Molecular Collections: An ontology for reasoning about fluids

Draft ° Please do not redistribute.
Comments welcome.
Submitted for publication, Feb 1990

John W. Collins
Kenneth D. Forbus

Qualitative Reasoning Group
Beckman Institute
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, Illinois, 61801, USA

Abstract

An important problem in qualitative physics is reasoning about thermodynamic systems. To identify a system as a refrigerator, for example, one must show that it includes a closed thermodynamic cycle which transports heat from a cold region to a warmer one. To draw such conclusions requires thinking of fluids in terms of "pieces of stuff" whose movements can be tracked around the system. Capturing the ability to reason about pieces of stuff in this way is clearly essential for creating intelligent computer-aided engineering systems. This paper presents the *molecular collection* (MC) ontology, which captures many important intuitions about pieces of stuff. We claim that this ontology is *parasitic* on the contained-stuff ontology, in the sense that descriptions in terms of MC must be derived from contained-stuff descriptions. We describe the motivation underlying the molecular collection ontology and specify the laws that govern the behavior of MCs. We describe an implemented algorithm that uses these laws to generate MC envisionments. We show how the molecular collection ontology can be used to analyze several examples, including a refrigerator and a simple steam plant. Finally, we describe some open questions and planned future work.

1 Introduction

An important problem in qualitative physics is reasoning about thermodynamic systems. To identify a system as a refrigerator, for example, one must show that it includes a closed thermodynamic cycle which transports heat from a cold region to a warmer one. The

ability to detect a closed thermodynamic cycle requires tracing the mass flows around a system, showing that "the stuff" which was in one place travels through the system in such a way as to again be in that initial place. Simply tracing the connectivity of the system isn't sufficient, however. To establish heat transport, one must determine that "the stuff" is absorbing heat when it is in contact with the colder place and giving up heat when it is in contact with the warmer place. That is, one must reason about the properties of "the stuff" that is travelling through the system.

As Hayes [15] pointed out, one of the most tricky parts in reasoning about fluids is figuring out what "the stuff" is. What criteria should be used in individuating liquids? Hayes proposed two ontologies for reasoning about liquids: the *contained-stuff* ontology and the *piece-of-stuff* ontology. In the contained stuff ontology, one considers the liquid in a place to be an object. For example, a river is considered to be whatever water resides in its banks and bed. In this ontology, a river is the same river it was a century ago (shifts in where it flows notwithstanding). If the river is dammed up the part that was downstream vanishes, and if the dam breaks the river reappears. Alternately, the piece-of-stuff ontology defines an individual of fluid substance as a particular collection of molecules. In this ontology, one cannot step in the same river twice, since at any instant what you touch is a different collection of molecules. What comprises the river is constantly changing, as little pieces of water, each retaining their identity, pass by you on their flow to the sea.

Hayes noted that neither ontology alone suffices to explain common sense reasoning about liquids. Similarly, neither ontology alone is sufficient for intelligent computer-aided engineering. The contained stuff ontology provides the information needed to establish flows, and thus ascertain the global behavior of a system. However, problems like recognizing a system as a refrigerator require a different perspective. Under the contained-stuff ontology, there is no sense in which stuff moves from place to place—the fluid in a container is simply the fluid in the container, whose amount may be replenished or diminished by various processes, but without the mechanism of moving stuff we intuitively associate with such processes. In setting up quantitative analyses, one sees descriptions of elemental pieces of stuff over and over again in fluid and thermodynamics textbooks (e.g.[27,21]) To formalize these analyses, we need some form of the piece-of-stuff ontology.

This paper describes a specialization of the piece-of-stuff ontology, the *molecular collection* ontology, which supports the kinds of reasoning outlined above. Section 2 describes the basic idea of molecular collections, including how they are distinguished, what their properties are, and their relationship with contained-stuffs. Section 3 provides a precise formulation of the laws governing molecular collections. Section 4 describes an implemented algorithm for constructing envisionments of molecular collections, and Section 5 illustrates

this algorithm through the analysis of several examples. Finally, Section 6 describes some open questions and our plans for future extensions.

2 The nature of molecular collections

Almost every qualitative model of fluids uses the contained-stuff ontology (c.f. [8,2]). One noted exception is the work of Gambardella *et al.* [13], which uses a cellular automaton model to simulate properties of fluids and elastic objects. While the way their model derives its conclusions is interesting, (e.g., the surface of a liquid in a container being horizontal arises from the interaction of local rules associated with the "molecules" which comprise the object, all of which are explicitly represented.) this scheme does not seem to capture the notion of an "elemental piece of fluid" that engineering analyses use. Despite the obvious need for it, effective formulations of the piece-of-stuff view have been surprisingly difficult to achieve. Why?

We claim the relationship between the piece-of-stuff and contained-stuff ontologies is not that of two "separate but equal" schemes for representing the world. Instead, we claim that the piece of stuff ontology is *parasitic* on the contained-stuff ontology, in that a description of a system in terms of contained-stuffs is a prerequisite to computing a description of it in terms of pieces of stuff. The reason is that global information is required to establish the behavior of a piece-of-stuff. Consider a little piece-of-stuff, say 10^8 molecules, in the middle of a can of water. What is happening to it? Is it moving? Is it heating up? Without information about the surrounding fluid, we cannot say.

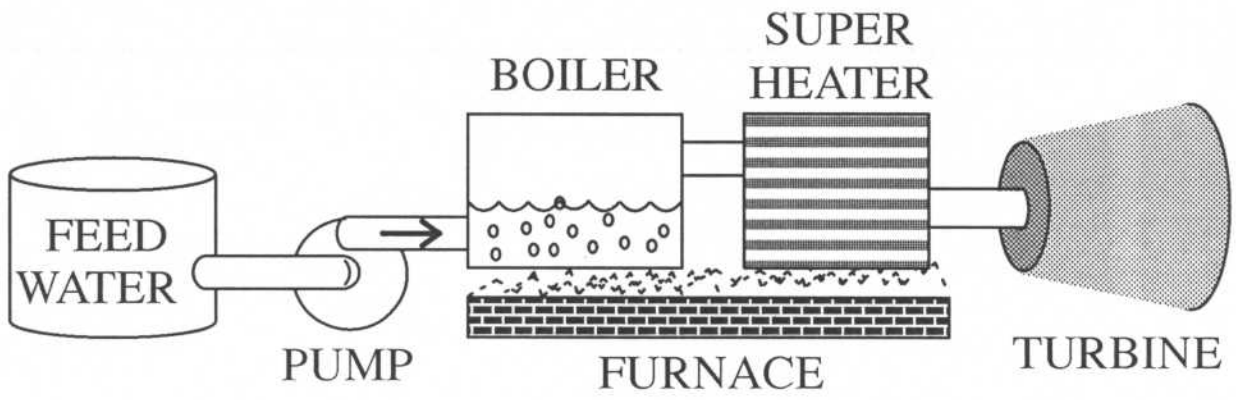
In classical physics the notion of gradient provides a local reflection of such global conditions. But establishing the gradient requires a global view of the physical system. This global view is exactly what the contained-stuff ontology provides. In particular, the contained-stuff representation supports establishing the paths and conditions needed to reason about flows and state changes. By starting with the results of the contained-stuff description, the piece-of-stuff representation does not need the ability to derive those conclusions.

Consider as a concrete example the system shown in Figure 1. This partial schematic of a Navy Propulsion plant provides an illustration of the importance of the MC ontology. A tricky question about this system is, "Given an increase in feedwater temperature, what happens to the steam temperature at the superheater outlet?".[1] The representation developed in this paper provides a basis for answering this question.

Figuring out how fluid moves requires knowing the mass properties of the fluid, viewed

[1] Understanding what happens in this situation is one of the hardest problems given at the U.S. Navy Surface Warfare Officers' School, in NewPort, R.I.

