Demonic operators and monotype factors

Eindhoven University of Technology

Department of Mathematics and Computing Science

Demonic Operators and Monotype Factors

by

Roland Backhouse and Jaap van der Woude

92/11

Computing Science Note 92/11
Eindhoven, May 1992

Copies can be ordered from:
Mrs. F. van Neerven
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

Demonic Operators and Monotype Factors

Roland Backhouse and Jaap van der Woude
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513,
5600 MB Eindhoven,
The Netherlands.

May 26, 1992

Abstract

This paper tackles the problem of constructing a compact, point-free proof of the associativity of demonic composition of binary relations and its distributivity through demonic choice. In order to achieve this goal a definition of demonic composition is proposed in which angelic composition is restricted by means of a so-called "monotype factor". Monotype factors are characterised by a Galois connection similar to the Galois connection between composition and factorisation of binary relations. The identification of such a connection is argued to be highly conducive to the desired compactness of calculation.

Nothing delights a mathematician more than to discover that two things, previously regarded as entirely distinct, are mathematicaly identical. *W. W. Sawyer*

La mathématique est l'art de donner le même nom à des choses différentes. *[J.] H. Poincaré*

The term "Galois connexion" was coined by Oystein Ore [20] almost fifty years ago in order to describe a particularly simple and elegant relationship between a pair of functions. Specifically, if $\mathcal{A}$ and $\mathcal{B}$ are two partially-ordered sets and $f \in \mathcal{A} \longrightarrow \mathcal{B}$ and $g \in \mathcal{B} \longrightarrow \mathcal{A}$ are functions mapping the two sets to each other then we say that f and g are *Galois connected* if and only if for all $a \in \mathcal{A}$ and $b \in \mathcal{B}$

$$(1) \qquad f.a \leq_{\mathcal{B}} b \quad \equiv \quad a \leq_{\mathcal{A}} g.b$$

The importance of the notion was recognised at a very early stage in mathematically-oriented computing science literature. As long ago as 1964 Hartmanis and Stearns [12] developed an alternative, but entirely equivalent, formulation of Galois connections called "pair algebras" which they applied to a data-refinement problem – the state assignment problem in sequential machines. [1] Seven years later, Conway [8] published a book on finite-state machines in which a very important element was the chapter on so-called "factor theory" and its subsequent application to the construction and analysis of so-called "biregulators". Conway did not refer to the work of Hartmanis and Stearns, nor to Galois connections, but there are clearly recognisable, formally establishable, parallels between his "L-R factorisations" of a regular language and Hartmanis and Stearns' "m-M decompositions" of a finite-state machine.

Although both the textbook by Hartmanis and Stearns and Conway's little monograph offer beautiful examples of the economy and elegance of abstract mathematics this aspect of their work seems not to have received the recognition that it deserves: in the case of Conway his theory of factors seems to have been completely disregarded, the only reference to this

[1]Although they did not use the term in the original paper describing their theory Hartmanis and Stearns briefly acknowledge the relevance of Galois connections in a footnote in their textbook [13] in which they said: "For related mathematical concepts see the discussion of Galois connections between partially ordered sets in [7]." Simons [21] formally establishes the equivalence between Galois connections and pair algebras.

chapter of his book of which we are aware being a paper [3] drawing a connection between factors and the failure-function method used in the well-known Knuth-Morris-Pratt string-searching algorithm [16]; in the case of Hartmanis and Stearns, the fact that the state-assignment problem is an interesting non-trivial example of data refinement seems to have escaped the attention of all those working in this now blossoming area.

In recent years there has been a reawakening of interest in Galois connections in computing science journals and conference publications. Of particular note are the textbook on Continuous Lattices [11] in which a substantial section is devoted to the topic, and Hoare and He's formulation of so-called "weakest pre- and post-specifications" via Galois connections with relational composition. (Their weakest pre- and post-specifications are the same as Conway's left and right "factors" and Dilworth's [10] left and right "residuals".) Other recent references are [14, 18, 17].

In our work on developing a relational theory of datatypes [1, 2, 4] we have come to recognise the importance and ubiquity of Galois connections. They abound particularly in the calculus of relations and their simple form lends itself superbly to compact calculation. With some practice they are very easy to spot, and their mastery is an indispensable precursor to the mastery of the orders-of-magnitude more complex notion of an "adjunction" in category theory.

This paper exploits a Galois connection that arises naturally in the study of demonic composition and choice in order to prove certain algebraic properties of these operators. The proofs illustrate well, in our view, how early recognition of a Galois connection can significantly shorten and simplify otherwise complicated calculations. The main concern of the paper is not the theorems that are proved — none of our results is in any way new — but with the "ergonomics" of mathematical calculation — how to choose suitable notation, and how to formulate definitions and calculation rules in such a way that seemingly difficult calculations become straightforward. In this sense the paper is a small contribution to a broad debate of central importance to the further development of mathematical practice.

1 The Algebraic Framework

In order that the reader be able to follow our calculations a limited knowledge of the axiomatic calculus of relations due to (among others) de Morgan, Schröder and Tarski is needed. Full accounts appear in several monographs; we will make do here with just a summary of precisely those properties we need in our calculations.

We work within the context of an algebra $(\mathcal{A}, \sqcup, \sqcap, \top\!\!\top, \bot\!\!\bot, \circ, I)$ consisting of a set $\mathcal{A}$ on which are defined three binary operators $\sqcup$, $\sqcap$ and $\circ$. Further the set $\mathcal{A}$ includes three (distinct) constants $\top\!\!\top$, $\bot\!\!\bot$ and I. The set $\mathcal{A}$ is assumed to be a complete, completely-distributive lattice under the operations $\sqcup$ and $\sqcap$ with top and bottom elements $\top\!\!\top$ and $\bot\!\!\bot$, respectively. The ordering relation on the elements will be denoted by $\sqsubseteq$ and its converse by $\sqsupseteq$. (Specifically, $X \sqsubseteq Y \ \equiv\ X \sqcup Y = Y$.) The set $\mathcal{A}$ also forms a monoid under the "composition" operator $\circ$. That is, $\circ$ is associative and has unit I. These two algebras are connected by the fact that composition distributes universally, from both the left and the right, over $\sqcup$. (Composition does not, however, distribute everywhere over $\sqcap$.)

