The Thought Experiment Approach to Qualitative Physics

David L. Hibler
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Gautam Biswas*
Department of Computer Science
Box 1688, Station B
Vanderbilt University
Nashville, TN 37235

Abstract

This paper discusses the application of the thought experiment methodology to qualitative reasoning. Problem solving using this technique involves simplification of the original problem, solution of the simplified problem, and generalization of the results obtained. Our emphasis in this work is to demonstrate the effectiveness of this approach in addressing complexity and grain size issues that affect qualitative simulation. The thought experiment methodology is presented formally, the implementation of a problem solver called TEPS is briefly discussed, and the methodology is compared with related techniques such as approximation, aggregation, and exaggeration.

1 Introduction

The use of qualitative models in explaining the behavior of physical systems is an area of continuing research among the AI community. This paper develops the thought experiment methodology in the qualitative reasoning framework. The technique developed can be applied to a wide variety of problem solving situations.

Naive physics or common sense problem solving is characterized by the lack of precise and complete knowledge of all the constraint relations among parameters relevant to a physical situation under study. This results in analysis being performed on weakly constrained systems which causes two major problems: (i) a large multiplicity of possible solutions many of which may be physically incorrect, and (ii) the corresponding computational complexity in deriving the behavior of even simple physical systems. It might be argued that constraint relationships may be more easily obtained for human-engineered systems that have been built for a specific purpose. (Assuming nothing is broken, at least the system designers should be able to specify the complete and precise constraints). However, this is true only for very simple mechanical devices. As devices become more complex, the loose coupling of the component sub-systems makes the complete set of parameter relations harder to define.

*This work was supported by a summer grant from the Vanderbilt University Research Council.

For example, in an automobile engine it is not difficult to derive the relations that specify the location of each piston relative to the others, however, if one considers other components, such as the fuel pump, water pump, generator, and battery, the proliferation in the number of components and their weak coupling makes constraints harder to generate.

Conventional qualitative simulation methods provide no solutions for dealing with the multiplicity of solutions and the computational complexity that arise because of the underconstrained nature of the problems. In a recent paper, Falkenhainer and Forbus [1988] suggest a number of modeling assumptions to deal with the complexity issues in large engineering systems. This paper presents an alternate approach that we feel is more formal, and applies in a broader perspective.

Imaginary, simplified situations are often analyzed by human problem solvers in order to understand the principles behind more realistic situations. In Polya's words, "To sum up, we used the less difficult, less ambitious, special, auxiliary problem as a stepping stone in solving the more difficult, more ambitious, general, original problem." [Polya, 1957] In physics, this technique is referred to as a *thought experiment* [Prigogine and Stengers, 1984]. We formalize this heuristic method and use it for qualitative physics problem solving.

The thought experiment methodology uses two steps: Simplification and Generalization to solve problems that are too complex for direct qualitative simulation. It also deals with the grain size issue in a way that is largely automatic. The simplification step involves finding an imaginary physical system that is similar to the system under consideration, but, in some sense, is simpler to interpret and solve. Such a simplified system is called a prototype of the original system. The problem is solved for the prototype.

Generalization is also a two step process that involves: (i) Conjecture and (ii) Verification. A conjecture hypothesizes the solution to the original problem based on the solution obtained for the prototype. The next step is to verify this conjecture to formulate an acceptable hypothesis for the current problem. Verification may use formal methods or heuristic techniques. An acceptable hypothesis resulting from verification is called a generalization. If verified heuristically, a generalization is treated as a *default assumption,* i.e., it is accepted un-

til contradictory evidence disproves it at some later time. Simplification and generalization are related in that procedures for finding prototypes probably have associated generalization methods.

We emphasize that the methodology adopted for solving the prototype problem is basically independent of the general thought experiment technique. Extensive work by different groups has produced several suitable methods, such as the *Confluence* method of De Kleer and Brown [1984], the *Qualitative Process Theory* (QPT) of Forbus [1984], and the *QSIM* method of Kuipers [1986]. Our problem solver uses the Forbus aaproach for prototype problem solving. Instead of competing with these and other methods of qualitative reasoning, the thought experiment technique actually aims at extending the nature and size of the problems to which these methods can reasonably be applied. The price we pay for this extension is that all qualitative solutions that satisfy the constraints of the problem are no longer guaranteed. In fact, in some cases, simplification just eliminates irrelevant detail. In other cases, simplification may eliminate legitimate possibilities which would overwhelm the problem solver.