Figure 1: The SWOS Problem



with respect to the components of the system. Looking solely at a particular collection of molecules, there is no way to establish the pressure differences between system components that imply the direction of flow. Although it must play a role in the solution of the problem, the piece-of-stuff ontology is inadequate. To determine facts like flow direction, the contained-stuff ontology must be used. Given a contained-stuff description, we can talk about pressure as a function of location rather than trying to find the pressure on an arbitrary collection of molecules.

Now let us begin defining the molecular collection ontology. We assume as our starting point a contained-stuff representation of thermodynamic substances and processes, written in Qualitative Process theory [8,9]. As argued elsewhere [24], QP theory provides a useful basis for formalizing thermodynamics concepts. For example, thermodynamic processes map onto QP's representation of physical processes, with restrictions like *isothermal* and *adiabatic* reflected in restrictions on `Ds` values. Our formulation of molecular collections does not rely too heavily on the particular details of the representation of thermodynamic properties, and what assumptions we do make are explained in context.

In Hayes' work [15] no restriction is made as to the size of a piece-of-stuff. We define the *molecular collection* (MC) ontology by stipulating that the collection be so small that we can assume it is never distributed over more than one place. This tiny piece-of-stuff is viewed as a collection of molecules—as opposed to a single molecule—so that it possesses macroscopic properties such as temperature and pressure. Call the arbitrary collection of molecules to be considered as a unit `MC`.

Any ontology must divide the world into individuals. For reasoning it is important that

the number of individuals be few. The contained-stuff ontology partitions a fluid system into a few discrete objects using the natural boundaries provided by containment. But the contained-stuff ontology fails to preserve molecular identity. Considering individual molecules would be prohibitive and unnecessary, since all the billions of them in a particular region act more or less alike. By considering the possible behaviors of a representative collection of molecules, we constrain the possibilities for the whole by considering only one individual.

By assuming MC to be sufficiently small to travel as a unit, we drastically simplify its dynamics. In particular, if MC approaches a fork in the fluid path through which it is flowing, it takes one branch or the other, without splitting up. If MC were large enough to split up, we would then have to consider what happens when smaller MCs rejoined. And given that an MC could split into smaller MCs, why not again, until it was smeared around the volume of working fluid which comprises the system?

It is instructive to compare MCs with contained-stuffs further. We view contained-stuffs as having an amount, which is directly influenced by processes representing flows and phase changes. The amount of an MC is constant by definition. Conversely, the location of an MC can vary, while the location of a contained-stuff is defined to be constant. (We are only considering the fluid with respect to the coordinate frame of the physical system which contains it; the fact that the freon in an automotive air conditioner has a component of its velocity due to the motion of the car is irrelevant for our analyses.) Contained-stuffs can vanish or appear, as the amount of substance in the particular state which defines them changes. MCs cannot be created or destroyed (ignoring nuclear processes). Thus MC makes explicit the notions of continuity of space and conservation of matter, while the contained-stuff ontology makes explicit the notions of continuity at fixed locations, and determines overall system behavior.

In figuring out how MC behaves, the critical observation is that each active process specifies a fragment of MC's history. Processes operate on objects, some of which (in fluid systems) will be contained-stuffs. We can associate laws with each process to describe what, if anything, its activity implies about the location and phase of MC. For example, liquid flow implies that when MC is in liquid form in the source, it can move into the path of the flow, and end up in the destination of the flow without changing state. Evaporation implies that MC will undergo a liquid-to-gas phase transition within the same location. By combining these partial histories, we can compute the full spatial extent of MC's travels and its associated phase transitions (if any).

Table 1: Two Ontologies for Fluids

Property	Ontology	
	Contained-Stuff	Piece-of-Stuff
Defining Criteria	Containment	Molecular Constituents
Amount	Variable	Constant
Existence	Fleeting	Permanent
Location	Constant (w.r.t. container)	Variable
Emphasis	Determines Overall Process Activity	Conservation of Matter, Continuity of Space

3 The laws of molecular collections

Now we precisely define the properties of MC and describe the laws governing its behavior. These laws are used in Section 4 to generate MC envisionments.

We begin with the following simplifying assumptions. First, we only consider single-substance fluid systems. Thus MC is always made of the same substance as the fluid which surrounds it. Second, we assume that the places where fluids exist within a system can be adequately modeled by a combination of abstract containers connected by paths. That is, we ignore the detailed geometry of containers and fluid paths. Furthermore, when reasoning in the contained-stuff ontology, we do not treat fluid in a path as a contained-stuff, and do not give it any explicit existence. (This means we must model a heat exchanger as two containers in contact, rather than two fluid paths in contact.) What is required to remove these simplifying assumptions is considered in Section 6.

Intuitively, we know that MC is a small handful of molecules, moving along in its fluid environment, its properties changing continously according to its surroundings. How should we quantize this behavior to yield appropriate qualitative states? To simplify what follows, we use MC to both refer to the individual itself and to a slice, describing the individual at a particular time (see [15] for the definition of slices).

3.1 Preliminaries

We need the following properties of contained-stuffs:

Definition 1 (Substance definition) *The function* substance *maps from a contained-stuff or* MC *to the substance it is made of.*

Definition 2 (Phase definition) *The function* phase *maps from a contained-stuff or* MC *to its current phase. The only allowed values are* liquid *or* gas. phase *is undefined when its argument does not physically exist.*

Definition 3 (C-S definition) *The function* C-S *maps from a substance, state, and container to the contained-stuff made of that substance in that state in that container. The functions* substance *and* phase *are defined for a contained-stuff as follows:*

$$\texttt{substance}(\texttt{C-S}(sub, phase, can)) = sub$$
$$\texttt{phase}(\texttt{C-S}(sub, phase, can)) = phase$$

Only minimal assumptions about connectivity are required:

Definition 4 (Path connections) *The predicate* Path-Connects(p,c1,c2) *holds exactly when the path* p *connects containers* c1 *and* c2.

We also need to know when and where flows occur. We assume the following predicates to provide this information:

Definition 5 (Flow definition) *The predicate* Flow-Thru(p) *holds for path* p *exactly when there is fluid flowing through* p. *The predicate* Flow-Between(s,d,p) *holds exactly when there is a fluid flow from source* s *to destination* d *via path* p. *It follows that:*

$$\forall s, d, p[\texttt{Flow-Between}(s, d, p) \Rightarrow \texttt{Flow-Thru}(p)]$$

where p is a path connecting containers s and d.

We define two special kinds of paths, *pumps* and *compressors*:

Definition 6 (Active paths) *The predicate* Pump(p) *holds exactly when* p *is a pump. The predicate* Compressor(p) *holds exactly when* p *is a compressor. Furthermore,*

$$\forall p[\texttt{Pump}(p) \Rightarrow \texttt{Fluid-Path}(p)]$$
$$\forall p[\texttt{Compressor}(p) \Rightarrow \texttt{Fluid-Path}(p)]$$

Defining these components as paths means we cannot reason about the internal structure of pumps or compressors (e.g., what happens as MC swirls around the blades of a compressor, or when it enters or leaves the cylinder of a reciprocating pump). In the examples which have concerned us so far such reasoning is unimportant, and we believe in any case that most of the laws stated here can be easily modified if a richer underlying spatial description were used for describing structure.

Information about phase changes derived from the contained-stuff model is also needed to derive information about MCs. The following predicates provide this information:

Definition 7 (Phase change information) *The predicate* `Evaporating-Liquid(l)` *holds exactly when some process or processes representing evaporation or boiling are operating on contained-liquid* l. *The predicate* `Condensing-Gas(g)` *holds exactly when some process or processes representing condensation are operating on contained-gas* g.

Typically one would include `Evaporating-Liquid` and `Condensing-Gas` in the `relations` field of the appropriate process (or processes, if these phenomena is decomposed into several cases).

3.2 Location

Clearly one important property of MC is where it is. There are two aspects to location: the part of the physical structure which currently contains it, and the contained-liquid which surrounds it. Since we do not distinguish contained-stuffs in paths, it is useful to disentangle these notions.