An interpretation is that in which $\mathcal{A}$ is the set of binary relations over some (anonymous) universe, $\sqcup$ is set union, $\sqcap$ is set intersection and $\circ$ is (angelic) relational composition. The top element $\top\!\!\top$ is the universal relation, the bottom element $\bot\!\!\bot$ is the empty relation and the unit I is the identity relation.

In order not to confuse meta-language with object-language we prefer to refer to the elements of $\mathcal{A}$ as *specs* rather than relations.

The subset of $\mathcal{A}$ consisting of those elements below I are called the *monotypes*. (They can be interpreted as partial identities, i.e. identities on subsets of the universe.) Thus,by definition,

(2) $\quad \mathit{monotype}.A \ \equiv\ I \sqsupseteq A$

An important property of monotypes is that composition coincides with intersection. That is, for all monotypes A and B,

(3) $\quad A \circ B \ =\ A \sqcap B$

This property can be formally derived in the calculus of relations but depends on other axioms for which we have no further use in the paper.

We also assume additional properties that validate the introduction of two so-called "domain operators", the *left domain operator* $<$ and the *right domain operator* $>$. Both are unary operators written as postfixes to their arguments. The specific properties that we assume are as follows. First, domains of specs are monotypes: For all specs R,

(4) $\quad monotype.R{<} \quad$ and $\quad monotype.R{>}$

(Note that the infix dot denotes function application and that unary operators always take precedence in our formulae over binary operators. Thus one should parse "$monotype.R{<}$" as "$monotype.(R{<})$".)

Second, the domain operators are defined by Galois connections between the lattice of all specs and the sublattice of the monotypes: For all specs R and monotypes A,

(5) $\quad A \sqsupseteq R{>} \;\equiv\; \top\!\top \circ A \sqsupseteq R$

and

(6) $\quad A \sqsupseteq R{<} \;\equiv\; A \circ \top\!\top \sqsupseteq R$

(In words, $R{>}$ is the least monotype A such that $\top\!\top \circ A \sqsupseteq R$, and dually for $R{<}$.)

From these definitions it is clear that properties of one domain operator can easily be dualised to properties of the other by reversing the order of the arguments in a composition. We shall therefore only state additional properties of the right domain operator and leave the reader to supply the dual property of the left domain operator.

The first of these additional properties is that the closed elements of the domain operators are precisely the monotypes.

(7) $\quad monotype.A \;\equiv\; A = A{>}$

Next, $R{>}$ is the least monotype A such that $R \circ A = R$: For all specs R and all monotypes A,

(8) $\quad A \sqsupseteq R{>} \;\equiv\; R \circ A = R$

Finally, we have three properties for which we have no verbal summary: For all specs R and S,

(9) $\quad (R \circ S){>} \;=\; (R{>} \circ S){>}$

For all specs S and monotypes A,

(10) $\quad (S \circ A)> \;=\; S> \circ A$

For all specs R and S,

(11) $\quad \exists(A : \mathit{monotype}.A : R = S \circ A) \;\equiv\; R = S \circ R>$

The interpretation of the domain operators in the calculus of binary relations is that $R>$ is the set of pairs (y, y) such that there exists an x with xRy. Vice-versa, $R<$ is the set of pairs (x, x) such that there exists a y with xRy. With these interpretations all the above properties can be readily verified. Compacter, axiomatic proofs can also be constructed but, with some exceptions, we have provided insufficient information about our axiom system to permit the reader to complete this exercise. From now on, however, we proceed entirely axiomatically and do not refer to the relational interpretation in any proofs.

This completes our brief introduction to the algebraic framework.

2 Defining Demonic Composition

Weakest precondition semantics [9] gives an axiomatic description of nondeterministic programs admitting several models. One of these is a relational model with "non-standard" relational composition and union: the so-called *demonic composition* and *demonic choice* (see e.g. [19, 6]). In this section we motivate and then define demonic composition. In section 6 we define demonic choice.

To motivate the definition of demonic composition let us briefly summarise its operational interpretation in the relational model. We consider some set $X_\perp$ consisting of "program states" X augmented with a distinguished element $\perp$ representing non-termination. A (non-deterministic) program is a relation between elements of $X_\perp$ and X that is total on X. (Note that we use the functional-programming convention of having input on the right and output on the left rather than the opposite way around as is conventional in imperative programming.) The demonic composition $R\,;\,S$ of two relations R and S is then the normal relational composition $R \circ S$ of R and S but excluding computations of S that can lead to non-termination.

This definition, with its implicit case analysis on $\perp$, leads to ugly calculations. A slight abstraction leads to a nicer definition. Specifically, we forget about the detailed structure of $X_\perp$ and define the demonic composition of R and S to be the normal relational composition $R \circ S$ but excluding computations of S beginning in states x such that states in Sx may lie outside the right domain of R.

Interpreting this specification literally we are required to specify $R\,;S$ formally via two clauses: The first clause states that it is "the usual composition" but with a restricted right domain. I.e.

$$(12) \qquad R\,;S \;=\; R \circ S \circ R\&S$$

where

$$(13) \qquad \mathit{monotype}.(R\&S)$$

Thus $R\&S$ is the "restriction" on the right domain.

The second clause states that the said restriction should include only those states x such that R is defined on all the S-results Sx. Replacing "states x" by "monotypes B", we formulate this second clause as the requirement that $R\&S$ satisfy the specification:

$$(14) \qquad \forall(B : \mathit{monotype}.B : \; R\&S \sqsupseteq B \;\equiv\; R{>} \sqsupseteq (S \circ B){<})$$

In words, $R\&S$ is the largest monotype B such that $R{>} \sqsupseteq (S \circ B){<}$.

