meaning. However, for duplicating known results, the researcher should demonstrate that the new approach, or the extension of an older approach, exhibits generality rather than specificity to the problem under consideration. We were in effect asked to do likewise.

But, let us dispense with the philosophy of science and experimentation. Instead, let us immediately turn to the results of our research that appear to be the most significant. If any of the cited results pique one's curiosity, answer a question one has considered, or complete one's understanding in some area, then one might keep in mind that much credit goes to our colleague Overbeek for making the original request. In addition, credit goes to Smullyan for having written a marvelous and provocative book, and to Statman for having provided a solution to one of Smullyan's questions. Finally—and this observation predictably gives us great satisfaction—substantial credit goes to our automated theorem-proving programs that were so instrumental in obtaining our results.

5.3. New Results

We shall divide this subsection into further subsections, each focusing on a specific topic; most of the material in this section concerns new results, from what we currently know. Rather than adhering to the typical formal approach, we shall simply discuss the various discoveries in a manner that often exhibits our delight at what we found and our excitement about future research.

5.3.1. The Significant Discoveries

As observed, the most significant contribution we have made to combinatory logic appears to be the introduction of the concepts of *kernel*, *superkernel*, and a systematic method, called the *kernel method*, for searching for fixed point combinators (see Section 5.3.2). From what we know, no other method exists for systematically searching for a proof of the presence of the strong fixed point property when given a set P of combinators; we are currently attempting to verify that fact. The kernel method can also be used for systematically searching for a proof that some given set P of combinators satisfies the weak fixed point property; as far as we know, no other systematic approach exists for such a search. In addition—although only in an indirect way—this method can also be used to suggest that a set P fails to satisfy either the weak or the strong fixed point property.

The kernel method can be applied by a person rather easily, and can be applied by an automated theorem-proving program with virtually no effort. When applied by our newest automated theorem-proving program OTTER, one need allow less than 1 CPU minute before obtaining the results that essentially settle the issue concerning the presence or absence of both the weak and the strong fixed point properties. The effectiveness and value of the kernel method compared to a straightforward approach is often dramatic. For example, when we recently studied, in the context of the strong fixed point property, the set P consisting of B and W^1 with

$Bxyz = x(yz)$

and

$W^1xy = yxx,$

our attempt to determine, without recourse to the use of the kernel method, whether the property is present or absent did not succeed. In contrast, when we instructed OTTER to use the kernel method to study this problem, the program succeeded in proving that the set P does satisfy the strong fixed point property, and—fascinating to us—succeeded in finding a proof in less than 2 CPU seconds. In Sections 5.3.3 through 5.3.6, we present substantial additional evidence that strongly suggests that the kernel method is a powerful aid for studying sets P when the object is to prove that the strong fixed point property or the weak fixed point property is present or absent. The evidence is taken from our successful use of the kernel method for a number of studies.

The remaining important results focus on various specific sets of combinators and answer a number of questions, some of which were open before we attacked them directly and indirectly with the kernel method.

If the set P consists of a single regular combinator, or if P consists of the combinators B and L alone, or if P consists of the combinators L and Q alone, then P cannot satisfy the strong fixed point property. On the other hand, where

$Nxyz = xzyz,$

$Hxyz = xyzy,$

and

$Bxyz = x(yz),$

the set consisting of B and N alone and the set consisting of B and H alone each satisfies the strong fixed point property. From what we have found, the combinator N has not been studied before.

Statman showed—by constructing the combinator we call Θ_4—that the set P consisting of B and W alone also satisfies the strong fixed point property, but "satisfies" is clearly an understatement. Indeed, with the kernel method, we show how one can construct from the elements of P alone an infinite set $\Theta(BW)$ of irreducible fixed point combinators such that there exists an infinite subset of $\Theta(BW)$ with the property that all of its elements are distinct even in the presence of extensionality. Even further, we also show how one can construct an infinite class of infinite sets of irreducible fixed point combinators containing $\Theta(BW)$ as one of its elements, where the construction depends on B and W alone. For the set $\Theta(BW)$ of fixed point combinators, there exists a natural one-to-one onto mapping between its members and the kernels from which they can be constructed. The elements of the infinite set $\Theta(BW)$ can be enumerated, where the rules for the enumeration rest solely with the generators of the superkernels of the infinite sequence of superkernels from which the kernels that correspond to the elements of $\Theta(BW)$ are derived.

If we focus exclusively on $\Theta(BW)$, among its elements we find combinators of length 8, 11, 14, 17, and the like—and no others. If we instead focus on the infinite class of infinite sets, we can give the following summary. No fixed point combinators can be constructed from B and W alone whose length is

less than 8, where the length of an expression is the count of the symbols in the expression, ignoring parentheses. From B and W alone, one can construct exactly 5 irreducible fixed point combinators of length 8, 5 of length 9, and none of length 10; and for any length greater than or equal to 11, one can construct more than 14.

If a set P satisfies the strong fixed point property, then P must contain a replicator and an isolator (for the definitions, see Section 5.1). To obtain a set of combinators that is complete for all of combinatory logic, one can consider the set consisting of N, B, and K alone. The completeness is established by noting that

$$BN(BB)xyz = xz(yz),$$

which says that $BN(BB)$ is an S, and by recalling that the set consisting of S and K alone is complete. If we consider the fact that $BN(BB)$ acts as S does and recall that the set consisting of B and N satisfies the strong fixed point property, and if we then note that the set consisting of S alone fails to satisfy the strong fixed point property—which follows from the fact that no single regular combinator by itself can satisfy that property—we indeed have an excellent example of how complex combinatory logic is. In fact, the set consisting of S alone does not come close to satisfying that property; this set even fails to satisfy the weak fixed point property.

5.3.2. The Kernel Method Revisited

At this point we focus heavily on the kernel method, whose formulation—together with the introduction of the related concepts—may be the most significant result of our research. In this section, the significant points concern the precise details for the use of the kernel method for searching for fixed point combinators, the ease with which the method can be applied, the successes obtained with it regarding the answers to previously open questions, and the remarkably small computer times required to obtain those answers.

Definitions and Framework

To provide a general framework for the material in this section, we repeat the following informal definitions; for the formal treatment, see Section 4.5. A kernel with respect to the set P of combinators is an expression Γ such that Γ reduces to $f\Gamma$, where f is an arbitrarily chosen combinator and where all reductions are with elements of P. In other words, since f is arbitrary, casually speaking, Γ reduces to $x\Gamma$ for all combinators x. If Γ is a kernel—which means that Γ reduces to $f\Gamma$—then, obviously, $\Gamma = f\Gamma$, which in turn implies that Γ satisfies the equation

$$y = xy$$

for the weak fixed point property. Actually, since the reducibility of one expression to another is a stronger condition than the relation of the two being equal, proving that some expression Γ is a kernel is ordinarily more difficult than proving that Γ satisfies the weak fixed point property—equivalently, proving that $\Gamma = f\Gamma$ for an arbitrarily chosen combinator f. However, when one is studying the strong fixed point property, we shall see that attempting to prove that Γ is a kernel rather than simply proving that $\Gamma = f\Gamma$ is the preferred approach.

To complete the framework for the following material, we note that there exist kernels of three types—simple, semisimple, and the rest. Rather than repeating the formal definitions given in Section 4.5, we instead give the following summary of the differences between these three types of kernel. Crucial for the differences is the concept of isolating a term, as, for example, when one says that the expression is reduced until f is isolated. For f to be isolated in an expression, the expression must either consist of f alone, or be of the form $f\Delta$ for some term Δ. A simple kernel Γ is a kernel such that, to reduce each of its elements Γ_f to $f\Gamma_f$, no reductions are required after f isolates. A semisimple kernel Γ is a kernel such that, to reduce each of its elements Γ_f to $f\Gamma_f$, all reductions (if any) that are used after one or more occurrences of f isolate are required to satisfy the pseudo 1's rule, defined in Section 4.5. One can quickly show that every simple kernel is a semisimple kernel since no reductions occur after f isolates. Therefore, three types of kernel exist—simple, semisimple, and those that are neither.

To apply the kernel method, one starts with a set P of combinators for which one wishes to determine whether or not P satisfies the strong—or, in some cases, the weak—fixed point property. For either type of problem, in addition to the equations that give the actions of the elements of P and the axiom of reflexivity, one also includes the inequality that corresponds to assuming that the weak fixed point property fails to hold for P. For most studies, the kernel method can be applied rather easily by hand, and very easily by an automated theorem-proving program. One can apply a two-stage version of the method or a three-stage version. Before we turn to a discussion of the two versions, let us give an overview of how the kernel method works and why it is so successful.

Overview of the Kernel Method

The kernel method approaches the question of attempting to prove that the strong fixed point property holds for a given set P of combinators in a manner that might be termed opposite from what is the most natural approach. The most natural approach—for a mathematician or logician, or for an automated theorem-proving program—to proving that the strong fixed point property holds is to assume that it does not, and then seek a proof by contradiction. In contrast, the kernel method approaches the problem by first focusing heavily on attempting to prove that the weak fixed point property holds for P; if it succeeds, then the kernel method attempts to extend the proof to one that shows that the strong fixed point property holds for P.

However, for the first objective, the method does not simply attempt to find a y such that

$$y = xy.$$

Rather, the method first seeks to find a kernel—a y such that y reduces to xy. By first focusing on the weak fixed point property rather than on the strong fixed point property, the kernel method, in the following sense, reverses matters. In particular, the strong fixed point property implies the weak fixed point property, and not conversely. Instead of being hindered by this reversal of focus, the kernel method derives much of its power by this action. By using the notation in which Θ denotes a fixed point combinator, one can quickly see that the strong fixed point property,

$$\Theta x = x(\Theta x),$$

implies the weak fixed point property,

$y = xy$,

by simply setting y to Θx.

The kernel method also derives much of its effectiveness and power from the fact that it focuses, in effect, on only the left side of the equation

$yx = x(yx)$.

Specifically, this method searches for a kernel—an expression Γ that reduces to $f\Gamma$ for an arbitrarily chosen combinator f—and, if one is found, attempts to expand the kernel to the construction of a fixed point combinator. In particular, the method attempts to expand the kernel Γ to an expression of the form Θf with Θ containing no occurrences of f. If it succeeds, then the proof that the set P of combinators under study satisfies the strong fixed point property merely requires certain obvious finishing touches.

By limiting its focus to one side of the equation for the strong fixed point property, the kernel method avoids the increased complexity and potential of going astray that one encounters when focusing on both sides of the equation, as one would ordinarily do when using the typical approach to address the question of proving that the strong fixed point property holds for a given set of combinators. The kernel method also benefits from pursuing a simpler objective, that of first proving that the weak fixed point property holds, which it does in the first stage of the two-stage version. The corresponding search, especially with the strategies we suggest, is far narrower than the general search required by starting with the assumption that the strong fixed point property fails to hold. Since the method, if it does succeed in this phase of its search, then in effect discards all of the false leads, the search from a kernel to the goal of constructing a fixed point combinator—which is the object of the second stage of the two-stage version of the kernel method—also involves a relatively narrow search. In other words, by partitioning the search for a fixed point combinator into the two stages we have just described, the kernel method offers marked advantages over the more direct search that proceeds from assuming that the strong fixed point property is absent.

When we turn our attention in Section 5.3.3 to presenting various results concerning the set P of combinators consisting of B and W alone, we shall discuss a mapping that naturally maps each fixed point combinator to a unique kernel. For certain subdomains of the domain focusing on B and W alone, the mapping is one-to-one; for others, it is many-to-one. For each set P of combinators to be studied, the kernel method in effect relies on such a mapping, tacitly assuming that to every fixed point combinator that one might consider there corresponds a kernel from which it can be constructed. The corresponding conjecture is still open. Although, for various sets P of combinators, many fixed point combinators naturally map to a single kernel—with appropriate restrictions on the expansion path that leads from a kernel in the direction of a fixed point combinator—one can obtain an inverse one-to-one mapping to that which maps fixed point combinators to kernels. Specifically, if the expansions are confined to satisfy the 1's rule, and if one always chooses to expand the shortest term that can be expanded—ignoring those terms that merely lead to a combinator that is uninteresting because it obviously reduces to one that will be found otherwise—then it appears that, if any combinators can be constructed, a unique fixed point combinator will be found as the image of the kernel.

Of course, there exist kernels that do not lead to the construction of a fixed point combinator, which one would predict since there exist sets P that satisfy the weak fixed point property but that do not satisfy the strong fixed point property. For example, consideration of the kernel $Lf(Lf)$, where f is an arbitrarily chosen combinator and L is the only combinator to be used in the attempted construction, cannot lead to a fixed point combinator. One need not abandon such kernels entirely. Instead, one can use other combinators to construct a fixed point combinator whose kernel is still $Lf(Lf)$. One can construct BML or SLL or $N(BBL)L$, for example, each of which naturally has $Lf(Lf)$ as its corresponding kernel. In fact, we often study cases in which the search for a fixed point combinator fails to complete the desired construction, but, by finding one or more kernels, does succeed in establishing that the weak fixed point property holds. Such a discovery often leads us to related studies, studies focusing on adjoining additional combinators.

More Background

We need a little more background before we can focus directly on the two-stage version of the kernel method, and then on the three-stage version. Syntactically speaking, a kernel, with respect to some given set P of combinators, is an expression Γ that reduces to $f\Gamma$ by applying some or all of the elements of P as reductions, where f is an arbitrarily chosen combinator. We could instead say that a kernel is an expression Γ that reduces to $x\Gamma$ for any combinator x if we chose to use the shorthand notation that is used, for example, in group theory when one says that $ex = x$ for the identity element e and for all x. Such a shorthand notation is also used, for example, in ordinary arithmetic when one says that the product of 1 and x is x. Notation of this type is shorthand because one cannot directly consider either ex in group theory or multiply 1 and x in arithmetic; in either case, the appropriate operation is actually defined on pairs of objects in the system, and not on an object and a variable.

The same is true in combinatory logic; one cannot apply one combinator to a variable, but must, instead, apply one combinator to another combinator. Therefore, technically, we must say that, for Γ to be a kernel, Γ reduces to $f\Gamma$ for an arbitrarily chosen combinator f, rather than saying that Γ reduces to $x\Gamma$. Although we have found kernel schemata, every kernel that we have encountered has the property that it is a function of f—or, speaking informally, a function of x. In other words, since f is an arbitrary combinator, any kernel Γ has the property that Γ reduces to $x\Gamma$—if we are permitted to use a shorthand way of speaking as is done in so many areas of algebra, for example—which implies that every kernel Γ has the property that $\Gamma = x\Gamma$.

A simple and familiar (by now) example of a kernel is the expression $Lf(Lf)$ for the combinator L with

$$Lxy = x(yy).$$

By applying L as a reduction, one immediately proves that $Lf(Lf)$ indeed reduces to $f(Lf(Lf))$, which says that $Lf(Lf)$ is actually a simple kernel. As we learned in Section 4.5, a kernel Γ is simple if Γ reduces to $f\Gamma$ without requiring any reductions after f isolates. For a second example of a kernel, we consider UUf for the combinator U with

$$Uxy = y(xxy).$$

Again, a single reduction with U suffices, and we therefore have a second example of a simple kernel. For a slightly more interesting example, we consider the expression $W(Bf)(W(Bf))$ for the combinators B and

W with

$$Bxyz = x(yz)$$

and

$$Wxy = xyy.$$

To prove that this expression is a kernel—indeed, a simple kernel—one must apply as reductions W followed by B. We shall encounter many additional examples of kernels in this section and its subsections, and some of those kernels will be quite intriguing to study.

For example, as a preview, the expression $Bf(W(Bf))(W(Bf))$ is a kernel that is not simple but is, instead, semisimple (see Section 4.5). In particular, to prove that $Bf(W(Bf))(W(Bf))$ is a kernel, one must apply a reduction with W after f isolates. If the set P of combinators consists of B, W, and N with

$$Nxyz = xzyz,$$

then one can use this semisimple kernel, by expanding with the elements of P, to lead to the construction of the fixed point combinator $B(N(BB(NBW))W)B$. We thus have proved—although not in the manner we would usually do so—that P satisfies the strong fixed point property. The proof we would ordinarily give rests on simply citing the fact that the subset of P consisting of B and W alone satisfies the strong fixed point property—which, with the construction of the infinite set $\Theta(BW)$ of fixed point combinators, we have proved in abundance. Of course, if a subset of a set of combinators satisfies the strong fixed point property, then the set itself does also.

As an aside, we did not discover the concept of kernel—which is obviously related to the concept of weak fixed point property—because of any knowledge about the weak fixed point property. Only after we had studied kernels for some time did we recognize the connection to the weak fixed point property, which, ironically, shows that we are indeed novices in the field of combinatory logic. One can quickly see that the concept of a kernel is related to the weak fixed point property by noting that if a kernel Γ reduces to $f\Gamma$, then $\Gamma = f\Gamma$. But one can then use Γ as the y to satisfy for all x there exists a y such that $y = xy$, which is the equation for the weak fixed point property. Because of this observation, one might even talk about the reducible weak fixed point property for a set P of combinators to mean that one can find a kernel Γ for the set P.

The point here is that a kernel is obviously a function of the combinator f that occurs in it, just as the y that satisfies the equation for the weak fixed point property is a function of x. Although we have not yet succeeded in proving that sets that satisfy the weak fixed point property also satisfy the reducible weak fixed point property, we conjecture that such is the case. With the perspective the preceding remarks provides, we can now turn to the details of applying the kernel method for searching for fixed point combinators.

Revisiting the Two-Stage Version

In the two-stage version of the kernel method, the first stage has the objective of finding *kernels* with respect to P, where P is some given set of combinators under study. The first stage begins its search for kernels by expanding the inequality $y \neq fy$ which is obtained from the assumption that the weak fixed point property fails to hold for P, where the expansions are confined to elements of P. The search

continues by expanding the inequalities that are thus obtained. All expansions are applied to the right side only of an inequality. If any kernels are found, then one has a proof that the weak fixed point property holds for P—for all x there exists a y such that $y = xy$.

The second stage of the two-stage kernel method seeks to use the kernels, one at a time, found in the first stage to attempt to construct a fixed point combinator with respect to P. More precisely, the second stage of the method seeks to find expressions of the form Θf by expanding a kernel with the elements of P. The researcher examines the results (if any) that are obtained, and discards those with the property that Θ contains one or more occurrences of f. Of course, if one is using an automated theorem-proving program, then correct use of the ANSWER literal permits the researcher to simply and easily examine its argument—which contains the object that has been constructed by the method—to see whether the construct indeed contains no occurrences of f (see Section 2.6). If any expressions Θf remain such that Θ contains no occurrences of f, we can correctly conclude that the corresponding Θ is a fixed point combinator with respect to P. In other words, if the two-stage version of the kernel method leads to the construction of an expression of the form Θf with f an arbitrarily chosen combinator and with Θ containing no occurrences of f, then we have proved that the set P satisfies the strong fixed point property.

To see why this is so, we give the following outline of the corresponding proof. First, $\Gamma = x\Gamma$ since Γ is a kernel and f is an arbitrary combinator and, therefore, any combinator x can be used in place of f. Second, since Γ expands to Θf and f is an arbitrary combinator, $\Gamma = \Theta x$, and $x\Gamma = x(\Theta x)$. Finally, again using the fact that $\Gamma = x\Gamma$, we can complete the proof by applying transitivity of equality twice.

If the use of the two-stage method does not lead to finding any kernels in its first stage after a moderate amount of effort by the researcher or—even more significantly—after a small amount of CPU time has been used by a theorem-proving program, then one can conjecture with confidence that the weak fixed point property is not satisfied by P. Similarly, if stage 1 succeeds in finding kernels but stage 2, after a small amount of CPU time, fails to construct a fixed point combinator, then again with confidence one can conclude that, although the weak fixed point property does hold, the strong does not. Of course, since we have not yet proved the appropriate theorems focusing on the completeness of the kernel method, nor proved that the kernel method leads to a decision procedure, one cannot be certain that P fails in this regard. Even further, when no kernels are found—since the strong fixed point property implies the weak fixed point property—one can also conjecture with confidence that P fails to satisfy the strong fixed point property.

When we study a set P of combinators, the imprecise phrase "a small amount of computer time" usually means less than 10 CPU seconds. The fact that we expect the kernel method to succeed so quickly suggests that, in our opinion, the method is extremely powerful; obviously, we find it difficult to remain objective. Therefore, to form an independent opinion of the power of the kernel method, one might consider the following question, but first without recourse to the kernel method or to a review of the earlier results, and then with its use; we give its answer in Section 5.3.5.

If the set P consists of the two combinators B with

$$Bxyz = x(yz)$$

and W^1 with

$W^1xy = yxx,$

does P satisfy the strong fixed point property? Even further, before turning to the use of the kernel method, one might glance at a clock to time one's first attempt. That attempt might, for example, proceed by assuming that no fixed point combinators can be constructed from the set P, which we can express as

$y \neq fy,$

and then seeking a proof by contradiction. In part to enable a comparison to be made with the kernel method—and, to be forthright, in part to show why we are so delighted with the kernel method—we note that our newest theorem-proving program answered the question under discussion in less than two CPU seconds. For those who are curious, our attempt to answer the question without recourse to using the kernel method failed.

Revisiting the Three-Stage Version

Perhaps more appealing than the two-stage version of the kernel method is the three-stage version. In the first stage, the object is to find superkernels with respect to the set P of combinators under consideration. A superkernel Γ is a kernel such that no reduction with an element of P can be applied except one involving the first combinator that occurs in Γ. In the second stage of the three-stage version of the kernel method, the object is to use the superkernels found in the first stage, one at a time, to find kernels. In contrast to the first stage in which the search is conducted by applying expansions with elements from P where each expansion is required to apply to a single term, the second stage employs global expansions with the elements of P. A global expansion is an expansion that is applied to a term and all like terms simultaneously. The third stage of the three-stage version of the method is similar to the second stage of the two-stage version; its object is to construct fixed point combinators from the kernels found in either the first or the second stage.

The first stage of the three-stage version of the kernel method also searches for kernel schemata. A kernel schema is an expression Δ that contains one or more variables such that, if one replaces those variables with variable-free terms, one obtains kernels. In addition, each of the kernels so obtained is not required to reduce to a superkernel, but it can be used to globally expand to obtain additional kernels. For example, the expression $\Omega\Omega z$ with $\Omega = W(B(Bf))$ is a kernel schema from which one obtains the superkernel $\Omega\Omega\Omega$, the pseudo-superkernel $\Omega\Omega(\Omega\Omega)$, and the like. The superkernel $\Omega\Omega\Omega$ plays an important role for constructing fixed point combinators from B and W alone, as we shall discuss in Section 5.3.3. The reason that the kernel $\Omega\Omega(\Omega\Omega)$ is not a superkernel is that it can be reduced with W in a manner that does not involve the first of its symbols. The reason this kernel is called a pseudo-superkernel is that it acts like a superkernel—it does not reduce to another superkernel.

Successes with the Kernel Method

We have successfully applied the kernel method to a variety of sets P of combinators; we shall discuss in detail the specific results in later subsections. Among those successes, the most noteworthy focuses on alternative answers to Smullyan's original question concerning the possible construction of a fixed point combinator from B and one other regular combinator. On the one hand, we were able to construct from the combinators B and W fixed point combinators that are not equal to Θ_4—the combinator

Statman discovered—and whose distinctness is maintained even when extensionality is present. We shall discuss these combinators in some detail in Section 5.3.3. On the other hand, we were able to find various regular combinators to replace W such that, when each is considered with B alone, one can construct fixed point combinators. We shall discuss those regular combinators—N and H and others—and the corresponding results in Section 5.3.4 and later sections, where N and H satisfy the following equations.

$$Nxyz = xzyz$$
$$Hxyz = xyzy$$

From a general viewpoint, perhaps more significant is the mapping that one finds by using the kernel method. Specifically, to each fixed point combinator—by observing the appropriate constraints—one can identify a unique kernel from which the fixed point combinator can be obtained by expansion with the elements of the set P under consideration. If viewed from the theorem that says that the presence of the strong fixed point property implies the presence of the weak fixed point property, the preceding remark says that to every fixed point combinator Θ there corresponds a kernel Γ such that Θf reduces to Γ, where Θ is a y satisfying the condition for all x there exists a y such that

$$yx = x(yx)$$

and where Γ is a y satisfying the condition for all x there exists a y such that

$$y = xy.$$

In other words, speaking informally, the corresponding conjecture asserts that for every fixed point combinator, we can find a reducible weak fixed point combinator to which the fixed point combinator naturally maps. Or—still speaking informally—in the context of the quantifiers, there exists a mapping that interchanges the quantification, with of course a corresponding change in the relevant equation. The mapping is not always one-to-one; more than one fixed point combinator may naturally map to a single kernel (see Section 5.3.3).