In many ways the thought experiment approach is similar to *analogical reasoning.* However, it is more restricted in that it does not involve the process of extrapolation of phenomena from one domain to another. Also, the solution to the problem for the prototype is not known until it is attempted as part of the thought experiment. In some sense it also resembles *approximation* techniques, but, as we discuss later, conceptually the simplification process is very different from approximation.

This paper presents a formal description of the thought experiment methodology for qualitative reasoning, and discusses a problem solver called TEPS (Thought Experiment Problem Solver) designed to encompass a number of other kinds of problem solvers. An example problem illustrates the application of this technique.

2 Thought Experiments: A Formal Description

The thought experiment methodology is formally developed as a state space problem solver.

2.1 State Space

States are described using *predicates.* The arguments of the predicates refer to domain objects and qualitative variables used to describe these objects. To avoid being side tracked, we postpone a discussion on qualitative variables (in terms of *quantity spaces* or *landmark* values) until Section 3. Predicates used for state description are called *descriptors.* For a given problem domain we assume a fixed set, Z, of possible descriptors. These form the descriptor space.

The qualitative state space, *Q* is defined as the power set of Z, (the set of all possible subsets of *Z),* i.e., *Q = P(Z),* where *P* denotes the power set function. Note that a qualitative state is just a set of descriptors without regard to the consistency or semantic correctness of the set. The only requirement is that it comes from the set of descriptors, Z, so as to be syntactically acceptable to the problem solver. A nonsensical or contradictory state may be flagged during the simulation process, though depending on the design of the problem solver, nonsensical input may produce nonsensical states as answers.

As a next step, we define descriptions of states in terms of *A,* a set of adjectives. With each adjective *i* in *A* is associated an adjective function, say F_i : *Q*—>*Q.* F, has the property: *q' — F_i(q) ==> q' C q,* i.e., the subset of descriptors of *q* which satisfy adjective *i* is *q'.* A state *q* can have more than one adjective; however, the focus is on *distinct* adjectives. Two adjectives are distinct if they always describe different parts for all states, i.e., *i* and *j* are distint if *Fi(q)* and *Fj(q)* are distinct for all *q.*

A description of a state is defined as a set of distinct adjectives which describe it, i.e., *D : Q —► P(A).* This can be looked upon as replacing different parts (subsets) of a state by corresponding adjectives that describe those parts. Any higher order structure built from states has a description which consists of the corresponding structure with states replaced by their descriptions. Two states with the same descriptors are considered identical. At the finest level of description the adjectives are the descriptors themselves and the description of a state is simply the state itself.

2.2 Thought Experiments

Assume a problem solver with the necessary information to reason about states in Q using qualitative simulation. The problem solver acting on a state *q* produces a directed graph of states starting at *q.* This directed graph represents the evolution of the state *q* with time. *Results,* a subset of the state graph, are obtained by a well defined method. This subset is ordinarily the leaves of the graph, and represents the collection of final states. Thus the problem solver has an associated results function: *R : Q —> P(Q). R(q)* is the set of states that constitute the result of the qualitative simulation. A more concise form of stating the results is *D(R(q)),* a set of descriptions of the states in *R(q).* Note that if several states in *R(q)* have the same description, this description occurs only once in *D(R(q)).*

A simplification, 5, is a function *S : Q —> Q* which maps states to simpler states. A prototype of *q* is *S(q),* where *S* is a simplification.

Direct qualitative simulation goes from a qualitative state, *q,* directly to *R(q). A* given description of *R{q)* is *D(R(q)). A* thought experiment goes through the following sequence:

q -> S(q) -> R(S(q)) -> C(R(S(q))).

The first step is the simplification, the next is the solution of the prototype and the last is the conjecture (with verification). The expectation is that *C(R(S(q))) — D(R(q)).*

The major components of the thought experiment scheme are explained in greater detail below.

Simplifications

Conceptually, simplifications can be derived from *simplification hierarchies.* *A* simplification hierarchy is a subset of *Q* that is partially ordered from simpler to more complex states. A simplification can then be described as a function *S* : *Q* —> *Q* which maps a state in a given level to a simpler level of the hierarchy, if one exists.

