TEPS: THE THOUGHT EXPERIMENT APPROACH TO QUALITATIVE PHYSICS

David L. Hibler * and Gautam Biswas **

Department of Computer Science
University of South Carolina
Columbia, S.C. 29208.

Department of Computer Science
Vanderbilt University
Box 1688, Station B
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. 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,

*This work is based on D. Hibler's Ph.D. dissertation research.

**This research was supported in part by a summer grant from the Vanderbilt University Research Council.

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 until 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 approach 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 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.

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 $F_i:Q\rightarrow Q$. F_i has the property: $q' = F_i(q) \rightarrow q' \subset q$, 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 distinct if $F_i(q)$ and $F_j(q)$ are distinct for all q.

A **description** of a state is defined as a set of distinct adjectives which describe it, i.e., $D:Q\rightarrow P(A)$. 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.

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: $R:Q\rightarrow P(Q)$. $R(q)$ 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, S, is a function $S:Q\rightarrow Q$ 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:

$$q\rightarrow S(q)\rightarrow R(S(q))\rightarrow C(R(S(q))).$$

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 $S:Q\rightarrow Q$ 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 an 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 [1988a]. A more detailed discussion of this issue is presented in Section 4.

Following QPT [Forbus, 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.

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 S. Unfortunately S 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 S, i.e., $T=\overline{S^{-1}}$. (The bar above S^{-1} indicates that T may be an approximation and not a true inverse in the mathematical sense). We define $T:P(Q)\rightarrow P(Q)$ such that $T(X)$ is the set of all states q such that $S(q)$ is in X. 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 S requires that we define S 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 S. 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 $C_1(R(S_1(q)))=C_2(R(S_2(q)))$ this increases our confidence 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., fluid mechanics, heat and the refrigeration process), (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

$$int(c) \rightarrow surf(c) \rightarrow ext(c)$$

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), 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. The present version of TEPS uses a simple additive mechanism, always choosing the highest ranked simplification which applies.

Two kinds of simplification principles have been implemented in the current TEPS prototype. The first deals with *Feature Simplification* applied to identical objects. The second more powerful principle, is termed *Simplification by Abstraction*. This principle defines a set of rules based on an abstraction hierarchy of the problem space descriptors that help determine which simplifications are best suited for a particular problem, and how successive simplification results can be related to each other. Feature simplification based on identical objects is discussed in Hibler and Biswas [1989a].

The Simplification by Abstraction principle uses two default rules that relate to: (i) the maximum abstraction level, and (ii) description inheritance. Given a simplification hierarchy based on abstractions, the obvious question is what level of abstraction in the hierarchy would produce the "best" simplification. The idea here is to choose one that makes problem solution easy, i.e., an underlying motivation is "economy of computations." For particular problem types, specific rules that dictate the most appropriate level of simplification may exist. (Humans usually acquire this information from problem-solving experience).

However, in situations where a specific rule cannot be found, a default strategy may be to use maximum possible abstraction which should correspond to the greatest simplification. If the thought experiment based on this simplification is successful, probably a minimum amount of computational effort has been expended in solving the problem. Even if the thought experiment is not successful, we still may have made progress due to the inheritance principle that is discussed later.

Unfortunately, the maximum abstraction principle may not be sufficient to generate the necessary simplification because the simplification hierarchy (a subset of Q) may only be partially ordered from simpler to more complex states. A partial (as opposed to total) order implies that there may be more than a single "simplest" state. In such situations, rules to guide the choice of the appropriate simplification need to be developed. An appropriate heuristic in this situation may suggest picking the simplification which relates to the description type desired as the answer to the problem.

The second default principle for abstraction is description inheritance, which helps relate simplifications to one another. This rule states that any applicable description of states (or sets of states) at a more abstract level in a simplification hierarchy can be assumed to apply at more specific (less simplified) levels unless contradicted by results or information at that level. The inherited information can help in forming conjectures at the more refined levels. Also, if simplification at a particular level fails to produce a solution, the problem solver can retract to previous more abstract levels for which the problem solution can be obtained without significant amounts of recomputations. The next section discusses an example that deals with the abstraction or grain size issue in the problem solving process.