Comparing the Kernel Method to the Standard Approach

Of a different flavor, but equally significant, either version of the kernel method offers far more efficiency when searching for fixed point combinators than if one chooses the more typical route. To see why this is so, we note that the typical route begins by denying that the strong fixed point property holds for the set P under consideration. This route is typical of automated theorem proving, but it is also typical of mathematics and logic. Next, one proceeds to apply reductions to the inequality—obtained from the assumption concerning the absence of the strong fixed point property—and the inequalities that are deduced. If one is using an automated theorem-proving program, then one would ordinarily use the inference rule paramodulation, for it builds in the concept of equality. Reductions must be applied to both sides of the various inequalities because one is in fact searching for a combinator Θ such that

$$\Theta x = x(\Theta x)$$

for all combinators x.

Since one might easily miss the need for applying reductions to both sides of the various inequalities, let us briefly cover that point immediately. In particular, let us imagine that we are presented with a

candidate combinator Θ for the sought-after fixed point combinator. Our task would then be to prove that

$$\Theta x = x(\Theta x).$$

We would most likely invoke the Church-Rosser theorem which states that, when two expressions are equal, then a third exists to which the two both reduce. Therefore, if indeed

$$\Theta x = x(\Theta x),$$

then both Θx and $x(\Theta x)$ reduce to some to-be-determined common third expression. If we were asked to be more precise, we would actually consider Θf and $f(\Theta f)$ for some arbitrarily chosen combinator f. The need for this replacement of x by f arises from the fact that one cannot technically talk about applying Θ to x since one applies one combinator to another, and not to a variable. For a more familiar example, one does not multiply x by 4 in ordinary arithmetic; the operation of multiplication is applied to pairs of numbers, not to a pair consisting of a variable and a number. So, taking these remarks and observations into account, we would attempt to reduce both Θf and $f(\Theta f)$ repeatedly until some common third expression is encountered. In other words, a natural proof that

$$\Theta f = f(\Theta f)$$

proceeds by reducing both sides; very seldom does Θf, for example, reduce to $f(\Theta f)$.

Having clarified the point about the need to reduce both sides of various inequalities if one is reasoning backward, say, from the assumed falseness of the theorem, or the need to reduce both sides of various equalities if one is reasoning forward, we can now show why use of the kernel method offers one a distinct advantage over the typical approach. Specifically, we showed earlier why, when one finds a kernel that expands to Θf for some Θ containing no occurrences of f, such an event amounts to having a proof that Θ is indeed a fixed point combinator. Contained in that discussion is the fact that the kernel method is, in effect, required to focus on one side of the equality

$$\Theta f = f(\Theta f)$$

only. Such a focus of attention is far more effective than that concerned with both sides, for the latter obviously requires the examination of a substantially greater number of paths in most cases. Equally serious, the latter presents the danger of reducing the two sides in such a way that one never arrives at a common third expression. For example, if one applies the same reductions to both Θf and $f(\Theta f)$ repeatedly, then one will never obtain a reduction of both to the expression $f\Gamma$ where Γ is the kernel—that we believe always exists—from which the fixed point combinator Θ can be obtained by expansion, nor, in fact, to any common third expression. The preceding analysis gives us a bonus, for it shows why, from the viewpoint of automated theorem proving, one should not use the version of paramodulation that simultaneously replaces all like terms.

An additional factor contributing to the effectiveness of the kernel method when compared to other more straightforward approaches rests with its narrow perspective. In stage 1 of the two-stage version, for example, the object of the search is a kernel. The search for such an object is often far narrower, and the path far shorter, than that for a fixed point combinator. Having found a kernel, the second stage then ignores all of the paths that have not yet been completed—and even considers the kernels one at a time—and again pursues a narrower and shorter path, that from the kernel to the fixed point combinator compared to that from the axioms to the fixed point combinator. Added to what we have said is the use of a

number of strategies that make the various searches even narrower (see Section 4.5.1 where we presented the two-stage kernel method). All of these observations apply equally to the three-stage version of the kernel method as well. We thus have an explanation for the ease and effectiveness offered by the use of either version of the kernel method for attempting to prove that the weak or the strong fixed point property holds for some given set P of combinators.

The Kernel Method and a Mapping

One final point of substantial interest with regard to the kernel method. From what we have found so far, it appears that to every fixed point combinator Θ there naturally corresponds a single kernel. In most cases, one finds the image of a given fixed point combinator Θ by appending f to Θ, where f is an arbitrarily chosen combinator, and then repeatedly applying reductions to Θf with the requirement that each reduction must satisfy the 1's rule or satisfy the pseudo 1's rule. Intuitively, a reduction satisfies the 1's rule if the term being reduced involves the first symbol in the expression being reduced, ignoring parentheses and commas. Formally, a reduction satisfies the 1's rule if the term being reduced has a position vector consisting of all 1's (see Section 2.3 or Section 3.4 for examples). The pseudo 1's rule is similar to the 1's rule except that one ignores all leading f's when deciding whether the constraint is satisfied. Unfortunately, there do exist cases for which one must eventually also apply reductions that violate even the pseudo 1's rule.

Of course, the kernel that naturally corresponds to the fixed point combinator under study may not be a simple kernel. For such an example, one considers the fixed point combinator $S(QM)(QM)$ whose naturally corresponding kernel is $QMf(QMf)$, which is a semisimple kernel.

The kernel that naturally corresponds to a fixed point combinator Θ is the first kernel one encounters when reducing Θf for an arbitrarily chosen combinator f, where all reductions must satisfy the 1's rule or the pseudo 1's rule. As expected, we conjecture that to every fixed point combinator Θ there exists a kernel Γ such that Θf reduces to Γ reduces to $f\Gamma$. Phrased in the context of the weak and strong fixed point properties, we conjecture that the strong fixed point property implies the reducible weak fixed point property. The existence of combinators that are eliminators and combinators that are permuters may present severe obstacles, and the conjecture may require appropriate modification. Also, the existence of fixed point combinators that do not have a normal form—combinators such as $LO(LO)$—may present additional obstacles. In particular, $LO(LO)$ with

$$Lxy = x(yy)$$

and

$$Oxy = y(xy)$$

is such that repeated reductions with L and O do not terminate. Nevertheless, a natural kernel corresponding to this fixed point combinator exists, the kernel $LO(LO)f$.

The mapping of fixed point combinators to kernels is not one-to-one in all cases. Therefore, if one wishes to reverse matters and to find a natural unique fixed point combinator (if such a combinator exists for the combinator set under study) that corresponds to a given kernel—in other words, if one wishes to apply a mapping that can be thought of as the inverse of that which maps fixed point combinators to

kernels—one must impose additional restrictions on the path that leads from the kernel to the fixed point combinator. To see why we must say "if such a combinator exists for the combinator set under study", one need only consider the kernel $Lf(Lf)$. The set consisting of L alone, although it obviously satisfies the weak fixed point property as proved by the given kernel, does not satisfy the strong fixed point property, which follows from Theorem 5 of Section 4.6. On the other hand, if one is allowed to adjoin combinators, then there do exist fixed point combinators for which $Lf(Lf)$ is the natural corresponding kernel. For example, the fixed point combinator BML has as its natural corresponding kernel $Lf(Lf)$. We shall make additional and clarifying comments about this topic and related topics as we proceed through the next few subsections.

Because of the ease with which the following problem was solved, we now come to what may be very persuasive evidence of the power of the kernel method. Specifically, as this section was being written, we "rediscovered the existence of the combinator H" with

$$Hxyz = xyzy.$$

Since we wished to include in this report the results concerning the set consisting of B and N alone—among which is an alternative answer to the question posed by Smullyan discussed earlier—we decided to instruct our program to use the kernel method to study the set obtained by replacing N by H.

In 0.2 CPU seconds, the program found the kernel $\Omega\Omega\Omega$ with $\Omega = H(B(Bf))$. Of course, this kernel is very familiar—familiar in that we have seen two clones of it, one with H replaced by N, and one with H replaced by W. Surely, because of the success with B and W followed by the success with B and N, consideration of this new kernel by the kernel method would lead to the construction of a fixed point combinator, which in turn would prove that the set P consisting of B and H alone satisfies the strong fixed point property. Surely, if we instructed the program to attempt the desired construction, we would have yet a third solution to the Smullyan problem. In fact, such is the case.

After 60 CPU seconds, the program constructed

$$B(B(H(B(BH)(BB))(H(BH)(BB))H)B)B,$$

which is indeed an impressive fixed point combinator. Without the kernel method, we cannot even make a good guess as to the number of hours of a researcher's time—perhaps weeks is a better estimation—that would be required to construct this fixed point combinator from H and B alone.

For a quite different example, still focusing on H and B alone, we suggest one examine the kernel $HH(B(Bf))(HH)$. The discovery of this kernel would also be sufficient to prove that the weak fixed point property holds for H and B. We do not in any way mean to imply that we would have found this kernel without the kernel method; quite the contrary, for it seems counterintuitive that this expression satisfies any interesting equation, much less the equation for the weak fixed point property. However, since the kernel method found this expression, why not reward the method with the request that it attempt to construct from this kernel a fixed point combinator?

The method in fact succeeded. It found

$$H(B(H(HB))B)B(HH),$$

which one possibly could have constructed without the kernel method—unless, of course, one had seen

the kernel, but not known it was one, and had then been asked to start with it. In that case, one might simply have considered the given expression (kernel) useless, and decided that the request to use the given expression could not lead to success.

As the final piece of evidence of the kind we have been giving, we turn to other combinators that are regular and noneliminating and involve three distinct variables, whose right side is fully left associated, and that replicate one variable—exactly one of the variables occurring on the right side of the corresponding equation appears more than once. We in fact confine our attention to a subset of that set, to those such that only one variable replicates and replicates exactly once. The combinators N and H are examples of what we have in mind. Of the remaining, only the combinator whose right side is $xyyz$ has the property that, when considered with B, the resulting pair fails to satisfy the strong fixed point property.

If one were hasty, one might conclude that, because of so many distinct solutions, the original question posed by Smullyan—which was open until early 1986—was in fact an easy one to answer. We suspect the opposite is true. From what we know, the reason for finding these distinct solutions rests solely with the power of the kernel method, and not with the lack of difficulty of finding an answer to the original question.

5.3.3. The Combinators *B* and *W*

The most startling result regarding the combinators *B* and *W* concerns the unexpected wealth of fixed point combinators that one can construct from the two. In particular, one can construct an infinite set $\Theta(BW)$ of irreducible fixed point combinators from *B* and *W* alone such that $\Theta(BW)$ contains an infinite subset with the property that no two of its elements are equal even in the presence of extensionality. Even more impressive, one can construct an infinite class of infinite sets of irreducible fixed point combinators from *B* and *W* alone. The members of the set $\Theta(BW)$ can be enumerated where the enumeration strongly connects them to the respective generators of the superkernels from which the set $\Theta(BW)$ is obtained. Indeed, $\Theta(BW)$ can be partitioned into an infinite sequence of finite subsets such that the enumeration of all of $\Theta(BW)$ begins by assigning the integers 1 through 5 to the members of the first set in the sequence, continues by assigning the integers 6 through 19 to the members of the second set in the sequence, and, in general, proceeds by assigning to the members of the n-th set in the sequence the integers $k+1$ through p where $p-k$ is equal to the number of ways to associate $n+3$ letters. For any given set in the infinite sequence, all of its members have the same length, the number of symbols occurring in the combinator excluding parentheses; the length of the combinators in the n-th set in the sequence is $3(n+2)-1$. The generator of the superkernel from which the members of the n-th set in the sequence are constructed directly and indirectly contains $n+3$ symbols, one of which is the combinator *W* and one of which is the arbitrarily chosen combinator *f*. The final point regarding the enumeration of the combinators in $\Theta(BW)$ asserts that the n-th superkernel is the $n+2$ power of its generator. Essentially all of these results were obtained by relying heavily on the kernel method.

The connection between the combinators in $\Theta(BW)$ and the individual kernels that lead to their construction rests with the following observation. For the set $\Theta(BW)$ of fixed point combinators, there exists a natural one-to-one onto mapping from its members to the set $\Gamma(BW)$ of kernels from which they can be constructed. In other words, speaking casually, there exists a one-to-one mapping of the y that satisfies there exists a y such that for all x

$$yx = x(yx)$$

and the y that satisfies for all x there exists a y such that

$$y = xy.$$

However, as we already noted, the y that the kernel method finds—if successful—to prove that the weak fixed point property holds actually satisfies the more restrictive condition that y reduces to xy. One can obtain an inverse mapping from the kernels in $\Gamma(BW)$ to the elements of $\Theta(BW)$ if appropriate restrictions are placed on the expansion path that leads from a kernel to its fixed point combinator.

The set $\Theta(BW)$ is a member of the infinite class of infinite sets we mentioned earlier. For most of the other infinite sets in the infinite class, the mapping from fixed point combinators to kernels is many-to-one. From $\Theta(BW)$, one can construct an infinite set that acts as a companion to $\Theta(BW)$. The construction consists of replacing the last occurrence of the combinator *W* in each combinator in $\Theta(BW)$ by *BW*. If we borrow from sets outside of $\Theta(BW)$, we can give the following summary regarding the lengths of the fixed point combinators that can be constructed from *B* and *W* alone. No fixed point combinators can be constructed from *B* and *W* alone whose length is less than 8. From *B* and *W* alone, one can construct

exactly 5 irreducible fixed point combinators of length 8, 5 of length 9, and none of length 10; and for any length greater than or equal to 11, one can construct more than 14.

To put into perspective the propensity for constructing fixed point combinators offered by the set consisting of B and W alone, as late as the spring of 1986, it was not known whether one could find a single regular combinator that, when considered with the combinator B, would give the set P of combinators consisting of B and the other combinator the power to satisfy the strong fixed point property. Then, after Statman's discovery of the combinator we call Θ_4, one might naturally have thought that Θ_4 was unique. Indeed, when we notified Smullyan by letter of our discovery of four additional fixed point combinators constructible from B and W alone—we give the five fixed point combinators shortly—he was quite surprised.

To add to the perspective regarding the richness of the set P consisting of B and W alone, we recall that the set consisting of S and K alone is complete, where S and K satisfy the following equations.

$$Sxyz = xz(yz)$$

$$Kxy = x$$

In particular, from S and K, one can construct any combinator that one wishes. But no corresponding claim exists concerning the set consisting of B and W. Nevertheless, with regard to the strong fixed point property—if we speak informally—these two combinators have almost all the power one might ever wish for. So that one can gain some understanding concerning the source of all of this power, and so that one can get a view of the history of the research that produced the results, let us rush through the discoveries as they were made.

The First Infinite Set of Fixed Point Combinators

The fixed point combinator Θ_4, which Statman discovered, is one of a family of five such combinators that can be constructed from B and W alone. Here are the five combinators, each preceded by the kernel to which it naturally corresponds.

$$\Gamma_1 = W(B(Bf))(W(B(Bf)))(W(B(Bf)))$$
$$\Theta_1 = B(B(B(WW)W)B)B$$

$$\Gamma_2 = W(BBBf)(W(BBBf))(W(BBBf))$$
$$\Theta_2 = B(B(WW)W)(BBB)$$

$$\Gamma_3 = BWB(Bf)(BWB(Bf))(BWB(Bf))$$
$$\Theta_3 = B(B(WW)(BWB))B$$

$$\Gamma_4 = BW(BBB)f(BW(BBB)f)(BW(BBB)f)$$
$$\Theta_4 = B(WW)(BW(BBB))$$

$$\Gamma_5 = B(BWB)Bf(B(BWB)Bf)(B(BWB)Bf)$$
$$\Theta_5 = B(WW)(B(BWB)B)$$

In the absence of extensionality, all five combinators Θ_1 through Θ_5 are distinct; in its presence, all five are equal. All but the fourth are new discoveries resulting from our research. Each of the five fixed point

combinators contains five occurrences of B and three occurrences of W. Of the kernels, only Γ_1 is a superkernel, and a superkernel from which the other four kernels can be derived by global expansion.

From the superkernel Γ_1, one can obtain an infinite sequence of superkernels—we shall show precisely how one obtains this set at the end of this section, but we first give the more significant properties of this set. As commented earlier, from Γ_1—which is the cube of the expression $W(B(Bf))$—one can obtain by repeated global expansion with B four additional kernels such that, from each of the five, one can construct exactly one fixed point combinator. From the next superkernel—which is the fourth power of $W(B(B(Bf)))$—in the sequence, one can obtain by global expansion 13 additional kernels. From each of these 14 kernels—if one expands the smallest possible term in the kernel that contains an occurrence of the generator of the kernel—one can construct exactly one fixed point combinator. Here are the 14 fixed point combinators.

$B(B(B(B(W(WW))W)B)B)B$
$B(B(B(W(WW))W)B)(BBB)$
$B(B(B(W(WW))W)(BBB))B$
$B(B(B(W(WW))(BWB))B)B$
$B(B(W(WW))W)(BB(BBB))$
$B(B(W(WW))(BWB))(BBB)$
$B(B(W(WW))W)(B(BBB)B)$
$B(B(W(WW))(BW(BBB)))B$
$B(B(W(WW))(B(BWB)B))B$
$B(W(WW))(BW(BB(BBB)))$
$B(W(WW))(B(BWB)(BBB))$
$B(W(WW))(BW(B(BBB)B))$
$B(W(WW))(B(BW(BBB))B)$
$B(W(WW))(B(B(BWB)B)B)$

From the next superkernel—which is the fifth power of $W(B(B(B(Bf))))$—one can obtain by global expansion 41 additional kernels. If one observes the appropriate strategy to restrict the expansions, from each of the 42 kernels one can construct exactly one fixed point combinator. One can continue in this fashion and generate an infinite sequence of superkernels such that, if one derives the corresponding kernels and then applies the appropriate strategy for constructing fixed point combinators, the corresponding sequence of sets of fixed point combinators contains 5, 14, 42, and, in general, the number of ways one can associate n letters for $n \geq 4$. The n elements to be associated are the symbols in the generator of the superkernel—for example, in Γ_1, the symbols W, B, B, and f.

By focusing on the symbols in the generator of a superkernel of a set in the infinite sequence and analyzing their final position for each kernel derived from the superkernel, one obtains a natural enumeration of the kernels (see Section 4.4.3). The enumeration of the kernels in turn induces an enumeration of the corresponding fixed point combinators. Indeed, the kernels listed earlier—Θ_1 through Θ_5—reflect this enumeration (see Section 4.4.3 for the full details). More to the point perhaps, we chose the notation and naming conventions for the combinators in the class $\Theta(BW)$ because of this enumeration and the mapping of fixed point combinators to kernels. All of the fixed point combinators within a given set in this

infinite sequence have the same length—8, 11, 14, and, in general $3k$-1, where k is the power of the generator in which the superkernel is expressed.

An Infinite Companion Set of Fixed Point Combinators

Next we come to an infinite set of fixed point combinators that act as companions to those we have just discussed. We discovered this next set in our attempt to find combinators whose lengths would fill in the some of the gaps in the first set, the gaps between 8, 11, 14, and the like. The combinators in this infinite set fall naturally into subsets based on the length of the combinator. The subsets consist of 5 combinators of length 9, 14 of length 12, 42 of length 15, and—as one might guess—n of length $3k$, where the superkernel from which a subset is directly and indirectly constructed has the form Ω^k with Ω its generator, and where n is the number of ways to associate k+1 letters. The subsets are, respectively, the companion sets to the members in the infinite sequence whose union is $\Theta(BW)$, the members consisting, respectively, of combinators of length 8, 11, 14, and the like.

To see how each companion set is obtained, let us focus first on the set consisting of Θ_1 through Θ_5. One simply modifies the corresponding kernel by replacing W by BW in all occurrences of the generator and expands to the corresponding fixed point combinator, or, equivalently, directly replaces in the combinator itself the last occurrence of W by BW. Here are the five pairs, each pair consisting of an element from among Θ_1 through Θ_5 with its companion.

$\Theta_1 = B(B(B(WW)W)B)B$
companion = $B(B(B(WW)(BW))B)B$

$\Theta_2 = B(B(WW)W)(BBB)$
companion = $B(B(WW)(BW))(BBB)$

$\Theta_3 = B(B(WW)(BWB))B$
companion = $B(B(WW)(B(BW)B))B$

$\Theta_4 = B(WW)(BW(BBB))$
companion = $B(WW)(B(BW)(BBB))$

$\Theta_5 = B(WW)(B(BWB)B)$
companion = $B(WW)(B(B(BW)B)B)$

In general, one obtains the companion set to one of the finite sets in the infinite sequence discussed earlier by replacing the occurrence of W by BW in all occurrences of the generator of the appropriate kernel and then expanding to the fixed point combinator. Instead, and perhaps simpler, one can obtain the companion set by replacing the last occurrence of W by BW in the actual combinator itself to which one is constructing the companion.

A Many-to-One Mapping

Even though we have discussed two infinite sets of fixed point combinators, we have not yet exhausted the riches that can be extracted from the consideration of B and W alone. In addition, we have not yet discussed the case in which a kernel is the natural correspondent of more than one fixed point

combinator—the case in which the mapping from fixed point combinators to their respective kernels is many-to-one rather than one-to-one. We can attack both issues simultaneously by revisiting the superkernel from which, directly and indirectly, one can construct the 14 combinators each of which has length 11. As we noted, the superkernel is the fourth power of $W(B(B(^{9}f)))$. For notational convenience, let us denote $W(B(^{9}(Bf)))$ by Ω, obviously not to be confused with our normal use of $\Omega = W(B(Bf))$ in the context of the superkernel Γ_1. The superkernel under study can thus be written as $\Omega\Omega\Omega\Omega$.

Earlier in this section we noted that exactly one fixed point combinator can be constructed from this superkernel and from each of the kernels derived from it, provided that the expansions are applied to the smallest possible term in the kernel that contains an occurrence of the generator of the kernel. In other words, we supplied at that point in the discussion a hint of what we now say explicitly. Specifically, if one expands the superkernel $\Omega\Omega\Omega\Omega$ with the intention of constructing fixed point combinators, one will succeed. However, in contrast to Γ_1, in this case one can construct four fixed point combinators. Each of the four that one constructs maps naturally to the single superkernel $\Omega\Omega\Omega\Omega$. To be even clearer, the four are obtained by ignoring the constraint of applying expansions only to the smallest term in the kernel that contains an occurrence of the generator Ω.

To see how one can construct four fixed point combinators from $\Omega\Omega\Omega\Omega$, we first examine Θ_6,

$B(B(B(B(W(WW))W)B)B)B$,

which is the first combinator in the set consisting of 14 combinators of length 11. Of course, as indicated, observing the constraint of expanding the smallest term containing Ω leads to the construction of this fixed point combinator. This combinator is named Θ_6 because the combinators of length 8 are numbered, respectively, Θ_1 through Θ_5, and the given combinator is the first member of the set of combinators of length 11. This combinator is constructed by using the strategy we discussed in Section 4.5.1 and in its subsections. Specifically, one treats Ω as a constant, expanding until a single occurrence of Ω remains, then replacing Ω by its definition $W(B(B(Bf)))$, and then expanding further with B and W. In addition, all expansions are required to satisfy the 1's rule, and each term that is expanded after the definition replacement must contain an occurrence of f. Here is the initial segment of the expansion path that leads to the construction of Θ_6.

$\Omega\Omega\Omega\Omega$, $W\Omega\Omega\Omega$, $WW\Omega\Omega$, $W(WW)\Omega$

To complete the construction, one merely replaces Ω by its definition $W(B(B(Bf)))$ and applies B as an expansion four times, which, as expected, produces an expression of the form Θf with Θ containing no occurrences of f—the combinator Θ_6.

To construct the other three fixed point combinators from $\Gamma_6 = \Omega\Omega\Omega\Omega$, one applies the same approach used to construct Θ_6, except, of course, pursuing paths that do not favor the shortest term that is expandable. An analysis of the approach shows that all four fixed point combinators so obtained differ only in their respective middle segments. After all, the initial segment of each is identical and consists of four occurrences of B whose presence is explained by the need to free f since Ω contains four symbols to be incorporated into the corresponding fixed point combinator. The final segment of each is identical, consisting of the combinators in Ω in order with the exception of f. In other words, the differences between the four combinators rests totally with the expansion path that one traverses to obtain an

expression in which the generator Ω occurs exactly once. Therefore, we need list only the four expressions, each of which contains a single occurrence of Ω, for one can complete the construction by replacing Ω by its definition and expanding with *B* four times. Here are the four expressions.