There exist a number of ways for embedding a state *q* in a simplification hierarchy. For example, one can induce a hierarchy by embedding any *feature* of the state in a hierarchy involving that feature. The term feature includes a state descriptor contained in the state, a subset of state descriptors in the state, or a argument to a state descriptor in the state. This feature hierarchy then induces a simplification hierarchy for the entire state.

Feature simplification could be built into the problem solver in a number of ways. One example involves *identical* objects. Given that the problem solver has type information about the kinds of arguments a predicate takes it can distinguish arguments which refer to objects. It can then determine whether a state contains identical objects by checking if the objects are in otherwise identical descriptors. This embeds the problem in a hierarchy involving the number of identical objects, and simplification involves reducing that number. For example, given a system with *n* identical objects *(n* large), the simplification procedure might suggest reducing the problem to one with only two identical objects.

Another method involves *numerical* arguments. If type information indicates that an argument to a descriptor is numerical then that value can be embedded in a numerical hierarchy such as zero, infinitesimal, finite, and infinite. The thought experiment method in this case would be similar to the exaggeration technique of Weld [1988]. A more detailed discussion of this issue is presented in Section 4.

Following QPT [1984], the problem solver contains modules describing generic objects *(individual views)* and causal processes *(processes).* Any such module may contain information that is useful for feature simplification. For example, an individual view describing a generic building structure may contain descriptors for a prototype building that could replace those for the more complicated building structure. Furthermore, if the problem domain imposes hierarchies on individual views, then this might form the basis for simplification. To continue the example, a spectrum of levels ranging from a very simple generic structure with just two rooms, to arbitrarily complex structures with multiple rooms can be defined. The level of complexity of the structures would depend on the number of rooms, and the spatial configuration of the rooms with respect to each other. These techniques are termed *Simplification by Abstraction* to contrast them from direct feature simplification. Examples of the use of abstraction hierarchies appear in Hibler and Biswas [1989].

Conjectures

A conjecture is a guess about a description of *R(q)* based on *R(S(q)).* The most precise description of *R(q)* would be *R(q)* itself, however, it is our belief that a less precise description often suffices to describe and explain behavior in qualitative problem solving. We must, therefore, add to the thought experiment process the ability to specify the type of description desired for the result. This is done by specifying a set of separate adjectives and associated adjective functions, which forms the *description basis.* The conjecture uses *R(S(q))* and the given description basis to find *D(R(q)).*

One obvious way to form conjectures is to use the inverse of the simplification 5. Unfortunately 5 is usually a many-to-one function, so the inverse of a state results in a set of states. Let *T* be the function on sets of states induced by the inverse of 5, i.e., *T* — *S~1.* (The bar above *S~J* indicates that *T* may be an approximation and not a true inverse in the mathematical sense). We define *T* : *P(Q)* -> *P(Q)* such that *T(X)* is the set of all states *q* such that *S(q)* is in A'. One conjecture method is to use the composition of *D* with *T* for the conjecture. Thus, we assume *D(R(q))* = *D(T(R(S(q))))).*

The function T need not be defined on *R(S(q)).* Therefore, if the feature involved in the simplification changes from *S(q)* to *R(S(q))* it may not be clear what the correct inverse should be. Even if it is, it is often better to pick a different *inverse-like* function that has better properties, e.g., it is more specific. Note that the construction of *T* from 5 requires that we define 5 not only for the particular state *q* which was to be simplified, but for all qualitative states *q* that are meaningful for that problem. There could be many such definitions.

Our experiences indicate that the easiest way to form conjectures is to get away from inverses and use *cross descriptions.* *A* cross description is a description basis which is applicable across the levels of the hierarchy which generates 5. For cross descriptions the conjecture is *D(R(q))* = *D(R(S(q))),* i.e., the same description applies to *R(q)* and *R(S(q)).* As an example, consider a problem with 20 objects in a row, and the description refers to object 20. Directly applying a simplification that reduces the number of objects to less than 20 does not produce a cross description that can be applied across the simplification hierarchy. On the other hand, reformulating the problem so that the description refers to the last object in the row, makes the description general enough so that it applies to rows of arbitrary length, and, therefore, across the hierarchy.

