Setting up Large-Scale Qualitative Models

Brian Falkenhainer and **Kenneth D. Forbus**
Qualitative Reasoning Group
Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Avenue, Urbana, Illinois 61801

Abstract

A qualitative physics which captures the depth and breadth of an engineer's knowledge will be orders of magnitude larger than the models of today's qualitative physics. To build and use such models effectively requires explicit *modeling assumptions* to manage complexity. This, in turn, gives rise to the problem of selecting the right qualitative model for some purpose. This paper addresses these issues by describing a set of conventions for modeling assumptions. *Simplifying assumptions* decompose a domain into different grain sizes and perspectives which may be reasoned about separately. *Operating assumptions* reduce the complexity of qualitative simulation by focusing on particular behaviors of interest. We show how these assumptions can be directly represented in Qualitative Process theory, using a multi-grain, multi-slice model of a Navy propulsion plant for illustration. Importantly, we show that model selection can often be performed automatically via partial instantiation. We illustrate this technique with a simple explanation generation program that uses the propulsion plant model to answer questions about physical and functional characteristics of its operation.

1 Introduction

A long-range goal of qualitative physics is to develop systematic models that capture the breadth and depth of human reasoning about the physical world. Such models will be crucial for future intelligent computer-aided design and tutoring systems. Clearly, they will need to be orders of magnitude larger than today's models. Furthermore, they must capture phenomena at several levels of detail, and from a variety of perspectives. Building such models raises several new issues for qualitative modeling:

1. *Organization problem:* How can we organize a model that captures phenomena at a variety of grain sizes and perspectives?
2. *Relevance problem:* Generating all possible states becomes intractable as the size of system modeled grows. Even if we could generate them all, often we only care about a subset of the behavior. How can we use qualitative simulation in a more focused way to answer questions?
3. *Selection problem:* As models get larger, complete instantiation becomes both undesirable and impossible. No one understands spilling a cup of coffee via quantum mechanics. Furthermore, some of the perspectives in a model will be mutually incompatible. How can an appropriate subset of a model be selected for reasoning, given a particular question?

This paper addresses each of these issues. In particular, we claim the key idea in solving all of them is a set of conventions for explicitly representing *modeling assumptions*. We introduce explicit *simplifying assumptions* to solve the organization problem by providing "scoping", delimiting when descriptions are and are not applicable. We introduce *operating assumptions* to describe standard behaviors or default conditions. We illustrate how, using these conventions, the selection problem can in some cases be solved automatically via partial instantiation. These conventions are illustrated using a multi-grain, multiple perspective high-level model of a Navy propulsion plant. We demonstrate our solution to the model selection problem by showing how, in the context of a tutoring system, the form of a question can be analyzed so that the appropriate set of modeling assumptions can be automatically computed.

In the next section we outline our perspective on qualitative modeling, showing the need for explicit modeling assumptions to control model instantiation and use. Section 3 gives a brief tour of the steam plant and its qualitative model which provides our motivating example. Section 4 describes our conventions for modeling assumptions, and Section 5 shows how they are used to organize the steam plant model. Section 6 describes our algorithm for automatically computing a minimal set of simplifying assumptions for a given query. Finally, we discuss directions for future research.

2 The Modeling Process

We call the system or situation being modeled the *scenario*, and its qualitative model the *scenario model*. The simplest way to build a scenario model is to create a model of that specific scenario for a particular purpose. While useful systems may be built this way, it is also easy to generate ad hoc models of dubious merit, where the model must be thrown away whenever the scenario or purpose changes slightly. An indirect route is more robust – build first a general-purpose *domain model*, which describes a class of related phenomena or systems. Ideally, a scenario model can be built by instantiating and composing descriptions from the domain model. Developing a domain model requires more initial work, but it simplifies generating models for a range of scenarios. Furthermore, ad hoc aspects of models are more likely to be discovered if the same descriptions are re-used in a variety of settings.