The initial challenge we set ourselves in this paper is to construct an *axiomatic* proof of the associativity of demonic composition. (Later we consider the additional challenge of showing that it distributes through demonic choice.) In particular, we deny ourselves any appeal to extensionality properties based on the existence of "points" in the domains of specs. (An example of an extensionality axiom that is independent of the axioms on which our calculations are based is that the spec I is the union of all pairs (x, x) where x ranges over the set of all "points".) This makes our task more difficult but more rewarding in that the validity of the theorem we prove extends to more models.

3 Preliminary Analysis

Before embarking on the task of proving that demonic composition is indeed associative let us examine the more elementary consequences of its specification.

The immediate question is whether there is a solution to the conjunction of (13) and (14) viewed as equations in $R\&S$. To see that this is indeed the case — at a glance — we observe that both the functions $<$ and $S\circ$ are universally $\sqcup$-junctive, hence so is their composition and thus

(15) $\quad \sqcup(B : \mathit{monotype}.B \wedge R> \sqsupseteq (S \circ B)< : B)$

solves (14). It also solves (13) since it is clearly a monotype. We may thus conclude that the binary operator & does indeed exist.

Knowing this formula is however of little help in any calculations involving $R\&S$ since the inevitable first step in any such calculation will be to return to (14). More progress can be made if one is aware that being universally $\sqcup$-junctive is equivalent to having a certain sort of adjoint. Note that the requirement on $R\&S$ — for all monotypes B and all specs R and S,

(16) $\quad R\&S \sqsupseteq B \quad \equiv \quad R> \sqsupseteq (S \circ B)<$

— is almost a Galois connection between the function $(R \mapsto R\&S)$ and the function $(B \mapsto (S \circ B)<)$. That it is not so can be solely attributed to the occurrence of the right domain operator on the right side of (16).

We can dismiss this obstacle by noting that, for all monotypes A we have $A> = A$ and, in particular, $(R>)> = R>$. Consequently,

(17) $\quad R\&S \quad = \quad R>\&S$

where, for all monotypes A and B,

(18) $\quad A\&S \sqsupseteq B \quad \equiv \quad A \sqsupseteq (S \circ B)<$

Property (17) tells us that the left operand of & may always, without loss of generality, be assumed to be a monotype. Property (18) says that — with the said assumption — the function $\&S$ is adjoint to the function $(B \mapsto (S \circ B)<)$. I.e. in the lattice of monotypes there is a Galois connection between $\&S$ and the composition of the two functions $<$ and $S\circ$.

4 Monotype Factors

The recognition of a Galois connection is a very crucial observation and unleashes a welcome gush of properties. In this case the gush becomes a minor flood if one is already familiar with the Galois connection between composition and "factors" in the calculus of relations. (The term "factor" is that coined by Conway [8] in his study of regular languages. Elsewhere the terms "residual" [7] and "weakest pre-/post-specification" [15] are used for the same concept.) Specifically, right factors are defined by the Galois connection

(19) $\quad S\backslash R \sqsupseteq T \quad\equiv\quad R \sqsupseteq S \circ T$

Comparing the right sides of (18) and (19) we see that they are almost identical but for the additional application of the right domain operator. To facilitate exploitation of the similarity it pays to rewrite the left side of (18) so that the arguments A and S appear in the same order as R and S in (19). Let us therefore introduce the binary operator $\backslash$ defined by

(20) $\quad S\backslash A \quad=\quad A\&S$

for all monotypes A and specs S. Then, substituting in (18), $S\backslash A$ is completely characterised by the Galois connection

(21) $\quad S\backslash A \sqsupseteq B \quad\equiv\quad A \sqsupseteq (S\circ B)<$

We call $S\backslash A$ a *monotype factor.*

Using the well-documented properties of spec factors as a guide one quickly establishes a number of properties of monotype factors. Some of these are listed in the table overleaf alongside the corresponding properties of spec factors. (Several of these properties are predicted purely from the fact that one has a Galois connection, in particular all the cancellation properties. The second junctivity property, both for monotype and spec factors, combines properties of composition with the defining Galois connections, and the two properties labelled "miscellaneous" are peculiar to composition and monotypes.)

Note that, in order to keep our formulae compact, a shorthand for quantified expressions has been exploited in the table. Specifically, if $\mathcal{V}$ is some bag of values all of the same type and f is a function on elements of that type then $f.\mathcal{V}$ denotes the bag obtained by applying f to each element of that bag. Moreover, if $\mathcal{V}$ is some bag and $\oplus$ is some idempotent, associative, and commutative operator we write $\oplus\mathcal{V}$ for $\oplus(v : v \in \mathcal{V} : v)$. An example of this shorthand is the first junctivity property. Spelt out in full the property reads:

$$\sqcap(A : A \in \mathcal{A} : S\backslash A) \;=\; S\backslash \sqcap(A : A \in \mathcal{A} : A)$$

This convention will be used in several places in the text.

At this point we are faced with a predicament. Equation (12) introduced the notation $R\&S$ but now we have another notation for the same quantity, namely $S\backslash R{>}$. Should we continue our calculations using the original notation or should we switch to the new form?

For us there is no doubt that the latter is the better choice. The notation $R\&S$ was purely ad hoc, invented on the spur of the moment in order to fulfill an initial goal. The notation $S\backslash A$, however, is deliberately chosen in order to suggest an analogy with division in ordinary arithmetic. In particular, the order of the arguments in $S\backslash A$ is designed to facilitate the use of the cancellation properties in table 1 (specifically, the arguments that are cancelled should be adjacent to each other), which from experience with calculations with Galois connections are very useful.