W(*WW*)Ω
W(*BWW*)Ω
W(*W*(*BW*))Ω
W(*BW*(*BW*))Ω

Fixed Point Combinators That Were Missing

A quick glance at the four expressions shows that we have in fact answered one of the previously unanswered questions concerning the construction of fixed point combinators, or perhaps extended one of the answers is a better way to put it. In particular, we have found a means for constructing fixed point combinators of length 13. Admittedly, for such a construction, we must abandon the one-to-one mapping of fixed point combinators to their natural kernels. Nevertheless, another gap in the lengths of such combinators is filled in. As for the remaining gaps—for example, combinators of length 16, 19 and, in general, $3k+1$ for $k \geq 5$—the obvious guess is the right one. Specifically, we can focus on the superkernels of the form W(B(B(B(Bf)))) to the fifth power and the like, relax the constraint of expanding only the shortest term that contains at least one occurrence of the generator of the superkernel under study, and construct the desired objects.

To see one such example—and to complete one's understanding of how it all works—let us focus on ΩΩΩΩΩ, but now where Ω = *W*(*B*(*B*(*B*(*Bf*))))). Because of the preceding discussion, we need give only the partial expressions that one obtains when pursuing an expansion path from ΩΩΩΩΩ to the construction of fixed point combinators. In other words, to complete a construction, one merely replaces the generator by its definition and applies *B* the appropriate number of times—in this case, five. To add to the growing picture, in contrast to the single fixed point combinator that corresponds to the cube of *W*(*B*(*Bf*)) and to the four fixed point combinators that correspond to the fourth power of *W*(*B*(*B*(*Bf*))), the following 18 expressions lead to fixed point combinators that correspond to and are directly constructible from ΩΩΩΩΩ with Ω = *W*(*B*(*B*(*B*(*Bf*))))).

W(*W*(*WW*))Ω
W(*BW*(*WW*))Ω
W(*W*(*BWW*))Ω
W(*W*(*W*(*BW*)))Ω
W(*BW*(*BWW*))Ω
W(*BW*(*W*(*BW*)))Ω
W(*W*(*B*(*BW*)*W*))Ω
W(*W*(*BW*(*BW*)))Ω
W(*W*(*W*(*B*(*BW*))))Ω
W(*BW*(*B*(*BW*)*W*))Ω
W(*BW*(*BW*(*BW*)))Ω
W(*BW*(*W*(*B*(*BW*))))Ω

$W(W(B(BW)(BW)))\Omega$
$W(W(BW(B(BW))))\Omega$
$W(BW(B(BW)(BW)))\Omega$
$W(BW(BW(B(BW))))\Omega$
$W(W(B(BW)(B(BW))))\Omega$
$W(BW(B(BW)(B(BW))))\Omega$

Of the 18 combinators that can be constructed, the first is that which we name Θ_{20} because it is the first of length 14—its natural kernel correspondent is the third superkernel from the infinite sequence of superkernels discussed earlier in this section—and because there exist 5 such combinators directly and indirectly derived from the first of the superkernels, Γ_1, and 14 derivable from the second superkernel, Γ_6.

The combinators in the set whose natural kernel is Γ_{20} can be divided according to their length—one of length 14, three of length 15, five of length 16, five of length 17, three of length 18, and finally one of length 19. All 18 share a common initial and a common final segment. The main difference between them rests with the middle segment, that which is determined by the expansion path that leads from the superkernel to an expression in which the generator of the superkernel appears once. In particular, from Γ_{20}, one can already fill in the gaps for combinators of length 16 and of length 19. Of course, for combinators of length 22, 25, and the like, one simply moves to the fourth, fifth, and later superkernels in the infinite sequence whose first element is Γ_1.

A Simple Way to Study Certain Sets of Combinators

The preceding discussion concerning the construction of more than one fixed point combinator from a superkernel applies equally to the kernels that are derived from that superkernel. Indeed, if for some reason—which might be far from obvious—one wishes to study these various combinators, one can do so in the following manner. One begins by selecting a superkernel from the infinite sequence of superkernels and expresses it as an appropriate power of a generator, which one can denote for that study as Ω to permit one to easily use the preceding examples. One then simultaneously expands all occurrences of the generator—a process we call global expansion—to derive the full set of kernels that reside in the family of the chosen superkernel. Next, one selects any of the kernels—including the superkernel itself—and pursues all expansion paths that lead from the selected kernel to an expression in which the generator occurs exactly once. However, each expansion along such a path must involve a term that contains at least one occurrence of the generator Ω. This constraint is designed to avoid following a path that eventually leads to an uninteresting combinator, one that is equal even without extensionality to a combinator obtained along a path that observes the given constraint. The search can be further restricted by requiring that all expansions satisfy the 1's rule. Then one replaces Ω by its definition, and continues expanding.

At this point, the expansions can again be limited to those that satisfy the 1's rule. Each expansion must involve a term containing an occurrence of f, where f is the arbitrarily chosen combinator that occurs in the kernel. Similar to the earlier constraint, this restriction is designed to avoid constructing uninteresting combinators. One reaches the final destination when one obtains an expression of the form Θf with Θ containing no occurrences of f, which means that Θ is a fixed point combinator. Except for Γ_1, the mapping from kernel to fixed point combinator(s) is one-to-many.

The following shortcut can be taken because the kernels within a set are so closely related. In the simplest case, if one has constructed all of the fixed point combinators that correspond to a superkernel, one can begin with the expressions that result from expanding that superkernel to the point at which the generator occurs exactly once. For any other kernel in the set of kernels derivable from the superkernel, the same expressions are relevant. Therefore, one merely replaces the generator of the superkernel by the generator of the kernel under study, and expands to the various fixed point combinators that naturally correspond to that kernel. Just to prevent one from trying to leap too far, one cannot simply move to the set of fixed point combinators whose natural kernel correspondent is the superkernel that began this phase of the study. Speaking informally, the reason such an action fails is that some of the expansions with B that occur on the path from an expression containing a single occurrence of the generator of the superkernel already occur—for a kernel other than the superkernel—on the global expansion path from the superkernel to that kernel.

We can easily illustrate this point about expansions with B occurring before and after the point at which the generator of a superkernel and a derived kernel occur, respectively, once. We focus on Θ_1 and Θ_4 and their respective natural kernel correspondents.

$\Gamma_1 = W(B(Bf))(W(B(Bf)))(W(B(Bf)))$
$\Theta_1 = B(B(B(WW)W)B)B$

$\Gamma_4 = BW(BBB)f(BW(BBB)f)(BW(BBB)f)$
$\Theta_4 = B(WW)(BW(BBB))$

On each of the two respective paths from the kernel to the fixed point combinator that corresponds to it, two expansions with W occur to produce an expression in which the corresponding generator occurs exactly once. However, to obtain Θ_4 from $WW\Omega_4$ requires a single expansion of B in contrast to the three expansions with B that are required to obtain Θ_1 from $WW\Omega_1$. The explanation for this difference rests with the fact that the arbitrarily chosen combinator f is, so to speak, freer in Γ_4 than it is in Γ_1. In particular, since $\Omega_1 = W(B(Bf)), f$ is more deeply ensnared in Ω_1 than it is in $\Omega_4 = BW(BBBf)$. Some of the work of freeing f is done by the two global expansions with B that occur when deriving Γ_4 from Γ_1.

Therefore, although B occurs five times in both Θ_4 and Θ_1, the source of their occurrences is different for the two combinators. Specifically, in Θ_4, one occurrence of B results from expanding its kernel, two result from deriving its kernel from the corresponding superkernel, and two result from the occurrences of B in that superkernel. In contrast, in Θ_1, three occurrences of B result from expanding its kernel—which is the superkernel itself—and two result from the occurrences of B in the superkernel itself. Summarizing, we know of no simple and direct way to simply move from a fixed point combinator to another when their respective kernels are different even though both kernels are derived from the same superkernel. One can, however, resort to the shortcut we gave earlier and reduce the amount of work to be done, which brings us to what might be termed "fine print". After touching on the appropriate remarks concerning this fine print, we can complete our treatment of the set P of combinators consisting of B and W alone.

Fine Print

As is well known, fine print is sometimes used to hide information, but it is also sometimes used to keep information from being a distraction. Our intention is the latter—the following remarks, if made earlier, might well have served no immediate purpose, but simply functioned as a distraction.

The point we must make now is that we have not attempted to prove that each facet of our approach to constructing fixed point combinators has the appropriate completeness property. In particular, by following our approach precisely, one might fail to discover a fixed point combinator. One or more of the restrictions we observe during the various phases of our search might indeed be too restrictive. Of course, we have checked individual cases with one of our computer programs, but such a check does not replace a formal proof. Therefore, one might exercise some caution with regard to the property of our approach that focuses on finding all of the desired objects.

On the other side of the issue, that concerning soundness, our method is in fine shape. When one applies the kernel method and succeeds in constructing what is claimed to be a fixed point combinator, the claim is easily verified and always holds. As we commented, one is required to inspect, for example, the Θ in the expression Θf that results from a successful completion of stage 2 of the two-stage version of the method. However, such an inspection is a trivial task. Therefore, taking these remarks into account, some reserve is warranted if one is intent upon finding all fixed point combinators constructible from a given set P of combinators, when the kernel method is the means for the corresponding search.

Additional Infinite Sets of Fixed Point Combinators

Having given both a warning about what remains to be done and an implicit invitation to embark on the corresponding research, we can now complete our extraction of the riches that are offered by B and W. Specifically, in addition to the various infinite sets of combinators we have discussed, still others exist. When we mentioned earlier that the kernel method could find kernel schemata, we had in mind expressions such as $\Omega_1\Omega_1 z$ with $\Omega_1 = W(B(Bf))$. But we also had in mind expressions such as $\Omega_6\Omega_6 xy$, where $\Omega_6 = W(B(B(Bf)))$.

From the first of the two expressions, by instantiation one can obtain the superkernel Γ_1—one simply replaces z by Ω_1. From the second expression, one can obtain the superkernel Γ_6. From the first of the two superkernels, as we have discussed, one can construct, directly and indirectly, Θ_1 through Θ_5. From the second superkernel one can construct, directly and indirectly, Θ_6 through Θ_{19}, and more. The additional fixed point combinators arise from the fact that the mapping from fixed point combinators to their natural kernels is many-to-one for sets of kernels derived from those superkernels in the infinite sequence that are later in that sequence than Γ_1 is.

We can use the two kernel schemata to yield far more by replacing, respectively, z and x and y by the appropriate expressions. Specifically, if one replaces z in the first schema by any expression purely in terms of Ω_1, then, regardless of the association of the occurrences of that generator, one obtains a kernel. From that kernel, one can obtain kernels by applying global expansions; such an action will produce results that parallel the derivation of the kernels Γ_2 through Γ_5. Although we have not investigated the corresponding case, it appears that similar actions with regard to the second schema produce results that parallel Γ_6 through Γ_{20}.

For example, if one focuses on the first schema and on the instance $\Omega\Omega(\Omega\Omega)$ with $\Omega = W(B(Bf))$, one finds that $\Omega\Omega(\Omega\Omega)$ acts like a superkernel. However, it is not a superkernel because there exist reductions that apply to it but that do not satisfy the 1's rule; it can be thought of as a pseudo-superkernel. If one attempts to construct fixed point combinators from this pseudo-superkernel, one meets with success. Indeed, from it—rather than constructing a single fixed point combinator—one can construct nine such combinators. The source of this fecundity seems to be the same as that for the fourth power of $W(B(B(Bf)))$, which we reviewed earlier when we showed how one can obtain four fixed point combinators from a single superkernel. Succinctly stated, more than one fruitful path exists from the pseudo-superkernel to the point at which its generator occurs exactly once.

One might expect to find even wilder behavior if one studies $\Omega\Omega(\Omega\Omega\Omega)$ or $\Omega\Omega(\Omega(\Omega\Omega))$ or $\Omega\Omega(\Omega\Omega\Omega\Omega)$ and the like, with $\Omega = W(B(Bf))$. How extreme is the mapping one finds when one then turns to the consideration of instances of the second schema we have not studied. Perhaps the most appropriate utterance we can make relies on our earlier metaphor when we referred to these studies as exploring the *B*-and-*W* mine. Using that metaphor, one might say that the *B*-and-*W* mine is one of the richest mines ever found.

Odds and Ends

To complete this section, we have a few loose ends to tie up. For example, we promised to say how one can generate the infinite sequence of superkernels that begins with Γ_1. As may be transparent by now, one takes the generator $W(B(Bf))$, inserts a *B* to the right of *W*, and then fully right associates the result. The result is the generator of the next superkernel. One must, however, take the fully left-associated fourth power of the new generator. For the third superkernel in the sequence, one simply takes the generator of the second superkernel, inserts a *B* to the right of *W*, fully right associates, and then takes the fifth power of the result. The general pattern for the superkernels in this infinite sequence is even rather musical—the n-th element has a generator that contains n+1 occurrences of *B*, and the superkernel is equal to its generator taken to the power n+2.

The next topic focuses on the superkernel $W(Bf)(W(Bf))$. For the person who has immediately turned to this section on highlights, the existence of this superkernel might indeed be a shock. Such a person might ask: Why has this superkernel been omitted from the infinite sequence? One might suggest that it should be the first element in the sequence, rather than the cube of $W(B(Bf))$. After all, the requirements of the pattern for such superkernels are met by $W(Bf)(W(Bf))$ and, even further, one can begin with it and generate the superkernel we have been giving the first place in the sequence. The answer—for the person encountering $W(Bf)(W(Bf))$ for the first time or for the person who finds a review helpful—is that this superkernel cannot be used by the kernel method for constructing a fixed point combinator from *B* and *W* alone. The fault does not rest with the kernel method; rather, it rests with the superkernel itself. Although it does globally expand to yield an additional kernel—specifically, $BWBf(BWBf)$—it is not the natural kernel correspondent of any fixed point combinator constructible from just *B* and *W* alone.

To see what this superkernel lacks, one can either review our earlier comments on the subject, or simply show that the possible expansions with *B* and with *W* will not lead to a Θ containing no occurrences of *f*, the arbitrarily chosen combinator. In other words, if Θ is a fixed point combinator

constructed from B and W alone, which means that

$$\Theta x = x(\Theta x)$$

for all combinators x, Θf will never reduce to $W(Bf)(W(Bf))$. Therefore, since we are mainly interested for this aspect of combinatory logic in constructing fixed point combinators, we treat the superkernel $W(Bf)(W(Bf))$ as undesirable—although it certainly can be put to good use if other combinators are present, combinators such as M or S or N.

$$Mx = xx$$
$$Sxyz = xz(yz)$$
$$Nxyz = xzyz$$

For example—and here we have a small preview of the next section—the expression $B(N(BBW)W)B$ is a fixed point combinator whose natural kernel is $W(Bf)(W(Bf))$. Finally, the superkernel $W(Bf)(W(Bf))$ is in a sense the minimal y that satisfies the weak fixed point property for the set consisting of B and W alone, at least with regard to this pattern of superkernels.

For the next topic, we return to a very brief visit to the infinite set of combinators we enumerate beginning with Θ_1 and to the companion infinite set. We think of these two infinite sets as companion sets because they partition into the same type of subset—the first subset of each consists of 5 elements, the second of 14, the third of 42, and we know the full story about the remaining subsets. We saw how to obtain the companions of a subset of combinators; we simply replace the final occurrence of W in each by BW. However, companion is the better word—the word clone would not be a good choice.

The significant difference to note concerns the schema $\Omega\Omega z$ of which the superkernel Γ_1 is an instance. In particular, if one applies the first stage of the two-stage kernel method to the appropriate study, one does find $\Omega\Omega z$; however, one does not find its correspondent within the companion set, for it does not exist. Indeed, one simply finds the superkernel that can be obtained from Γ_1 by replacing all occurrences of W by BW. The explanation for this divergent behavior can be found if one reduces both superkernels—if one considers the path that must exist for any kernel Γ showing that Γ reduces to $f\Gamma$. In the one case, the third occurrence of the generator never is involved in the reduction. Therefore, one can replace the third occurrence by the variable z and still preserve the reduction property under discussion, which is why we call such expressions kernel schemata. In contrast, the companion superkernel, because of the occurrence of B before the occurrence of W in its generator, does involve the third occurrence of its generator. Therefore, one cannot make the corresponding replacement of a generator by a variable to obtain a kernel schema.

We can now turn to the topic of the lengths of fixed point combinators constructible from B and W alone. We have proved, by an exhaustive search with one of our computer programs, that none exists of length less than 8, and exactly five exist of lengths 8 and 9—we are uninterested in those that trivially reduce to others even when extensionality is absent. None exists of length 10, and many exist of length n with n greater than 10.

We close our comments concerning B and W with the suggestion that perhaps all of the fixed point combinators that can be constructed from these two combinators alone are captured by one of the sets we have discussed here or by focusing on an appropriate instance of $\Omega\Omega z$ and related schemata. In other

words, perhaps future mining of the *B*-and-*W* will only clarify the picture and add important detail to what has been said here, but will not lead to the discovery of yet another independent vein from which new gold can be extracted.

5.3.4. The Combinators *B* and *N*

Since we have explored the *B*-and-*W* mine as much as we intend to for this report—in particular, extracted and refined the ore taken from various rich veins, and given the location of other perhaps richer veins that can be mined in the future—we now turn to the exploration of the *B*-and-*N* mine. Immediately, one might ask about the reason for studying the set of combinators consisting of *B* and *N* with $Bxyz = x(yz)$ and $Nxyz = xzyz$. The source for our interest rests entirely with the discussion of the square consisting of the four combinators—*B*, *W*, *S*, and *L*—discussed in Section 1.

B W
S L

In that discussion, we noted that, of the four sets obtained by choosing one element from column 1 and one from column 2, the set consisting of *B* and *W* satisfies the strong fixed point property, the set consisting of *B* and *L* does not, the set consisting of *S* and *W* does not, and the set consisting of *S* and *L* does.

Two two-element sets remained unclassified, that consisting of *B* and *S*, and that consisting of *W* and *L*. In Section 5.3.6, we show that the set consisting of *W* and *L*—although, because of the existence of the kernel $Lf(Lf)$, it does obviously satisfy the weak fixed point property—fails to satisfy the strong fixed point property. Unfortunately, we have not yet settled the case for the set consisting of *B* and *S*; we can only conjecture, based on preliminary experiments with the kernel method, that this set fails to satisfy both the strong and the weak fixed point property. The connection to the combinator *N* can be traced to our tendency to imitate what has worked before, and to our failure to determine whether the set consisting of *B* and *S* satisfies the strong fixed point property. In particular, we did try to "force" a kernel to exist for *B* and *S* alone, which failed; the failure, as we shall immediately see, led directly to the discovery, and then the study of the combinator *N*.

We began our attack on the set consisting of *B* and *S* alone by starting with Γ_1, which is the cube of $W(B(Bf))$, and attempting to modify that kernel appropriately to produce a kernel for *B* and *S*. After trying the simplest approach—that of uniformly substituting *S* for *W*—and recognizing that an attempt to reduce the resulting expression Δ to $f\Delta$ fails to isolate *f*, we studied the effects of inserting an additional occurrence of *B* in the generator and adding one more occurrence of the generator to permit a sequence of reductions to isolate *f*. The resulting expression does in fact permit one to apply reductions until *f* isolates, but the occurrences of the generator are not associated in the desired manner in the result. Indeed, if $\Omega = S(B(B(Bf)))$, and if one reduces $\Omega\Omega\Omega\Omega$ until *f* isolates, one obtains $f(\Omega(\Omega\Omega)\Omega)$.

In view of the two failures to force a kernel to exist for the set consisting of *B* and *S* alone, perhaps the prudent action would have been to simply abandon the corresponding line of research. Instead, we decided that, since we could not suitably modify the kernel Γ_1—the kernel whose study had led us to the construction of the infinite set $\Theta(BW)$ of fixed point combinators—perhaps we could "modify" the combinator *S*. Our conjecture for the failure of the combinator *S*, when considered with *B*, to yield a kernel rests with the association of the variables on the right side of the equation for *S*. In particular, if the right

side were fully left associated, perhaps all would go as desired, which it in fact does. Specifically, if we replace S, with $Sxyz = xz(yz)$, by N, with $Nxyz = xzyz$, and return to the study of a clone of Γ_1—which we shall simply call Γ with Γ equal to the cube of $N(B(Bf))$—we find that Γ does reduce to $f\Gamma$.

In other words, we have proved that the set consisting of B and N at least satisfies the weak fixed point property. With this success in hand, since the study of Γ_1 proved so fruitful and since Γ so closely resembles Γ_1, how could we resist studying Γ, and how could we resist attempting to prove that the superkernel Γ leads to the construction of fixed point combinators from B and N alone? Here are the most significant results obtained from that study.

5.3.4.1. Significant Results Concerning *B* and *N*

The set P of combinators consisting of B and N alone does indeed satisfy the strong fixed point property. In fact, one can construct an infinite set $\Theta(BN)$ of irreducible fixed point combinators that shares a number of the important properties with $\Theta(BW)$. For example, one can extract from $\Theta(BN)$ an infinite subset such that no two of its elements are equal even in the presence of extensionality. Where the classes whose union is $\Theta(BW)$ consist, respectively, of 5 combinators of length 8, 14 of length 11, 42 of length 14, 132 of length 17, and the like, the corresponding classes in $\Theta(BN)$ consist of 2 combinators of length 13 and 3 of length 14, 14 of length 14, 42 of length 17, 132 of length 20, and the like.

However, as strongly suggested by the fact that the first class of combinators in $\Theta(BN)$ does *not* consist of combinators of a common length, one should expect—especially if one wishes to imitate the enumeration that we gave for $\Theta(BW)$—to encounter problems that do not exist when studying B and W. The reason for such an expectation rests with the fact that W is *not* a permuter but N is. In particular, the first three variables on the right side of the equation for N are x, z, and Y, which means that their order is a nontrivial permutation of the variables that appear of the left side of the equation.

The first problem one encounters concerns the fact that the mapping from fixed point combinators to their natural kernels is many-to-one for all kernels that are used to generate $\Theta(BN)$, which is not the case for all of the kernels that generate $\Theta(BW)$. In particular, even for the first kernel Γ, one can obtain, by applying the second stage of the kernel method, many fixed point combinators; on the other hand, from each of $\Gamma_1(BW)$ through Γ_5(BW), one obtains a single fixed point combinator. Then, since the mapping from fixed point combinators to Γ, which is the cube of $N(B(Bf))$, is many-to-one, we have the problem of how to choose from among the combinators, or the problem of how to order the full set of combinators.

One might immediately suggest that we approach the problem as we did with Γ_6 to which four combinators naturally map, and choose the shortest combinator from those that are available. In particular, from Γ_6, one can construct four fixed point combinators—one of length 11, two of length 12, and one of length 13—each of which has Γ_6 as its natural kernel. Such an approach—selecting the shortest fixed point combinator that naturally maps to the kernel under consideration—does not suffice. Let us quickly show why this solution is not sufficient, and also show how prolific the superkernel Γ is.

In contrast to Γ_1 from which the single fixed point combinator Θ_1 is constructed by expansion with B and W, from Γ, by expanding with B and N, one can construct at least 52 fixed point combinators among which are the following four.

$B(B(N(BB(N(BBN)N))N)B)B$

$B(B(N(BB(N(N(BB))N))N)B)B$
$B(B(N(N(B(BB)(N(BBN))))N)B)B$
$B(B(N(N(B(BB)(N(N(BB)))))N)B)B$

Therefore, the first superkernel Γ with respect to the set consisting of B and N behaves rather like the second superkernel Γ_6 of the sequence of superkernels that leads to the construction of $\Theta(BW)$. What adds to the complication of the study of B and N when compared to that of B and W—in particular, if one wishes to enumerate the corresponding fixed point combinators—is that two of the given four combinators have length 13. Therefore, when studying B and N, we cannot simply choose the shortest combinator from among those that naturally map to a kernel.

The superkernel Γ equal to the cube of Ω, where $\Omega = N(B(Bf))$, is an instance of the kernel schema $\Omega z\Omega$. We thus have another similarity since the superkernel $\Gamma_1(BW)$ is an instance of a kernel schema, but a schema in which the variable z occurs last. The schema $\Omega z\Omega$ yields many kernels that in turn lead to the construction of additional fixed point combinators from B and N alone.