So far, we have stated the commonplace view of modeling in qualitative physics. Qualitative process theory

[3] organizes domain models around *processes*, which can be automatically instantiated to form scenario models. Device-centered ontologies [1; 14] provide catalogs of devices, which can be composed to build scenario models. (Kuiper's QSIM [8] does not provide any abstraction or organizing structure for domain models itself, but one could imagine using it with either ontology.) Unfortunately, as we have attempted to build more realistic models, we have discovered that this view is inadequate.

This view breaks down in two ways for complex domain models. First, higher fidelity models are simply bigger than lower fidelity models. Representing fluids in detail, for instance, requires geometric information about the placement of portals, descriptions of head at every distinguishable place, models of fluid resistance in pipes, and so forth. We have built such models, (which turn out to be several times larger than than the models in [3]), and even on simple situations they swamp our machines.

Only part of the problem is technological. Even if our computers ran infinitely fast, for most purposes we simply don't need or want such detailed answers. When we do need the details, it is typically about a very narrow range of behaviors. This scaling problem becomes even more acute when faced with modeling the kind of propulsion plant studied in STEAMER [5], which used a numerical model that contained hundreds of parameters. The stock AI answer is "hierarchy", but how should it be done?

The second breakdown comes from the use of multiple perspectives. In some cases, a feed tank is best viewed as an infinite capacity liquid source. In other cases, it should be viewed as a container which may be emptied (perhaps with dire consequences). One cannot consistently use both views at once. One solution would be to create multiple, distinct models, one for each perspective and purpose. Doing so would significantly raise the difficulty of the selection problem, and make knowledge acquisition and maintenance nearly impossible. We must find ways for incompatible perspectives to peacefully coexist in a single domain model.

These issues have been addressed before in qualitative physics, albeit partially and informally. de Kleer and Brown, for instance, describe *class-wide assumptions*, which roughly correspond to our use of simplifying assumptions. However, this notion has never been formalized nor explicitly used as part of their programs or models [7]. So far, the device ontology in qualitative physics has inherited a limitation from System Dynamics [10] upon which it is based: the process of mapping from the "real-world" scenario to a device model lies outside the theory.

Qualitative Process theory was designed with such problems in mind. The descriptions of the domain model are automatically instantiated by a QP interpreter, thus – in theory – providing the means for modeling assumptions to be explicitly represented. This paper describes a set of conventions for exploiting this power.

3 A steam plant model

Since steam plants are not everyday systems, we begin with a brief description of steam propulsion plants, and the highlights of our model. Figure 1 shows an abstract view of propulsion plants adapted directly from Navy training

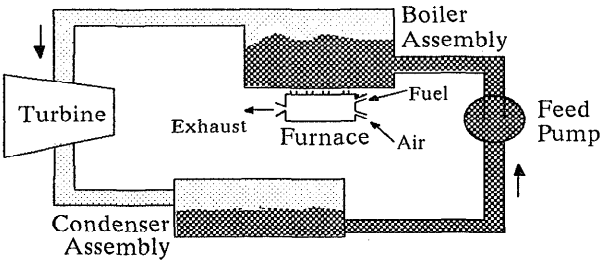

Figure 1: Simplified model of a navy steam-powered propulsion plant.

materials [9]. The primary components operate in the following fashion:

- *Boiler assembly.* The boiler assembly takes in distilled water and fuel and produces superheated steam. Most surface ships use several boilers, but this can be ignored. The heat is supplied in most ships by an oil-buring furnace. The steam produced by the boiler is fed through the *superheater*, which increases its temperature in order to get more work out of it.
- *Turbines.* The superheated steam then enters the turbines, which produce work (by driving the ship's propellers), resulting in the temperature, pressure, and kinetic energy of the steam dropping.
- *Condenser assembly.* The steam exhausts from the turbine to the condenser, where it is cooled by circulating sea water and condensed again into liquid.
- *Feed Pumps.* A series of pumps transport the condensate back to the boiler assembly, where the cycle begins again.