Fortunately, very little rewriting is required. It suffices to rewrite the definition of demonic composition:

(22) $\quad R\,;S \;=\; R \circ S \circ S\backslash R{>}$

where

(23) $\quad monotype.(S\backslash R{>})$

Several of the properties in table 4 can be reformulated in ways that prove to be particularly valuable to our specific aims. The first is the cancellation property

(24) $\quad A \;\sqsupseteq\; (S \circ S\backslash A){<}$

which, in view of (8), has the equivalent formulation:

(25) $\quad A \circ S \circ S\backslash A \;=\; S \circ S\backslash A$

	Monotype Factors	Spec Factors
Definition	$S \searrow \in \text{monotype} \longleftarrow \text{monotype}$	$S\backslash \in \text{spec} \longleftarrow \text{spec}$
	$S \searrow A \sqsupseteq B \;\equiv\; A \sqsupseteq (S\circ B)<$	$S\backslash R \sqsupseteq T \;\equiv\; R \sqsupseteq S\circ T$
Junctivity Properties	$\sqcap(S \searrow \mathcal{A}) \;=\; S \searrow (\sqcap\mathcal{A})$	$\sqcap(S \backslash \mathcal{R}) \;=\; S \backslash (\sqcap\mathcal{R})$
	$\sqcap(\mathcal{S} \searrow A) \;=\; (\sqcup\mathcal{S})\searrow A$	$\sqcap(\mathcal{S} \backslash R) \;=\; (\sqcup\mathcal{S})\backslash R$
Constants	$I \;=\; S \searrow I$	$\top\!\!\top \;=\; S \backslash \top\!\!\top$
	$I \;=\; \perp\!\!\perp \searrow A$	$\top\!\!\top \;=\; \perp\!\!\perp \backslash R$
	$I \searrow A \;=\; A$	$I \backslash R \;=\; R$
Cancellation Properties	$A \;\sqsupseteq\; (S\circ S\searrow A)<$	$R \;\sqsupseteq\; S\circ S\backslash R$
	$S \searrow (S\circ B)< \;\sqsupseteq\; B$	$S\backslash(S\circ T) \sqsupseteq\; T$
	$S\searrow A \;=\; S\searrow(S\circ S\searrow A)<$	$S\backslash R \;=\; S\backslash(S\circ S\backslash R)$
	$(S\circ S\searrow(S\circ B)<)< \;=\; (S\circ B)<$	$S\circ S\backslash(S\circ R) \;=\; S\circ R$
Miscellaneous	$T\searrow(S\searrow A) \;=\; (S\circ T)\searrow A$	$T\backslash(S\backslash R) \;=\; (S\circ T)\backslash R$
	$A\circ A\searrow B \;=\; A\circ B$	$A\circ A\backslash R \;=\; A\circ R$

Note: R, S and T denote arbitrary specs, A and B denote monotypes, $\mathcal{S}$ and $\mathcal{R}$ denote arbitrary sets of specs, and $\mathcal{A}$ denotes an arbitrary set of monotypes.

Table 1: Monotype Factors versus Spec Factors

The second is that the monotype transformer $S\setminus$ is universally $\sqcap$-junctive. Since, however, for monotypes the $\sqcap$ operator coincides with composition the monotype transformer $S\setminus$ is universally composition-junctive and, more particularly, for all monotypes A and B,

(26) $\quad S\setminus(A \circ B) \;=\; S\setminus A \circ S\setminus B$

Finally, the two properties labelled "miscellaneous" can be usefully combined into one giving:

(27) $\quad A \circ (S \circ A)\setminus B \;=\; A \circ S\setminus B$

for all monotypes A and B, and all specs S.

We conclude this section with one obvious consequence of (22) — at least obvious to the experienced "speculist" — which crops up so frequently in our calculations that we presume to anticipate its usefulness. Specifically:

(28) $\quad R;S \;=\; R \circ S \circ (R;S)>$

The property is just an instance of (11).

5 The Proof of Associativity

Now let us turn to the task in hand — proving that demonic composition is associative. We consider the two terms $R;(S;T)$ and $(R;S);T$, and expand each using (22) very cautiously in order not to allow the formulae to grow too big. First, we obtain

$$\begin{array}{ll}
 & R;(S;T) \\
= & \qquad \{\text{ definition: (22) }\} \\
 & R \circ (S;T) \circ (S;T)\setminus R> \\
= & \qquad \{\text{ definition: (22) }\} \\
 & R \circ S \circ T \circ T\setminus S> \circ (S;T)\setminus R>
\end{array}$$

(Note that the outermost occurrence of ";" has been expanded first. Expanding the innermost occurrence leads to a larger formula.)

This is a pleasing result becauses it expresses $R;(S;T)$ in terms of a restriction on the right domain of $R \circ S \circ T$. Now for the other term:

$$\begin{array}{ll}
 & (R\,;\,S)\,;\,T \\
= & \qquad \{\ \text{definition: (22)}\ \} \\
 & (R\,;\,S)\ \circ\ T\ \circ\ T\backslash(R\,;\,S){>} \\
= & \qquad \{\ \text{Applying (22) for a second time would introduce an} \\
 & \qquad\ \ \text{undesirable restriction on the } \textit{left} \text{ domain of } T\text{, not on} \\
 & \qquad\ \ \text{the right. We search around for something more suitable.} \\
 & \qquad\ \ \text{Aiming for (25) we apply (28)}\ \} \\
 & R\ \circ\ S\ \circ\ (R;S){>}\ \circ\ T\ \circ\ T\backslash(R;S){>} \\
= & \qquad \{\ \text{cancellation: (25), } A,S\ :=\ (R;S){>},T\ \} \\
 & R\ \circ\ S\ \circ\ T\ \circ\ T\backslash(R;S){>}
\end{array}$$

Thus $(R\,;\,S)\,;\,T$ has also been expressed in terms of a restriction on the right domain of $R\ \circ\ S\ \circ\ T$ and we can infer that

$$\begin{array}{lll}
(29) & & R;(S;T)\ =\ (R\,;\,S)\,;\,T \\
 & \Leftarrow & T\backslash S{>}\ \circ\ (S;T)\backslash R{>}\ =\ T\backslash(R;S){>}
\end{array}$$