Definition 8 (Location of MC) *The function* `location` *maps from* MC *to the container or fluid path in which it resides.*

Definition 9 (Fluid location of MC) *The function* `surrounding-stuff` *maps from* MC *to the contained-stuff which surrounds it, if any.*

When the location of MC is a container, its `surrounding-stuff` is defined as the contained-stuff in that container having the same state as MC:

Law 1 (Locations and stuffs)

```
Container(location(MC)) ⇒
    Exists(surrounding-stuff(MC))
    ∧ surrounding-stuff(MC) = C-S(substance(MC),phase(MC),location(MC))
```

3.3 Previous properties

If we view MC as a slice which specifies the properties of the molecular collection, then associated with it is some temporal extent. This temporal extent, as usual, could be an instant or an interval. We assume an Allen-style semantics for time [1], wherein temporal extents can meet with no intervening time. Thus it makes sense to talk about "the previous MC" for any given MC slice. We continue leaving time implicit for simplicity. However, we need to discuss changes in some properties from one slice to another:

Definition 10 (Previous properties) *The function* `previous-location` *maps from an* MC *slice to the location of the* MC *slice which meets it. Similarly, the function* `previous-phase` *maps from an* MC *slice to the phase of the directly-preceeding* MC.

The laws introduced in this section are used in Section 4 to compute MC envisionments. Computing envisionments requires thinking about states instead of slices, and all these laws can be so construed without any changes whatsoever. The only complication is that an envisionment state can have many equally valid predecessors, but since an episode in a history represents a single piece of actual space-time, there can only be (barring lack of knowledge) one previous meeting epsiode [10].

Our theory must make explicit which distinctions are to be made concerning MC states. If states were only distinguished by their present location and phase, then there would be exactly one MC state for each contained-stuff. We need a more fine-grained distinction in order to reason about the effects of a cold MC flowing into a hot contained-stuff, for example. We could additionally differentiate states based upon this temperature inequality; however, we take a different tack. We instead consider the previous-location and previous-phase of MC. Thus two MCs which enter a contained-stuff through different paths are distinguishable by their previous-locations. This introduces the possibility of multiple MC states having the same location and phase, but with a different past. These in turn could transition to a common MC state, which demonstrates that specifying MC's previous location and phase in no way guarantees a unique past. We postpone further discussion of this point until Section 3.7.

A number of laws constrain transitions from one episode to another. We introduce the notion of adjacency to rule out transitions between remote or disconnected parts of the fluid system:

Definition 11 (Adjacency) *Two locations are* Adjacent *exactly when it is possible for* MC *to move directly from one to the other.*

This is enforced by the constraint that the current and previous locations are adjacent:

Law 2 (Continuity of location)

$$\texttt{Adjacent(location(MC),previous-location(MC))}$$

A container is considered adjacent to itself, to allow for phase transitions, equilibration, and quiescence:

Law 3 (Container adjacency)

$$\forall c[\texttt{Container}(c) \Rightarrow \texttt{Adjacent}(c, c)$$

Since paths can only be connected to containers and not to each other, it is impossible for two paths (or two containers) to be adjacent;

Law 4 (Adjacency nogoods)

$$\forall p1, p2[\texttt{Path}(p1) \wedge \texttt{Path}(p2) \wedge [p1 \neq p2] \Rightarrow \neg\texttt{Adjacent}(p1, p2)]$$
$$\forall c1, c2[\texttt{Container}(c1) \wedge \texttt{Container}(c2) \wedge [c1 \neq c2] \Rightarrow \neg\texttt{Adjacent}(c1, c2)]$$

A path can only be adjacent to itself if there is no flow through it:

Law 5 (Path dynamics)

$$\forall p[\texttt{Path}(p) \wedge \texttt{Flow-Thru}(p) \Rightarrow \neg\texttt{Adjacent}(p, p)]$$

A container which is not connected to either end of a path is not adjacent to the path:

Law 6 (Path continuity)

$$\forall p, c1, c2, c3[\texttt{Path-Connects}(p, c1, c2) \wedge \texttt{Container}(c3) \wedge [c3 \notin \{c1, c2\}] \Rightarrow \neg\texttt{Adjacent}(p, c3)]$$

These laws, combined with flow laws introduced below, suffice to pin down MC's possible locations.

3.4 Continuous properties of MC

Like other kinds of physical objects, molecular collections have a variety of continuous properties. The values of these properties are included as constituents of an MC's state. These properties are described using the quantity space representation of QP theory [8]. Their intended semantics is that of the thermodynamical property with the same name, e.g., the temperature of MC is specified by the function temperature. The inequalities involving these quantities and their derivatives are determined by the surrounding fluids and by the processes which contain MC within their "sphere of influence." Most of the rest of the laws of molecular collections serve to define these interactions.

Definition 12 (Quantities of MC) *Molecular collections are assumed to possess the following quantities:* mass, heat, temperature, volume, height, *and* pressure.

Table 2: Quantities in the two Ontologies

Quantity Type	Ontology	
	Contained-Stuff	MC
mass (L & G)	Influenced	Constant
heat (L & G)	Influenced	$\propto_Q$ temp
temp (L & G)	heat/mass	Equilizes
pressure (L)	$\propto_Q$ depth	Inherited
height (L)	(level) $\propto_Q$ volume	Vertical Position
volume (L)	$\propto_Q$ mass	Constant
pressure (G)	heat/volume	Inherited
volume (G)	volume(Container)	heat/pressure

3.5 Interactions with the surrounding stuff

We assume the molecules which comprise MC remain a fixed collection over time. Thus the mass of MC cannot change; this enforces conservation of matter:

Law 7 (Conservation of Mass) *For all* MC *slices,*

$$\mathtt{Ds[mass(MC)]} = 0$$

While MC never loses or gains molecules from the stuff around it, the stuff does have strong direct and indirect effects. When MC is in a container, its pressure is determined by the surrounding stuff:

Law 8 (Pressure Inheritance)

$$\begin{aligned} &\mathtt{Container(location(MC))} \Rightarrow \\ &\quad \mathtt{Ds[pressure(MC)]} = \mathtt{Ds[pressure(surrounding\text{-}stuff(MC))]} \\ &\quad\ \wedge\mathtt{pressure(MC)} = \mathtt{pressure(surrounding\text{-}stuff(MC))} \end{aligned}$$

Additional constraints are provided by the active processes, such as liquid-flow and evaporation. The rules for each type of process are discussed in the sections below.

3.6 Fluid Flow Processes

When reasoning about contained-stuffs it is useful to distinguish between several varieties of fluid flow (see [2] for details). Most of these distinctions are irrelevant from the MC

perspective. The reason is that the differences between different varieties of fluid flow all affect the properties of the contained-stuffs they affect. Since most of MC's properties are derived from properties of its surrounding stuffs, these differences will be inherited correctly. As stated above, we assume the predicates Flow-Between and Flow-Thru are implied by any variety of liquid or gas flow process, either pumped or pressure-driven.

Roughly, what a flow implies for MC is this: When MC gets caught up in a flow, it changes locations first from the source container to the flow path and then into the destination container. A change in location means a change in episode, hence a flow gives rise to at least three MC episodes. (More than three episodes can result due to thermal mixing, as described below.)