Our model captures the first few "high-level" models of the steam plant, with various perspectives. Some questions that can be answered with the model currently are illustrated in Figure 2. We have focused only on the main steam cycle, ignoring support systems such as lubrication and distillation. We only represent the highest levels of structural abstraction, throwing away all geometric information. Even so, we believe this is the largest qualitative model built to date. The domain model includes definitions of 8 object types, 23 views, and 14 processes. (Expanding these into horn clauses yields 1566 "axiom-equivalents".) During the partial instantiation computation on the plant model, 21 processes, 55 views, and 79 quantities are created. (This works out to 8617 instantiated horn clauses in the ATMS database.) A Symbolics machine has never lasted through a total envisionment of the full model. But using the techniques described in this paper, the envisionments typically take a few minutes.

4 Modeling Assumptions

Conceptually, we view setting up and using a scenario model as a process of filtering potential instantiations and behaviors. Ideally, the "raw input" takes the form of a true structural description, whose terms are physical objects such as pipes, tanks, sumps, butterfly valves, and so

Figure 2: Some questions the model can answer
Here are some answers generated by an implemented query system using the steam plant model. The questions were formulated in a specialized query language. The explanations are automatically generated by the program. The size of the subset of the model instantiated for the query is listed after each question.

Q: What affects the efficiency of the plant?
The efficiency of the plant is affected positively by the work rate of the turbine. It is also affected negatively by the energy input to the plant.
15 quantities, 41 inequalities, 3 process, 3 views, and 3 situations.

Q: What is causing black smoke to rise from the furnace?
Black smoke is rising from the furnace because it is not the case that the fuel/air ratio of the furnace is less than the F/A saturation point for the furnace.
24 quantities, 65 inequalities, 8 processes, 15 views, and 21 situations.

Q: How many mass flows are there?
There are 4 mass flows:
1. A flow of water from the condenser to the feed pump.
2. A flow of steam from the turbine to the condenser.
3. A flow of steam from the boiler to the turbine.
4. A flow of water from the feed pump to the boiler.
25 quantities, 83 inequalities, 7 processes, 20 views, and 15 situations.

on. The output is a scenario model, which is then analyzed.

The first filter in the modeling process is a mapping from the raw input into a set of *structural abstractions*, which capture the essentials of that system relevant to a particular analysis. For example, a collection of pipes and valves might be recast as an abstract fluid path, which may or may not be aligned.[1] If parts of the structure do not play a role in the behaviors of interest, then those parts may be thrown away. For example, in considering the thermodynamic properties of the main steam cycle in a propulsion plant, one ignores the multitude of drain valves and pipes, since they are only used during system startup and shutdown.

The next filter selects the relevant grain size and perspectives by specifying a set of *simplifying assumptions*. Answering a student's questions about the overall functioning of the plant, for instance, does not require instantiating a detailed model of lubrication flow. We take as our inspiration Sussman's *slices* notion [11], where results from multiple perspectives could be combined in synthesizing engineered systems. In Sussman's system the language for specifying perspectives was domain-dependent (i.e., electronic circuits), and instantiation decisions were made by hand. By contrast, our techniques should work for any phenomena expressible in QP theory, and we also address the problem of automatic perspective selection.

At this stage the model is ready for analysis. Often this analysis requires qualitative simulation, which itself can be tuned by imposing *operating assumptions* to filter out irrelevant behaviors. For teaching basic plant operation, for instance, the steady-state behavior is crucial, while the intermediate states between "cold iron" (i.e., completely off) and normal operation are irrelevant.

Here we describe how modeling assumptions can be organized for model and behavior selection. We assume structural abstractions as inputs, and ignore the problem of computing them from structural descriptions.

4.1 Simplifying assumptions

A common technique for managing complexity is to ignore irrelevant details. A prerequisite for ignoring details in setting up a model is some means to "turn off" certain aspects of it. Consequently, we require every description in the domain model to depend on explicit simplifying assumptions (except for those which are always to be instantiated). These take the form CONSIDER(⟨*specifier*⟩). The collection of CONSIDER assumptions form the groundwork of any particular analysis. For instance, in the steam plant model we provide the ability to selectively instantiate thermal properties with the following description:

```
(defView (Thermal-Physob ?physob)
   Individuals
      ((?physob :type Physob
                :conditions
                   (CONSIDER
                      (Thermal-Properties ?physob))))
   Relations
      ((Quantity (Temperature ?physob))
       (Quantity (Tboil ?physob))
       (Quantity (Tfreeze ?physob))
       (Greater-Than (A (Tboil ?physob))
                     (A (Tfreeze ?physob)))
       (not (Less-Than (A (Temperature ?physob))
                       zero))))
```

The thermal properties of an object will be instantiated exactly when this CONSIDER assumption is believed.