Since $BN(BB)$ is an S, the set consisting of N, B, and K is complete for combinatory logic. The combinator N can be defined in terms of familiar combinators—N behaves like $B(BW)C$—which one can see by using the following equations for those familiar combinators.

$Bxyz = x(yz)$
$Wxy = xyy$
$Cxyz = xzy$

In contrast, S behaves like $B(BW)(BBC)$.

If we replace W by N in the square of combinators discussed in Section 1.3 to obtain

$B\ N$
$S\ L$

for study, we have a pleasing parallel. From B and N, one can construct—in abundance—fixed point combinators; from B and L, one cannot; from S and N, one cannot; but from S and L one can. For such a set of four combinators, we enjoy commenting—as we did in Section 1.3—that the replacement of either of B or N respectively by S or L loses the strong fixed point property, but replacement of both restores it. We thus again encounter the complexity, challenge, and intrigue of combinatory logic—metaphorically, two negatives make a positive, again. Incidentally, in Section 5.3.6, we show that the set consisting of the combinators N and L fails to satisfy the strong fixed point property, which parallels the behavior of the set consisting of W and L; the corresponding formal proofs of the two results are unalike.

With regard to less precise information, from what we have found in the literature, the combinator N has apparently not been studied; in particular, neither the logician Smullyan nor the logician Barendregt have studied the combinator N. As we shall indicate with examples—many of which we give in Section 5.3.5—for the use of kernels for constructing fixed point combinators, N appears to be far more useful than S is. With this summary of the significant results in hand, let us now turn to a more detailed account of what we found in our study of the set P consisting of B and N alone.

5.3.4.2. Fixed Point Combinators Constructible from *B* and *N*

As always, to study a set of combinators—in this case, the set *P* consisting of *B* and *N* alone—we apply the kernel method. Within the first few seconds of CPU time, the first stage of the two-stage method yields a number of kernels and kernel schemata. The first it finds—which might, and indeed should, naturally cause one to wonder how much guidance was given to the kernel method—is $N(B(Bf))z(N(B(Bf)))$. The implied question deserves an immediate answer—the only guidance we gave the method was that of using the 1's rule strategy, requiring all expansions to satisfy the 1's rule.

Now, if we borrow from our earlier notation and write $\Omega = N(B(Bf))$, then the first result obtained in stage 1 is the kernel schema $\Omega z\Omega$, which might remind one of a kernel schema we studied extensively earlier in the context of *B* and *W*—the schema $\Omega(BW)\Omega(BW)z$ with Γ_1 as an instance, where $\Omega(BW) = W(B(Bf))$. If we next instantiate the variable *z* in this *B*-and-*N* schema to Ω, we obtain the superkernel $\Omega\Omega\Omega$ with respect to the set *P* consisting of *B* and *N*, a superkernel which resembles $\Gamma_1(BW)$ except for the uniform replacement of *W* by *N*. Let us, therefore, call this new superkernel $\Gamma_1(BN)$.

Consistent with our objective of illustrating how research might be conducted, we next pose the three questions that led us further into the exploration of the *B*-and-*N* mine, questions that arise from our practice of imitating what has worked for other studies. First, can one use the second stage of the two-stage kernel method to construct from $\Gamma_1(BN)$ one or more fixed point combinators from *B* and *N* alone? Second, can one use the first stage of the three-stage kernel method to globally expand $\Gamma_1(BN)$ to obtain other kernels with respect to *B* and *N*? Third, can one apply the techniques used to extract additional gold from the *B*-and-*W* mine to extract additional gold from the *B*-and-*N* mine, if indeed the *B*-and-*N* mine does contain gold?

Of these three questions, the second can be attacked most easily. In fact, if one examines the results obtained by OTTER when that program applies the first stage of the two-stage kernel method to the study of *B* and *N*, one finds among the first 17 kernels and kernel schemata five expressions that tell an important part of the story. In particular, one finds $\Delta z\Delta$, where Δ ranges over the following five terms.

$\Omega_1(BN) = N(B(Bf))$
$\Omega_2(BN) = N((BBBf))$
$\Omega_3(BN) = BNB(Bf)$
$\Omega_4(BN) = BN(BBB)f$
$\Omega_5(BN) = B(BNB)Bf$

As one correctly guesses, by instantiating the variable *z* in the kernel schema under discussion to each of the expressions $\Omega_1(BN)$ through $\Omega_5(BN)$, we obtain five kernels. The second of the three questions is, therefore, answered in the affirmative—each of the five kernels can be obtained by applying global expansions with *B* to $\Gamma_1(BN)$, if we include the empty set of expansions. In other words, we can parallel our approach to the study of *B* and *W* and obtain, respectively, $\Gamma_1(BN)$ through $\Gamma_5(BN)$. Since no other useful kernels can be obtained from the superkernel $\Gamma_1(BN)$, the connection between the sets consisting respectively of *B* and *N* and *B* and *W* begins to tighten.

Before attempting to extend the parallel further by searching for additional superkernels, let us turn to the first of the three questions, that concerned with expanding kernels to the successful construction of

fixed point combinators. To answer that question—and, as usual, we assume tacitly that the kernel method is complete in the sense that is obviously behind the remark we are making—we first recall that the natural kernel that corresponds to a given fixed point combinator Θ is the first kernel one finds when one reduces Θf subject to the following constraint. One appends f to the combinator under consideration, where f is an arbitrarily chosen combinator, and then applies reductions with the elements of the set P under study, always choosing, when a choice is possible, the reduction that applies to a term furthest left in the expression being reduced.

Immediately, when we apply stage 2 of the two-stage kernel method, we encounter a difference between the power of $\Gamma_1(BW)$ and $\Gamma_1(BN)$. Specifically, from $\Gamma_1(BN)$, one can directly construct, among others, the following four fixed point combinators, two of which have length 13, and two of which have length 14.

$B(B(N(BB(N(BBN)N))N)B)B$
$B(B(N(BB(N(N(BB))N))N)B)B$
$B(B(N(N(B(BB)(N(BBN))))N)B)B$
$B(B(N(N(B(BB)(N(N(BB)))))N)B)B$

In contrast, the application of the second stage of the two-stage kernel method to $\Gamma_1(BW)$ yields the single fixed point combinator

$\Theta_1(BW) = B(B(B(WW)W)B)B,$

which has length 8.

Having succeeded in constructing four fixed point combinators, each of which has as its natural kernel the superkernel $\Gamma_1(BN)$—and recalling that we remarked earlier that one can in fact construct at least 52 such combinators—next in order is the consideration of the four kernels $\Gamma_2(BN)$ through $\Gamma_5(BN)$ derivable from $\Gamma_1(BN)$. Since each of the as-yet-unstudied kernels is, respectively, the cube of

$\Omega_2(BN) = N((BBBf))$
$\Omega_3(BN) = BNB(Bf)$
$\Omega_4(BN) = BN(BBB)f$
$\Omega_5(BN) = B(BNB)Bf,$

and since each of $\Gamma_2(BW)$ through $\Gamma_5(BW)$ is, respectively, the cube of

$\Omega_2(BW) = W((BBBf))$
$\Omega_3(BW) = BWB(Bf)$
$\Omega_4(BW) = BW(BBB)f$
$\Omega_5(BW) = B(BWB)Bf,$

one can, by recalling what occurs for B and W, quickly hazard a guess concerning fixed point combinators constructible from the four kernels under study. The natural guess is that the four kernels $\Gamma_2(BN)$ through $\Gamma_5(BN)$ have the same power that $\Gamma_1(BN)$ does—at least 52 combinators can be constructed from each of the four kernels. We conjecture that that guess is correct, but we have not yet succeeded in constructing all of the required combinators. In addition, one naturally guesses that each set conjectured to contain 52 combinators also contains two of length 13 and two of length 14 that are related to the corresponding combinators cited earlier in the context of the discussion of $\Gamma_1(BN)$. As strongly suggested by our

application of the kernel method, that guess is *not* correct—at least, not for all four kernels under consideration. Since we made the same guess, we were in no way prepared for this discovery. Specifically, when stage 2 of the two-stage kernel method is applied to $\Gamma_2(BN)$, among the fixed point combinators, one does find two of length 13 and two of length 14. However, when stage 2 is applied to constructing fixed point combinators from $\Gamma_3(BN)$ through $\Gamma_5(BN)$, no combinators of length 13 are found.

To avoid any suspense and to provide a glimpse of how mischievous B and N can be, here are the five fixed point combinators that we choose to be $\Theta_1(BN)$ through $\Theta_5(BN)$.

$B(B(N(BB(N(BBN)N))N)B)B$
$B(N(BB(N(BBN)N))N)(BBB)$
$B(N(N(B(BB)(N(N(BB)))))(BNB))B$
$N(N(B(BB)(N(N(BB)))))(BN(BBB))$
$N(N(B(BB)(N(N(BB)))))(B(BNB)B)$

With these five fixed point combinators, we see that the first of the three questions posed earlier is also answered in the affirmative. Of course, since expansion applied to three of the kernels under study does not lead to any combinators of length 13 or less, we have no choice about including combinators of length greater than 13 in the first class of those classes whose union is $\Theta(BN)$, the class of combinators whose natural kernels are, respectively, $\Gamma_1(BN)$ through $\Gamma_5(BN)$.

Given the fact that the kernel method fails to construct fixed point combinators of length 13 for three of the first five kernels $\Gamma_1(BN)$ through $\Gamma_5(BN)$, one naturally guesses that the picture for B and N—regarding the aspect of the study focusing on invariants—mirrors, only in a distorted fashion, the picture for B and W. By studying the following four fixed point combinators, one can gain some insight into how much distortion is applied to the picture for B and W to obtain the picture for B and N.

$B(B(B\ \ (WW)\ \ W)B)B$
$B(B(B(B\ \ (W(WW))\ \ W)B)B)B$
$B(B(B(B(B\ \ (W(W(WW)))\ \ W)B)B)B)B$
$B(B(B(B(B(B\ \ (W(W(W(WW))))\ \ W)B)B)B)B)B$

These four combinators are, respectively, the first, sixth, twentieth, and sixty-second elements of $\Theta(BW)$. In other words, these four combinators have as their respective natural kernels $\Gamma_1(BW)$, $\Gamma_6(BW)$, $\Gamma_{20}(BW)$, and $\Gamma_{62}(BW)$. In addition, except for the first of the four combinators—which is the only fixed point combinator that naturally maps to $\Gamma_1(BW)$—these four are the ones selected from sets of fixed point combinators, where each set maps to a single kernel. To see how the picture is distorted when one replaces the set of combinators consisting of B and W with the set consisting of B and N, let us give a somewhat detailed discussion of part of the recommended study of the four cited fixed point combinators selected from $\Theta(BW)$. By including the details, we prepare the way for one to understand how the pair B and N behaves in comparison to the pair B and W.

A glance at the first of the four combinators, $\Theta_1(BW)$, immediately gives one some of the invariants pertinent to the entire set of five combinators consisting of $\Theta_1(BW)$ through $\Theta_5(BW)$. Indeed, if we again write $\Theta_1(BW)$ as

$B(B(B\ \ (WW)\ \ W)B)B$,

its three segments—initial, middle, and final—in effect summarize the important facts. Specifically, the first five fixed point combinators in the set $\Theta(BW)$ each contain five occurrences of B and three of W. In addition, to obtain any of the other four combinators from $\Theta_1(BW)$, one replaces the final segment of $\Theta_1(BW)$ by the generator of the kernel corresponding to the desired combinator—of, course, ignoring f—and removes the appropriate number of occurrences of B from the initial segment. For example, to obtain $\Theta_5(BW)$ whose kernel is generated by $B(BWB)Bf$, one takes

$$B(B(B\ \ (WW)\ \ W)B)B$$

and removes two occurrences of B from the initial segment, repeats the middle segment, and replaces the final segment by the significant part of the generator of the appropriate kernel to obtain

$$B\ \ (WW)\ \ (B(BWB)B).$$

We can apply the same type of analysis to any of the classes whose union is $\Theta(BW)$ and discover some of the invariants for the members of that class. Even further, such an analysis leads to the discovery of some of the invariants for all of the fixed point combinators whose natural kernel is one of the kernels that corresponds to an element in the class under consideration. For example, if we analyze $\Theta_6(BW)$ written in its segmented form

$$B(B(B(B\quad (W(WW))\ \ W)B)B)B,$$

we can correctly conclude that each member of $\Theta(BW)$ whose kernel is obtainable by global expansion applied to $\Gamma_6(BW)$ has as its middle segment $W(WW)$, contains seven occurrences of B and four occurrences of W, and has an initial and final segment that are completely determined by the generator of the corresponding kernel.

In addition, we can select a kernel from those that lead to the construction of the infinite set $\Theta(BW)$ and immediately give some of the properties of the fixed point combinators that are *not* members of $\Theta(BW)$ but that have as their natural kernel a kernel that is derived from the same superkernel as the selected kernel is. For example, recalling that from each of the kernels $\Gamma_6(BW)$ through $\Gamma_{19}(BW)$ one can construct a set of four combinators consisting of one of length 11 and two of length 12 and one of length 13, one immediately finds that each of the 56 contains exactly four occurrences of W. With a bit more care, one finds that an analysis of the changes one makes to obtain the segmented representation of $\Theta_6(BW)$ through $\Theta_{19}(BW)$ shows what changes are required to obtain the segmented representation for the remaining 42 combinators that naturally map to one of the 14 kernels $\Gamma_6(BW)$ through $\Gamma_{19}(BW)$.

In particular, one can partition the 42 combinators into three sets of combinators such that each contains 14 members, one for each of the 14 kernels. The method that can be used to obtain each set is essentially that which we applied to

$$B(B(B(B\quad (W(WW))\ \ W)B)B)B$$

to obtain the additional 13 combinators, each of which corresponds to one of the kernels obtainable by applying global expansions to $\Gamma_6(BW)$. However, one applies the method to each of the following three combinators one at a time.

$$B(B(B(B\quad (W(BWW))\ \ W)B)B)B$$
$$B(B(B(B\quad (W(W(BW)))\ \ W)B)B)B$$

$B(B(B(B \quad (W(BW(BW))) \quad W)B)B)B$

The given three combinators are those of length 12, 12, and 13 such that, when considered with $\Theta_6(BW)$, they form the set of four fixed point combinators that can be obtained by applying stage 2 of the two-stage kernel method to $\Gamma_6(BW)$.

Having reviewed in some detail how the pair B and W behaves—with particular attention to some of the invariants—we can now give the corresponding analysis of how the pair B and N behaves. To begin with—in contrast to the first five fixed point combinators in $\Theta(BW)$—the first five fixed point combinators $\Theta_1(BN)$ through $\Theta_5(BN)$, if written in segmented notation, do not share a common middle segment. Here are the first five combinators of $\Theta(BN)$ written in segmented notation.

$B(B(N \quad (BB(N(BBN)N)) \; N)B)B$
$B(N \quad (BB(N(BBN)N)) \; N)(BBB)$
$B(N \; (N(B(BB)(N(N(BB))))) \; (BNB))B$
$N \; (N(B(BB)(N(N(BB))))) \; (BN(BBB))$
$N \; (N(B(BB)(N(N(BB))))) \; (B(BNB)B)$

As one can see, the first two of the five share a common middle segment, and the last two share a common but different middle segment. The explanation for this pairing—and for the fact that the first five combinators in $\Theta(BW)$ share a common middle segment—may rest with the fact that one can think of the combinator B as a composition combinator. In particular, one can replace Bxy by x circle (as an operator) y for all x and y, and then show that circle is associative. One can apply such a replacement to rewrite combinators in which B occurs in terms of circle. If, for example, one rewrites $\Theta_1(BW)$ through $\Theta_5(BW)$ in this manner, one finds that the five combinators present the five ways one can associate four letters; for this example, the four so-called letters are WW, W, B, and B. Perhaps one can heavily use the circle notation to extract various secrets, for example, to explain why exactly five irreducible combinators of length 8 exist constructible from B and W alone. Application of the same type of analysis might reveal vital information concerning how B and N work together.

Just as the segmented forms of $\Theta_1(BW)$ through $\Theta_5(BW)$ reflect the nature of the generator of the corresponding kernel and what is needed to free the combinator from f—which can be seen by examining the final and initial segments, respectively—so also do the corresponding forms for $\Theta_1(BN)$ through $\Theta_5(BN)$. However, in contrast to the pair B and W for which one merely applies an appropriate number of B's to free f, for the pair B and N, the situation is clearly more complicated. The fact that N is a permuter and W is not may be the explanation.

If one next attempts to extend the comparison of B and N with B and W by focusing on other fixed point combinators by selecting, for example, a fixed point combinator whose natural kernel is $\Gamma_2(BN)$, and if one compares it to the corresponding fixed point combinator whose natural kernel is $\Gamma_1(BN)$, one finds that both contain the same number of occurrences of B and the same number of occurrences of N, and that the two combinators differ only in the initial and final segments. Here are four pairs of fixed point combinators, where each pair consists of a combinator that naturally maps to $\Gamma_1(BN)$ and its correspondent that naturally maps to $\Gamma_2(BN)$.

$B(B(N(BB(N(BBN)N))N)B)B$

$B(N(BB(N(BBN)N))N)(BBB)$

$B(B(N(BB(N(N(BB))N))N)B)B$
$B(N(BB(N(N(BB))N))N)(BBB)$

$B(B(N(N(B(BB)(N(BBN))))N)B)B$
$B(N(N(B(BB)(N(BBN))))N)(BBB)$

$B(B(N(N(B(BB)(N(N(BB)))))N)B)B$
$B(N(N(B(BB)(N(N(BB)))))N)(BBB)$

To find the corresponding four-element sets for the other three kernels, $\Gamma_3(BN)$ through $\Gamma_5(BN)$, one *cannot* simply imitate what one does for the 14 kernels $\Gamma_6(BW)$ through $\Gamma_{19}(BW)$. One must, instead, find a way to cope with the fact that $\Theta_3(BN)$ through $\Theta_5(BN)$ each have length 14, and not length 13 as do $\Theta_1(BN)$ and $\Theta_2(BN)$. In other words, the fixed point combinators that one can construct from B and N that appear to play the role that the first five fixed point combinators in $\Theta(BW)$ play are not nearly as well behaved as one would prefer. Nevertheless, we can and shall show how one can enumerate an infinite set $\Theta(BN)$ that, in a general sense, mirrors some of the important properties possessed by the infinite set $\Theta(BW)$.

5.3.4.3. Enumerating Fixed Point Combinators Constructible from B and N

If one next turns to a possible enumeration of the fixed point combinators constructible from B and N alone, an enumeration that parallels the enumeration of $\Theta(BW)$, one encounters two obvious obstacles. The first obstacle rests with the fact that, even for each of the first five kernels $\Gamma_1(BN)$ through $\Gamma_5(BN)$, it appears that one can construct at least 52 irreducible fixed point combinators. Such abundance is in sharp contrast to the fact that one can construct only a single irreducible fixed point combinator from each of $\Gamma_1(BW)$ through $\Gamma_5(BW)$. In other words, one is immediately faced with choosing from among a set of combinators in the same way we chose from the four combinators whose natural kernel is $\Gamma_6(BW)$. The rule that led us to choosing which combinator to call $\Theta_6(BW)$ requires following an expansion path that focuses on the shortest term that can be expanded, from among those paths that lead to an irreducible combinator. From a mechanistic viewpoint, the rule we used says to choose the shortest of the four combinators whose natural kernel is $\Gamma_6(BW)$, which is the combinator of length 11. We use the same rule to choose $\Theta_7(BW)$ through $\Theta_{19}(BW)$, all of which have length 11, and also to choose the 42 combinators of length 14 that come next in the enumeration.

The rule we use for choosing which combinators to be $\Theta_1(BN)$ through $\Theta_5(BN)$ is more complicated, as one might expect since two of the five sets from which we wish to choose contain two combinators of length 13 and since the other three sets contain no combinators of length 13. And here we encounter the second obstacle to enumerating the set we intend to call $\Theta(BN)$—the first class of combinators contains five elements as does the first class for B and W, but, unlike the class for B and W, the five combinators constructible from B and N are *not* all of the same length. Since we cannot apply the rule that we used for choosing, for example, which combinators to call $\Theta_6(BW)$ through $\Theta_{19}(BW)$—no unique shortest combinator is available for our enumeration of $\Theta(BN)$—we must use a more complicated rule.

The rule we use says to choose the combinator whose expansion path from its kernel most quickly removes occurrences of the arbitrary combinator f, if a unique combinator can be obtained in this way; to break a tie, should one occur, one chooses the combinator obtained by pursuing an expansion path that involves the shortest term that can be expanded. Of course, one ignores expansion paths that lead to reducible combinators. The motivation for our rule can be traced to the desire to produce as quickly as possible an expression of the form Θf for some Θ that is free of any occurrences of f. With this rule—as noted earlier—the following five combinators are chosen in order as $\Theta_1(BN)$ through $\Theta_5(BN)$.

$B(B(N(BB(N(BBN)N))N)B)B$
$B(N(BB(N(BBN)N))N)(BBB)$
$B(N(N(B(BB)(N(N(BB)))))(BNB))B$
$N(N(B(BB)(N(N(BB)))))(BN(BBB))$
$N(N(B(BB)(N(N(BB)))))(B(BNB)B)$

For the next step of the enumeration of the infinite set we are calling $\Theta(BN)$, we need the next class of fixed point combinators, which means we need the next superkernel and the kernels that are obtainable from it by global expansion. To obtain the next superkernel, which we shall call $\Gamma_6(BN)$, we can apply the technique used to complete the construction of $\Theta(BW)$. Therefore, for the second superkernel, we simply take the generator of $\Gamma_1(BN)$, which is $N(B(Bf))$, insert a B to the right of N, fully right associate the result to obtain a new generator $\Omega_6(BN)$, take the fully left-associated fourth power of $\Omega_6(BN)$, and obtain the next superkernel $\Gamma_6(BN)$.

When one applies global expansion with B to this superkernel—as with $\Gamma_6(BW)$—one obtains 13 additional kernels, which we, as expected, call $\Gamma_7(BN)$ through $\Gamma_{19}(BN)$. If one applies stage 2 of the two-stage kernel method to the sixth through the nineteenth kernels one at a time, one obtains from each a set of fixed point combinators. One can then apply our rule for choosing, from each set considered in the obvious order, one fixed point combinator to be called, respectively, $\Theta_6(BN)$ through $\Theta_{19}(BN)$.

We find it instructive to suggest, at this point, that one make a guess about the lengths of the 14 fixed point combinators to be assigned the sixth through the nineteenth positions in $\Theta(BN)$. Our guess was that each would have length 19. We reasoned that one could apply an expansion path similar to that used to construct $\Theta_1(BN)$. In particular, we were convinced that one could construct from, for example, $\Gamma_6(BN)$ a fixed point combinator by beginning with an expansion with B to isolate N from the second occurrence of the generator $N(B(B(Bf)))$. We were right; one can construct a fixed point combinator in this fashion. Indeed, one can pursue an expansion path that begins in this manner and proceeds rather like that leading to the construction of $\Theta_1(BN)$. If one follows this approach, one does in fact construct a fixed point combinator of length 19.

However—and here we have an excellent example of why we strongly suggest that theorem-proving programs *not* be designed to imitate the way a person might attack problems requiring reasoning—shorter combinators can be constructed from $\Gamma_6(BN)$. For example, a person could easily—and we in fact did—apply the type of reasoning to produce the sixth fixed point combinator that was used to construct the first. By way of clarification, we still recommend that a person imitate earlier success; however, we recommend against a theorem-proving program imitating a mathematician or logician. By following this suggestion, we suspect that the team that results—that consisting of the scientist and the computer

program—can be far more effective than either would be alone.

Perhaps, during the time between our suggesting that one make a guess about the lengths of the sixth through the nineteenth combinators of $\Theta(BN)$ and this point in the report—even though little actual time has no doubt elapsed—a guess has been made. If one has not made such a guess—but one wishes to do so—note that we are about to give the sixth through the nineteenth kernels, each coupled with the fixed point combinator that naturally maps to it and that is, respectively, the sixth through the nineteenth combinator in our enumeration of $\Theta(BN)$.