It might be argued that the composition of a description *D* with a simplification inverse *T* is essentially the same as constructing a cross description, but this is usually not true because *D* does not refer to the entire state. In the example with the 20 objects, since the final result is desired only for object 20, it really does not matter how the mapping is performed on the first 19 objects, so in defining *D,* no commitment has to be made on these mappings. Therefore, the cross description mapping will be inherently simpler.

Verification

Verification can be rigorous or heuristic. It could even be empirical. If a rigorous proof cannot be established, a possible method for verification is to use different simplifications and see if they produce the same result. If $C_1(R(S_1(q)))$ = $C2(R(S2(q)))$ this increases our confi-

dence in the result.

3 TEPS - A Thought Experiment Problem Solver

The basic steps of TEPS implemented in Prolog are outlined below: (i) determine the input description in terms of a set of state descriptors, and the description basis that specifies the information desired in the final state of the system, (ii) apply Simplification, i.e., pick an appropriate simplification procedure from a list of simplifications, and create a prototype problem, (iii) perform Envisionment, i.e., apply the QPT simulator by identifying and firing all active processes at each step till the graph of states cannot be extended, (iv) Generalize, i.e., find an appropriate conjecture procedure, apply, and try to verify, and (v) output some or all state descriptors of the final state as requested. A detailed example is presented below, but a more complete description of the algorithm and the implementation is available in Hibler [1988].

The problem solver has the following: (i) a set of processes (ideally this would be very extensive, but, for now we restrict it to a particular domain of interest, e.g., static electricity and sets of pendulums), (ii) a set of simplification procedures (these include conditions for triggering the procedures, i.e., applicability conditions), (iii) a set of conjecture procedures (these include conditions for triggering the procedures, in particular, which simplification they might correspond to), and (iv) a set of verification procedures (they should include specification of the conjecture procedures for which they are appropriate). The verification procedure may call qualitative simulation in order to test the conjecture in other cases.

The qualitative simulation method is implemented as a generalization of QPT [Forbus, 1984]. A major difference is that the QPT focuses on numerical-valued parameters. Even though qualitative values are used for these parameters the approach tends to be too specialized for our purposes. We replace the quantity space with an attribute space. An attribute space is a directed graph. The nodes of the graph represent qualitatively significant regions. An edge between two nodes indicates that the regions are adjacent in terms of the possibilities for change of attribute of an object. The direction of the edge indicates direction of change. Thus edges serve as directional derivatives. For example, consider a solid conducting rod in the static electricity domain. Rather than consider geometrical regions based on the shape of the rod, or locational coordinates, a more elegant description that suffices for most qualitative static electricity problem solving is to consider the division of space into three topological regions: the *interior, surface,* and the *exterior* of the conductor, with

int(c) —> surf(c) —> ext(c)

defining the adjacency regions. A more complete discussion of the attribute space approach and it's effectiveness appear in Hibler and Biswas [1989]. For comparison purposes, our attribute space approach is somewhat similar to the Kuipers and Byun [1988] qualitative methods designed for a robot learning a spatial environment.

A major issue that arises in the description of TEPS as a general problem solver is the issue of *control.* In other words, given a number of possible simplifications (e.g., the ones discussed in Section 2; others are discussed in Hibler and Biswas [1989]), how does the program decide which simplification is the most appropriate in a particular situation? One way to answer this question is to check whether appropriate conjecture and verification procedures can be applied after the results of the prototype system have been derived. This implies that the simplification and generalization methods are *closely tied,* and heuristics can be defined that rank the suitability of simplification and generalization procedures based on the problem definition and desired results. We have investigated a number of simplification and conjecture techniques in Hibler and Biswas [1989]. However, our emphasis in this paper is to demonstrate the effectiveness of the basic methodology.

Even for the class of simplifications involving identical objects the problem solver can solve a wide variety of problems. We demonstrate this for the domain of static electricity in Hibler [1988]. For illustrative purposes, we discuss the problem solving steps for one of these problems. This problem involves a line of pendulums hanging from a support under the force of gravity. Each pendulum consists of a sphere hanging by a non conducting thread or string. Each sphere has the same radius, each pendulum has the same length, and the separation between adjacent pendulums is the same. A certain amount of charge is placed on one or more of the spheres. The problem is to determine properties of the final state of the array of pendulums. The final state consists of the final position of each pendulum and the final distribution of charge among the pendulums. It is assumed that each pendulum motion is damped due to viscosity of the air or the nature of the supporting thread so that a static configuration is reached quickly.