Representing simplifying assumptions imposes new responsibilities on the domain modeler. The model must be organized so that local decisions about relevance force a coherent subset of the model to be constructed. For instance, if thermal properties are considered in one part of a steam plant, they should also be considered in connected parts. Such coherence can be enforced by establishing logical dependencies between CONSIDER assumptions. For example, we divide our model into *operating blocks* and *functional blocks* to control granularity. An operating block corresponds to a system or subsystem which must be considered at a uniform level of detail. A functional block is like an operating block, but only has input-output behavior – its internal details are hidden at that resolution. If we are focusing on a particular level of a system, we want to treat its components as functional blocks. This is enforced by a rule in the model whose content is:

```
∀s∀c[System(s) ∧ Consider(Operating-Block(s))
                ∧ Has-Part(s,c)
     ⇒ CONSIDER(Functional-Block(c))]
```

Simplifying assumptions can also control perspectives. For example, in some circumstances it is appropriate to consider the thermal properties of all contained stuffs at a given level of detail. In our model this is expressed by the assertion CONSIDER(thermal-properties), whose consequence is:

[1] A fluid path is aligned exactly when all of its valves are open.

```
CONSIDER(thermal-properties)
   ⇒ ∀st[Contained-Stuff(st)
           ⇒ CONSIDER(thermal-properties(st))]
```

In other cases we want to focus on just particular substances inside certain containers. We say this by `CONSIDER(thermal-in(sub,can))`, where `sub` is a substance and `can` is a container. The implication of this assumption is

```
CONSIDER(thermal-in(sub,can))
  ⇒ ∀s[State(s)
      ⇒ Consider(thermal-properties(C-S(sub,s,can)))]
```

That is, if we are thinking about water in the boiler, we must consider both liquids and steam.

4.2 Operating assumptions

Engineers constantly use default assumptions about behavior to manage complexity. For example, when trying to figure out how a system containing a heat exchanger works, engineers tend to assume that the fluid in the hot leg is hotter than the fluid in the cold leg. If the system is operating as intended, making this assumption saves effort because the other two alternatives (i.e., the temperatures being equal or the cold leg temperature being higher than the hot leg temperature) need not be considered. If the system is not operating as intended then the engineer's predictions will be wrong and the analysis must be reperformed to consider the other alternatives.

Several kinds of operating assumptions are useful. The simplest are local restrictions over the space of possible behaviors. For instance, one might assume that the temperature in the boiler is higher than that of the condenser. More typically, collections of restrictions are gathered to describe *operating modes* of the system. The collection of assumptions about heat exchangers above can be collected into an individual view to form the "normal mode" of the device[2]. A steam plant has several operating modes, starting from "cold iron" and ending in "full steam", and each subsystem has modes as well. Forcing a system to be in a particular mode dramatically reduces the number of predicted behaviors.

Not all operating assumptions are organized into modes. In analyzing a new thermal system, for instance, one may first focus on its steady-state behaviors. Our model defines the concept of a system `s` being in steady state with respect to a given thermodynamic property `q` as follows:

```
Steady-State(s,q) ⇒
    ∀p∀st[Has-Part(s,p) ∧ Contained-Stuff(st)
          ∧ Container(st)=p ⇒ D[q(st)]=zero ]
```

Two important caveats must be remembered when using operating assumptions. First, they must respect the simplifying assumptions in force. For example, it is inconsistent to both force the boiler's temperature to be constant and to not consider the thermal properties of the boiler. The easiest way to ensure such consistency is to only include operating assumptions in descriptions which contain the appropriate simplifying assumptions as prerequisites. Second, care must be taken not to rule out possible behaviors which are actually important for the task. In the initial stages of a design, for instance, it may be useful to suppress fault models and concentrate on steady-state behavior, but it could be disastrous to continue making those assumptions in later stages. No modeling discipline can completely prevent such mistakes. The advantage of our conventions is that such assumptions are at least explicit in the analysis, rather than implicit (say, in the choice of one domain or scenario model over another).