N(B(B(Bf)))(N(B(B(Bf))))(N(B(B(Bf))))(N(B(B(Bf))))
B(B(B(N(BB(B(NN)N))N)B)B)B

N(B(BBBf))(N(B(BBBf)))(N(B(BBBf)))(N(B(BBBf)))
B(B(N(BB(B(NN)N))N)B)(BBB)

N(BBB(Bf))(N(BBB(Bf)))(N(BBB(Bf)))(N(BBB(Bf)))
B(B(N(BB(B(NN)N))N)(BBB))B

N(BB(BBB)f)(N(BB(BBB)f))(N(BB(BBB)f))(N(BB(BBB)f))
B(N(BB(B(NN)N))N)(BB(BBB))

N(B(BBB)Bf)(N(B(BBB)Bf))(N(B(BBB)Bf))(N(B(BBB)Bf))
B(N(BB(B(NN)N))N)(B(BBB)B)

BNB(B(Bf))(BNB(B(Bf)))(BNB(B(Bf)))(BNB(B(Bf)))
B(B(N(N(B(BB(NN))))(BNB))B)B

BNB(BBBf)(BNB(BBBf))(BNB(BBBf)(BNB(BBBf))
B(N(N(B(BB(NN))))(BNB)(BBB)

BN(BBB)(Bf)(BN(BBB)(Bf))(BN(BBB)(Bf))(BN(BBB)(Bf))
B(N(N(B(BB(NN))))(BN(BBB)))B

B(BNB)B(Bf)(B(BNB)B(Bf))(B(BNB)B(Bf))(B(BNB)B(Bf))
B(N(N(B(BB(NN))))(B(BNB)B))B

BN(BB(BBB))f(BN(BB(BBB))f)(BN(BB(BBB))f)(BN(BB(BBB))f)
N(N(B(BB(NN))))(BN(BB(BBB)))

BN(B(BBB)B)f(BN(B(BBB)B)f)(BN(B(BBB)B)f)(BN(B(BBB)B)f)
N(N(B(BB(NN))))(BN(B(BBB)B))

B(BNB)(BBB)f(B(BNB)(BBB)f)(B(BNB)(BBB)f)(B(BNB)(BBB)f)
N(N(B(BB(NN))))(B(BNB)(BBB))

B(BN(BBB))Bf(B(BN(BBB))Bf)(B(BN(BBB))Bf)(B(BN(BBB))Bf)
N(N(B(BB(NN))))(B(BN(BBB))B)

B(B(BNB)B)Bf(B(B(BNB)B)Bf)(B(B(BNB)B)Bf)(B(B(BNB)B)Bf)

$N(N(B(BB(NN))))(B(B(BNB)B)B)$

Each of the given combinators has length 14, which was indeed a result we did not expect. We therefore have an example of a set P of combinators—that consisting of B and N alone—such that there exist two kernels with respect to P with the property that, although the two kernels are derived from different superkernels, the length of the smallest fixed point combinator that corresponds to each of the two kernels is the same. Specifically, we can choose any of $\Gamma_3(BN)$ through $\Gamma_5(BN)$ for one kernel, and choose any of $\Gamma_6(BN)$ through $\Gamma_{19}(BN)$ for the other. After all, each of $\Theta_3(BN)$ through $\Theta_{19}(BN)$ has length 14, and each fixed point combinator whose length is less than 14 has as its natural kernel a kernel other than the specific ones under consideration.

As one might suspect, we cannot say at this time how rare such a phenomenon is. We do, obviously, have a sharp contrast to the behavior of the kernels, with respect to the set consisting of B and W, that we have studied. As we discovered, expansion applied to the kernels from which one can construct $\Theta(BW)$ yields—if one selects a shortest fixed point combinator for each kernel—an infinite class of finite sets which consist, respectively, of combinators of length 8, 11, 14, 17, and the like. An immediate question that most likely arises concerns the corresponding properties of the third, fourth, and later classes whose union is $\Theta(BN)$. Let us now turn to that question; by doing so, we can provide the information that one can use to complete the enumeration of $\Theta(BN)$.

As one might expect, we shall apply the approach we used for obtaining $\Gamma_6(BN)$. For the third superkernel $\Gamma_{20}(BN)$, we simply take the generator of $\Gamma_6(BN)$, which is $N(B(B(Bf)))$, insert a B to the right of N, fully right associate the result to obtain a new generator $\Omega_{20}(BN)$, and take the fifth power of $\Omega_{20}(BN)$. When one applies global expansion with B to this superkernel—as with $\Gamma_{20}(BW)$—one obtains 41 additional kernels, which we, as expected, call $\Gamma_{21}(BN)$ through $\Gamma_{61}(BN)$. One can apply stage 2 to the 42 kernels, one at a time, to construct fixed point combinators, and then apply our rule for choosing from each set considered in the obvious order one fixed point combinator to be called, respectively, $\Theta_{20}(BN)$ through $\Theta_{61}(BN)$. To complete the enumeration of $\Theta(BN)$, one can, as with B and W, continue to obtain the next superkernel, the kernels derivable from it by global expansion, and the appropriate fixed point combinators, and assign those combinators their proper place in $\Theta(BN)$.

To close this section with a capsule summary of some of the differences between and some of the invariants shared by the sets consisting, respectively, of B and W and B and N, we give the segmented representation of the first, sixth, nineteenth, and sixty-second fixed point combinators in $\Theta(BW)$ followed by that for the corresponding combinators in $\Theta(BN)$.

$$
\begin{array}{l}
B(B(B\quad(WW)\quad W)B)B \\
B(B(B(B\quad(W(WW))\quad W)B)B)B \\
B(B(B(B(B\quad(W(W(WW)))\quad W)B)B)B)B \\
B(B(B(B(B(B\quad(W(W(W(WW))))\quad W)B)B)B)B)B
\end{array}
$$

$$
\begin{array}{l}
B(B(N\quad(BB(N(BBN)N))\quad N)B)B \\
B(B(B(N\quad(BB(B(NN)N))\quad N)B)B)B \\
B(B(B(B(N\quad(BB(B(N(NN)N))\quad N)B)B)B)B \\
B(B(B(B(B(N\quad(BB(B(N(N(NN))N))\quad N)B)B)B)B)B
\end{array}
$$

In contrast to using superkernels to derive kernels which are then used to construct fixed point combinators of lengths 8, 11, 14, and the like—as occurs with $\Theta(BW)$—use of the superkernels one obtains for B and N leads to the construction of fixed point combinators of lengths 13, 14, 17, 20, and the like. Consistent with what one might guess if one were to extrapolate from the study of B and W, one can obtain a sequence of finite classes of kernels such that the number of elements in a class is equal to the number of ways to associate n letters, where n is the number of symbols in the generator of the superkernel from which the kernels in a class are derived. All of the combinators that are so constructed are irreducible. In fact, this irreducibility property is present even for the fixed point combinators that do not participate in the enumeration—those combinators we choose not to select from those whose natural kernel correspondent is in one of the finite classes of kernels on which we are focusing.

5.3.4.4. Additional Fixed Point Combinators Constructible from B and N

With regard to the third question, that concerned with exploring and extracting additional fixed point combinators from the B-and-N mine in a manner similar to the way we explored and extracted additional gold from the B-and-W mine, we can answer that question in the affirmative—we can apply similar techniques. Two techniques, other than that used to obtain the sequence of superkernels discussed in the preceding section, were used to extract gold from the B-and-W mine, each of which is a candidate for extracting fixed point combinators from the B-and-N. The first of the two techniques succeeds in constructing an infinite set of fixed point combinators that act as companions to the members of $\Theta(BW)$. As discussed in Section 4.7, the technique consists of replacing, in each generator of each of the relevant kernels, W by BW to obtain 5 fixed point combinators of length 9, 14 of length 12, 42 of length 15, ... If one attempts to apply this technique to the study of B and N—specifically, to the kernels that generate $\Theta(BN)$—the sought-after companion set of fixed point combinators is found. However, in contrast to working for all of $\Theta(BW)$, this technique does not produce companions for the first five combinators in $\Theta(BN)$.

By focusing on the superkernel $\Gamma_1(BW)$, which is the cube of $W(B(Bf))$, we can quickly see why the given technique succeeds for all of $\Theta(BW)$. When one applies W as the first step in the set of reductions that proves that $\Gamma usb1(BW)$ is in fact a kernel, the third occurrence of the generator is not part of the term being reduced. Therefore, when one replaces W by BW in all occurrences of the generator, the reduction with W, which follows the reduction with B, has room to apply; indeed, its application involves the third occurrence of the generator. In contrast, we see why the replacement of N by BN in all occurrences of the generator of $\Gamma_1(BN)$ fails to produce a kernel. Specifically, when one applies N as a reduction on the way to proving that $\Gamma_1(BN)$, which is the cube of $N(B(Bf))$, is a kernel, the term that N reduces involves all occurrences of the generator of the kernel. Therefore, the replacement of N by BN produces an expression that does not allow, after B is applied as a reduction, N to apply; too few terms exist for N to apply.

At this point, one might—simply and understandably—suspect that no hope exists for constructing a companion set to $\Theta(BN)$, regardless of the mechanism to be used or the properties the companion set might possess. Indeed, until the writing of this particular paragraph, we thought this was the case. However, if one were unusually persistent and attempted to carefully analyze what goes wrong, keeping in mind how we discovered the combinator N, one could discover—as we have just discovered—that some type of companion set does in fact exist. In particular, if one begins by focusing on $\Gamma_6(BN)$, which is the

fourth power of $N(B(B(Bf)))$, rather than on $\Gamma_1(BN)$, and then replaces N by BN in all occurrences of the generator, one indeed obtains a kernel that is a companion to $\Gamma_6(BN)$. Currently, we have not studied the relationship of the companion combinators, and, therefore, simply give our conjecture concerning their properties.

Where

$\Theta_6(BN) = B(B(B(N(BB(B(NN)N))N)B)B)B,$

its companion is

$B(B(B(N(N(B(BB(NN))))(BN))B)B)B.$

This companion to $\Theta_6(BN)$ is chosen by applying the same rule as that used to select the elements of $\Theta(BN)$—in particular, when possible, choose the shortest combinator that naturally maps to the kernel under consideration. We conjecture that, if one ignores the first five combinators, each companion to one of the fixed point combinators in the set $\Theta(BN)$ contains one more occurrence of B. In contrast to the companions for the elements of $\Theta(BW)$, which one can obtain by merely replacing the final occurrence of W by BW, the companions to the sixth and later elements of $\Theta(BN)$ exhibit a replacement of the last occurrence of N by BN, but also exhibit a modification to their middle segments. If we were content to dispense with the requirement of selecting the shortest fixed point combinator constructible from the kernel under consideration, then it appears that we could produce a companion set of combinators such that each companion is obtained from its mate by replacing the final two occurrences of N by BN.

We close our current examination of the fixed point combinators constructible from B and N alone by focusing on the final technique we used for extracting fixed point combinators from the B-and-W mine. Just as other instances of $\Omega_1(BW)\Omega_1(BW)z$ mark the entrance to galleries in the B-and-W mine whose exploration proved profitable, the study of various instances of $\Omega_1(BN)z\Omega_1(BN)$ lead to a wide variety of discoveries. Indeed, since N is a permuter and W is not, if one thoroughly explores the B-and-N mine, one will certainly encounter phenomena that are different from those one finds in the B-and-W mine. For example, when applying the second stage of the two-stage kernel method to the construction of the four combinators (given earlier) that naturally correspond to $\Gamma_1(BN)$, one must expand terms that contain no occurrences of f. Such is not the case for B and W. In addition, if one looks closely at the result of those expansions of terms that are free of f, and what it leads to, one finds that the study of B and N indeed contains some surprises.

Specifically, one finds intermediate expressions—between the kernel and the constructible fixed point combinator—that one might ordinarily discard on the grounds that they cannot possibly lead to interesting and irreducible combinators. The combinator

$B(B(N(N(B(BB)(N(BBN))))N)B)B$

is a perfect example; on the expansion path that leads to its construction, one encounters an expression containing a term that is free of f and that reduces. Nevertheless, by the time the construction has been completed, one obtains—as can be seen by inspection—a combinator that is irreducible. In other words, the pair B and N do behave quite differently from the pair B and W.

An additional example of this different behavior is provided by a brief comparison of the properties of the minimal kernel $W(Bf)(W(Bf))$ with the properties of the minimal kernel

$N(B(Bf))(N(B(B(Bf))))(N(B(B(Bf))))$. Although both kernels are in fact superkernels, the first of the two is useless for the construction of fixed point combinators when the construction is confined to the combinators mentioned in it, where the second of the two is useful for the corresponding construction. Perhaps the explanation rests with the fact that $N(Bf)(N(Bf))$ is not a kernel. However, since we discovered N in our attempt to force B and S to have a kernel, let us see what can be done with the kernel $W(Bf)(W(Bf))$ when one of S or N is adjoined to the set of combinators to be used in the search for fixed point combinators. By taking this action, we shall also gain a small glimpse of the relative power of S versus that of N.

From the set consisting of B, W, and S, the only fixed point combinator found by the kernel method whose natural kernel is $W(Bf)(W(Bf))$ is $B(SWW)B$. In contrast, from the set consisting of B, W, and N, the kernel method constructs both $B(N(BBW)W)B$ and $B(N(N(BB))W)B$ as fixed point combinators whose natural kernel is $W(Bf)(W(Bf))$.

As for the first kernel $N(B(Bf))(N(B(Bf)))(N(B(Bf)))$ in the sequence that leads to the construction of $\Theta(BN)$, although it is obviously of use, this kernel also presents an anomaly just as the kernel $W(Bf)(W(Bf))$ does. In particular, the two of the five fixed point combinators $\Theta_1(BN)$ through $\Theta_5(BN)$, derived from this superkernel and the four kernels obtained by applying global expansion to it, each have length $4k+1$ for $k = 3$, in contrast to the remaining elements of $\Theta(BN)$ each of which has length $4k+2$ for $k \geq 3$.

If we pursue this line of inquiry somewhat further, we find that an analysis of the fixed point combinators starting with $\Theta_6(BN)$ leads to the discovery of another property shared by $\Theta(BN)$ and $\Theta(BW)$. The property concerns the differences between elements selected from the adjacent finite sets whose union is, respectively, $\Theta(BN)$ and $\Theta(BW)$. With the exception of the first set of five elements in $\Theta(BN)$, the elements of the $n+1$-st set differ from the elements of the n-th set by containing two more occurrences of B and, respectively, one more occurrence of N or W. We thus see why the sets whose union is $\Theta(BW)$ consist of combinators of lengths 8, 11, 14, and the like, and those whose union is $\Theta(BN)$ consist of combinators of lengths 14, 17, 20, and the like, if we ignore $\Theta_1(BN)$ and $\Theta_2(BN)$. Both N and W, when used as expansions to construct the appropriate fixed point combinators, remove or collapse occurrences of the generator of the corresponding kernel. However, for the first set of five kernels that are relevant to $\Theta(BN)$, insufficient room exists for N to be applied in this manner; instead, N must be used after B is applied to separate the first symbol of a generator from the remaining symbols.

If the combinator N does in fact offer more power for constructing fixed point combinators than the combinator S does, the explanation may be the following. In many of our experiments that succeed in constructing a kernel, we find that the kernel is expressible as the fully left-associated power of some generator, and the generator is itself a fully right-associated expression. Obviously, fully left-associated expressions more readily expand with a combinator such as N than with a combinator such as S since the right side of the equation for N is fully left associated and that for S is not; in the obvious sense, N can be thought of as the fully left-associated version of S.

From this observation, recalling that $BN(BB)$ is an S, we quickly have a basis for the suspicion that the combination of B and N is more powerful than is S, when the object is that of constructing fixed point combinators. For example, we have demonstrated that the set consisting of the combinators B and N satisfies the strong fixed point property, and satisfies it abundantly. On the other hand—although we have

no formal proofs—we conjecture that the set consisting of *B* and *S* fails to satisfy the strong fixed point property, and even fails to satisfy the weak fixed point property.

In addition to our earlier remarks, perhaps part of the explanation for the power of the combination of *B* and *N* rests with the fact that the combinator *B* is especially useful in expanding fully right-associated expressions, expressions such as the type of generator we frequently encounter. Therefore, the complete system consisting of the combinators *B*, *N*, and *K* may be easier to use for constructing fixed point combinators than is the complete system consisting of *S* and *K*.

What other marvels await one who explores the *B*-and-*N* mine thoroughly we cannot say, for we have only begun our own exploration of the *B*-and-*N*.

5.3.5. Experiments with the Kernel Method

In this section, we show how the general-purpose automated theorem-proving program OTTER was used to implement the two-stage kernel method, and how the kernel method was used to search for fixed point combinators from pairs of proper combinators. The choice of most of the pairs of proper combinators on which we focus was motivated by Smullyan's question (first answered by Statman) on the possibility of constructing a fixed point combinator from *B* and one other regular combinator. (We recall that a *combinator is regular if the first variable on the right side of its reduction rule or equation is the same as* the first variable on the left side of the reduction rule and if that variable occurs exactly once on the right side of the reduction rule. The *order* of a proper combinator is the number of variables on the left side of its reduction rule.)

5.3.5.1. Implementation of the Kernel Method

Each of the two stages of the kernel method requires a separate computer run with the program. We use the combinators *B* and *W* to illustrate the method. Stage 1 uses the following input clauses.

(1) EQUAL(x,x)
(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))
(3) EQUAL(a(a(W,x),y),a(a(x,y),y))
(4) ¬EQUAL(y,a(f,y)) | ANSWER(y)

The program was directed to do the following for stage 1:

a. Paramodulate (without the 1's rule) from the right sides of clauses (2) and (3) into the right sides of negated equality clauses, producing new negated equalities.

b. At each iteration, paramodulate into the negated equality with the fewest number of symbols.

c. When a negated equality is found to contradict clause (1), (2), or (3), output the kernel (which is in the ANSWER literal) so that it can be used in stage 2.

d. Stop the search after a specified number of seconds (rather than when the first contradiction is found—the normal termination criterion).

The set of kernels from a stage 1 search was then collected. A typical set was large, containing sometimes as many as several hundred kernels, so a means was required for selecting promising kernels. We used the rule: Discard kernels that are reducible with a nonreplicating combinator (such as *B*); then

discard kernels that are reducible via a reduction that is not a head reduction—not a reduction that satisfies the 1's rule. The result was typically a set of less than ten kernels.

Stage 2 was run once for each kernel until fixed point combinators were found, or until the allotted time was exhausted. From stage 1, the kernel $W(Bf)(W(Bf))$ and the kernel schema $W(B(Bf))(W(B(Bf)))z$ were used. Use of the kernel $W(Bf)(W(Bf))$ does *not* lead to fixed point combinators because this kernel lacks the needed power; but use of the kernel schema *does* lead to fixed point combinators because its use yields instances that are kernels with sufficient power.

Where FPF(y) means that y is a fixed point of f, a stage 2 search with the kernel schema uses the following input clauses.

(1) ¬FPF(a(y,f)) | ANSWER(y)
(2) EQUAL(a(a(a(B,x),y),z),a(x,a(y,z)))
(3) EQUAL(a(a(W,x),y),a(a(x,y),y))
(4) FPF(a(a(a(W,a(B,a(B,f))),a(W,a(B,a(B,f)))),x))

The program was directed to do the following for stage 2:

a. Paramodulate, using the 1's rule, from the right sides of clauses (2) and (3) into positive unit clauses whose predicate is FPF.

b. At each iteration, paramodulate into the positive FPF unit clause with the fewest number of symbols.

c. When a positive P unit clause is found to contradict clause (1), output the possible fixed point combinator (which is in the ANSWER literal).

d. Stop the search after a specified number of seconds (rather than when the first contradiction is found—the normal termination criterion).

The set of possible fixed point combinators from a stage 2 search was then collected. One can see that a member, say Θ when it is free of f, of the set is in fact a fixed point combinator by noting that the program has found a Θ such that, for arbitrary f, $\Theta f \twoheadrightarrow \Gamma \twoheadrightarrow f\Gamma$, where $a \twoheadrightarrow b$ means "a reduces to b". Since $\Theta f \twoheadrightarrow \Gamma$, we have $f(\Theta f) \twoheadrightarrow f\Gamma$. Since Θf and $f(\Theta f)$ both reduce to $f\Gamma$, we have $\Theta f = f(\Theta f)$.

5.3.5.2. Application of the Kernel Method

We have applied the kernel method to various cases. One of the more interesting is that in which we conducted a systematic study, searching for fixed point combinators from pairs of combinators consisting of B and one other proper combinator. The other combinator was required to be noneliminating and required to replicate exactly one of its variables. Of the cases in which the other combinator is regular we considered those of orders 1, 2, and 3; there exist none of order 1, 2 (W and L) of order 2, and 30 (including S, N, and H) of order 3. Of the cases in which the other combinator is irregular, we considered those of orders 1 and 2; there exists one (M) of order 1, and 10 (including O and W^1) of order 2.

5.3.5.3. Summary of Results

When we use the term "sage" we are borrowing the term from Smullyan [Smullyan84]; he uses this term for a fixed point combinator.

Table 1. Sages from B and One Other Regular Combinator

Combinator/Rule	Fixed Point Combinator with B
$Wxy = xyy$	$B(B(B(WW)W)B)B$
$Lxy = x(yy)$	none exists (Theorem 9)
$Nxyz = xzyz$	$B(B(B(N(BNB)x)N)B)B$
$Hxyz = xyzy$	$H(B(H(HB)B)B)(HH)$
$N_1xyz = xyyz$	none exists (Theorem 9)
$N_2xyz = xzzy$	$B(B(N_2N_2(N_2N_2))B)B$
$W^*xyz = xyzz$	$B(B(W^*(W^*(W^*B))B)(W^*B))B$
$N_3xyz = xzyy$	$B(B(B(N_3BB)(N_3(N_3N_3N_3)))B)B$
$N_4xyz = x(yz)z$	$B(B(N_4(N_4(N_4B)(N_4(N_4B)))N_4)B)B$
$N_5xyz = x(zy)z$	? (kernels found)
$N_6xyz = x(zz)y$	? (kernels found)
$N_7xyz = x(yy)z$	none exists (Theorem 9)
$N_8xyz = x(zy)y$	? (kernels found)
$N_9xyz = x(yz)y$	? (kernels found)
$Sxyz = xz(yz)$	? (no kernels found)
$S_1xyz = xz(zy)$	? (no kernels found)
$S_2xyz = xz(yy)$	? (kernels found)
$S_3xyz = xy(yz)$	none exists (Theorem 9)
$S_4xyz = xy(zy)$	none exists (Theorem 9)
$S_5xyz = xy(zz)$	none exists (Theorem 9)
$\_xyz = x(\_\_\_)$†	none exists (Theorem 9)
$\_xyz = x(\_(\_\_))$†	none exists (Theorem 9)
†6 cases	

Table 2. Sages from B and One Irregular Combinator

Combinator/Rule	Fixed Point Combinator with B
$Mx = xx$	? (no appropriate kernels found)
$W_0xy = xyx$	? (kernels found)
$W^1xy = yxx$	$B(B(B(W^1W^1)(BW^1))B)B$
$W_2xy = yxy$	$B(W_2x(B(BW_2)W_2))B$
$W_3xy = yyx$	? (kernels found)
$W_4xy = xxy$	none exists (Theorem 9)
$Oxy = y(xy)$	? (no kernels found)
$L_1xy = x(yx)$	none exists (Theorem 9)
$L_2xy = y(xx)$	? (kernels found)
$L_3xy = x(xy)$	none exists (Theorem 9)
$L_4xy = y(yx)$	? (no kernels found)

Table 3. Some Kernels for Which Stage 2 Failed (See Tables 1 and 2)

Combinator/Rule	Kernel with B
$N_5xyz = x(zy)z$	$N_5(Bf)x(N_5(Bf))$
$N_6xyz = x(zz)y$	$N_6(Bf)(N_6(Bf))(N_6(Bf))$
$N_8xyz = x(zy)y$	$N_8(Bf)(N_8(Bf))(N_8(Bf))$
$N_9xyz = x(yz)y$	$N_9(Bf)(N_9(Bf))(N_9(Bf))$
$S_2xyz = xz(yy)$	$S_2(Bf)(S_2(Bf))(S_2(Bf)(S_2(Bf)))$
$Mx = xx$	$M(BfM)$
$W_0xy = xyx$	$W_0W_0(B(Bf)W_0)$
$W_3xy = yyx$	$W_3(B(Bf)W_3)W_3$
$L_2xy = y(xx)$	$L_2(B(Bf)L_2)(L_2L_2)$

Tables 1 and 2 summarize the results for the regular and irregular combinators, respectively. In each table, the rightmost column has a representative fixed point combinator constructed from B and the other combinator, a reason why no fixed point combinator exists, or a question mark indicating that we do not know whether the strong fixed point property holds. The entries with "?" indicate whether any kernels were found in stage 1. Each of the last two entries in Table 1 represents six cases.