Example Problem: Simplification by Abstraction

To illustrate the use of the Simplification by Abstraction principle in TEPS, consider the problem reasoning about fluids as given by Collins and Forbus [1987]. They discuss several problems, such as pumped flow, refrigeration systems, and part of a navy propulsion plant, involving heat and fluids in motion in mechanical systems. To analyze these problems they define two different ontologies: the "contained-stuff" ontology, and the "molecular-collection" (MC) ontology. MC is a specialization of Hayes [1984, 1985] "piece-of-stuff" ontology and "contained-stuff" is a generalization of his "contained-liquid" ontology.

Collins and Forbus first analyze the system of interest using the contained-stuff ontology. This describes the situation in terms of a few discrete objects consisting of the fluid in locations specified by natural boundaries. For example, a typical refrigerator, illustrated in Figure 1, has an evaporator, a compressor, a condenser, and an expansion valve that constitute natural locations. At each location the "stuff" could be liquid or gas or both. Using this contained-stuff view they determine which processes such as flow, boiling, and condensation are active. Rules based on these processes help to determine what happens in the MC ontology. In the MC ontology, they examine one molecular collection as it moves through the system. Rules connected with the flow process from the contained-stuff view determine the possible motion of the molecular collection. Other rules determine the way in which other properties of the molecular collection can change. For example, when a fluid boils, it absorbs heat; during condensation it emits heat.

Collins and Forbus show how this method of alternate views can be used to determine that a refrigerator pumps heat uphill to a higher temperature. They also suggest that this method can be the basis for a differential qualitative analysis to solve such problems as determining the result of increasing feedwater temperature to the boiler of a propulsion plant. (This is a problem given at the Surface Warfare Officers' School.)

The thought experiment approach to the refrigerator problem is outlined below. It illustrates how the problem solver picks the right grain size for addressing different aspects of the problem. For brevity, we discuss in outline the problem-solving steps executed by TEPS; interested readers can find details of the implementation in [Hibler and Biswas, 1989c].

TEPS starts with a description of the refrigerator's fluid system and the specification of the unknown for the problem, i.e., heat flow from the evaporator to the condenser. In the first step, the unknown causes instantiation of a description basis. This frame is set up to give a reasonably wide classification of possible behaviors for the "flow" of a quantity from one "place" to another. It includes alternatives, such as positive flow, negative flow, zero flow, and oscillating flow. Several description bases may be possible. These would be totally ordered from finest to coarsest. If a particular level of description were not specified the system

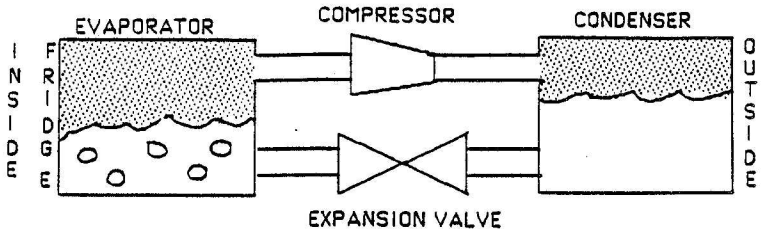

FIGURE 1 : REFRIGERATION UNIT

Locations: Evaporator, Compressor, Condenser, Expansion Valve

Qualitative description of these four "objects"

PROBLEM:

What is the heat flow from evaporator to condenser?