To be more specific, consider a certain amount of charge, c, is placed on the first of the series of pendulums. There are a total of 30 pendulums and all but the first have zero initial charge. The aim is to determine various descriptions of the final state of the pendulums.

How might a human describe what happens? Using reasonable assumptions, a human might decide that the initial charge on the first pendulum causes a polarization of the charge on the second. Thus, even though the second pendulum has no net charge the result is an attraction between these adjacent pendulums. These pendulums move toward each other. Assuming they touch and are conductors, charge is shared equally. The two pendulums now have like charges and, therefore, repel. Having analyzed this, the result can then be propagated down the line of pendulums. (See Figure 1).

Although we can easily describe what happens, a direct qualitative simulation using conventional techniques is hopelessly involved. The reason is that relative timing information on the motion of each pendulum absent, and, therefore, too many different possibilities have to be considered at once. To explain using an analogous

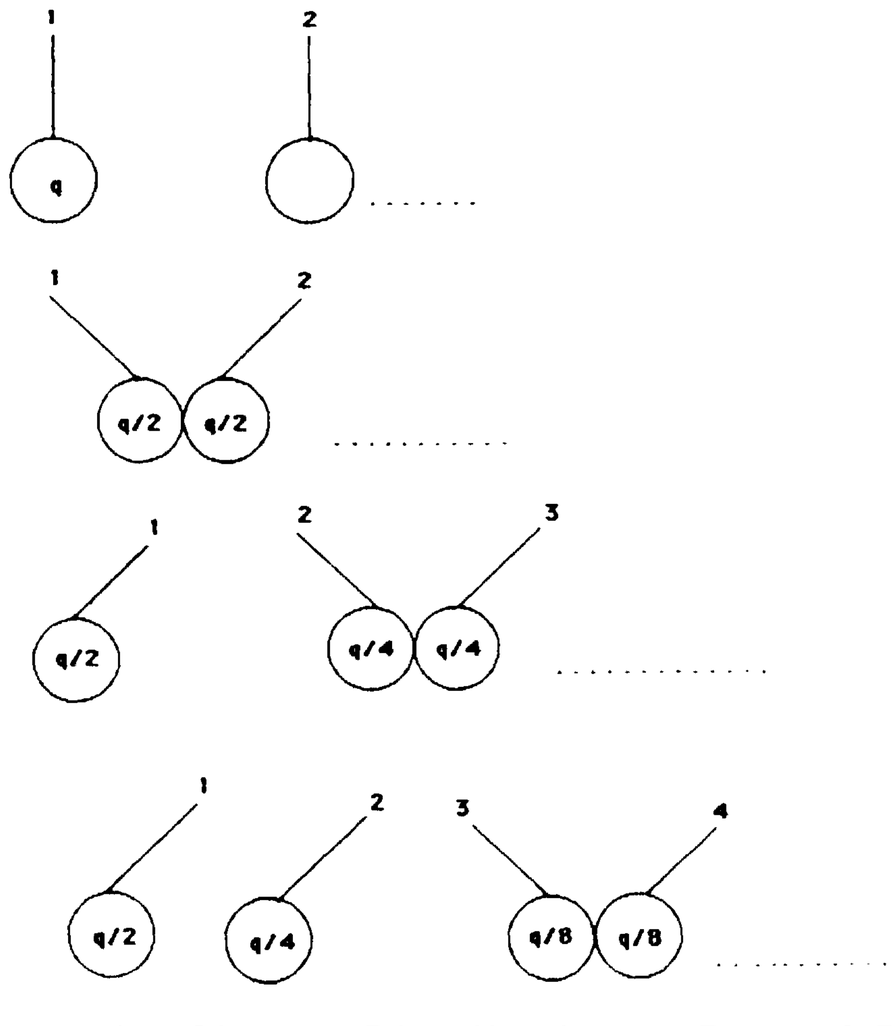

Figure 1: Charge Distribution - Row of Pendulums.

example, a qualitative simulation of a horse race without a relative ranking of the speeds of the horses, would indicate that every possible ordering (or permutation) of horses was a possible outcome for the race.

For this example, the input to TEPS for this problem is given below.

problem(state: [conductor(p),pendulum(p),mobile(p), location(p,center(p)),charge(p(I),c),number(p,N,30)],
unknowns: [charge(p,X)]).