The antecedent of (29) is established in two steps. First, we calculate that

$$\begin{array}{ll}
 & T\backslash S{>}\ \circ\ (S;T)\backslash R{>} \\
= & \qquad \{\ \text{definition: (22)}\ \} \\
 & T\backslash S{>}\ \circ\ (S\circ T\circ T\backslash S{>})\backslash R{>} \\
= & \qquad \{\ \text{cancellation: (27)} \\
 & \qquad\quad \text{with } A\ :=\ T\backslash S{>}\ ,\quad S\ :=\ S\circ T\ ,\ B\ :=\ R{>}\ \} \\
 & T\backslash S{>}\ \circ\ (S\circ T)\backslash R{>} \\
= & \qquad \{\ (S\circ T)\backslash A\ =\ T\backslash(S\backslash A),\ \text{junctivity: (26)}\ \} \\
 & T\backslash(S{>}\circ S\backslash R{>})
\end{array}$$

Now comparing the above with (29) we see that it suffices to prove

$$(30)\qquad (R;S){>}\ =\ S{>}\ \circ\ S\backslash R{>}$$

This task is completed as follows:

$$\begin{array}{ll}
 & (R;S){>} \\
= & \qquad \{\ \text{definition: (22)}\ \} \\
 & (R\ \circ\ S\ \circ\ S\backslash R{>}){>} \\
= & \qquad \{\ \text{domains: (9)}\ \} \\
 & (R{>}\ \circ\ S\ \circ\ S\backslash R{>}){>}
\end{array}$$

$$
\begin{array}{ll}
= & \quad \{ \text{ cancellation: (25) } \} \\
 & (S \circ S\backslash R{>}){>} \\
= & \quad \{ \text{ domains: (10), } S\backslash R{>} \text{ is a monotype } \} \\
 & S{>} \circ S\backslash R{>}
\end{array}
$$

6 Demonic Choice

The benefit of the little theory we have developed begins to pay dividends when we extend our problem further to the investigation of whether demonic composition distributes through demonic choice (to be defined shortly).

In this section we prove that demonic composition distributes both from the left and from the right over an arbitrary choice of specs. This is more general than the results of Berghammer [5] and van der Woude [22] both of whom only proved distributivity through a *finite, non-empty* choice of specs. Unlike in the previous section we are very brief in our discussion of the calculations. Hopefully by now the calculations speak for themselves!

For an arbitrary set $\mathcal{S}$ of specs define the demonic choice $\Box\mathcal{S}$ by

(31) $\quad \Box\mathcal{S} \;=\; \sqcup\mathcal{S} \circ \sqcap(\mathcal{S}{>})$

The motivation for this definition is that, in the relational model of weakest-precondition semantics discussed earlier, $\Box\mathcal{S}$ excludes all computations that may lead to non-termination.

Observe that

(32) $\quad \Box\mathcal{S} \;=\; \sqcup(\mathcal{S} \circ \sqcap(\mathcal{S}{>}))$

and, for non-empty $\mathcal{S}$,

(33) $\quad (\Box\mathcal{S}){>} \;=\; \sqcap(\mathcal{S}{>})$

Property (32) is just universal distributivity of composition over cup. Property (33) has a simple proof.

$$
\begin{array}{ll}
 & (\Box\mathcal{S}){>} \\
= & \quad \{ \text{ (31), (10) } \} \\
 & (\sqcup\mathcal{S}){>} \circ \sqcap(\mathcal{S}{>}) \\
= & \quad \{ \text{ For monotype } A,\ A\circ \text{ is positively } \sqcap\text{-distributive } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \sqcap((\sqcup\mathcal{S})> \circ \mathcal{S}>) \\
= & \quad \{ \ > \text{is monotonic}, \quad A \circ B \ = \ A \sqcap B \ \} \\
 & \sqcap(\mathcal{S}>)
\end{array}
$$

Theorem 34 $\quad R \ ; \ \Box\mathcal{S} \ = \ \Box(R \ ; \ \mathcal{S})$

Proof If $\mathcal{S}$ is empty the theorem is trivially true (since both left and right sides evaluate to $\perp\!\!\perp$). In the case of non-empty $\mathcal{S}$ we begin by expanding both sides using the definitions of demonic composition and choice.

$$
\begin{array}{ll}
 & R \ ; \ \Box\mathcal{S} \\
= & \quad \{ \ \text{definition: (22)} \ \} \\
 & R \circ \Box\mathcal{S} \circ (\Box\mathcal{S})\backslash R> \\
= & \quad \{ \ \text{definition: (31), and (33)} \ \} \\
 & R \circ \sqcup\mathcal{S} \circ (\Box\mathcal{S})> \circ (\Box\mathcal{S})\backslash R>
\end{array}
$$

and

$$
\begin{array}{ll}
 & \Box(R \ ; \ \mathcal{S}) \\
= & \quad \{ \ (32) \ \} \\
 & \sqcup((R \ ; \ \mathcal{S}) \circ \sqcap((R \ ; \ \mathcal{S})>)) \\
= & \quad \{ \ (28) \ \} \\
 & \sqcup(S : \ S \in \mathcal{S} : \ R \circ S \circ (R \ ; \ S)> \circ \sqcap((R \ ; \ \mathcal{S})>)) \\
= & \quad \{ \ > \text{is monotonic}, A \circ B \ = \ A \sqcap B, \bullet \ \mathcal{S} \text{ is non-empty} \} \\
 & \sqcup(S : \ S \in \mathcal{S} : \ R \circ S \circ \sqcap((R \ ; \ \mathcal{S})>)) \\
= & \quad \{ \ \text{composition is universally } \sqcup\text{-junctive} \ \} \\
 & R \circ \sqcup\mathcal{S} \circ \sqcap((R \ ; \ \mathcal{S})>)
\end{array}
$$