We first define the upstream and downstream ends of a path during a flow process:

Definition 13 (Upstream/Downstream) *The function* upstream-end *of a fluid path* p *defines where the flow is coming from. The function* downstream-end *of* p *indicates where the flow is going. Both* upstream-end *and* downstream-end *are undefined for times when no flow occurs through* p.

$$\forall c1, c2, p[\texttt{Flow-Between}(c1, c2, p) \Rightarrow \texttt{upstream-end}(p) = c1]$$
$$\forall c1, c2, p[\texttt{Flow-Between}(c1, c2, p) \Rightarrow \texttt{downstream-end}(p) = c2]$$

We must force MC to change locations as a result of a flow; if MC is in a path where stuff is flowing, it must have come from the source container and must next flow into the destination container:

Law 9 (Flow movement)

$$\forall p \ [\texttt{Flow-Thru}(p) \Rightarrow$$
$$[\texttt{previous-location(MC)} = p \Rightarrow \texttt{location(MC)} = \texttt{downstream-end(p)}]$$
$$\wedge \ [\texttt{location(MC)} = p \Rightarrow \texttt{Flowing(MC)}$$
$$\wedge \ \texttt{previous-location(MC)} = \texttt{upstream-end}(p)]]$$

Some quantities can change within the path of an active flow process. For example, in a normal flow path (i.e., no pumps), pressure decreases as MC flows downstream. These facts have already been worked out by QPE using the contained-stuff ontology, and are stored as relations between quantities at the two ends of the flow. These differences represent the gradients which exist in the flow path, and are used to inherit the derivatives of MC's quantities as it flows through the path:

Law 10 (Gradient Implications)

$$\forall p, \forall q \in \{\texttt{Pressure}, \texttt{Height}\} : [\texttt{location(MC)} = p \wedge \texttt{Flow-Thru}(p) \Rightarrow$$
$$\texttt{Ds[q(MC)]} = \texttt{As[q(downstream-end}(p)) - \texttt{q(upstream-end(p))]]}$$

If the underlying model of structure is detailed enough to include portals, the quantities involved should be taken to be the corresponding quantities of the portals connecting the flow. See [2] for details.

3.7 Thermal Equilibration

There are two ways for MC to enter a new contained-stuff. First, it can arrive as a consequence of that stuff being the destination of a flow. Second, it can arrive as a consequence of changing its phase: When MC boils, its surrounding stuff changes from the contained liquid to the contained-gas of that container. (We require all phase changes to happen within containers.) In both cases, it is reasonable to assume that MC's pressure maintains equilibrium with its surroundings. However, the same assumption should not be made regarding temperature. A pressure gradient exists through a flow path, allowing MC to gradually transition between the two pressures at the ends of the path. But the temperature of MC upon exiting a flow path will not necessarily equal the temperature of its newly-surrounding fluid. For instance, a cold MC entering a hot contained-stuff will eventually warm up enough to reach thermal equilibrium with its surroundings. In fact, the effect of a temperature difference between MC and its surroundings is often of central importance; such is the case in the refrigeration model described in Section 5.2. Therefore we must explicitly represent the effects of MC reaching thermal equilibrium.

We represent this possibility by defining two new predicates, `Equilibrating` and `Equilibrated`, which hold exactly when MC is achieving thermal equilibrium or has achived it, respectively. These predicates are wholly determined by the constituents of MC's state—namely, its current and previous location and phase. An `Equilibrated` MC is one whose previous-location and previous-phase are the same as its corresponding present values. This gives us an MC to represent the "prototypical collection of molecules" in a contained-stuff. Any MC which has just entered a contained-stuff is `Equilibrating`. Hence an interval over which one is true rather than the other comprises a distinct episode in MC's history. In fact, MC will always be in exactly one of the three behaviors: `Flowing`, `Equilibrating`, `Equilibrated`.

Intuitively, what happens is this. Upon first entering a contained-stuff—as a result of either a fluid-flow or a phase transition—MC is labeled as *equilibrating*, during which time MC's temperature approaches that of its surroundings. After some interval of time, MC becomes *equilibrated*, such that its temperature simply follows that of its surroundings.[2]

The ability to determine whether MC is `Equilibrating` or `Equilibrated` was a primary

[2] In fact we will force these transitions by disallowing the direct escape of an equilibrating MC from its surrounding contained-stuff. This is explained in detail in Section 4.

reason for looking at the previous-location and previous-phase properties. However, we wish to know more than whether MC is Equilibrating; we want to know "from where" it is Equilibrating, since this will determine how MC reacts to its surrounding stuff. The previous values for location and phase provide this information.

We now define these two predicates. If MC has just arrived in a container, then it must be *equilibrating*:

Law 11 (Equilibrating on entry)

$$\texttt{Container(location(MC))} \wedge \texttt{location(MC)} \neq \texttt{previous-location(MC)} \Rightarrow \texttt{Equilibrating(MC)}$$

MC must also equilibrate after a phase change, since it enters a new contained-stuff, which might have a different temperature:

Law 12 (Equilibrating on phase change)

$$\texttt{phase(MC)} \neq \texttt{previous-phase(MC)} \Rightarrow \texttt{Equilibrating(MC)}$$

After residing within a contained-stuff for a sufficient period of time, MC eventually becomes *equilibrated*:

Law 13 (Eventual equilibration)

$$\texttt{phase(MC)} = \texttt{previous-phase(MC)} \wedge \texttt{location(MC)} = \texttt{previous-location(MC)} \Rightarrow \texttt{Equilibrated(MC)}$$

Of course, with the given resolution of our qualitative information, we have no way of knowing how long this takes. If MC's temperature is the same as that of its surrounding stuff, then we stipulate that the Equilibrating episode lasts for only an instant; otherwise, the episode lasts for an interval of time. Furthermore, since MC is very small, we stipulate that equilibration occurs long before MC could reach another path and leave the container. We now specify some consequences of equilibration:

Law 14 (Equilibrated temperature) MC*'s equilibrated temperature is inherited from the surrounding stuff:*

$$\texttt{Equilibrated(MC)} \Rightarrow \texttt{Ds[temperature(MC)]} = \texttt{Ds[temperature(surrounding-stuff(MC))]} \wedge \texttt{temperature(MC)} = \texttt{temperature(surrounding-stuff(MC))}$$

Note that MC must reach thermal equilibrium before it can leave its surrounding contained-stuff. This relates its temperature to temperatures in the contained-stuff ontology. In the case of liquid flow, MC's temperature is constant during its travels from one container to another. Thus by knowing the relative temperatures of the two contained-liquids, we can infer the relation between MC's temperature and that of its new surrounding-stuff. This in turn determines how MC's temperature changes while Equilibrating:

Law 15 (Temperature equalization) MC*'s equilibrating temperature gradually equalizes with the temperature of the surrounding stuff:*

$$\texttt{Equilibrating(MC)} \Rightarrow \texttt{Ds[temperature(MC)]} = \texttt{As[temperature(surrounding-stuff(MC))} - \texttt{temperature(MC)]}$$

3.8 Height equilibration

Changes in location can also mean changes in height. Rather than introducing a separate set of distinctions for this equilibration, we simplify the MC history by assuming that the height of a liquid MC reaches its equilibrium by the time that its temperature equilibrates. An MC's equilibrated height is inherited from the level of the surrounding contained-liquid:

Law 16 (Equilibrated height for liquid)

$$\begin{aligned}&\texttt{Equilibrated(MC)} \wedge [\texttt{phase(MC)} = \texttt{LIQUID}] \Rightarrow \\ &\quad \texttt{Ds[height(MC)]} = \texttt{Ds[level(surrounding-stuff(MC))]} \\ &\quad \wedge\ \texttt{height(MC)} = \texttt{level(surrounding-stuff(MC))}\end{aligned}$$

3.9 Gasses

In the contained-stuff ontology, a contained-gas inherits its volume from its container, while heat and mass are directly influenced by flow and other processes. These three quantities (volume, heat, and mass) are the independent variables, which together determine the remaining (dependent) quantities of temperature and pressure.

The quantities of a gaseous MC must be constrained diferently. In order to maintain our premise that MC never divides itself between two paths, it is necessary to adopt a partial volume view of gasses, instead of the more traditional view based on partial pressures. Thus a gaseous MC does not fill its container, but instead occupies a small (infinitesimal) volume determined by the local temperature and pressure. Because MC has constant mass, the two quantities of heat and temperature will always correlate except during a phase change. Thus the two quantities which are the dependent variables in the contained-stuff

ontology—namely temperature and pressure—become the independent variables for gasses in the MC ontology.