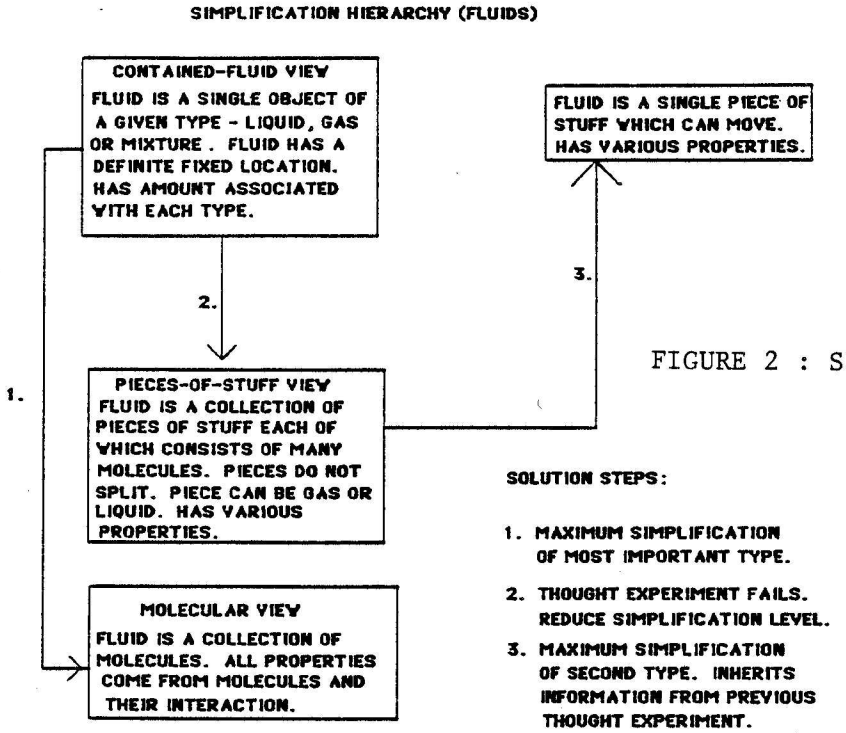

FIGURE 2 : SIMPLIFICATION STEPS

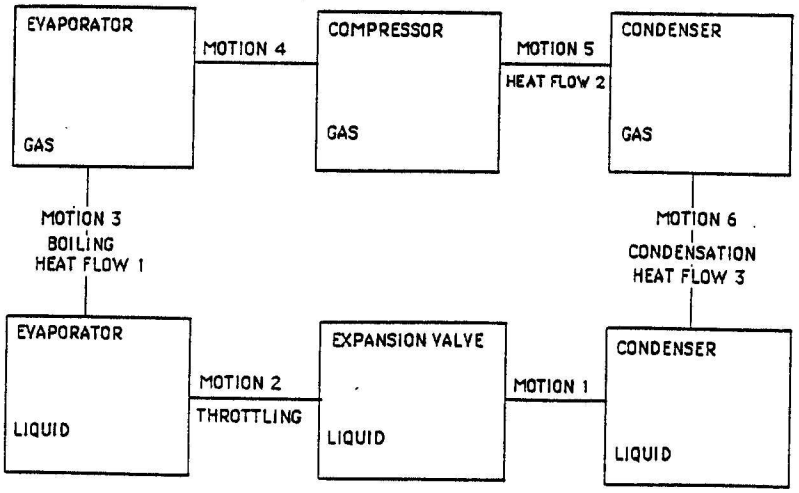

FIGURE 3 : SECOND SIMPLIFICATION

would attempt the finest (most refined) level, but if verification at this level failed coarser descriptions would be attempted.

The overall flow of control (i.e., choice of simplification steps) is shown in Figure 2. The second step is the first iteration of a thought experiment. An individual view involving fluids indicates a feature hierarchy for fluids that can be used for simplification. This ranges from a molecular view to a collection of "pieces-of-stuff" (each piece small but consisting of many molecules), to a contained-fluid view. In addition, the individual view for fluids contains a weight associated with this feature to indicate its relative importance with respect to other possible feature hierarchies that may be applied for simplification. The system need not know which simplifications are incompatible a priori. It simply tries simplifications in the order of their importance. If two simplifications are similar (e.g., they come from the same hierarchy), and the more important one is found to be incompatible, preconditions for the second one will not be satisfied after the first is found incompatible. Thus the system avoids unnecessary repetitions. Using the default abstraction principle and the weighting heuristic, the above problem simplifies to a contained-fluid problem.

A compressor can be modeled in different ways. A simple model assumes that it maintains a constant fluid flow between intake and outlet as long as the pressure difference opposing the flow between the two is less than some maximum. The expansion valve can be modeled by requiring that it allows no fluid flow if the pressure difference tending to cause the flow is less than a certain minimum. Above this minimum the fluid flow is qualitatively proportional to pressure. These relations come from the individual views for the compressor and expansion valve, respectively. A relation from an individual view for fluid in a closed container would state that the pressure in the container is qualitatively proportional to the amount of fluid. Combining the previous relations the system is able to determine that the refrigerator will reach a steady state in which there is higher pressure in the condenser than in the evaporator.