[2]Discussions with engineers indicate that most process designers tend to have detailed models for only one or two operating modes of a system, hence normal mode makes sense in many cases. But for systems with many defined operational regions, the idea of a normal mode doesn't make much sense.

5 Organization of the model

Here we return to the steam plant model, and show how these ideas are used in its organization.

5.1 Granularity

The model has three distinct levels of granularity (see Figure 3), which we describe here.

Unheated closed thermodynamic cycle: The propulsion cycle is treated as a black box, with heat flowing in and work flowing out. This level is useful for describing global properties of the system, such as efficiency. This level is predicated on `CONSIDER(Operating-Block(Steam-Plant))`.

Contained stuffs: Working fluids are explicitly represented using the *contained stuffs* ontology [6]. At minimum the volumetric properties (e.g., amount and pressure) are represented, but thermal properties (e.g., temperatures, thermal mixing) can also be included, according to the chosen perspective. This description is locally predicated on statements like `CONSIDER(Operating-Block(boiler))` or globally established by `CONSIDER(Volumetric-Properties)`.

Boiler assembly: The boiler is the heart of the plant, so we include an additional level of detail about it. This level represents the furnace explicitly, including the effects of fuel/air ratio on heat production rate and efficiency. This level is predicated on `CONSIDER(Operating-Block(furnace))`. The furnace may be examined independently or in the context of the rest of the plant. When being examined independently, idealized sources and sinks are instantiated to provide an "exterior" for the system.

5.2 Perspective

Perspectives allow irrelevant parts of a model to be turned off. Not all perspectives are consistent with every level of granularity. In our model, the following perspectives are supplied:

Volumetric properties: As mentioned above, this perspective is mandatory with the contained-stuff level. A special process describes the volumetric effects of phase changes without invoking thermal properties.

Thermal properties: Heats, temperatures, and thermal effects of mixing are considered in this perspective. A thermal perspective may be introduced for any component or system `S` by asserting `CONSIDER[Thermal-in(water,S)]`.

Boiler fault models: In operating a plant it is important to keep the water level within a certain range. Too low, and the boiler can melt. Too high, and water

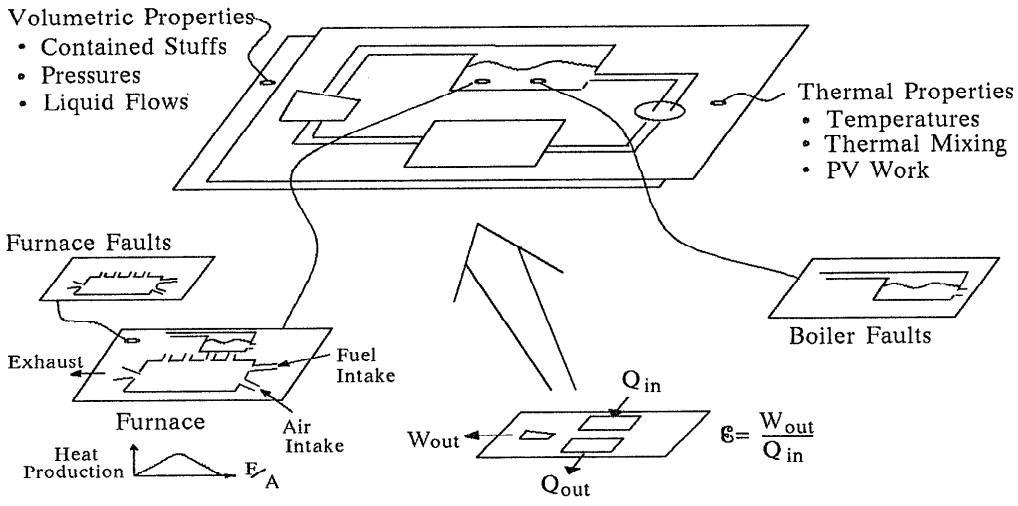

Figure 3: Differing views of the propulsion plant.