The MC version of the ideal gas law is as follows:

Law 17 (Ideal gas law) *The* volume *of a gaseous* MC *is influenced positively by its* heat *and negatively by its* pressure*:*

$$\begin{aligned}&\texttt{phase(MC)} = \texttt{GAS} \Rightarrow \\ &\quad (\texttt{Ds[heat(MC)]} = 0 \Rightarrow \texttt{Ds[volume(MC)]} = -\texttt{Ds[pressure(MC)]}) \\ &\quad \wedge(\texttt{Ds[pressure(MC)]} = 0 \Rightarrow \texttt{Ds[volume(MC)]} = \texttt{Ds[heat(MC)]}) \\ &\quad \wedge(\texttt{Ds[pressure(MC)]} = -\texttt{Ds[heat(MC)]} \Rightarrow \texttt{Ds[volume(MC)]} = \texttt{Ds[heat(MC)]})\end{aligned}$$

Adiabatic Expansion and Compression: Processes which do not involve a flow of heat are called *adiabatic*. When a gas is expanded or compressed adiabatically, it does work on (or is worked upon by) its surroundings. MC is assumed to expand or compress adiabatically during gas flow, as well as when its surrounding contained-stuff undergoes adiabatic expansion or compression:

Definition 14 (Adiabatic expansion and compression) *The predicate* Adiabatic-Expansion *holds exactly when a contained-gas undergoes adiabatic expansion. The predicate* Adiabatic-Compression *holds exactly when a contained-gas undergoes adiabatic compression.*

Whether or not a contained-gas is undergoing adiabatic expansion or compression can be ascertained by the processes which are acting upon it and how its continuous properties are changing. Specifically, if a contained-gas is the object of an expansion (compression) process, and is not the source or recipient of a heat-flow process, then the expansion (compression) is adiabatic. Thus we assume that information about the adiabatic expansion and compression of contained-stuffs is available for MC reasoning.

Law 18 (Adiabatic expansion) MC *undergoes adiabatic expansion when either:*

$$\begin{aligned}&\exists p[\texttt{Fluid-path}(p) \wedge \neg\ \texttt{Compressor}(p) \wedge \texttt{Flow-Thru}(p) \\ &\quad \wedge\ \texttt{location(MC)} = p \wedge\ \texttt{phase(MC)} = \texttt{GAS}] \\ &\vee\ [\texttt{Adiabatic-Expansion(surrounding-stuff(MC))} \wedge \texttt{Equilibrated(MC)}]\end{aligned}$$

Furthermore,

$$\texttt{Adiabatic-Expansion(MC)} \Rightarrow \texttt{Ds[temperature(MC)]} = -1$$

The law for adiabatic compression is similar, but doing work on the fluid requires a compressor:

Law 19 (Adiabatic compression) MC *undergoes adiabatic compression when either:*

$\exists p$ [Compressor(p) $\wedge$ Flow-Thru(p) $\wedge$ location(MC)= $p\wedge$ phase(MC) = GAS]
$\vee$ [Adiabatic-Compression(surrounding-stuff(MC)) $\wedge$ Equilibrated(MC)]

Furthermore,
Adiabatic-Compression(MC) $\Rightarrow$ [Ds[temperature(MC)] = 1]

3.10 Phase changes

Recall that phase changes are assumed to only occur within containers (Section 3.7). Since we are not considering the solid phase here, only two types of phase transitions are possible: evaporation and condensation. For the purposes of this representation, we are considering boiling as a special case of evaporation. A phase transition is represented in MC's description as differing values for current and previous phase; since phase transitions can only occur within containers, state and location should not be allowed to change simultaneously:

Law 20 (Location/phase change isolation)

[previous-phase(MC) $\neq$ phase(MC)] $\Rightarrow$ [previous-location(MC) = location(MC)]
[previous-location(MC) $\neq$ location(MC)] $\Rightarrow$ [previous-phase(MC) = phase(MC)]

Evaporation We begin with evaporation. Recall that the predicate Evaporating-Liquid indicates that a contained-liquid is undergoing evaporation or boiling (Section 3.1). Whether or not MC is evaporating is determined entirely by its surroundings:

Law 21 (Conditions for evaporation)
[Equilibrated(MC) $\wedge$ Evaporating-Liquid(surrounding-stuff(MC))
$\Leftrightarrow$ Evaporating(MC)]
$\wedge$ [$\neg$ Container(location(MC)) $\Rightarrow \neg$ Evaporating(MC)]

A non-evaporating piece of liquid has constant volume, and its heat and temperature are correlated:

Law 22 (Non-evaporating liquid)
$\neg$ Evaporating(MC) $\wedge$ phase(MC) = LIQUID $\Rightarrow$
Ds[volume(MC)] = 0
$\wedge$ Ds[heat(MC)] = Ds[temperature(MC)]

As part of the constraints to filter out inconsistent transitions, we must ensure that the previously-surrounding liquid is really evaporating:

Law 23 (Evaporation phase compatibility)

$$[\texttt{previous-phase(MC)} = \texttt{LIQUID}] \wedge [\texttt{phase(MC)} = \texttt{GAS}] \Rightarrow$$
$$\texttt{Evaporating-liquid(C-S(substance(MC),LIQUID,location(MC)))}$$

When MC is evaporating, its quantities are changing in certain ways:

Law 24 (Consequences of MC evaporation)

$$\texttt{Evaporating(MC)} \Rightarrow \texttt{Ds[temperature(MC)]} = 0 \wedge \texttt{Ds[heat(MC)]} = 1$$
$$\wedge\ \texttt{Ds[volume(MC)]} = 1 \wedge \texttt{Ds[height(MC)]} = 1$$

Condensation The laws of condensation are very similar to those of evaporation. Whether or not MC is condensing or not is completely determined by its surroundings:

Law 25 (Conditions for condensation)

$$[\texttt{Equilibrated(MC)} \wedge \texttt{Condensing-Gas(surrounding-stuff(MC))} \Leftrightarrow \texttt{Condensing(MC)}]$$
$$\wedge\ [\neg\ \texttt{Container(location(MC))} \Rightarrow \neg\ \texttt{Condensing(MC)}]$$

A non-condensing gaseous MC has correlated heat and temperature:

Law 26 (Non-condensing gas)

$$\neg\texttt{Condensing(MC)} \wedge \texttt{phase(MC)} = \texttt{GAS} \Rightarrow \texttt{Ds[heat(MC)]} = \texttt{Ds[temperature(MC)]}$$

As with evaporation, to ensure consistent transitions we must ensure that the previously-surrounding gas is really condensing:

Law 27 (Condensation phase compatibility)

$$\texttt{previous-phase(MC)} = \texttt{GAS} \wedge \texttt{phase(MC)} = \texttt{LIQUID} \Rightarrow$$
$$\texttt{Condensing-gas(substance(MC),GAS,location(MC))}$$

When a piece of gas condenses, it loses heat, shrinks and falls, while its temperature is assumed to remain constant:

Law 28 (Consequences of MC condensation)

$$\texttt{Condensing(MC)} \Rightarrow \texttt{Ds[temperature(MC)]} = 0 \wedge \texttt{Ds[heat(MC)]} = -1$$
$$\wedge\ \texttt{Ds[volume(MC)]} = -1 \wedge \texttt{Ds[height(MC)]} = -1$$

3.11 Summary of episode types

We have defined the following possible behaviors for MC: Equilibrating, Equilibrated, Flowing, Evaporating and Condensing. The first three episode types: Equilibrating, Equilibrated and Flowing form a taxonomy—MC will always be in exactly one of these episode types. Phase transitions (i.e., Evaporating and Condensing) can occur only during Equilibrated episodes.

Figure 2 shows a portion of a hypothetical scenario in which a container is connected to four active fluid paths—two are flowing liquid and two are flowing gas. The large arrows indicate the processes acting on the contained-stuffs. In addition to the flows, the liquid in the container is boiling. The small circles represent MC states, while the small arrows indicate the possible transitions which can occur among them. The table identifies the types of episodes occuring in each MC state. This figure is included to summarize the kinds of behavior in which MC can participate.