The first part is a specification of the initial state *q.* The number predicate indicates that there are 30 objects each of which has the properties indicated for the generic object p. This representation for identical objects triggers the simplification procedure before a more involved analysis is initiated. The predicate involving the specific object p(I) is carried through unchanged to the prototype. Simplification eliminates the number predicate and constructs a prototype involving two pendulums, *S(q). A* qualitative simulation, *R(S(q)),* is performed on this prototype. Once specific processes and individual views are defined, the qualitative simulation process *R* is very similar to the method of Forbus [1984].

The conjecture procedure associated with the simplification determines a description basis from the specification for unknowns. Type information for the charge predicate indicates to the system that the charge is numerical. The adjective set is *A* — {+, -, 0}, *f+(q)* is the set of all descriptors in *q* of the form charge(p(X),Y), where Y is positive. *f-(q)* and *fo(q)* are defined similarly. *D(R(S(q)))* turns out to be {{+}}, i.e., for all states in *R(S(q))* all objects have positive charge. (On the other hand, a result ({+, -}, {0}) would imply that for some states some objects have positive charge and some objects have negative charge, and for other states all objects have zero charge).

Using cross descriptions, we conjecture *D(R(q))* to be {{-f}} too. Verification checks that the same description holds for the case involving three and then four pendulums and produces the answer: [result(allstates,charge(p,[-f]))]. In future, an induction technique wil be developed to formally verify the result. If the location of the pendulums were the desired result, the answer would be expressed in terms of left, right, and center.

The TEPS problem solver is set up to produce results at different levels of detail. In the problem specification given above, if the unknowns predicate is modified to read: charge(p(30,X)), and the description basis is changed from +,—,0 to *exact,* a more detailed solution is attempted.

In this case, the envisionment on the prototype *S(q)* determines that charge on p(2) is c/2, half of the original charge on p(I). (For a detailed step by step solution see Hibler [1988]). The specification of an *exact* description basis, and the fact that amount of charge on a specific pendulum is desired, triggers a different conjecture procedure than the one discussed above. This procedure is based on *dimensionless* ratios, and results in the listing of all dimensionless ratios between the final derived quantity and the original quantities in the prototype. For this problem, it is ½, and it relates to charge on the pendulums. The procedure then proceeds to conjecture that the ratio for the actual problem is a simple function of the prototype. A library of simple functions are built into the system. Verification then entails examining enough values of N to determine the function's parameters and to confirm its correctness for at least two additional values. Suppose verification fails for all functions, the system attempts to produce results at a coarser level of description. In this case the result produced would be + , i.e., there exists some amount of positive charge on the last pendulum.

The program is set up to work with arbitrary N (e.g., number(p,N,large)). The problem is then solved symbolically, and cross description used to conjecture the general solution. The solution for pendulum N of N pendulums is $(½)^{N-1}$

From the above discussion, the effectiveness of thought experiment methodology in dealing with complexity arising from a large number of identical objects has been demonstrated. The specification of the description basis and the description hierarchies produce results at the desired level of granularity. However, if the solution process becomes computationally infeasible or a solution cannot be derived (probably because of lack of sufficient details), TEPS sutomatically attempts to generate solutions at coarser levels of detail. Thus the manner of handling the grain size issue is largely automatic. Note that results produced by the problem solver can be speculative, especially if a formal verification procedure cannot be applied. Also, as the concept of complexity is expanded beyond just large numbers to include other situations, like complexity of processes and individual views or system description in terms of large numbers of interacting parameters, the simplification-conjecture strategy may produce inaccurate results. But they may still be com-

patible with intuitive and default notions that humans formulate when they reason with incomplete knowledge. This issue is discussed in more detail in the next section.

4 Discussion

How does the thought experiment method compare with other techniques for handling complex problems? Falkenhainer and Forbus [1988] discuss the use of *simplifying assumptions* based on *CONSIDER* statements to decompose a domain into different grain sizes and perspectives which may be reasoned about separately. The thought experiment method is more general than the *CONSIDER* method. Changing grain size and perspective can both be considered simplifications in our method. However, simplifications can generate other kinds of changes, therefore, overall it is more powerful. This power is purchased at the price of some disadvantages. When more extreme simplifications (as in the pendulum problem) are made, the correspondence between problem and prototype is more complex. Also, the results of this type of thought experiment often represent reasonable beliefs about the behavior of the system and may not be as strong as the results of a qualitative simulation.