droplets are entrained into the superheater. Since steam is moving through the superheater faster than sound, these water droplets can cause tremendous damage. Asserting `Consider(Fault(fluid-level,boiler))` instantiates `level` as an explicit quantity, qualitatively proportional to amount, and installs these problems as a possibility.

Furnace fault models: The fuel/air ratio in the furnace is also tightly controlled. If the mixture is too rich, black smoke comes out of the stacks, which is bad if you want not to be seen. If the mixture is too lean, white smoke appears. In either case, fuel efficiency drops dramatically. These problems are instantiated as possibilities by `Consider(Fault(Exhaust-type,Furnace))`.

6 Model selection for question-answering

Our conventions for modeling assumptions provide a framework for organizing large-scale qualitative models, but this effort is useless unless we can effectively select just the right aspects of a model to use for a particular task. The general model selection problem is extremely hard, and depends critically on the nature of the task. Consequently, we focus on a particular task, namely answering questions in an instructional setting. We only address the problem of selecting appropriate simplifying assumptions; the problem of ascertaining the right operating assumptions is beyond the scope of this paper.

An intelligent tutoring system consists of several components, including a student model, dialogue manager, and domain expert [13]. Given a question, our task is to find a subset of the model that (a) suffices to answer the question and (b) minimizes extraneous details. A simple question about whether or not phase changes happen in the boiler should not be answered with a soliloquy on the possibility of black smoke rising from the stack. Furthermore, we would like to insulate the tutoring system from the internals of the model as much as possible. Ideally, we would like to create a set of question-answering routines that would work with any QP models. Such routines could form the core of a generic tutoring system which, given a QP model and appropriate nomenclature and display information, could produce reasonable explanations (in the manner of [2]).

We make only the plausible assumption that these routines can identify which parts of a query are descriptions which must be supplied by the qualitative model. These descriptions might be specifications of quantities, such as efficiency, or relationships, such as a liquid flow occurring.

Our algorithm assumes the qualitative simulator uses an ATMS. (It could be modified, at greatly reduced efficiency, to work with another kind of TMS.) In an ATMS, a fact can be asserted as true or false, with the usual meaning, except once asserted, such facts can never be retracted. A fact may also be *assumed*, which means it may or may not appear as part of some context (i.e., *environment*). Our qualitative simulator, QPE [4], exploits this distinction by not instantiating descriptions when their prerequisites (such as CONSIDER assumptions) are asserted false, since they could never hold in any consistent context.

Given the structural description for a particular scenario, and a list of query descriptions $\mathcal{Q}_\mathcal{L}$, we find the minimal appropriate set of simplifying assumptions as follows:

1. Expand the structural description using the domain model. This involves finding instances of process and view instances, as well as creating theoretical entities such as contained-stuffs.
2. Assume (not assert!) every possible CONSIDER statement.
3. Create a new node, QUERY, justified by the conjunction of the descriptions in $\mathcal{Q}_\mathcal{L}$.
4. Find the environment in the label for QUERY which

has the minimum number of CONSIDER assumptions. Return these CONSIDER assumptions as the result.

Envisionment can now proceed, beginning with the expansion process again, but with the minimal appropriate simplifying assumptions asserted as true, and any CONSIDER assumptions not believed as consequences of them asserted as false. The query system illustrated in Figure 2 used this algorithm to determine what aspects of the model to instantiate.

7 Discussion

The establishment of conventions for modeling assumptions is crucial for the organization and use of large-scale domain models. We introduced simplifying assumptions, in the form of CONSIDER statements, as a means of selecting grain size and perspective. We described how operating assumptions, such as steady-state, could be specified to filter possible behaviors. We have tested these techniques by building a multi-grain, multi-perspective model of a Navy propulsion plant which is significantly larger than any previous qualitative model. We further showed how a particular part of the model selection problem, finding a minimal appropriate set of simplifying assumptions, could be solved automatically by analyzing a partial instantiation of a model with respect to a particular question.