4 Reasoning about Molecular Collections

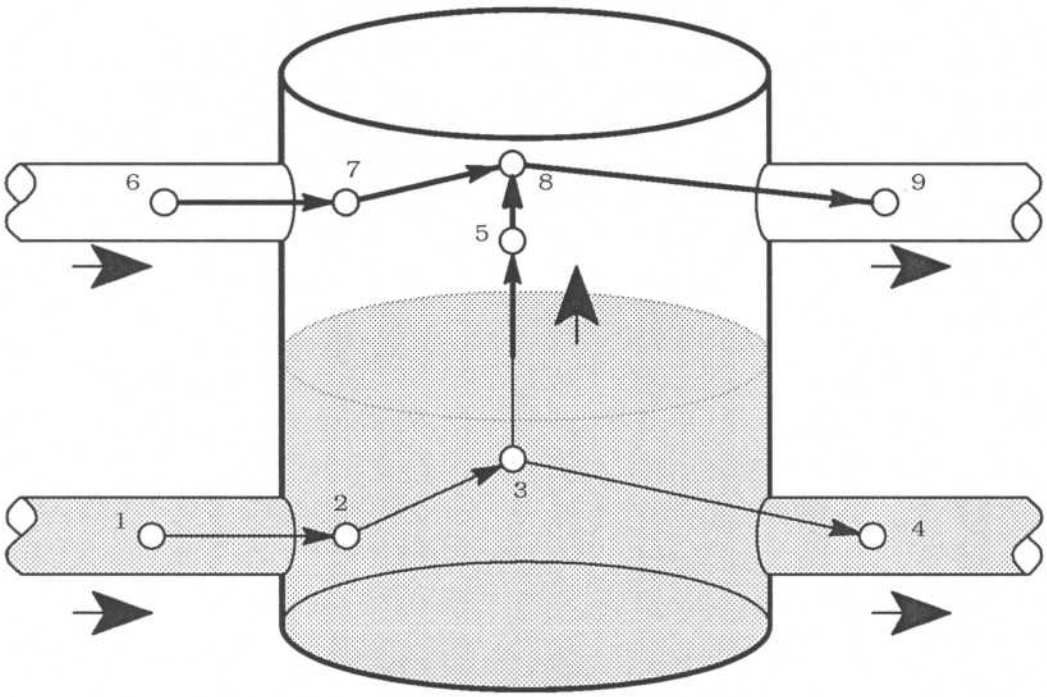
At this point we have introduced a number of definitions and laws which define what MCs are, and what their behavior is under various circumstances. This section explores how these laws can be used to envision the possible behaviors MC may experience in a fluid system, and what kinds of conclusions can be drawn from MC envisionments. We begin by motivating the choice of envisioning as the style of reasoning to use, then describe our algorithm, and finally discuss some conclusions that can be drawn by inspecting MC envisionments.

4.1 Why envision MCs?

The motivation for the MC ontology is to capture the sorts of intuitions associated with "following a little piece-of-stuff around" a system. To do this we must be able to figure out what states an MC might be in, and predict what can happen next. But the information given in real tasks is often fragmentary, which suggests that generating alternate behaviors is important.

To be concrete, suppose we were analyzing a new design for a refrigerator. We presumably would be given a drawing of the parts, to some level of detail. We would generally receive information about assumed operating conditions (e.g., the environment to which heat is being dumped might always be warmer than 40° F, and the demanded internal temperature was never lower than 32° F). Often our "input" would include some clues

Figure 2: A hypothetical scenario for MC



Episode Type	MC State S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Flowing	√			√		√			√
Evaporating			√						
Condensing									
Equilibrating		√			√		√		
Equilibrated			√					√	

about what processes are presumed to occur, perhaps noted on the diagram by arrows indicating directions of mass and heat flows.

One of the first things we might ask ourselves is whether or not this design was at all plausible. This question can be re-phrased as, "is there any possible behavior under which this system does indeed behave like a refrigerator?" Given a QP domain model which adequately captures our intuitions, one can quickly generate a constrained envisionment to see if the presumed pattern of activity is possible. That is, the structural description could be elaborated into a QP scenario model, and an envisionment generated under the *operating assumptions* [7] that the indicated flows are all active and the system is in steady-state. If the envisionment is empty, clearly the system cannot work as intended. If the envisionment is non-empty, then one must ask whether or not any of these states satisfies the behavioral requirements of refrigeration. As noted in Section 1, these conditions can be precisely characterized using the MC ontology. So for each state in the contained-stuff envisionment, one can generate predictions about MC's behavior, and answer yes if one of them satisfies the functional description of a refrigerator.

Given this (and similar) tasks, should we generate MC histories or envisionments? The recognition criteria only requires finding cycles of behavior. Since most engineering thermodynamic analyses focus on steady-state behaviors [16], the properties of MC will not change as a consequence of successive passes through the cycle. This suggests that the extra resolution gained by individuating states into episodes is not required. Furthermore, the lack of precise state information means that even at the qualitative level of resolution, there can be more than one contained-stuff state. Furthermore, each contained-stuff state could contain several cycles, if the system being analyzed consists of several disconnected fluid paths. This means we must check every MC state, since otherwise we may miss the relevant cycle. Taken together, these conclusions suggest using envisioning rather than history generation.

Before going on, two notes: First, for other tasks history generation will be better. Tracking how a system reaches steady-state, for example, or determining which steady-state a fluid system is approaching, would require the finer resolution of histories. Second, in [3] we mistakenly used the term "MC history" for what were actually "MC envisionments". We apologize for the error.

4.2 An MC envisioning algorithm

Roughly, a total envisioner works like this: The scenario is analyzed to determine all possible states, and then all legal transitions are found to connect these states into chains of possible behaviors. Since the molecular collection analysis is based on the contained

Figure 3: The MC envisioning algorithm

Given a domain model M_D using the contained-stuff ontology, scenario model M_S, and modeling assumptions A_M, the algorithm below computes the MC envisionment for a particular contained-stuff state S_{CS}. Notice that to analyze a different state, only steps 4 and 5 need to be repeated.

1. Generate the contained-stuff envisionment $\mathcal{E}$ of M_S using M_D and A_M.
2. Construct choice sets representing possible locations and phases of MC.
3. Use interpretation construction to generate all possible MC states.
4. Project the MC states onto some state S_{CS}selected from $\mathcal{E}$.
5. Construct possible transitions between pairs of MC states.

stuff analysis, things get a bit more complicated. Our algorithm is summarized in Figure 3. This section examines it step-by-step, pointing out the information requirements and analyzing its complexity. In what follows we assume an ATMS [4], although it could be re-formulated without one, perhaps at a substantial loss of efficiency.

4.2.1 Step 1: Generating the contained-stuff envisionment

The parasitic nature of the molecular collection ontology means that we must start with a contained-stuff description. We assume a domain model M_D has been defined which provides a suitable model of thermodynamic phenomena. In particular, we assume the predicates in Section 3.1 have been appropriately defined[3]. The structure of the system to be analyzed is specified by the scenario model M_S. We assume an envisioner, such as QPE [12], is used to apply M_D to M_S to find what individuals are implied by the structural description (such as instances of processes) and to generate possible contained-stuff behaviors.

Supplying task-specific modeling assumptions (A_M) can provide signficant constraint in this step [7]. Simplifying assumptions, which control perspective and level of detail, can prevent consideration of irrelevant properties. Ignoring geometry, for example, is often appropriate in the early stages of evaluating a design. Operating assumptions, such as the steady-state assumption, can reduce the size of an envisionment from hundreds or even

[3] In our implementation, this is accomplished by using an interface file that defines the necessary predicates in terms of the constructs of a particular domain model. Thus the MC envisioner can be applied to different domain models simply by changing interface files.

thousands of states to a mere handful. While it is hard to estimate the complexity of envisioning (c.f. [12]), this step can be quite rapid under the task constraints of interest.