With proper assumptions about the parameters working out the details (see [Hibler and Biswas, 1989c]) shows that the boiling process will occur in the evaporator and the condensation process will occur in the condenser. (The boiling point of a liquid is qualitatively proportional to the pressure.)

There is not enough information to calculate heat flow because this depends on information concerning the internal energy of the fluid entering and leaving the different locations. When the system attempts to apply the description basis to the singleton set of result states it finds no answer for heat flow. (This is different from zero heat flow.) Thus the first thought experiment fails.

At this point, still using the abstraction principle, the system attempts a thought experiment at the next level of abstraction, i.e., the pieces-of-stuff level (Figure 2). Information about the previous level, including process information, is saved temporarily. At this stage, two things happen. Description inheritance comes into action, and another type of simplification is possible.

The new simplification-type (Step 3, Figure 2) at the pieces-of-stuff level is a numerical one. The system attempts to solve the problem using one piece-of-stuff. Appropriate processes at the piece-of-stuff level are activated. As discussed, these processes can use the inheritance mechanism to determine results. The history of the piece-of-stuff can now be determined as it goes through the system (Figure 8). This analysis in terms of a single piece-of-stuff is commonly used by standard texts. "As usual, we consider the pressure and volume changes of a constant mass of fluid as it is conveyed from the liquid storage, where it is at the temperature and pressure of the condenser through the throttling valve, through the evaporator, into the compressor, and finally back to the condenser" [Zemansky and Dittman, 1981, p. 149]. Processes similar to those at the higher level determine what happens to the "piece-of-stuff" from the process description at the higher level. The individual view for the piece-of-stuff starts the piece-of-stuff arbitrarily with characteristics which are consistent with the information inherited from the higher level. Thus, as Zemansky says we could start with the fluid as a liquid located in the condenser, having pressure and temperature of the condenser. A motion process determines the motion of the piece-of-stuff based on the fluid flow process at the higher level. This takes the piece-of-stuff through the expansion valve. A boiling process examines the information from the higher level and determines that the fluid now boils at this location. A heat flow process triggered by the boiling determines that heat flows into the piece-of-stuff during boiling. As it goes around the cycle, the various processes determine that the piece-of-stuff vaporizes, rises, goes through the compressor, condenses, and emits heat returning to its original state. At each step, processes and individual views may examine the state, processes, and individual views at the higher level to determine what happens. This examination is built into the process and view structure. The description basis applied to this simulation

determines that the refrigerator pumps heat uphill to a higher temperature.

This problem solution illustrates that the use of an appropriate abstraction hierarchy and suitable heuristics, TEPS can handle problems at different levels of granularity and obtain desired results. A number of different levels of granularity may have to be tried, but the grain size at which the problem is solved is determined automatically by the system without external intervention. A similar analysis should be applicable to other types of problems.

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

[Collins and Forbus, 1987]
Collins, J.W., and K.D. Forbus, "Reasoning about Fluids via Molecular Collections", *Proc. Sixth National Conference on Artificial Intelligence*, Seattle, WA, pp. 590-594, July 1987.

[DeKleer and Brown, 1984]
DeKleer, 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.

[Hayes, 1984]
Hayes, P.J., "The Naive Physics Manifesto", from **Expert Systems in the Microelectronic Age**, D. Michie, ed., Edinburg Univ. Press, pp. 242-270, 1984.

[Hayes, 1985]
Hayes, P.J., "Naive Physics 1: Ontology for Liquids", from **Formal Theories of the Commonsense World**, Ablex Publishing, 1985.

[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, 1989a]
Hibler, D.L., and G. Biswas, "The Thought Experiment Approach to Qualitative Physics", to appear, *Proc. IJCAI-89*, Detroit, MI, August 1989.

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

[Hibler and Biswas, 1989c]
Hibler, D.L., and G. Biswas, "Thought Experiment Problem Solving Using Abstraction Hierarchies", *Technical Report CS-89-06*, Dept. of Computer Science, Vanderbilt University, Nashville, TN, 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, 1988a]
Weld, D. S., "Exaggeration", *Proc. of the 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.

[Zemansky and Dittman, 1981]
Zemansky, M.W. and R.H. Dittman, **Heat and Thermodynamics: An Intermediate Textbook**, McGraw Hill, New York, NY, 1981.