The issues we have addressed are relatively new, but we think we have made substantial progress on them. Much remains to be done, such as figuring out an automatic solution to selecting operating assumptions for an instructional context. We are currently extending our collection of generic query routines, with the long-range goal of providing a QP toolkit for building intelligent tutoring systems.

We are still a long way from building the kind of qualitative model we ultimately desire. We believe a qualitative model sufficient to support the full range of reasoning an intelligent tutor would need about the steam plant – the kind of model sought in the STEAMER project – would be about ten times larger than our current model. For example, there are at least three levels of detail below the finest grain of our current plant model which would be useful in intelligent tutoring systems. These new levels of detail will in turn require introducing new perspectives. To explain how a jet pump works, for instance, requires substantial geometric reasoning. No existing qualitative physics can handle the mixture of dynamics and geometry involved, and more research is needed to extend the range of phenomena we can cover.

The kind of analysis we have focused on here, explanation generation, is one of the simpler uses for a qualitative model. We suspect these ideas will prove useful for other types of analyses as well (viz Slices), but this remains to be explored. For many analyses, the mapping from structural description to structural abstraction is *the* crucial step; doing it incorrectly can prevent consideration of important phenomena (such as ignoring resonance phenomena in the design of structures). The discipline of explicit modeling assumptions must be extended to this part of the modeling process, so that we can build engineering problem solvers whose analyses are trustworthy.

8 Acknowledgements

John Collins provided valuable commentary and technical assistance. This research was supported by an IBM Graduate Fellowship, by the National Aeronautics and Space Administration, Contract No. NASA NAG-9137,, by the Office of Naval Research, Contract No. N00014-85-K-0225, and by an NSF Presidential Young Investigator Award.

References

[1] de Kleer, J. and Brown, J. "A qualitative physics based on confluences", *Artificial Intelligence*, **24**, 1984

[2] Forbus, K. and Stevens, A. "Using Qualitative Simulation to Generate Explanations" Proceedings of the Third Annual Conference of the Cognitive Science Society, August 1981.

[3] Forbus, K. "Qualitative process theory" *Artificial Intelligence*, **24**, 1984

[4] Forbus, K. "The Qualitative Process Engine", Technical Report No. UIUCDCS-R-86-1288, December, 1986. To appear, *International Journal of AI in Engineering*, 1988

[5] Hollan, J., Hutchins, E., and Weitzman, L., "STEAMER: An interactive inspectable simulation-based training system", AI Magazine, Summer, 1984.

[6] Hayes, P. "Naive Physics 1: Ontology for liquids" in Hobbs, R., Moore, R. (Eds.), *Formal Theories of the Commonsense World*, Ablex Publishing Corporation, Norwood, New Jersey, 1985.

[7] Iwasaki, I. and Simon, H. "Causality in device behavior", *Artificial Intelligence*, **29**, 1986.

[8] Kuipers, B. "Qualitative Simulation", *Artificial Intelligence*, **29**, September, 1986.

[9] U.S. Navy, *Principles of naval engineering*, NAVPERS 10788-B, Prepared by the Bureau of Naval Personnel, 1970.

[10] Shearer, J., Murphy, A. and Richardson, H. *Introduction to System Dynamics*, Addison-Wesley, 1971.

[11] Stallman, R.M., and Sussman, G.J. "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis", *Artificial Intelligence* **14** (1980) 1-39

[12] Stevens, A., Roberts, B., Stead, L. Forbus, K., Steinberg, C., Smith, B. "STEAMER: Advanced computer-aided instruction in propulsion engineering", BBN Technical report, July, 1981

[13] Wenger, Etienne, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Inc., 1987.

[14] Williams, B. "Qualitative analysis of MOS circuits", *Artificial Intelligence*, 24, 1984.