In this way both $R \ ; \ \Box\mathcal{S}$ and $\Box(R \ ; \ \mathcal{S})$ have been expressed as restrictions on the right domain of $R \circ \sqcup\mathcal{S}$ and it suffices to prove that these domain restrictions are equal. Now,

$$
\begin{array}{ll}
 & \sqcap((R \ ; \ \mathcal{S})>) \\
= & \quad \{ \ \text{notational convention} \ \} \\
 & \sqcap(S : \ S \in \mathcal{S} : \ (R \ ; \ S)>) \\
= & \quad \{ \ (30), (3) \ \} \\
 & \sqcap(S : \ S \in \mathcal{S} : \ S> \sqcap S\backslash R>) \\
= & \quad \{ \ \text{calculus} \ \} \\
 & \sqcap(\mathcal{S}>) \ \sqcap \ \sqcap(\mathcal{S}\backslash R>)
\end{array}
$$

$$\begin{array}{ll}
= & \quad \{\ \text{junctivity of } \backslash R{>}\text{, see table}\ \} \\
 & \sqcap(\mathcal{S}{>}) \ \sqcap \ (\sqcup\mathcal{S})\backslash R{>} \\
= & \quad \{\ (33) \bullet \ \mathcal{S} \text{ is non-empty, } (3)\ \} \\
 & (\Box\mathcal{S}){>} \circ (\sqcup\mathcal{S})\backslash R{>} \\
= & \quad \{\ (27)\ \} \\
 & (\Box\mathcal{S}){>} \circ (\sqcup\mathcal{S} \circ (\Box\mathcal{S}){>})\backslash R{>} \\
= & \quad \{\ (33), (31) \ \bullet \ \mathcal{S} \text{ is non-empty}\ \} \\
 & (\Box\mathcal{S}){>} \circ (\Box\mathcal{S})\backslash R{>}
\end{array}$$

$\Box$

Theorem 35 $\quad \Box\mathcal{S} \ ; \ R \ = \ \Box(\mathcal{S} \ ; \ R)$

Proof Again we note that the theorem is trivially true for empty set $\mathcal{S}$. For non-empty $\mathcal{S}$ the same strategy is repeated. First,

$$\begin{array}{ll}
 & \Box\mathcal{S} \ ; \ R \\
= & \quad \{\ \text{definition: } (22)\ \} \\
 & \Box\mathcal{S} \circ R \circ R\backslash(\Box\mathcal{S}){>} \\
= & \quad \{\ (33), (31) \ \bullet \ \mathcal{S} \text{ is non-empty}\ \} \\
 & \sqcup\mathcal{S} \circ (\Box\mathcal{S}){>} \circ R \circ R\backslash(\Box\mathcal{S}){>} \\
= & \quad \{\ \text{cancellation: } (25)\ \} \\
 & \sqcup\mathcal{S} \circ R \circ R\backslash(\Box\mathcal{S}){>}
\end{array}$$

and

$$\begin{array}{ll}
 & \Box(\mathcal{S} \ ; \ R) \\
= & \quad \{\ (32)\ \} \\
 & \sqcup((\mathcal{S} \ ; \ R) \circ \sqcap((\mathcal{S} \ ; \ R){>})) \\
= & \quad \{\ (28)\ \} \\
 & \sqcup(S : S \in \mathcal{S} : S \circ R \circ (S \ ; \ R){>} \circ \sqcap((\mathcal{S} \ ; \ R){>})) \\
= & \quad \{\ {>} \text{ is monotonic, } (3)\ \} \\
 & \sqcup(S : S \in \mathcal{S} : S \circ R \circ \sqcap((\mathcal{S} \ ; \ R){>})) \\
= & \quad \{\ \text{universal distributivity of composition over cup}\ \} \\
 & \sqcup\mathcal{S} \circ R \circ \sqcap((\mathcal{S} \ ; \ R){>})
\end{array}$$

Now we compare the two domain restrictions:

$$
\begin{array}{ll}
 & \sqcap((\mathcal{S}\ ;\ R)>) \\
= & \qquad \{\ (30)\ \} \\
 & \sqcap(R> \circ R\backslash(\mathcal{S}>)) \\
= & \qquad \{\ \text{monotypes distribute through non-empty cap}\ \} \\
 & R> \circ \sqcap(R\backslash(\mathcal{S}>)) \\
= & \qquad \{\ \text{junctivity of } \backslash\text{, see table}\ \} \\
 & R> \circ R\backslash \sqcap(\mathcal{S}>) \\
= & \qquad \{\ (33)\ \bullet\ \mathcal{S} \text{ is non-empty}\ \} \\
 & R> \circ R\backslash(\square\mathcal{S})>
\end{array}
$$

Since

$$
\begin{array}{ll}
 & R \circ R\backslash(\square\mathcal{S})> \\
= & \qquad \{\ R\ =\ R \circ R>\ \} \\
 & R \circ R> \circ R\backslash(\square\mathcal{S})>
\end{array}
$$

the theorem follows.

□

7 Discussion

Our concern here has not been to *establish* a mathematical theorem — that demonic composition is associative and distributes through demonic choice has been known for a long time[2] — but with economy and elegance of calculation. The exercise was prompted by discontent with our own and others' proofs using the axiomatic relational calculus. A useful by-product (and possibly the main contribution) of the exercise has been to identify the notion of monotype factor.

Performing this exercise has taught us some valuable lessons in efficient and economical calculation and we feel it is worthwhile to pass on some of those lessons to the reader. In order to make the discussion more concrete we briefly summarise aspects of the proofs given earlier by van der Woude [22] and Berghammer [5].

[2]Although we don't know for how long. Moreover, as remarked earlier, the theorems presented here are more general than the standard theorems in computing science texts since we do not exploit extensionality. It is thus not clear to us whether the specific theorems are indeed well-known.

Both van der Woude and Berghammer based their calulations on explicit, closed formulae for $R\,;\,S$. Specifically, van der Woude defined

$$(36) \qquad R\,;S \;=\; (R \circ S) \sqcap (\top\!\top \circ R)/S^{\cup}$$

($S^{\cup}$ being the converse of S) whilst Berghammer worked with the formula

$$(37) \qquad R\,;S \;=\; (R \circ S) \sqcap \neg(\neg(\top\!\top \circ R) \circ S)$$

(This latter formula was published earlier by Berghammer and Zierer [6].) It is well-known that $U/V \;=\; \neg(\neg U \circ V^{\cup})$ so it is clear that these two formula are equivalent. The equivalence of (36) to our own definition of $R\,;\,S$ is left as an exercise. (It should become clear after our discussion of monotypes versus vectors below.)