Table 3 contains representative kernels for the cases in which stage 1 found kernels but stage 2 failed to find fixed point combinators.

5.3.5.4. Details for Stage 2 Successes

When stage 2 finds a set of fixed point combinators, typically, many of the elements of the set reduce to other elements; therefore, we usually delete all reducible elements. The one exception is the set

of fixed point combinators constructed from the set consisting of the combinators B and W_2; in this set, all fixed point combinators we found were reducible. In the exception, we deleted all that reduce with B and all in which W_2 have more than two arguments—have terms of the form W_2xyz for some x, y, and z. There remained two fixed point point combinators, one of which reduced to the other; one was deleted.

We note that some of the kernels (kernel schemata) and some of the fixed point combinators contain variables.

Table 4. $Wxy = xyy$, Starting Kernel $W(B(Bf))(W(B(Bf))x$

Fixed Point Combinator	Natural Kernel
$B(B(B(WW)W)B)B$	$W(B(Bf))(W(B(Bf)))(W(B(Bf)))$
$B(B(B(W(W(W(BB))))W)B)B$	$W(B(Bf))(W(B(Bf)))(W(B(Bf))(W(B(Bf))))$
$B(B(W(B(W(W(BB)))W))B)B$	$W(B(Bf))(W(B(Bf)))(W(B(Bf))(B(Bf)))$
$B(B(W(W(B(W(BB))W)))B)B$	$W(B(Bf))(W(B(Bf)))(B(Bf)(B(Bf)))$
$B(W(B(B(W(W(BB)))W)B))B$	$W(B(Bf))(W(B(Bf)))(W(B(Bf))(Bf))$
$B(W(B(W(B(W(BB))W))B))B$	$W(B(Bf))(W(B(Bf)))(B(Bf)(Bf))$

Table 5. $Nxyz = xzyz$, Starting Kernel $N(B(Bf))x(N(B(Bf)))$

Fixed Point Combinator	Natural Kernel
$B(B(B(N(BNB)x)N)B)B$	$N(B(Bf))(N(B(Bf))x)(N(B(Bf)))$
$B(B(B(N(N(BN))B)N)B)B$	$N(B(Bf))(N(B(Bf))B)(N(B(Bf)))$
$B(B(N(BB(N(BNB)))N)B)B$	$N(B(Bf))(N(B(Bf))(B(Bf)))(N(B(Bf)))$
$B(B(N(B(BN(BB))N)N)B)B$	$N(B(Bf))(B(Bf))(N(B(Bf)))$

Table 6. $Hxyz = xyzy$, Starting Kernel $HH(B(Bf))(HH)$

Fixed Point Combinator	Natural Kernel
$H(B(H(HB)B)B)(HH)$	$HH(B(Bf))(HH)$
$B(B(H(H(B(BB)H))H)B)B$	$HH(B(Bf))(HH)$
$H(H(B(HBB))B)(HH)$	$HH(B(Bf))(HH)$
$H(H(H(B(BB)))B)(HH)$	$HH(B(Bf))(HH)$
$H(H(HH(HB))B)(HH)$	$HH(B(Bf))(HH)$
$H(H(H(H(BB)B))B)(HH)$	$HH(B(Bf))(HH)$
$H(H(H(H(HB)B))B)(HH)$	$HH(B(Bf))(HH)$
$H(H(H(H(HH)B))B)(HH)$	$HH(B(Bf))(HH)$

Table 7. $N_2xyz = xzzy$, Starting Kernel $N_2N_2(N_2N_2)(B(Bf))$

Fixed Point Combinator	Natural Kernel
$B(B(N_2N_2(N_2N_2))B)B$	$N_2N_2(N_2N_2)(B(Bf))$

Table 8. $W^*xyz = xyzz$, Starting Kernel $W^*B(Bf)(W^*B(Bf))x$

Fixed Point Combinator	Natural Kernel
$B(B(W^*(W^*(W^*B))B)(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$
$B(B(W^*W^*(W^*BB))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$
$B(W^*(B(W^*(W^*B)B))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(Bf))$
$B(W^*(B(W^*(W^*BB)))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(W^*(B(W^*BB)))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*W^*(B(W^*BB)(W^*B)))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$W^*(B(B(W^*(W^*B)B)(W^*B)))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)f)$
$W^*(B(B(W^*(W^*BB))(W^*B)))B$	$W^*B(Bf)(W^*B(Bf))(ff)$
$W^*(B(W^*(B(W^*BB))(W^*B)))B$	$W^*B(Bf)(W^*B(Bf))(Bff)$
$W^*(B(W^*(B(W^*BB)(W^*B))))B$	$W^*B(Bf)(W^*B(Bf))(ff)$
$B(B(W^*(BW^*(W^*B))B)(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$
$B(B(W^*(W^*(BW^*)W^*)B)(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$
$W^*(W^*(B(B(W^*BB)(W^*B))))B$	$W^*B(Bf)(W^*B(Bf))(ff)$
$W^*W^*(B(B(W^*BB)(W^*B))B)$	$W^*B(Bf)(W^*B(Bf))(ff)$
$B(W^*(B(B(W^*(W^*B)B))W^*)B)B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(Bf))$
$B(W^*(B(B(W^*(W^*BB)))W^*)B)B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(B(W^*(B(W^*BB)))W^*)B)B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(BW^*(B(W^*BB)))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(W^*(B(B(W^*BB))W^*))B)B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(B(W^*(W^*B)(W^*B))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$
$B(W^*(B(W^*B(W^*B)))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(W^*(W^*B)B)(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(W^*W^*(BB))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(W^*(BB)(W^*B))(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(B(W^*(W^*B)B)W^*)B)B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$B(W^*(B(W^*W^*(BB))W^*)B)B$	$W^*B(Bf)(W^*B(Bf))(Bf(Bf))$
$W^*(B(B(W^*B(W^*B))(W^*B)))B$	$W^*B(Bf)(W^*B(Bf))(ff)$
$B(B(W^*(B(W^*B)W^*)B)(W^*B))B$	$W^*B(Bf)(W^*B(Bf))(W^*B(Bf)(W^*B(Bf)))$

Table 9. $N_3xyz = xzyy$, Starting Kernel $N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$

Fixed Point Combinator	Natural Kernel
$B(B(B(N_3BB)(N_3(N_3N_3N_3)))B)B$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$
$B(B(N_3BB)(B(N_3(N_3N_3N_3))B))B$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$
$B(B(N_3(BB(N_3(N_3N_3N_3)))B)B)B$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$
$B(N_3BB)(B(B(N_3(N_3N_3N_3))B)B)$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$
$B(N_3(BB(B(N_3(N_3N_3N_3))B))B)B$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$
$B(N_3(B(BB(N_3(N_3N_3N_3)))B)B)B$	$N_3(B(Bf))(N_3(B(Bf)))(N_3(B(Bf)))$

Table 10. $N_4xyz = x(yz)z$, Starting Kernel $N_4(B(Bf))(N_4(B(Bf)))(N_4(B(Bf)))x$

Fixed Point Combinator	Natural Kernel ($R = N_4(B(Bf))$)
$B(B(N_4(N_4(N_4B)(N_4(N_4B)))N_4)B)B$	$RRR(R(R(B(Bf))))$
$B(N_4(B(N_4(N_4B)(N_4(N_4B)))N_4)B)B$	$RRR(R(R(Bf)))$
$N_4(B(B(N_4(N_4B)(N_4(N_4B)))N_4)B)B$	$RRR(R(Rf))$
$B(B(N_4(N_4B(N_4(N_4B)(N_4(N_4B))))N_4)B)B$	$RRR(R(R(R(B(Bf)))))$
$B(N_4(B(N_4B(N_4(N_4B)(N_4(N_4B))))N_4)B)B$	$RRR(R(R(R(Bf))))$
$B(N_4(N_4B(B(N_4(N_4B)(N_4(N_4B)))N_4))B)B$	$RRR(R(R(B(Bf)(Bf))))$
$B(N_4(N_4(BB(N_4(N_4B)(N_4(N_4B))))N_4)B)B$	$RRR(R(R(B(Bf)(Bf))))$

Table 11. $W^1xy = yxx$, Starting Kernel $W^1(B(Bf)W^1)(B(Bf)W^1)$

Fixed Point Combinator	Natural Kernel
$B(B(B(W^1W^1)(BW^1))B)B$	$W^1(B(Bf)W^1)(B(Bf)W^1)$
$B(B(W^1W^1)(B(BW^1)B))B$	$W^1(B(Bf)W^1)(B(Bf)W^1)$
$B(W^1W^1)(B(B(BW^1)B)B)$	$W^1(B(Bf)W^1)(B(Bf)W^1)$

Table 12. $W_2xy = yxy$, Starting Kernel $W_2x(B(BW_2)W_2)(Bf)$

Fixed Point Combinator	Natural Kernel
$B(W_2x(B(BW_2)W_2))B$	$W_2x(B(BW_2)W_2)(Bf)$

5.3.6. The Lack of Fixed Point Properties

In this section, we focus on those cases in which the set P of combinators under study fails to satisfy one or both of the fixed point properties.

In Section 3.1, we gave the definitions of *proper combinator*, *regular combinator*, *eliminator*, *replicator*, *permuter*, and *isolator* (*I* and *K* are included among isolators). In Theorem 4 of Section 4.6, we proved that each set of combinators that satisfies the strong fixed point property must contain at least one replicator and at least one isolator (not necessarily distinct from the replicator). In this section, we again use the notation $a \twoheadrightarrow b$ for "*a* reduces to *b*". We also use the symbol *f* for an arbitrarily chosen combinator. Therefore, we may assume that *f* has no particular properties—from the viewpoint of automated theorem proving, *f* is a Skolem constant. A few of the definitions and results in this section occur in other sections—they are repeated here for completeness of the presentation.

For illustrations, this section focuses on the following combinators and their associated reduction rules.

$Bxyz = x(yz)$
$Cxyz = xzy$
$Hxyz = xyzy$
$Ix = x$
$Kxy = x$
$Lxy = x(yy)$
$Mx = xx$
$Nxyz = xzyz$
$Qxyz = y(xz)$
$Q_1xyz = x(zy)$
$Sxyz = xz(yz)$
$Wxy = xyy$
$W^1xy = yxx$

The strong fixed point property asserts that

$\exists\Theta\forall x\,(\Theta x = x(\Theta x))$;

(Θ is called a *fixed point combinator*). The weak fixed point property asserts that

$\forall x\exists y\,(y = xy)$.

The weak fixed point property follows immediately from the strong fixed point property by letting *y* be Θx, for any fixed point combinator Θ.

Definition. For any combination *cd*, the *left component* is *c* and the *right component* is *d*. In particular, the left component of *cde* is *cd* and the right component is *e*.

5.3.6.1. Results Concerning the Weak Fixed Point Property

In this section, we study the weak fixed point property, and focus on sets for which it fails to hold.

Definition. A proper combinator is called an *isolator* if and only if the right side of its reduction rule is either a single variable or of the form *lr*, where *l* is a single variable. An equivalent definition is that a proper combinator is an isolator if the right side of its reduction rule is not of the form *cde* for any terms *c*, *d*, and *e*.

For example, the combinators *B*, *Q*, Q_1, *L*, *M*, *I*, and *K* are isolators.

Definition. Consider two terms *t* and *f*. The term *t* has 0 *leading f's* if and only if *t* is not of the form *fg* for some term *g*; the term *ft* has *n*+1 *leading f's* if and only if *t* has *n* leading *f*'s. For example, *f(f(ft))* has 3 leading *f*'s if *t* is not of the form *fg* for some term *g*.

Lemma 2. If *f* is an arbitrarily chosen combinator, then a reduction can never decrease the number of leading *f*'s; if a reduction increases the number of leading *f*'s, then an isolator is present.

Proof. Consider a term with some number of leading f's: $f(f(\cdots(ft)\cdots))$. The only terms to which a reduction can apply are t and subterms of t. Clearly, no reduction can decrease the number of leading f's. (Note that if $f = I$, then a decrease does occur; but f is arbitrary—in particular, f is not guaranteed to be I.) Assume that some reduction increases the number of leading f's. Then there must be a one-step reduction $t' \rightarrow ft''$, where t' has no leading f's. Clearly, an isolator is required; the proof is complete.

Theorem 8. If P is a set of proper combinators, and if M is the only isolator in P, then P fails to satisfy the weak fixed point property.

Proof. Assume, by way of contradiction, that P satisfies the weak fixed point property. The weak fixed point property asserts (with the obvious notational change) that $\forall f \exists t\,(t = ft)$. By the Church-Rosser property for applicative systems, there exists a term g such that $t \twoheadrightarrow g$ and $ft \twoheadrightarrow g$. Since f is arbitrary, the left component of all terms to which ft reduces must be f. Therefore, g must be of the form ft' for some t'. Since $ft \twoheadrightarrow ft'$, t must reduce to t'. Now we have that t reduces to both t' and ft'. The term ft' has one more leading f than t', so by Lemma 2, there must be at least one use of an isolator in the reduction $t \twoheadrightarrow ft'$. Since M is the only isolator, any isolation step that increases the number of leading f's must be $Mf \rightarrow ff$. Therefore, t reduces to a term t'' that consists of nothing but f's. Since $t'' = ft''$ and neither is necessarily reducible, a contradiction has been established and the proof is complete.

The following example shows that the analogue to Theorem 8 in which I is the only isolator does not hold. Consider the combinator U', with $U'xyz = yz(xxyz)$. (U' is derived from Turing's combinator U.) $U'U'I$ is a fixed point combinator, which can be demonstrated with a two-step reduction.

Corollary 1. If P is a set of proper combinators, and if M is the only isolator in P, then P fails to satisfy the strong fixed point property.

Proof. The proof follows immediately from Theorem 8 and the fact that the strong fixed point property implies the weak fixed point property.

Corollary 2. The set consisting of the combinators M, W, W^1, S, H, N, C satisfies neither the weak fixed point property nor the strong fixed point property.

Proof. The only isolator in the set is M. The proof follows immediately from Theorem 8 and Corollary 1.

5.3.6.2. Results Concerning the Strong Fixed Point Property

In this section, we study the strong fixed point property, and focus on sets for which it fails to hold.

Lemma 3. If Θ is a fixed point combinator, then there exists a sequence of terms $t_0, t_1, t_2, \ldots$ such that $\Theta f \twoheadrightarrow t_0 \twoheadrightarrow ft_1 \twoheadrightarrow f(ft_2) \twoheadrightarrow \cdots$.

Proof. Let t_0 be Θf. Since Θ is a fixed point combinator, $\Theta f = f(\Theta f)$. By the Church-Rosser property for applicative systems, there exists a term g_1 such that $\Theta f \twoheadrightarrow g_1$ and $f(\Theta f) \twoheadrightarrow g_1$. Since f is arbitrary, the left component of all terms to which $f(\Theta f)$ reduces must be f. Therefore, g_1 must be of the form ft_1 for some t_1. Since $\Theta f \twoheadrightarrow ft_1$, $f(\Theta f)$ must reduce to $f(ft_1)$. Now we have that $f(\Theta f)$ reduces to both ft_1 and to $f(ft_1)$. By a second application of the Church-Rosser property, there exists a term g_2 such that $ft_1 \twoheadrightarrow g_2$ and $f(ft_1) \twoheadrightarrow g_2$. The term g_2 must be of the form $f(ft_2)$ for some t_2. Successive applications of the Church-Rosser property establish the existence of the desired sequence such that $\Theta f \twoheadrightarrow t_0 \twoheadrightarrow ft_1 \twoheadrightarrow$

$f(ft_2) \twoheadrightarrow \cdots$, and the proof is complete (see Figure 1).

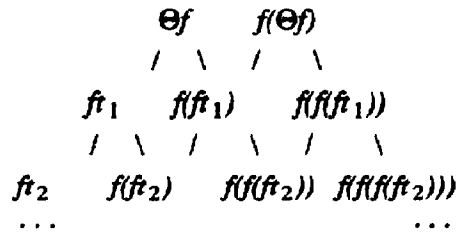

Figure 1. Diagram for Lemma 3.

Conjecture. If Θ is a fixed point combinator, then there exists a term Γ such that $\Theta f \twoheadrightarrow \Gamma \twoheadrightarrow f\Gamma \twoheadrightarrow f(f\Gamma) \twoheadrightarrow \cdots$. (The kernel method presented in Section 4.3 finds fixed point combinators by finding such a Γ. A proof of the conjecture would be of great value for proving the total completeness of the kernel method.)

Definition. A proper combinator is called *left restricted* if and only if the last variable on the left side of its reduction rule does not occur in the left component of the right side of its reduction rule.

For example, the combinators B, Q, Q_1, and L are left restricted.

Theorem 9. If P is a set of proper combinators, and if all members of P are left restricted, then P fails to satisfy the strong fixed point property.

Proof. Assume, by way of contradiction, that P satisfies the strong fixed point property. Then there exists a fixed point combinator Θ constructed from the members of P alone. By Lemma 3, there exists a term t such that $\Theta f \twoheadrightarrow ft$, for arbitrary f. Θf has the form cd, where c is free of f. We show that if a one-step reduction with a left-restricted combinator is applied to a combination cd, where c is free of f, then the result is a combination $c'd'$, where c' is free of f.

Case 1. The reduction involves c only or d only. Obvious.

Case 2. The reduction involves both c and d. The reduction rule for a left-restricted combinator has the form $Ax_1 \cdots x_n = lr$, where l is free of x_n. Since l is constructed from nothing more than $x_1, \cdots, x_{n-1}$, the term c' is constructed from nothing more than the subterms of c. In particular, f does not occur in c'.

Since Θ is free of f, we can conclude that, regardless of the number of reduction steps, Θf cannot reduce to ft. A contradiction has been established, and the proof is complete.

Corollary 3. The set consisting of the combinators B, Q, Q_1, L fails to satisfy the strong fixed point property.

Proof. Each member is left restricted. By Theorem 9, the set fails to satisfy the strong fixed point property.

Richard Statman's result [Statman86], which we give as the next corollary, on the nonconstructibility of a fixed point combinator from a single regular combinator is a corollary of the preceding theorem.

Corollary 4. If P is a set consisting of a single regular combinator, then P fails to satisfy the strong fixed point property.

Proof. Assume, by way of contradiction, that P satisfies the strong fixed point property. Let A be the single regular combinator. By Corollary 1, A must be an isolator. Therefore, the reduction rule for A has the form $Ax_1 \cdots x_n = x_1 r$, where r is free of x_1. Clearly, $n > 1$ and r is not empty. Since the left component of the right side of the reduction rule does not contain x_n, A is left restricted. By Theorem 9, P fails to satisfy the strong fixed point property, which establishes a contradiction and completes the proof.

Definition. A proper combinator is called *right terminating* if and only if the last variable on the left side of its reduction rule is the only variable that occurs in the right component of the right side of its reduction rule.

For example, the combinators L, W, M, and N are right terminating.

Theorem 10. If P is a set of proper combinators, and if all members of P are right terminating, then P fails to satisfy the strong fixed point property.

Proof. Assume, by way of contradiction, that P satisfies the strong fixed point property. Then there exists a fixed point combinator Θ constructed from the members of P alone. By Lemma 3, there exists a term t_1 such that $\Theta f \twoheadrightarrow ft_1 \twoheadrightarrow f(ft_2)$, for arbitrary f. The term Θf is of the form cd, where d contains nothing more than occurrences of f. We show that if a one-step reduction with a right-terminating combinator is applied to a combination cd, where d contains nothing more than occurrences of f, then the result is a combination $c'd'$, where d' contains nothing more than occurrences of f.

Case 1. The reduction involves c only. Obvious.

Case 2. The reduction involves d only. This case cannot occur, because d contains nothing more than occurrences of f.

Case 3. The reduction involves both c and d. The reduction rule for a right-terminating combinator has the form $Ax_1 \cdots x_n = lr$, where r contains nothing more than occurrences of x_n. The term c matches (unifies) with $Ax_1 \cdots x_{n-1}$, and d matches with x_n. The term c' is the appropriate instance of l, and d' is the appropriate instance of r. Since d matches with x_n and d contains nothing more than occurrences of f, d' contains nothing more than occurrences of d, and therefore, nothing more than occurrences of f.

Therefore, all terms to which Θf reduces have a right component that consists entirely of occurrences of f. Since $\Theta f \twoheadrightarrow ft_1 \twoheadrightarrow f(ft_2)$, we have that $t_1 \twoheadrightarrow ft_2$. But t_1 consists entirely of occurrences of f, which establishes a contradiction and completes the proof.

Theorem 11. The set of combinators consisting of L, M, N, H, W, W^1 fails to satisfy the strong fixed point property.

Proof. Assume, by way of contradiction, that Θ is a fixed point combinator constructed from members of the set. By Lemma 3, there exists a term t such that $\Theta f \twoheadrightarrow ft$ for arbitrary f. Clearly, there exists a *head reduction*—a reduction that satisfies the 1's rule—$\Theta f \twoheadrightarrow ft'$, where $ft' \twoheadrightarrow ft$. The last step of such a head reduction must be with an isolator. The only isolators are M and L; the last step cannot be with M, because that would make ft' be ff, which is not necessarily reducible; therefore the last step is with L. Let us consider a *head expansion* from ft' to Θf. We have just seen that the first step is with L, to produce Lfg for some term g. Any number of subsequent expansions with L produces $Lf'g'$ for some term g'. The term f' has the form $L(L(\cdots(Lf)\cdots))$, where the number of leading L's is the number of subsequent expansions with L. We may therefore assume that the current term is $Lf'g'$, and that the next step is not

with L. Let us now consider the next step of the head expansion.

Case 1. A head expansion with M produces $M(Lf')$, which cannot be further expanded with any of the available combinators.

Case 2. A head expansion with W produces WLf', which cannot be further expanded.

Case 3. A head expansion with W^1 produces $W^1f'L$, which cannot be further expanded.

Case 4. A head expansion cannot occur with N or H because $Lf'g'$ does not have enough arguments.

The only head expansion path that includes a term of the form Θf is by case 2, which can include WLf, but WL is not a fixed point combinator. This contradiction completes the proof of the theorem.

We note that Theorem 11 can be easily extended to cover other combinators and classes of combinator.

5.4. Open Questions

Our enjoyment of a paper, report, or book is markedly increased when we find that the authors have included open questions to answer, questions that can be used as the focus of research. The explicit presentation of such questions can stimulate interest in a field, and such an action can produce important advances. Indeed, as we remarked earlier in this report, Smullyan's question concerning the possible existence of a regular combinator that, when considered with the combinator B, permits one to construct a fixed point combinator caused us to study combinatory logic in the context of automated theorem proving.

With the intention of increasing the interest in both automated theorem proving and combinatory logic, and with the object of stimulating research focusing on questions whose answers are especially pertinent to the material presented in this report, we now turn to a discussion of various research topics and open questions. By definition, an open question is one whose answer has not yet been obtained. As one might expect, this section will consist of a number of somewhat disconnected paragraphs, many of which will be short. In the next section, we shall give some of our conjectures that are pertinent to the topics of this section.

5.4.1. Open Questions Focusing on Completeness

Question: If Θ is a fixed point combinator for the set P of combinators, must there exist a kernel Γ with respect to P such that, for every combinator f, Θf reduces to Γ_f, where Γ_f is an element of Γ?

Question: If Γ is a kernel with respect to the set P of combinators, will Γ always be found by applying expansions with elements of P to the right side of the inequality $y \neq fy$, an inequality obtained from assuming that the weak fixed point property does not hold for P?

Question: If the set P of combinators satisfies the weak fixed point property, must there exist a kernel Γ with respect to P?

The first topic for discussion in this entire section on open questions—and the topic that fascinates us the most—concerns the possible completeness of the kernel method. The topic consists of the three

open questions posed at the beginning of this subsection. For these questions, we again use the phrases "fixed point for P" and "kernel with respect to P" to mean that the corresponding object is constructible from the elements of P. Does there exist, for every set P of combinators and every combinator Θ that is a fixed point combinator for P, a kernel to which Θ naturally maps? If Γ is a kernel with respect to the set P of combinators, will Γ always be found by expanding the right side of the equation $y \neq fy$ with elements from P? If P is a set of combinators satisfying the weak fixed point property, must there exist a kernel Γ with respect to P?