Consider the problem of a building that has a sunny side and a shady side. Can our problem solver reach the intuitively reasonable conclusion that the warmest-rooms are on the sunny side? It is quite possible for a room on the sunny side to be cooler than one on the shady side. For example, a long hallway could carry heat to a poorly insulated room on the shady side, while a well insulated room on the sunny side might stay cool. A qualitative simulation because it generates all possibilities, would contain both predictions with no clue to indicate which one is more likely. TEPS solves this problem by first generating a prototype for a simple building. This prototype comes from a module (an individual view) which describes a generic building. The conjecture predicts the same pattern of variation for the temperatures in the prototype and in the original building. Thus the warmest rooms are on the sunny side. This example also shows that conjectures need not always be correct, but they do seem to correspond to default notions (e.g., birds fly) that humans use.

Other researchers have stated the importance of simplification. Iwasaki and Bhandari [1988] say "Abstracting a detailed description to produce a simpler description is essential in reasoning about a complex system". Aggregation of variables has been suggested and fits in our framework. Aggregation involves replacing the variables in a problem by other variables each of which depends on a collection of the original variables. Basically, any approximation technique can be used for simplification, and the approximation of the original system becomes the prototype. The exaggeration technique of Weld [1988a,1988b] uses extreme perturbations so that parameters have infinite or infinitesimal values. His transform, simulate, and scale correspond roughly to our simplify, solve, and conjecture. An advantage of the TEPS framework, is that it allows even more drastic versions of some of these methods to be used.

In summary, the thought experiment methodology is viable, it generalizes a number of problem solving schemes, and it adds a new dimension to qualitative problem solving. It's effectiveness in handling size complexity and grain size issues have been demonstrated.

References

[DeKleer and Brown, 1984] De Kleer, J. and J. S. Brown, "A Qualitative Physics Based on Confluences", *Artificial Intelligence,* vol. 24, pp. 7-83, 1984.

[Falkenhainer and Forbus, 1988] Falkenhainer, B. and K.D. Forbus, "Setting up Large-Scale Qualitative Models", *Proc. Seventh National Conference on Artificial Intelligence,* Minneapolis, MN, pp. 301-306, August 1988.

[Forbus, 1984] Forbus, K.D., "Qualitative Process Theory", *Artificial Intelligence,* vol. 24, pp. 85-168, 1984.

[Hibler, 1988] Hibler, D.L., Qualitative Physics for Complex Regular Systems, M.S. Thesis, Univ. of South Carolina, Columbia, S.C., 1988.

[Hibler and Biswas, 1989] Hibler, D.L. and G. Biswas, "TEPS: Applying the Thought Experiment Methodology to Qualitative Problem Solving", in review, *IEEE Trans, on PA MI,* 1989.

[Iwasaki and Bhandari, 1988] Iwasaki, Y., and Bhandari, I., "Formal Basis for Commonsense Abstraction of Dynamic Systems", *Proc. Seventh National Conference on Artificial Intelligence,* Minneapolis, MN, pp. 307-312, August 1988.

[Kuipers, 1986] Kuipers, B., "Qualitative Simulation", *Artificial Intelligence,* vol. 29, pp. 289-338, 1986.

[Kuipers and Byun, 1988] Kuipers, B., and Y. T. Byun, "A Robust, Qualitative Method for Robot Spatial Learning", *Proc. Seventh National Conference on Artificial Intelligence,* Minneapolis, MN, pp. 775-779, August 1988.

[Polya, 1957] Polya, G., How To Solve It, Doubleday &: Company, p. 196, 1957.

[Prigogine and Stengers, 1984] Prigogine,I., and I. Stengers, Order Out of Chaos, Bantam Books, p. 43, 1984.

[Weld, 1986] Weld, D. S., "The Use of Aggregation in Causal Simulation", *Artificial Intelligence,* vol. 30, pp. 1-34, 1986.

[Weld, 1988a] Weld, D. S., "Exaggeration", *Proc. Seventh National Conference on Artificial Intelligence,* Minneapolis, MN, pp. 291-295, August 1988.

[Weld, 1988b] Weld, D. S., "Comparative Analysis", *Artificial Intelligence,* vol. 36, pp. 333-373, 1988.