There are two main differences between the calculations given here and those of van der Woude and Berghammer. The first is that they both failed to spot and exploit the Galois connection underlying the definition of demonic composition. Its identification and the use of the factor notation to encourage the application of the cancellation rules streamlines the calculations considerably. The second is that the device used by van der Woude and Berghammer to restrict the domain of a spec is not composition with a monotype but, instead, intersection with a so-called (right) "vector". Let us explain this latter difference because it is also of fundamental importance.

Suppose $\mathbb{U}$ is a set and X is some subset of $\mathbb{U}$. Then there are two possibilities for representing X as a binary relation over $\mathbb{U}$. The choice made in this paper is to represent X as the monotype X_m where, for all $x,y \in \mathbb{U}$, $x\ X_m\ y \;\equiv\; x = y \wedge x \in X$. The choice made by Berghammer is to represent X by the so-called "vector" X_v where for all $x,y \in \mathbb{U}$, $x\ X_v\ y \;\equiv\; y \in X$. Recall that the defining characteristic of a monotype A is that $I \sqsupseteq A$. The defining characteristic of a vector V is that $V \;=\; \top\!\top \circ V$.

It is clear from the above that there is a (1-1) correspondence between monotypes and vectors. Specifically, we have, for all monotypes A and all vectors V,

$$(38) \qquad (\top\!\top \circ A)> \;=\; A \qquad \text{and} \qquad \top\!\top \circ V> \;=\; V$$

In addition we recall that the right domain operator was defined via a Galois connection between monotypes and vectors — see (5).

This close correspondence between monotypes and vectors makes the choice of which to use as representation of sets a particularly difficult one. Alternatively, one might argue that it doesn't make any difference which one chooses since calculations with monotypes can easily be converted into calculations with vectors and vice-versa! There is, however, one overriding argument why one should prefer monotypes to vectors and that is the dominant rôle of composition in programming applications. Let us explain.

A pattern of reasoning that appears repeatedly above can be summarised by the following schema:

$$\begin{array}{ll}
 & P \circ Q \\
= & \qquad \{ \text{ reason why } \; P \;=\; R \circ A \; \text{ for monotype } A \; \} \\
 & R \circ A \circ Q \\
= & \qquad \{ \text{ reason why } \; A \circ Q \;=\; S \; \} \\
 & R \circ S
\end{array}$$

Note that this calculation involves a silent use of the associativity of composition in the middle step. (Typically in our calculations the first step involved the expansion of the definition of demonic composition.)

The same calculation can be performed using vectors. Specifically, let V denote $\top\!\!\top \circ A$. Then the restriction $R \circ A$ on the right domain of R can equally be expressed as $R \sqcap V$ and the restriction $A \circ Q$ on the left domain of Q can be expressed by $Q \sqcap V^{\cup}$. The proof fragment becomes

$$\begin{array}{ll}
 & P \circ Q \\
= & \qquad \{ \text{ reason why } \; P \;=\; R \sqcap V \; \text{ for vector } V \; \} \\
 & (R \sqcap V) \circ Q \\
= & \qquad \{ \text{ For all } X,\, Y,\, Z, \\
 & \qquad\quad (X \sqcap \top\!\!\top \circ Y) \circ Z \;=\; X \circ (Y^{\cup} \circ \top\!\!\top \sqcap Z) \\
 & \qquad\quad X, Y, Z \;:=\; R, V, Q \; \} \\
 & R \circ (Q \sqcap V^{\cup}) \\
= & \qquad \{ \text{ reason why } \; Q \sqcap V^{\cup} \;=\; S \; \} \\
 & R \circ S
\end{array}$$

The invisible middle step — associativity of composition — has now been replaced by the application of a complicated and far-from-obvious calculation rule. Taking the step in a practical calculation (i.e. one in which R, V and/or Q are non-trivial expressions) becomes a non-trivial intellectual feat. (The rule can be made less complicated by splitting it into two simpler rules — see [6, theorem 2.1] — but that only makes the applicability of the step less obvious.)

The conclusion we would draw is that there is a substantial design element, having far-reaching consequences on ease of calculation, involved in the construction of a calculus. The choice of representation of basic concepts — here illustrated by the dichotomy between monotypes and vectors — is one such factor. The choice of notation that encourages instant recognition of calculational rules — here illustrated by the choice of the notation $R\backslash S$ to encourage recognition of the applicability of the cancellation rules — is a second factor. Last but not least, recognition of fundamental mathematical concepts and their formulation in the form of elegant calculational rules — here illustrated par excellence by the notion of a Galois connection — is a third factor in that design process.

References

[1] R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude. Polynomial relators. To appear: Proceedings of the 2nd Conference on Algebraic Methodology and Software Technology, May 22-25, 1991.

[2] R.C. Backhouse, P. de Bruin, G. Malcolm, T.S. Voermans, and J. van der Woude. Relational catamorphisms. In Möller B., editor, *Proceedings of the IFIP TC2/WG2.1 Working Conference on Constructing Programs*, pages 287–318. Elsevier Science Publishers B.V., 1991.

[3] R.C. Backhouse and R.K. Lutz. Factor graphs, failure functions and bi–trees. In A. Salomaa and M. Steinby, editors, *Fourth Colloquium on Automata, Languages and Programming*, pages 61–75. Springer-Verlag, LNCS 52, July 1977.

[4] R.C. Backhouse, T.S. Voermans, and J. van der Woude. A relational theory of datatypes. In preparation: copies of draft available on request, September 1991.

[5] R. Berghammer. Relational specification of data types and programs. Bericht Nr. 9109, Universität der Bundeswehr München, Fakultät für Informatik, September 1991.

[6] R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.

[7] Garrett Birkhoff. *Lattice Theory*, volume 25 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1948.

[8] J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, London, 1971.

[9] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics*. Springer-Verlag, Berlin, 1990.

[10] R.P. Dilworth. Non-commutative residuated lattices. *Transactions of the American Mathematical Society*, 46:426–444, 1939.

[11] G. Gierz, K. H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices*. Springer-Verlag, 1980.