4.2.2 Step 2: Generating choice sets

To generate MC states, we must first find the possible constituents of state. That is, we must figure out what locations are possible, what phases exist, and so forth. To fully exploit an ATMS, we use an implicit temporal notation. That is, the statements in our model simply refer to MC, as opposed to MC at some particular time. The following choice-sets are computed at this stage:

Locations Every container and fluid path in $\mathcal{E}$ are assumed to be possible locations of MC. The connectivity expressed in M_S is used to install adjacency information. Possible values for previous locations are also generated.

Phase If any contained-gasses exist, then Phase(MC) = GAS is assumed. If any contained-liquids exist, then Phase(MC) = LIQUID is assumed. Possible values for previous phases are also generated.

These four choice sets—current and previous values for location and phase—are combined in the next step to define the MC states. The constraints imposed by flow processes, phase transitions and adjacency are sufficient to prevent inconsistent combinations from being considered. Other aspects of MC's behavior—such as the type of episode it is in and the Ds values of its quantities—can be derived using the laws of Section 3. These laws are exploited computationally by transforming them into antecedent rules which either justify conclusions or install nogoods.

The complexity of this step is linear in the number of processes and phases in $\mathcal{E}$, because the rules to establish phases and episode types require matching only a single object. The laws for establishing adjacency and ruling out inconsistent combinations require pairs of antecedents, so they are quadratic in the total number of locations in M_S. Importantly, this step (and all the others save Step 1) are independent of the number of states in $\mathcal{E}$.

4.2.3 Step 3: Generating MC states

Once the choice sets have been established, a standard ATMS interpretation construction procedure is used to generate the set of possible MC states. Each state is represented by an ATMS *envirionment*, consisting of a consistent collection of assumptions from the choice sets above.

The general complexity of interpretation construction is at worst exponential, both in time and in the size of the result. The constraints of the molecular collection formulation yield much better performance than that, however. The time-complexity of interpretation construction depends on the details of an ATMS implementation, so we ignore them in favor of estimating the size of the result (e.g., the number of MC states). Each possible location can only have a small number of MC states. If the location is a path, then there are only three possible previous-locations, multiplied by two possible phases. If the location is a container, MC can either be equilibrated, undergoing a phase change (two more types of episodes), or equilibrating. MCs entering a container from different paths will have different equilibrating episodes, so if there are n paths into a container, there could be at most $3+n$ distinct MC states[4] for that container. Thus we have as an upper bound on the total number of states a small multiple of the number of distinct locations.

It should be emphasized that at this stage, these MC states are not specific to any single situation in $\mathcal{E}$; rather, they represent the union of all possible behaviors of MC for *every* contained-stuff situation. Finding what happens in a specific contained-stuff situation is the purpose of the next step.

4.2.4 Step 4: Projecting MC states onto a contained-stuff situation

The molecular collection ontology represents the "micro structure" of the behavior of contained-stuffs. Thus to carry out an MC analysis requires choosing some particular pattern of activity involving contained-stuffs. We restrict consideration to contained-stuff situations that last an interval of time, to respect the intuition that if the chosen situation only lasts for an instant, MC wouldn't have time to do anything.

Let S_{CS} be the situation selected from $\mathcal{E}$. To find the complete MC situations that can occur during S_{CS}, the environments corresponding to the MC states must be *unioned* with the environment corresponding to S_{CS}.[5]

Those unions which are successful (i.e., consistent) comprise the situations of the MC envisionment for S_{CS}. In addition to eliminating MC states which are not consistent with S_{CS}, this computation augments the consistent MC environments with the details from S_{CS} needed to completely determine MC's actual behavior (e.g., its inequality relations and derivatives for each MC situation).

The complexity of this step is linear on the number of MC states, which as we saw earlier is quite small. Obviously, the number of MC states surviving after this step can be no more

[4] One for each path flowing in to the container, plus one equilibrated for each of two phases, plus one which just crossed a phase boundary.

[5] Two environments are unioned by building a new environment from the union of the two sets of assumptions.

than the number which existed before it. In general, this step requires on the order of a few seconds of run time.

4.2.5 Step 5: Finding state transitions

The final step uses the `previous-location` and `previous-phase` information to connect consistent MC states together. First, each state environment is associated with some datastructure to hold information about transitions. Next, pairs of states are tested to see if one could be the previous state of the other. Given two MC states s_1 and s_2,[6] such that $s_1 \neq s_2$, s_1 can transition to s_2 exactly when:

$$\begin{aligned}&\texttt{phase(at(MC,}s_1\texttt{))} = \texttt{previous-phase(at(MC,}s_2\texttt{))}\\ &\quad\wedge\texttt{location(at(MC,}s_1\texttt{))} = \texttt{previous-location(at(MC,}s_2\texttt{))}\\ &\quad\wedge[\neg\texttt{Equilibrating(at(MC,}s_1\texttt{))} \vee \texttt{Equilibrated(at(MC,}s_2\texttt{))}]\end{aligned}$$

The first two constraints enforce the obvious requirements that the phase and location of MC in the first state agree with the corresponding previous values in the second state. The third constraint requires that if MC is equilibrating in the initial state, then it must be equilibrated in the end state. This enforces our requirement that MC must equilibrate before it leaves a contained-stuff. Note that it is impossible for MC to be equilibrated in both states, since that would make them indistinguishable—i.e., the same state.

By linking the MC states with all legal transitions, we have a complete MC envisionment. It should be noted that this envisionment can consist of several distinct components if the fluid system is disconnected.

The complexity of this step is no worse than quadratic in the number of MC states, since we could check each pair to see whether or not a transition can occur between them. However, this bound can be greatly improved by indexing the MC states by their locations. For each location we maintain two lists of MC states—one for MCs located there and one for MCs previously-located there. For each location, transitions are found by checking one list against the other. Only the phase and equilibration constraints need be verified, since the locations automatically match. Given n paths into a container, there are at most $n + 3$ MC states in each list. Thus the actual complexity of this step is quadratic in the average number of paths per container in the scenario.

[6] Although it is most convenient to avoid explicit temporal notations in actual implementations, we use them here for clarity.

4.3 Conclusions supported by the MC envisionment

Several important fluid and thermodynamic properties of a system can be ascertained by direct inspection of the MC envisionment. Phenomena such as global flow paths, branching, and cycles of flow are made explicit in the connectivity of the graph. In real fluid systems branching is very common. For example, steam coming out of a ship's boiler is often tapped off for several different purposes, such as driving the propulsion turbines, running generators to produce electricity, and powering the ship's laundry. The choice of which path to take will depend on the goal of the reasoning. Sometimes it is the properties of a specific path which are of interest (for instance, in checking whether or not the ship's laundry is receiving steam). In other cases all paths must be considered (for instance, in computing the enthalpy balance of a steam plant).

One of the most important classes of behavior are cycles. Identifying cycles is fundamental to recognizing function in thermodynamics. Combined with inequalities and derivative information, this information suffices to classify systems as refrigerators or heat pumps[24]. These properties are explored in more detail in the next section.

5 Examples

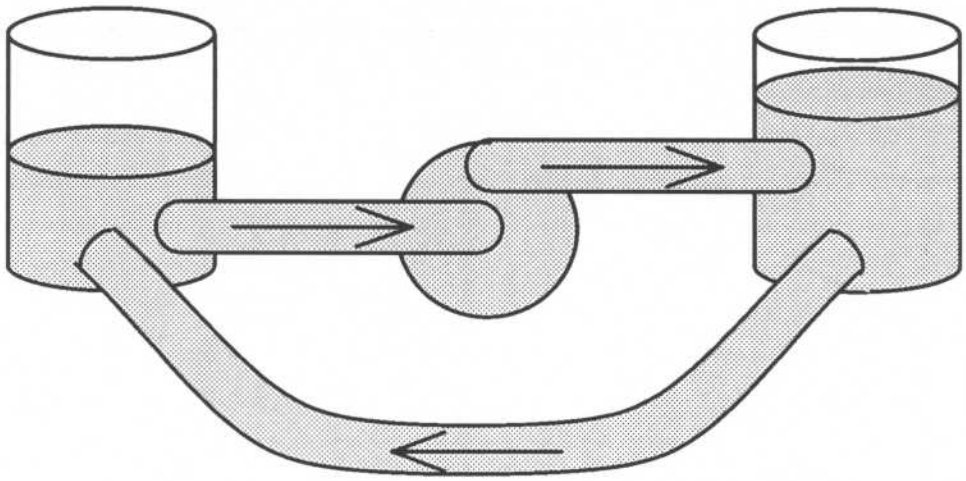
The MC–Envisioner has been tested on a number of examples of varying complexity. For concreteness this section describes several of them.