If the first two questions can be answered in the affirmative, then, for any set P of combinators that satisfies the strong fixed point property, we know that the kernel method will always succeed in finding a proof of that fact. If the second and third questions can be answered in the affirmative, then, for any set P of combinators that satisfies the weak fixed point property, we know that the kernel method will always succeed in finding a proof of that fact.

5.4.1.1. The First Open Question Focusing on Completeness

With regard to the first of the three open questions on completeness—if the question turns out to be answered in the affirmative—we noted earlier that one finds the natural kernel Γ to which some given fixed point combinator Θ maps by applying reductions to Θf, where each reduction must be with an element of the set P of combinators to which Γ and Θ are relevant, and where f is an arbitrarily chosen combinator. To be precise, one actually finds a kernel Γ such that Γ contains a Γ_f such that Θf reduces to Γ_f. When a choice of reductions to apply exists, one must choose that which applies to the leftmost term; we shall give an example pertinent to this point shortly. When a reduction applies, one is not allowed to avoid its use—to skip the reduction; we shall shortly elaborate on this point as well. By definition, the natural kernel that corresponds to Θ is the first expression Γ such that Θ reduces to Γ and Γ reduces to $f\Gamma$, where f is an arbitrarily chosen combinator.

To be totally candid, we note that this choice for the definition of *natural kernel* is based merely on our preference; obviously, in many cases, other interesting choices exist. For example, with our definition of natural kernel, the fixed point combinator $BM(QM)$ naturally maps to the kernel $M(QMx)$, rather than to the kernel $QMx(QMx)$. For a more interesting example, as we shall immediately see when we discuss the fixed point combinator $\Theta_4(BW) = B(WW)(BW(BBB))$, one has the choice of at least three kernels to which $\Theta_4(BW)$ reduces, namely, $\Gamma_4(BW)$, $\Gamma_2(BW)$, and $\Gamma_1(BW)$. Although we would like to find a deep theoretical justification for our preference for the natural kernel corresponding to a given fixed point combinator, we have not yet succeeded in doing so. Since the concept of kernel has not been studied before—at least, our current research suggests that this is the case—we cannot suggest another source for this topic.

To illustrate precisely what is meant by the first of the two questions concerning the completeness of the kernel method—and to attempt to provide some insight on how the question might be attacked—let us focus on the following example. Here we give the promised example concerning selecting the first kernel one encounters. Let the set P consist of the combinators B and W with $Bxyz = x(yz)$ and $Wxy = xyy$, and let the fixed point combinator be $\Theta_4(BW) = B(WW)(BW(BBB))$. For convenience, we shall refer to this fixed point combinator simply as Θ_4, rather than as $\Theta_4(BW)$, and take similar actions when we discuss various kernels.

One reduction step with B applied to $\Theta_4 f$ yields $WW(BW(BBB)f)$. Clearly, one next has a choice of applying a reduction with W to the entire expression, or applying a reduction with B to $BW(BBB)f$. However, when the object is to attempt to find the natural kernel to which Θ_4 naturally maps, one must apply W, rather than skipping this reduction step—even temporarily. Indeed, if one applies B first and then applies W, one will fail to find the desired natural kernel Γ_4, which is the cube of $BW(BBB)f$. Instead, once the rules have been violated by applying the reduction with B before that with W, one will eventually find either the kernel Γ_2 or the kernel Γ_1, where Γ_2 is the cube of $W(BBBf)$ and Γ_1 is the cube of $W(B(Bf))$. What one finds depends on whether one again applies a reduction with B before one applies a reduction with W.

To see precisely what can go wrong if one skips a reduction, we can focus on the same example. However, we must first address what some might consider a fine point. In particular, one cannot know that a kernel has been found—even the wrong kernel—until one has (so to speak) passed it. Only in retrospect, by examining the earlier expressions, does one determine that one of those expressions is a Γ such that a later expression has the form $f\Gamma$. Now, if one follows the rules by applying in order B, W, and W again as reductions, starting with $\Theta_4 f = B(WW)(BW(BBB))f$, one obtains Γ_4 equal to the cube of $BW(BBB)f$. Since one cannot know that Γ_4 is a kernel until one has applied in order B, W, B, B, and B as reductions to obtain $f\Gamma_4$, one could find the wrong kernel by applying to each copy of $BW(BBB)f$ one reduction with B. In other words, one could skip the reduction with W. In such an event, one would obtain Γ_2, which one would later prove to be a kernel by applying the appropriate reductions to obtain $f\Gamma_2$.

Finally, we can apply the same type of analysis to the same example to show that, if one does not select the first kernel one encounters on the way from reducing Θ_4 to some expression $f\Gamma$, one will not succeed in locating the natural kernel to which the fixed point combinator under consideration maps. For example, after reducing Θ_4 to Γ_4, a reduction with B applied to the first occurrence of $BW(BBB)f$ yields a perfectly good kernel Γ'_4—but not the desired one—which differs from Γ_4 only by having $W(BBBf)$ in place of the first occurrence of $BW(BBB)f$. Indeed, one can easily verify that Θ_4 reduces to Γ'_4 which in turn, by applying a reduction step that satisfies the pseudo 1's rule, reduces to $f\Gamma'_4$. Of course, as the notation suggests and as we saw earlier, it is also the case that Θ_4 reduces to Γ_4 which in turn reduces to $f\Gamma_4$. To find the desired kernel, one must, therefore, make a choice from among these two given kernels as well as any other candidates. Our rule says that one must always choose the kernel one encounters first, by applying reductions constrained by the given rules for their application.

We use the following approach—which has not failed yet—for avoiding the pitfalls we have just discussed and for finding the natural kernel to which a given fixed point combinator under study naturally maps. We append to the combinator Θ under study the symbol f, where f is an arbitrarily chosen combinator, and reduce Θf with the elements of the set P of combinators under consideration, where all reductions are subject to the 1's rule, until f isolates. Where we denote the expression we have at that point by $f\Delta$, we test the intermediate expressions to see whether in fact one of them is Δ itself. If we find such an expression, then, by setting $\Gamma = \Delta$, we have found the kernel Γ to which Θ naturally maps. In addition, we know, by definition, that $\Gamma = \Delta$ is a simple kernel.

On the other hand, if Δ is not a kernel, then we attempt to apply reductions subject to the pseudo 1's rule to $f\Delta$, checking after each reduction to determine whether the result is of the form $f\Gamma$ where Γ appears earlier on the reduction path. If we find such a Γ, if at least one reduction was applied after f isolates, and if all reductions satisfy the 1's rule or the pseudo 1's rule, we have found a semisimple kernel—and one that is not a simple kernel. Of course, consistent with our earlier remarks, we always choose the first kernel that we find. For example, if $BM(QM)$ is the fixed point combinator under consideration, we choose as its natural kernel $M(QMx)$ rather than $QMx(QMx)$ even though $BM(QM)$ reduces to both kernels. We continue to apply reductions subject to the pseudo 1's rule, even if more than one occurrence of f has been isolated from the expression to its right.

Since the natural kernel being sought may not even be semisimple, a search for kernels that is restricted to observing the pseudo 1's rule may fail to terminate. Therefore, after f isolates, one must simultaneously explore the full set of reduction paths. In particular, when the kernel being sought is not a simple kernel, one must explore reduction paths that include steps that violate the pseudo 1's rule; we shall shortly give an appropriate example. In our search for the natural kernel, if we are forced to violate the pseudo 1's rule, we always first consider the reduction path that focuses on applying the reduction to the leftmost term that reduces; we shall amplify this remark with the promised example. As we have commented, for every fixed point combinator we have studied, we have always succeeded in finding a kernel to which it naturally maps. In other words, we have never embarked on an infinite reduction path, which suggests—but does not prove—that the kernel method is indeed complete in the sense that to every fixed point combinator there corresponds a natural kernel to which it reduces.

For the promised example, let us consider the fixed point combinator $B(S(CSI)(CSI))B$. To study this combinator, we use the following equations.

$Bxyz = x(yz)$
$Cxyz = xzy$
$Ix = x$
$Sxyz = xz(yz)$

If we apply our procedure for finding the natural kernel that corresponds to this fixed point combinator, and if we pause at the point at which we encounter an expression with one isolated occurrence of f, we have the expression $f(\Xi(I\Xi))$, where $\Xi = CSI(Bf)$. As it turns out, the path from the given fixed point combinator to the expression just given passes through the expression $\Xi\Xi$. Therefore, if we apply a reduction with the combinator I to $f(\Xi(I\Xi))$, we obtain $f(\Xi\Xi)$, which proves that $\Xi\Xi$ is a kernel, and in fact the kernel that is the natural correspondent of the fixed point combinator $B(S(CSI)(CSI))B$. This kernel is not even semisimple.

The important aspect of the example under discussion is the fact that if, instead of skipping the reduction step with C that satisfies the pseudo 1's rule, one applies this step and continues applying reduction steps constrained by the pseudo 1's rule, one will fail to find the desired kernel. Even worse, as this example shows, steadfastly observing the pseudo 1's rule can cause one to embark on an infinite reduction path that never terminates with finding a kernel. In other words, one cannot simply begin with a fixed point combinator and apply reductions that satisfy the 1's rule until f isolates, and then apply only those reduction steps that satisfy the pseudo 1's rule, and expect to always find a kernel. In the given example,

because of the nature of $\Xi\Xi$, such an action will never lead to finding a kernel. Unfortunately, therefore, when one begins with a fixed point combinator, the search for its natural kernel may require pursuing a number of reduction paths simultaneously; of course, when the desired kernel is simple, then one need pursue only one reduction path, stopping when f isolates.

Before turning to the second open question on completeness, let us briefly discuss the possible relevance of the Church-Rosser theorem to the first open question. The Church-Rosser theorem says that if two expressions are equal, then a third expression exists such that the two equal expressions each reduce to it. Of course, the third expression may be identical to one of the two expressions under consideration—one of the expressions may reduce to the other. To set the stage for this aspect of the discussion of the possible relevance of the Church-Rosser theorem to the completeness of the kernel method, let us consider the following example.

Since the combinator M has the property that $Mx = xx$ for all combinators x, we see that $MB(MW) = BB(MW)$ and that $MB(MW) = MB(WW)$. Therefore, $BB(MW) = MB(WW)$. By Church-Rosser, there must exist a third expression, not necessarily distinct from the two under consideration, such that the two expressions of interest each reduce to it. That expression is $BB(WW)$. Now, if Θ is a fixed point combinator, then $\Theta f = f(\Theta f)$ for any arbitrarily chosen combinator f. By Church-Rosser, both Θf and $f(\Theta f)$ must reduce to a common third expression. As we have discussed earlier in this report, that third expression must have the form $f\Delta$ for some Δ. Just as one can see that the reduction path that reduces $f(\Theta f)$ to $f\Delta$ must reduce Θ to Δ, one can also see that the reduction path that reduces Θf to $f\Delta$ must reduce $f(\Theta f)$ to $f(f\Delta)$. Therefore, $f(\Theta f)$ reduces to $f\Delta$ and also to $f(f\Delta)$, so $f\Delta = f(f\Delta)$.

Next, let $\Delta_1 = \Delta$. By applying to $f\Delta_1$ and $f(f\Delta_1)$ the analysis that leads to finding Δ_1, we can find a Δ_2 such that both $f\Delta_1$ and $f(f\Delta_1)$ reduce to $f(f(f\Delta_2))$, and, therefore, such that Δ_1 reduces to $f\Delta_2$. We can continue in this fashion, applying the Church-Rosser theorem, to obtain an infinite set of expressions Δ_i, where the integer i gives the number of leading f's in the expression to which each of the so-called preceding pair of expressions reduces.

Unfortunately, nothing in the Church-Rosser theorem enables us to conclude that a kernel occurs in the sequence of Δ's. In particular, we cannot conclude that, for some i, $\Delta_i = \Delta_{i+1}$. Indeed, if the kernel method is not complete, continued application of the Church-Rosser theorem will give us an infinite sequence of Δ's such that no kernel occurs in that sequence. This discussion thus shows how the kernel method might escape from being complete and, in a distant fashion, also shows why we suspect that the kernel method is in fact complete. After all, Θf reduces to Δ and also to $f\Delta$, which says that $\Delta = f\Delta$, which in turn says that Δ must reduce to $f\Xi$ for some Ξ.

5.4.1.2. The Second Open Question Focusing on Completeness

With regard to the second completeness question—that focusing on the completeness of the algorithm for finding kernels by expanding the right side of $y \neq fy$—we can make only a few observations. First, one must permit expansion steps such that the term being expanded is a single variable. For example, to find the semisimple kernel $QMx(QMx)$ with $Qxyz = y(xz)$ and $Mx = xx$, one can begin by expanding the variable y in the term fy by applying M to obtain $xx \neq f(Mx)$. If, instead, one restricts the search for a kernel by requiring that all expansions satisfy the 1's rule restriction, then stage 1 of the two-stage kernel

method will not find $QMx(QMx)$.

Since the kernel $QMx(QMx)$ is the natural correspondent for the fixed point combinator $S(QM)(QM)$ with $Sxyz = xz(yz)$, preventing expansion steps from applying to single variables will prevent the kernel method from finding a proof that the set consisting of the combinators S, Q, and M satisfies the strong fixed point property. Perhaps one must always permit expansion of terms that are variables in order to find a kernel if that kernel is strictly semisimple—a kernel Γ such that its reduction to $f\Gamma$ requires one or more steps to be applied after f isolates, where those additional steps satisfy the pseudo 1's rule. The corresponding question is at present still open.

The second observation concerns the need to permit expansion steps to be applied to terms that contain no occurrences of f, the arbitrarily chosen combinator. For example, if the set P consists of the combinators L and O with $Lxy = x(yy)$ and $Oxy = y(xy)$, then one can show that P satisfies the strong fixed point property. In particular, one can prove that $LO(LO)x = x(LO(LO)x)$ for all combinators x, which says that $LO(LO)$ is a fixed point combinator, and also obviously implies that $LO(LO)x$ is a kernel—in fact, the natural kernel for $LO(LO)$. To find this kernel, one first expands fy with O to obtain $xf \neq Oxf$. One then expands with L to obtain $yyf \neq LOyf$, which requires expanding the term Ox.

For a more interesting and closely related example, the fixed point combinator $O(LO(LO))$ has as its natural kernel $O(LO(LO))x$, which reduces to $x(LO(LO)x)$ by reducing it with O, and then to $f(O(LO(LO))x)$ by reducing with L. If one simply attempts to apply the corresponding expansions, but in reverse order, to fy, one encounters an intriguing obstacle—neither the term fy nor any of its subterms is appropriate for expanding with L. However—and we cannot say how general this case is—one can instead first expand fy with O to obtain $xf \neq Oxf$, and then expand the occurrence of x on the right side of this inequality with L to obtain $x(yy)f \neq O(Lxy)f$. In other words, an expansion path exists that begins with fy of $y \neq fy$ and terminates with the discovery of the desired kernel, $O(LO(LO))x$. After the next observation, we shall return to this example in the context of an interesting question for those whose field is automated theorem proving.

The third observation concerns the need to continue along an expansion path even after one has succeeded in finding a kernel. For an appropriate example, we return to the study of the combinators B and W. When these two combinators are used to search for a kernel by applying stage 1 of the two-stage kernel method, the inequality $yy \neq W(Bf)y$ is encountered, and from it one extracts the kernel $W(Bx)(W(Bx))$. If the set P of combinators under study consists of S, B, and W, and if one is attempting to prove that P satisfies the strong fixed point property, then one can put this kernel to immediate and good use. Specifically, stage 2 of the two-stage kernel method can expand this kernel with S and then expand the result with B to obtain the fixed point combinator $B(SWW)B$, and one has the desired proof.

However, if one were asked to find a proof that rests on a fixed point combinator that does *not* begin with B—a request that might seem synthetic, but that provides us with a simple example—then truncating the search for kernels along the given expansion path would be a mistake. In particular, if one expands the right side of the inequality $yy \neq W(Bf)y$ with B to obtain $yy \neq BWBfy$, one can extract the kernel $BWBx(BWBx)$. From this kernel, one can immediately construct the fixed point combinator $S(BWB)(BWB)$, which is precisely what is needed to fulfill the hypothetical request.

A distantly related example that might merit exploration focuses on the two kernels $M(CCM(C(Bx)))$ and $M(C(C(Bx))M)$. The first kernel reduces to the second by applying C as a reduction, where $Cxyz = xzy$. The two kernels are simple, and the second is in fact a simple superkernel. The fixed point combinator $B(B(BM(CCM))C)B$ has the first kernel as its natural correspondent, and the fixed point combinator $C(B(B(B(BM)C)C)B)M$ has the second kernel as its natural correspondent.

The second kernel, $M(C(C(Bx))M)$, offers added interest, for one can construct the following four fixed point combinators from it—combinators that manifest an interesting property.

$C(B(B(B(BM)C)C)B)M$
$B(C(B(B(BM)C)C)M)B$
$B(B(C(B(BM)C)M)C)B$
$B(B(B(C(BM)M)C)C)B$

In addition to the fact that each of the four combinators naturally maps to the kernel $M(C(C(Bx))M)$, one can apply a simple algorithm to obtain the n-th from the n-1st for $n \geq 2$. One can obtain the second from the first by interchanging the first two symbols and by interchanging the last two. One can obtain the third from the second by ignoring the first and last symbols, and then applying the same type of interchange. Finally one can obtain the fourth from the third by ignoring the first two and last two symbols, and applying the same type of interchange.

We close this discussion of the second completeness question by giving the promised discussion oriented to research in automated theorem proving. When we considered as an example the fixed point combinator $O(LO(LO))$ and its natural kernel correspondent $O(LO(LO))x$, we commented on the (so to speak) absent term that one would ordinarily expand with L, if one could. We showed that, despite the absent term, another path exists for finding the kernel $O(LO(LO))x$—one expands fy with O, and then expands a subterm (x) of the result with L. In the context of automated theorem proving, one would say that the *into term* that is called for to enable one's program to paramodulate from the right side of L into a subterm of fy is missing. One who is very familiar with paramodulation might suggest that the functional reflexive axioms could be used to replace—if written in terms of the function a for applies—a(f,y) by a(f,a(y,z)). For a reminder, the appropriate functional reflexive axiom is

EQUAL(a(y,z),a(y,z))

if written in the standard clause notation. The functional reflexive axioms for a set of clauses are just those instances of the clause

EQUAL(x,x)

(for reflexivity of equality) obtained by replacing x by, for example, $a(y,z)$, $g(y)$, and the like for every function a and g that is present in the set of clauses. The use of functional reflexive axioms—even to fill the gap of a (so to speak) absent *into term*—is at best unpalatable. Their use, almost always, materially decreases the effectiveness of an automated theorem-proving program.

Now, since one ordinarily relies heavily on the use of strategy to restrict the application of whatever inference rules are being used, and since the set of support strategy is typically the choice of strategy, one encounters an additional complication. In particular, the ploy of having paramodulation apply to pairs of positive clauses—the ploy that one might use to avoid introducing functional reflexive axioms—does not

mesh well with placing in the set of support only the clause corresponding to $y \neq fy$. In other words, if one wishes to increase the likelihood of refutation completeness—which amounts to always having some path available to complete a proof by contradiction when the set of clauses under study is self-contradictory—one is faced with various choices, none of which is very appetizing.

The question or questions that one might ask—from the viewpoint of automated theorem proving—concern the discovery of the conditions, if any, under which the use of paramodulation for the search for kernels is refutation complete. For example, can one identify specific properties (of the equations for the combinators present in a problem) that preserve refutation completeness when the set of support consists only of the clause

¬EQUAL(y,a(f,y))

and when no functional reflexive axioms are adjoined? For a second question, if one does in fact include the functional reflexive axioms, can one formulate a strategy for sharply curtailing their use? As an aside, we find it rather piquant to discover that the functional reflexive axioms might indeed be relevant to the use of an automated theorem-proving program for any important application—and the study of combinatory logic with such a program might indeed prove to be important.

If one can show that the functional reflexive axioms are indeed necessary when the set of support consists of the single inequality resulting from assuming that the weak fixed point property does not hold, then one might use this result to show that the second completeness question must be answered in the negative. In other words, one might succeed in proving—to our disappointment—that the first stage of the kernel method requires actions equivalent to using the functional reflexive axiom for the function a; in particular, one might be forced to replace, for example, the variable y in the term fy by xy.

5.4.1.3. The Third Open Question Focusing on Completeness

With regard to the third completeness question, which focuses on the possibility that every set P of combinators that satisfies the weak fixed point property must also admit a kernel with respect to P, we can only give examples that might in some distant fashion suggest an attack on this open question. By studying the properties of the examples, one might see how to prove that the question can be answered in the affirmative; or, regretfully, one might see how to answer the question in the negative by constructing an appropriate counterexample. All of the examples concern functions whose domain is the full set of possible combinators and whose range is a set of objects, each of which is a fixed point for one of the combinators.

For the set of examples, let P be a set of combinators that satisfies the weak fixed point property. Then, for each combinator f, there must exist at least one fixed point of f that is constructible from the elements of P. Indeed, since for all combinators x there exists a y such that $y = xy$—when one chooses an arbitrary combinator f—the y that exists and that depends on f is a fixed point of f. Let us focus on the set P consisting of B and W alone—which we have repeatedly shown satisfies the strong fixed point property, and therefore the weak—and note that the following discussion can be applied to any choice of a set of combinators that satisfies the weak fixed point property. To complete the stage setting for the examples, let us also assume that we have an ordered, infinite set P' of combinators.

In our first example, we focus on the kernel $W(Bx)(W(Bx))$. For each combinator f in P', since $W(Bf)(W(Bf))$ reduces to $f(W(Bf)(W(Bf)))$, we immediately have $W(Bf)(W(Bf))$ as a fixed point of f, not to be confused with a fixed point combinator. In particular, if f and g are two distinct combinators, then a fixed point of f is ordinarily not a fixed point of g; here, we are focusing on fixed points of functions, rather than on fixed point combinators. If we hold the kernel constant, as f ranges over the elements of P', we obtain a one-to-one function Ξ that maps each element f of P' to a single fixed point of f. In addition, the elements of the range of Ξ are very tightly coupled—the only difference between two distinct elements in the range of Ξ is the uniform replacement of f by g, where f and g are elements of P'. Indeed, there exists a single reduction that reduces Γ_f to $f\Gamma_f$, where f ranges over the elements of P' and where Γ_f ranges over the corresponding set of fixed points. The reduction under discussion is that which proves that $W(Bx)(W(Bx))$ is a kernel. Even further, since each of the set of fixed points whose union is the kernel $W(Bx)(W(Bx))$ are P-reducible fixed points for the set P consisting of B and W alone, and since we have a fixed point for each combinator f, we also have a proof that P satisfies the weak fixed point property.

For our second example, we consider the kernel schema $\Omega\Omega z$ with $\Omega = W(B(Bx))$. As we study this example, we learn that a set of fixed points Δ_f—one for each f as f ranges over the elements of P—can have the property that there exists a single reduction that takes each Δ_f to $f\Delta_f$ without the set of fixed points forming a kernel. To see how this can occur, let us assume that the elements of P' are ordered. Let us also map each element to a fixed point of it by instantiating z first to B, second to BB, third to BBB, ... The resulting set of fixed points does *not* form a kernel, because one cannot, for two distinct combinators f and g, obtain one fixed point from the other by simply applying a uniform replacement of f by g. Nevertheless, since for each combinator f there exists a P-reducible fixed point of f, we again have a proof that the set P consisting of B and W alone satisfies the weak fixed point property—even though the proof does not rely on the use of a kernel.

A single reduction exists with the property that, for every combinator f, the corresponding instance Ξ of $\Omega\Omega z$ reduces to $f\Xi$. The mapping under discussion maps a combinator to a fixed point of that combinator, but the members of the set of fixed points in the range of this mapping are not tightly coupled in the way that they are when we use a kernel to induce the mapping. Summarizing, although a single common reduction is present for all of the members of this set of fixed points, the members do not form a kernel, showing that the property that focuses on a common reduction is not sufficient for defining a kernel. Also, in answer to a question that might naturally arise, we reject the use of the entire schema, for its use would not produce a one-to-one mapping from combinators to a fixed point of each; instead, its use would produce a many-to-one mapping, which is not of interest to us.