[12] J. Hartmanis and R.E. Stearns. Pair algebras and their application to automata theory. *Information and Control*, 7(4):485–507, 1964.

[13] J. Hartmanis and R.E. Stearns. *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, 1966.

[14] Horst Herrlich and Miroslav Hušek. Galois connections. In Austin Melton, editor, *Mathematical Foundations of Programming Semantics*, LNCS 239, pages 122–134. Springer-Verlag, 1985.

[15] C.A.R. Hoare and Jifeng He. The weakest prespecification. *Fundamenta Informaticae*, 9:51–84, 217–252, 1986.

[16] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6:325–350, 1977.

[17] J. M. McDill, A. C. Melton, and G. E. Strecker. A category of Galois connections. In David H. Pitt, Axel Poigné, and David E. Rydeheard, editors, *Category Theory and Computer Science*, LNCS 283, pages 290–300. Springer-Verlag, 1987.

[18] A. C. Melton, D. A. Schmidt, and G. E. Strecker. Galois connections and computer science applications. In David H. Pitt, Axel Poigné, and David E. Rydeheard, editors, *Category Theory and Computer Science*, LNCS 283, pages 299–312. Springer-Verlag, 1987.

[19] Thanh Tung Nguyen. A relational model of nondeterministic programs. *International J. of Foundations of Computer Science*, 2(2):101–131, June 1991.

[20] Oystein Ore. Galois connexions. *Transactions of the American Mathematical Society*, 55:493–513, 1944.

[21] Martin Simons. Galois connections and pair algebras. Unpublished draft, December 1991.

[22] Jaap van der Woude. Free style specwrestling: Demonic composition and choice. In *Lambert Meertens, CWI, Liber Amicorum, 1966-1991*. Stichting Mathematisch Centrum, Amsterdam, January 1991.

Acknowledgement

Thanks go to Huub ten Eikelder and Wim Feijen for comments on an earlier draft of this report. Thanks also to the continuing interest and enthusiasm of members of the Eindhoven speculist club.

In this series appeared:

89/1	E.Zs.Lepoeter-Molnar	Reconstruction of a 3-D surface from its normal vectors.
89/2	R.H. Mak P.Struik	A systolic design for dynamic programming.
89/3	H.M.M. Ten Eikelder C. Hemerik	Some category theoretical properties related to a model for a polymorphic lambda-calculus.
89/4	J.Zwiers W.P. de Roever	Compositionality and modularity in process specification and design: A trace-state based approach.
89/5	Wei Chen T.Verhoeff J.T.Udding	Networks of Communicating Processes and their (De-)Composition.
89/6	T.Verhoeff	Characterizations of Delay-Insensitive Communication Protocols.
89/7	P.Struik	A systematic design of a parallel program for Dirichlet convolution.
89/8	E.H.L.Aarts A.E.Eiben K.M. van Hee	A general theory of genetic algorithms.
89/9	K.M. van Hee P.M.P. Rambags	Discrete event systems: Dynamic versus static topology.
89/10	S.Ramesh	A new efficient implementation of CSP with output guards.
89/11	S.Ramesh	Algebraic specification and implementation of infinite processes.
89/12	A.T.M.Aerts K.M. van Hee	A concise formal framework for data modeling.
89/13	A.T.M.Aerts K.M. van Hee M.W.H. Hesen	A program generator for simulated annealing problems.
89/14	H.C.Haesen	ELDA, data manipulatie taal.
89/15	J.S.C.P. van der Woude	Optimal segmentations.
89/16	A.T.M.Aerts K.M. van Hee	Towards a framework for comparing data models.
89/17	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra.

90/1	W.P.de Roever- H.Barringer- C.Courcoubetis-D.Gabbay R.Gerth-B.Jonsson-A.Pnueli M.Reed-J.Sifakis-J.Vytopil P.Wolper	Formal methods and tools for the development of distributed and real time systems, p. 17.
90/2	K.M. van Hee P.M.P. Rambags	Dynamic process creation in high-level Petri nets, pp. 19.
90/3	R. Gerth	Foundations of Compositional Program Refinement - safety properties - , p. 38.
90/4	A. Peeters	Decomposition of delay-insensitive circuits, p. 25.
90/5	J.A. Brzozowski J.C. Ebergen	On the delay-sensitivity of gate networks, p. 23.
90/6	A.J.J.M. Marcelis	Typed inference systems : a reference document, p. 17.
90/7	A.J.J.M. Marcelis	A logic for one-pass, one-attributed grammars, p. 14.
90/8	M.B. Josephs	Receptive Process Theory, p. 16.
90/9	A.T.M. Aerts P.M.E. De Bra K.M. van Hee	Combining the functional and the relational model, p. 15.
90/10	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra, p. 30. (Revised version of CSNotes 89/17).
90/11	P. America F.S. de Boer	A proof system for process creation, p. 84.
90/12	P.America F.S. de Boer	A proof theory for a sequential version of POOL, p. 110.
90/13	K.R. Apt F.S. de Boer E.R. Olderog	Proving termination of Parallel Programs, p. 7.
90/14	F.S. de Boer	A proof system for the language POOL, p. 70.
90/15	F.S. de Boer	Compositionality in the temporal logic of concurrent systems, p. 17.
90/16	F.S. de Boer C. Palamidessi	A fully abstract model for concurrent logic languages, p. p. 23.
90/17	F.S. de Boer C. Palamidessi	On the asynchronous nature of communication in logic languages: a fully abstract model based on sequences, p. 29.

90/18	J.Coenen E.v.d.Sluis E.v.d.Velden	Design and implementation aspects of remote procedure calls, p. 15.
90/19	M.M. de Brouwer P.A.C. Verkoulen	Two Case Studies in ExSpect, p. 24.
90/20	M.Rem	The Nature of Delay-Insensitive Computing, p.18.
90/21	K.M. van Hee P.A.C. Verkoulen	Data, Process and Behaviour Modelling in an integrated specification framework, p. 37.
91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.

91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.
92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.

91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.