For our third example, we focus on the set of expressions $BWBf(W(Bf))$, where f ranges over the elements of P'. The first and third example differ only in the fact that $BWBf(W(Bf))$ does not possess the reducibility property of a kernel, for $BWBf(W(Bf))$ does not reduce to $f(BWBf(W(Bf)))$. In other words, we again have a one-to-one function that takes a combinator of P' and maps it to a single fixed point of that combinator, and the elements in the range of this function are tightly coupled in a manner similar to the coupling for the elements in the first example. Both the function that maps f to $W(Bf)(W(Bf))$ and that which maps f to $BWBf(W(Bf))$ are very well behaved. Although the set of fixed points in this example do

not form a kernel, the existence of this set is sufficient for proving yet again that the set consisting of the combinators B and W satisfies the weak fixed point property.

In contrast, the next function we discuss exhibits slightly wilder behavior. For this fourth function, we recall that—closely related to the set $\Gamma(BW)$—we can construct an infinite set of superkernels (containing the superkernels of $\Gamma(BW)$ as a subset) from B and W alone such that the i-th superkernel, for i = 1,2,3,..., is the i+1-st power of its generator Ω_i, where $\Omega_1 = W(Bx)$, $\Omega_2 = W(B(Bx))$, $\Omega_3 = W(B(B(Bx)))$, and the like. For our function, we map the i-th element of P' to the i-th element in the given infinite sequence of superkernels. In contrast to our first two functions, the elements in the range of our third function are not nearly so tightly coupled. Instead, if f and g are two consecutive elements of P', then the image of g can be obtained from the image of f by a procedure that we used earlier to generate the superkernels of $\Gamma(BW)$. In that procedure, we obtained the generator of the next superkernel from the generator of the preceding superkernel by inserting an occurrence of B and fully right associating, and we increased the exponent by 1. As in the preceding example, we have a set of fixed points that suffices to prove that the set consisting of the combinators B and W satisfies the weak fixed point property, even though that set of fixed points is not a kernel.

For our fifth and last function, we consider a very badly behaved mapping that takes an element f of P' to a single fixed point of f. We consider the elements of P' in order, starting with the first. For each element, we use a random number generator to generate an integer between 1 and 5. The integer is used to assign a single fixed point to the particular element of P' being mapped. The chosen integer selects from among the following five kernels, and assigns the fixed point that is obtained by choosing from that kernel the element with f in place of x, where f is the combinator being mapped.

$Lx(Lx)$
$W(Bx)(W(Bx))$
$M(BxM)$
$M(QMx)$
$QMx(QMx)$

The idea is to prevent the elements in the range of the function being constructed from forming a (single) kernel, even though each of those elements is a member of some kernel.

The main aspect of our examples is to show in a small way how far a set of fixed points of a set of combinators f can be from forming a kernel. By studying these examples and by extrapolating heavily, one might see a way to prove that every set P of combinators that satisfies the weak fixed point property also admits a kernel. Or—although we do not prefer this possibility—perhaps a study of these examples will enable one to find a counterexample.

5.4.2. Open Questions Focusing on Specific Sets of Combinators

Question: If the set P consists of the combinators B and M alone with $Bxyz = x(yz)$ and $Mx = xx$, does P satisfy the strong fixed point property?

Question: If the set P consists of the combinators B and S alone with $Sxyz = xz(yz)$, does P satisfy the weak or the strong fixed point property?

Question: If the set P consists of the combinators B and N_1 with $N_1xyz = xyyz$, does P satisfy the weak fixed point property?

In this section, we pose open questions that focus on specific sets of combinators in the context of one of the fixed point properties—the weak or the strong or both. The first question, posed by the logician Raymond Smullyan, asks about the presence or absence of the strong fixed point property for the set *P* consisting of the combinators *B* and *M* alone.

Where $Bxyz = x(yz)$ and $Mx = xx$, one can immediately see that *P* satisfies the weak fixed point property by noting that $M(BxM)$ is a kernel, and, therefore, $M(BfM) = f(M(BfM))$ for any arbitrarily chosen combinator *f*. Smullyan in fact proves that *P* satisfies the weak fixed point property by giving this particular expression on page 135 of his book *To Mock a Mockingbird* [Smullyan84], but, of course, he does not call the expression a kernel.

Stage 1 of the two-stage kernel method finds the following four kernels in less than 1 CPU second when applied by the automated theorem-proving program OTTER (Other Theorem-proving Techniques for Effective Research) on a Sun 3 workstation. The first of the four, which is the one in Smullyan's book, is found in .06 CPU seconds.

$M(BxM)$
$M(B(Bx)M)z$
$M(BBBxM)z$
$M(MBBxM)z$

If given more time, the first stage of the kernel method finds additional kernels with respect to *P*. However, none of the given four kernels—or, for that matter, none of the others found by the kernel method—leads to the construction of a fixed point combinator.

We can rather easily outline an argument showing why none of the given four kernels—or any kernels that fail to have *f* as the last symbol—could ever lead to a proof that the strong fixed point property is satisfied by *P*. In other words, we can outline an argument showing why—if one is ever to succeed in constructing a fixed point combinator by using the kernel method—one must first succeed in finding a kernel whose last symbol is *f*, where *f* is an arbitrarily chosen combinator. First, let us assume that some Θ exists such that Θ is a fixed point combinator for *P*. Then, if one is ever to succeed in applying the kernel method to construct Θ, there must exist a kernel Γ such that expansions with *B* and with *M* applied to Γ eventually yields Θf. Therefore, Θf must reduce to Γ—whether or not Γ is the natural kernel correspondent of Θ—which in turn reduces to $f\Gamma$. All of the reductions that are involved must be with *B* or with *M*. Finally, because the last variable on both sides of the equation for *B* is the same, and because this property holds for *M* as well, any reduction with *B* or with *M* applied to an expression ending in *f*—an expression such as Θf—must yield an expression ending in *f*. In particular, therefore, Γ must end in *f*, which rules out the use of the four given kernels or any like them. If one wishes to study this argument in its complete form, we recommend a study of the corresponding theorem, which we intend to include in the coming book.

Of course, since we have not yet proved that the kernel method is complete—which is the subject of the first open question discussed in Section 5.4.1—the line of attack on which we are focusing may not be

a wise choice. One can take a totally different approach to the study of the question of whether P satisfies the strong fixed point property, an approach that avoids reliance on the kernel method and, therefore, is unaffected by the possible incompleteness of that method. In particular, one can attempt to prove that P does *not* satisfy the property by finding an appropriate model, a model that satisfies B and M but violates the strong fixed point property

$$yx = x(yx),$$

where y is existentially quantified and x is universally quantified. Predictably, we prefer to study the question in terms of the kernel method; indeed, we have a proof that the strong fixed point property does not hold for P, a proof, however, that depends on the assumption that for every fixed point combinator there exists a kernel to which it reduces. We expect to include our (so to speak) partial proof in the proposed book mentioned earlier—or, even better—wish to include a proof that is complete.

In contrast to our remarks that all point toward the possibility that the set P consisting of B and M alone fails to satisfy the strong fixed point property, one might be inclined to think that P in fact does. After all, B is an isolator, and M is a replicator and an isolator. By Theorem 4 of Section 4.6, we know that, for a set of combinators to satisfy the strong fixed point property, it is necessary for P to contain an isolator and a replicator. On the other hand, we also know that the presence of such combinators is not sufficient for proving that a set satisfies that property. For example, by Theorem 6 of Section 4.6, we know that the set consisting of B and L alone with $Lxy = x(yy)$ fails to satisfy the property—fails even though B is an isolator and L is both an isolator and a replicator.

Since we have nothing more to contribute regarding the possibility of constructing a fixed point combinator from B and M, let us turn to the second two-part open question of this section. Does the set P consisting of B and S alone, with $Sxyz = xz(yz)$, satisfy the strong fixed point property? Does P satisfy the weak fixed point property? Our only contribution to these questions consists of the fact that the kernel method has not succeeded in finding any kernels with respect to P. We have not yet succeeded in proving that it is impossible for the kernel method to find any appropriate kernels, nor have we proved that our approach to applying the kernel method is complete, which is the second question of Section 5.4.1. Even further, we have not yet proved the theorem that asserts that if a set of combinators satisfies the weak fixed point property, then there must exist a kernel with respect to that set—a topic that is the focus for the third open question on completeness.

The next open question is very similar to the second of the preceding two. Let the combinator N_1 be such that $N_1xyz = xyyz$. Does the set P consisting of the combinators B and N_1 satisfy the weak fixed point property? Except for citing Theorem 9 of Section 5.3.6.2—which shows that the set under discussion cannot satisfy the strong fixed point property—we can do no more than parrot our remarks concerning the similar question focusing on B and S.

5.4.3. Additional Open Questions

Question: Does there exist a finite model satisfying B and L in which the strong fixed point property fails to hold?

Question: Does there exist a finite model satisfying W and L in which the strong fixed point property fails

to hold?

Question: Does there exist a finite model satisfying S and W in which the weak fixed point property fails to hold?

Question: What interesting conditions are sufficient to guarantee that a set of combinators satisfies the weak fixed point property?

Question: What interesting conditions are sufficient to guarantee that a set of combinators satisfies the strong fixed point property?

Question: What conditions are necessary, and what are sufficient, for a set P of combinators such that there exists a simple, alternatively semisimple, kernel with respect to P?

Question: Does there exist a kernel Γ *in which at least two distinct replicators occur such that the two replicators each participate in the reduction that proves that* Γ *is a kernel?*

Question: What properties of a fixed point combinator are necessary, and what are sufficient, for its natural kernel to be simple, alternatively semisimple?

Question: If Θ *is a fixed point combinator whose natural kernel is* Γ*, and if all combinators in* Θ *are regular, noneliminating, and nonpermuting, must* Γ *be a simple kernel?*

Question: Ignoring parentheses, what is the length of the shortest fixed point combinator expressed purely in terms of S and K?

Question: How large is the largest set of combinators that satisfies the strong, alternatively weak, fixed point property such that no proper subset does?

Question: Can one find other sets (than the two we have discussed) of combinators that can be arranged in a square with isolators down the diagonal and replicators down the antidiagonal such that the sets consisting of the elements in a row satisfy the strong fixed point property and such that all other sets consisting of one element from each column fail to satisfy the property?

The following observations are pertinent to the cited questions. We have proved with Theorem 6 of Section 4.6 that the set consisting of the combinators B and L alone cannot satisfy the strong fixed point property. Therefore, we know an appropriate model must exist, but we do not know if a finite model of the desired type exists. We would be interested in examining either a finite or an infinite model that satisfies both B and L but that fails to satisfy the equation for the strong fixed point property. Obviously, because of the kernel $Lx(Lx)$, we know that any set of combinators containing L must satisfy the weak fixed point property.

Because of additional theorems we have proved in Section 5.3.6, we know that one cannot construct a fixed point combinator from L and W alone, and we also know that the set consisting of S and W alone cannot satisfy the weak fixed point property. Therefore, corresponding models must exist, but again we do not know if the desired finite models exist. We would also be interested in examining either finite or infinite models with the appropriate properties.

As for interesting sufficient conditions, we are ruling out, for example, conditions such as that which asserts that the set contains B and W.

The shortest fixed point combinator known to us that is expressed purely in terms of S and K contains 15 symbols. We present this fixed point combinator using the following equations that express the combinators I, B, and W in terms of S and K.

$I = SKK, \quad B = S(KS)K, \quad W = SS(SK).$

$S(K(SSIW))B$

Finally, of the two squares that are relevant to the cited questions, the first has a first row of B and W and a second row of S and L. The second square is obtained from the first by replacing W by N.

5.5. Conjectures

When one poses open questions for research, as we did in the preceding sections, the sporting gesture is to make conjectures concerning the more important among those questions. In addition, since the experience one has had with an area of research usually disposes one to a specific side of a question from that area, the action of making conjectures provides a pleasant and exciting way to convey that experience. Therefore, we now make various conjectures.

5.5.1. Conjectures Concerning Completeness

The conjecture that is most appealing to us focuses on the first of the three completeness questions posed in Section 5.4.1. Specifically, we conjecture that the kernel method is complete. In particular, if P is a set of combinators that satisfies the strong fixed point property and Θ is a fixed point combinator expressed purely in terms of the elements of P, we conjecture that there must exist a kernel Γ with respect to P such that Θf reduces to Γ which in turn reduces to $f\Gamma$, where f is an arbitrarily chosen combinator. In other words, if Θ is a fixed point combinator expressed purely in terms of the elements of some set P of combinators, then, for each combinator f, we conjecture that there exists a kernel Γ with respect to P such that Γ contains a P-reducible fixed point Γ_f of f with the property that Θf reduces to Γ_f which in turn reduces to $f\Gamma_f$.

We also conjecture that one can find the natural kernel for such a Θ by first appending f to Θ, where f is any arbitrarily chosen combinator, and applying reductions with the elements of P such that all reductions satisfy the 1's rule. The object is to find an appropriate P-reducible fixed point Γ_f of f such that Θf reduces to Γ_f which in turn reduces to $f\Gamma_f$. When f isolates, one then pauses to examine the resulting expression to see whether it is of the form $f\Gamma$ for some Γ that has already occurred on the reduction path. If in fact the resulting expression has this form, then one has found the desired natural kernel, and the kernel is simple. If not, then one must pursue in an orderly fashion all of the reduction paths that emanate

from the expression for which f first becomes isolated. The additional search is required because the natural kernel of Θ may not even be semisimple. If one finds a number of candidate kernels—technically, candidate P-reducible fixed points of f—the natural kernel is that containing the P-reducible fixed point of the form $f\Gamma_f$, where Γ_f is the earliest such fixed point of f along all reduction paths.

With regard to the second open question posed in Section 5.4.1, we conjecture that all kernels with respect to a given set P of combinators can be found by applying expansions with the elements of P to the right side of the inequality $y \neq fy$. As we have repeatedly remarked, this inequality arises from assuming that the weak fixed point property fails to hold for P. We shall not be amazed—although we shall be disappointed—to find that one might be forced, in certain bizarre cases, to use the functional reflexive axiom

EQUAL(a(x,y),a(x,y))

when paramodulation is the inference rule one instructs the chosen automated theorem-proving program to apply. In other words, from the viewpoint of combinatory logic, one might be forced to replace y in fy by xy, and take a similar action more than once.

Finally, with regard to the third question of Section 5.4.1, we conjecture that when a set P satisfies the weak fixed point property, then there must exist a kernel with respect to P. In other words, we conjecture that all sets P of combinators that satisfy the weak fixed point property also satisfy the reducible weak fixed point property, and, even further, in such a way that one can find among the P-reducible fixed points of f for all combinators f a subset that forms a kernel with respect to P.

5.5.2. Conjectures Concerning Specific Sets of Combinators

We conjecture—as Smullyan does—that the set P of combinators consisting of B and M alone with $Bxyz = x(yz)$ and $Mx = xx$, although it does satisfy the weak fixed point property, fails to satisfy the strong fixed point property. As commented, the existence of the kernel $M(BxM)$ is sufficient for establishing that P satisfies the weak fixed point property. Our conjecture is in part influenced by the fact that we have a partial proof that P cannot satisfy the strong fixed point property, a proof that depends on the assumption that, to every fixed point combinator Θ expressed purely in terms of B and M, there exists a kernel Γ containing an element Γ_f such that Θf reduces to Γ_f which in turn reduces to $f\Gamma_f$. More generally, our partial proof rests on the completeness of the kernel method.

However, we do know of an expression purely in terms of B and M that might serve as a warning, an expression Δ_1 such that $\Delta_1 f$ reduces to $f\Delta_2 f$ which reduces to $f(f\Delta_3)$ ad infinitum. Specifically, if one begins with $B(BM(BM))B$ and reduces with B and M first by observing the 1's rule, and then by observing the pseudo 1's rule, one can travel an infinite reduction path that contains expressions with an ever-increasing number of occurrences of f, each isolated from the expression to its right. An analysis of the possible reduction paths that begin with this expression strongly suggests that it is not a fixed point combinator. Were one trying to construct a fixed point combinator from B and M alone, the expression $B(BM(BM))B$ might provide a starting point.

If one can find a fixed point combinator that can be expressed purely in terms of B and M—as our colleague Ross Overbeek points out—then one has proved that the kernel method is, sadly, incomplete. The missing link is, of course, our proof that no such fixed point combinators exist, a proof that depends

on the completeness of the kernel method. In particular—unless we have produced a flawed proof in some other way—the existence of such a combinator would assert that our proof does not hold, and the only flaw would be the use of the completeness of the kernel method.

We conjecture that the set consisting of the combinators B and S alone with $Bxyz = x(yz)$ and $Sxyz = xz(yz)$ fails to satisfy the weak fixed point property and, therefore, also fails to satisfy the strong fixed point property. We are greatly influenced by our view of the kernel method and by the fact that no kernels were found for this set of combinators after 60 CPU seconds when the program OTTER, running on a Sun 3 workstation, applied the first stage of the two-stage kernel method.

We also conjecture that the set consisting of the combinators B and N_1 alone with $Bxyz = x(yz)$ and $N_1xyz = xyyz$ fails to satisfy the weak fixed point property. Our view has the same basis as that given for the preceding conjecture.

5.5.3. Additional Conjectures

Currently, we have only one conjecture relevant to the open questions posed in Section 5.4.3. We conjecture that the natural kernel that corresponds to a fixed point combinator is always simple when the fixed point combinator is expressed purely in terms of regular, noneliminating, and nonpermuting combinators.

6. Conclusions

From our viewpoint, when a mathematician or logician expresses interest in a problem or an area, that interest defines the problem or area as significant and meriting research, provided of course that the person in question is an expert in the corresponding field. In other words, one needs little additional justification for working on the problem or area, or for considering such work as important.

A case in point is Smullyan's obvious interest in fixed point properties, as evidenced in his book [Smullyan84] and in a lecture he gave in Indianapolis in the spring of 1986. From comments in his book, from comments in private conversations with him, and from discussions with other logicians in the field of combinatory logic, what becomes very clear is that the presentation of a set of combinators coupled with the query about the presence or absence of either the weak or the strong fixed point property often results in no information. Obviously, the fault does not lie with the mind(s) of the scientist(s). Rather, the problem of proving that either fixed point property is present or absent is nontrivial, and fraught with arduous computation; the corresponding questions are in fact deep questions.

What we offer in this report is evidence that has been desired for many years by researchers in automated theorem proving. Specifically, researchers have hoped to find some area of mathematics or logic that would be so amenable to attack with an automated theorem-proving program that the probability of answering questions randomly selected from that field would be greater than one half. In addition, the hope was that the answers would typically be obtained quickly. To always succeed is not the real goal, although obviously a great one if achieved. However, to be able to frequently succeed is what has been desired by many.

We can now, apparently, fulfill this desire for fixed point problems randomly selected from combinatory logic. In particular, one can choose combinators at random, submit the corresponding fixed point

problem to our kernel method applied by our automated theorem-proving program OTTER using a Sun 3 workstation, and have a good chance of getting important answers in less than one CPU minute. Viewed as a contest, if somebody in an imaginary audience shouts out some combinators, and if an expert in combinatory logic competes with the kernel method to see who first produces an appropriate proof, the kernel method will win a high percentage of the time. In other words, by using the automated theorem-proving program OTTER to apply the kernel method, we have an example of an area, perhaps smaller than one would like, such that we can return meaningful information almost immediately.

The kernel method is *not* an algorithm for constructing fixed point combinators. Rather, the method provides a systematic means for reasoning about the constructibility of fixed point combinators—and therefore, about the presence or (indirectly) the absence of fixed point properties. Indeed, the kernel method is a global strategy for searching for proofs of both the weak and the strong fixed point property.

When the method proves that the weak fixed point property holds for some given set P of combinators, it does so by finding a set of P-reducible fixed points that are very tightly coupled—the set of fixed points is a kernel. More precisely, such a proof is based on finding a single P-reducible fixed point of the combinator f—where f is an arbitrarily chosen combinator—in such a way that one can correctly conclude that the corresponding P-reducible fixed points have been found for all such combinators f.

When the kernel method proves that the strong fixed point property is satisfied by a given set P of combinators, the proof rests on the construction of an appropriate fixed point combinator. That fixed point combinator is constructed by applying expansions with the elements of P to one of the kernels found by the method.

Of course, instead of applying the kernel method, one can adopt either of two standard approaches for searching for fixed point combinators. One can couple intuition with a somewhat random search. Or, one can assume that the strong fixed point property is not satisfied by the set of combinators under study, and then attempt to produce a proof by contradiction. The results given in this report, however, strongly suggest that use of the kernel method is far superior to either of the standard approaches one might employ.

References

[Barendregt81] Barendregt, H. P., *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, Amsterdam (1981).

[Curry58] Curry, H. B., and Feys, R., *Combinatory Logic I*, North-Holland, Amsterdam (1958).

[Curry72] Curry, H. B., Hindley, J. R., and Seldin, J. P., *Combinatory Logic II*, North-Holland, Amsterdam (1972).

[Hindley86] Hindley, J. R., and Seldin, J. P., *Introduction to Combinators and Lambda Calculus*, Cambridge University Press, Cambridge (1986).

[Lukasiewicz48] Lukasiewicz, J., "The shortest axiom of the implicational propositional calculus", in Proceedings of the Royal Irish Academy, Vol. 52, Section A, no. 3, pp. 25-33 (April 1948).

[McCune87] McCune, W., and Wos, L., "A case study in Automated theorem proving: Finding sages in combinatory logic", *Journal of Automated Reasoning* **3**, no. 1, pp. 91-108 (February 1987).

[RobinsonG69] Robinson, G., and Wos, L., "Paramodulation and theorem-proving in first-order theories with equality", pp. 135-150 in *Machine Intelligence 4*, ed. B. Meltzer and D. Michie, Edinburgh University Press, Edinburgh (1969).

[RobinsonJ65] Robinson, J. "A machine-oriented logic based on the resolution principle", *Journal of the ACM* **12**, pp. 23-41 (1965).

[Smullyan84] Smullyan, R., *To Mock a Mockingbird*, Alfred A. Knopf, New York (1985).

[Smullyan86] Smullyan, R., Private correspondence (1986).

[Smullyan87] Smullyan, R., Private correspondence (1986).

[Statman86] Statman, R., Private correspondence with Raymond Smullyan (1986).

[Wos65] Wos, L., Robinson, G., and Carson, D., "Efficiency and completeness of the set of support strategy in theorem proving", *Journal of the ACM*, Vol. 12, pp. 536-541 (1965).

[Wos67] Wos, L., Robinson, G., Carson, D., and Shalla, L., "The concept of demodulation in theorem proving", *Journal of the ACM*, Vol. 14, pp. 698-709 (1967).

[Wos73] Wos, L., and Robinson, G., "Maximal models and refutation completeness: Semidecision procedures in automatic theorem proving", pp. 609-639 in *Word Problems: Decision Problems and the Burnside Problem in Group Theory*, ed. W. Boone, F. Cannonito, and R. Lyndon, North-Holland, New York (1973).

[Wos80] Wos, L., Overbeek, R., and Henschen, L., "Hyperparamodulation: A refinement of paramodulation," in Proceedings of the Fifth Conference on Automated Deduction, Vol. 87, Lecture Notes in Computer Science, ed. R. Kowalski and W. Bibel, Springer-Verlag, New York, pp. 208-219 (July 1980).

[Wos84] Wos, L., Overbeek, R., Lusk, E., and Boyle, J., *Automated Reasoning: Introduction and Applications*, Prentice-Hall, Englewood Cliffs, N.J. (1984).

[Wos87] Wos, L., *Automated Reasoning: 33 Basic Research Problems*, Prentice-Hall, Englewood Cliffs, N.J. (1987).

Distribution for ANL-88-10

Internal:

J. M. Beumer (3)
F. Y. Fradin
H. G. Kaper
A. B. Krisciunas
E. L. Lusk
W. W. McCune
R. A. Overbeek
G. W. Pieper (37)
R. L. Stevens
L. T. Wos (40)

ANL Patent Department
ANL Contract File
ANL Libraries
TIS Files (3)

External:

DOE-OSTI, for distribution per UC-405 (66)
Manager, Chicago Operations Office, DOE
Mathematics and Computer Science Division Review Committee:
J. L. Bona, Pennsylvania State University
T. L. Brown, University of Illinois, Urbana
P. Concus, Lawrence Berkeley Laboratory
S. Gerhart, Micro Electronics and Computer Technology Corp., Austin, TX
H. B. Keller, California Institute of Technology
J. A. Nohel, University of Wisconsin, Madison
M. J. O'Donnell, University of Chicago
D. Austin, ER-DOE
H. Barendregt, Nijmegen University, The Netherlands
A. Bundy, University of Edinburgh, Scotland
M. McRobbie, Australian National University
R. Meyer, Australian National Laboratory
G. Michael, Lawrence Livermore Laboratory
R. Smullyan, University of Indiana