5.1 Pumped Flow

Figure 4 illustrates a scenario consisting of two open containers connected by a pump and a return fluid path. This example is worth considering because it includes cyclic flow which allows an equilibrium situation, where liquid exists in both containers and the flow rates have equalized. We choose this state as our base environment for examining MC.

Figure 5 shows the MC envisionment for the equilibrium situation. The MC envisionment is annotated with information about the type of process responsible for the movement, as well as the derivatives and state (phase) of MC at each place in the envisionment. This information becomes more useful for complex examples, such as in the refrigerator example below. Even though the situation is in steady state (i.e., all derivatives in the Contained-Stuff ontology are ZERO), the MC envisionment shows that each little piece-of-stuff in the system undergoes continuous change, both in position and in pressure. Note that even though MC is always in thermal equilibrium with its surroundings, it still must pass through an Equilibrating state upon entering a container. As stated earlier, Equilibrating in such situations only occurs for an instant.

Figure 4: A Simple Pumped-Flow Example



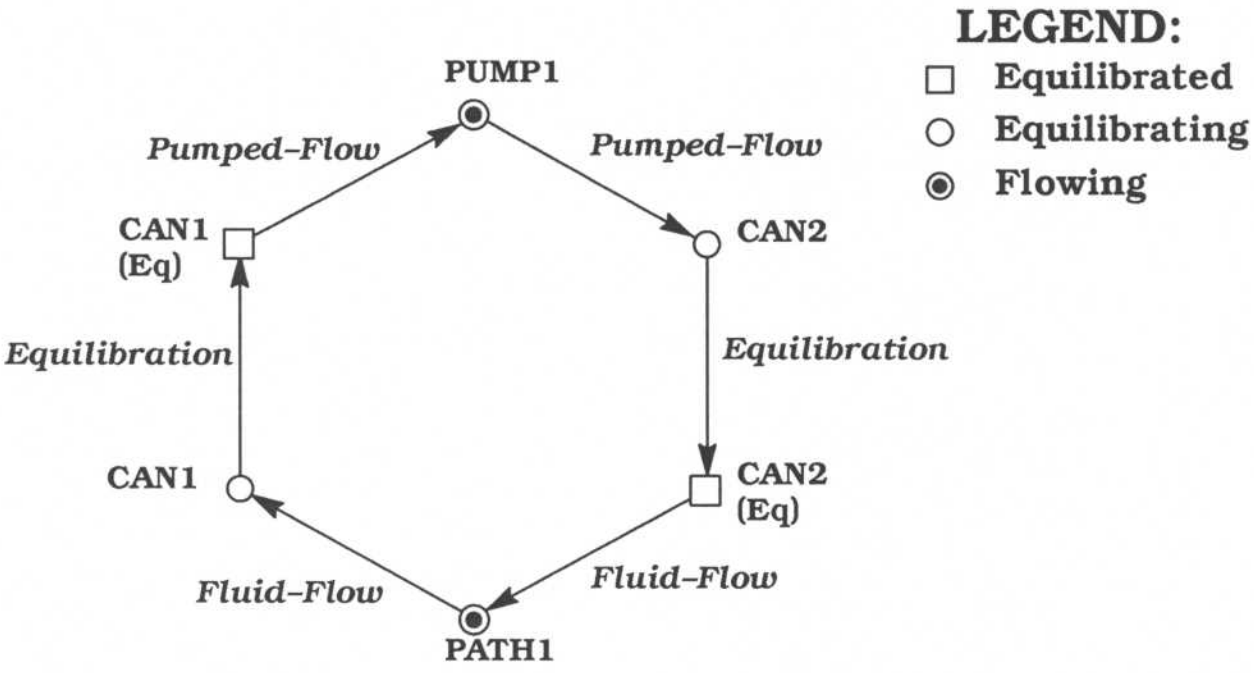
One conclusion which the MC ontology makes possible involves conservation of matter. Since there is no way for MC to enter or leave the system, one can conclude that the total amount of stuff in the system is constant.

5.2 A Refrigerator

One of the motivations for looking at the MC ontology was to allow reasoning about complex thermodynamic cycles such as that used in a refrigerator. Figure 6 shows a simple refrigerator involving six separate processes: two heat flows, two state changes (boiling and condensation), a compressor flow and a liquid flow. As in the pumped flow example, the situation selected for the MC envisionment is the steady state, where all flows have equalized. Unlike the pumped flow example, here we are matching flow rates with rates of boiling and condensation. The presence of phase transitions of both types puts MC through its hoops, in that every type of episode is encountered somewhere in the loop.

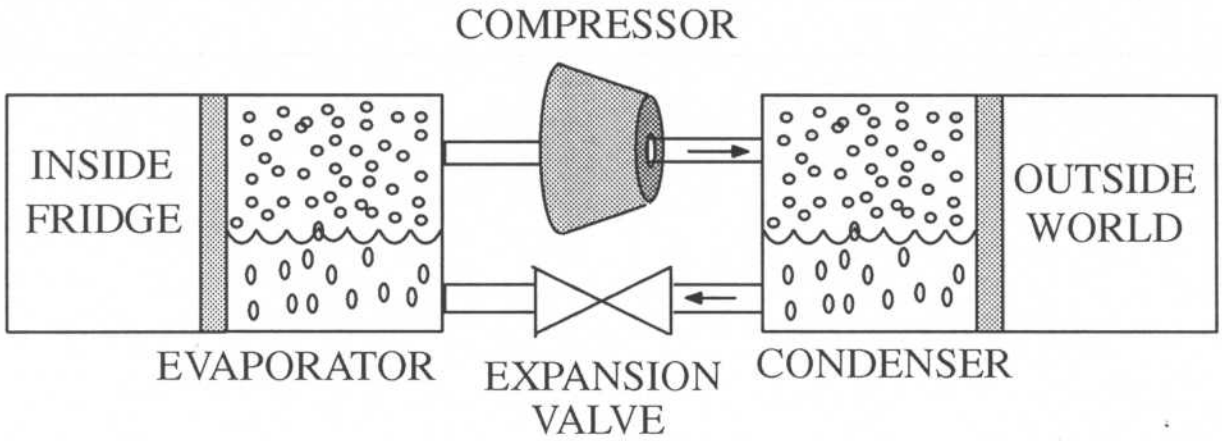
Figure 7 shows the MC envisionment. MC boils in the evaporator and then is pumped through the compressor to the condenser, where it returns to the liquid phase and is finally forced through the expansion valve back into the evaporator. This representation provides the foundation for an important class of engineering conclusions. Since MC gains heat during boiling and loses it during condensation, it must be moving more heat through the compresser than returns via the expansion valve, so there is a net heat flow from the

Figure 5: The MC Envisionment for the Pumped-Flow Example



Location	Can1	Pump	Can2	F-P
Ds[Heat]	0	0	0	0
Ds[Temperature]	0	0	0	0
Ds[Pressure]	0	1	0	-1
Ds[Volume]	0	0	0	0
Ds[Height]	0	0	0	0

Figure 6: A Refrigerator



evaporator to the condenser. Thus the refrigerator is pumping heat *uphill* to a higher temperature.

5.3 The SWOS Problem

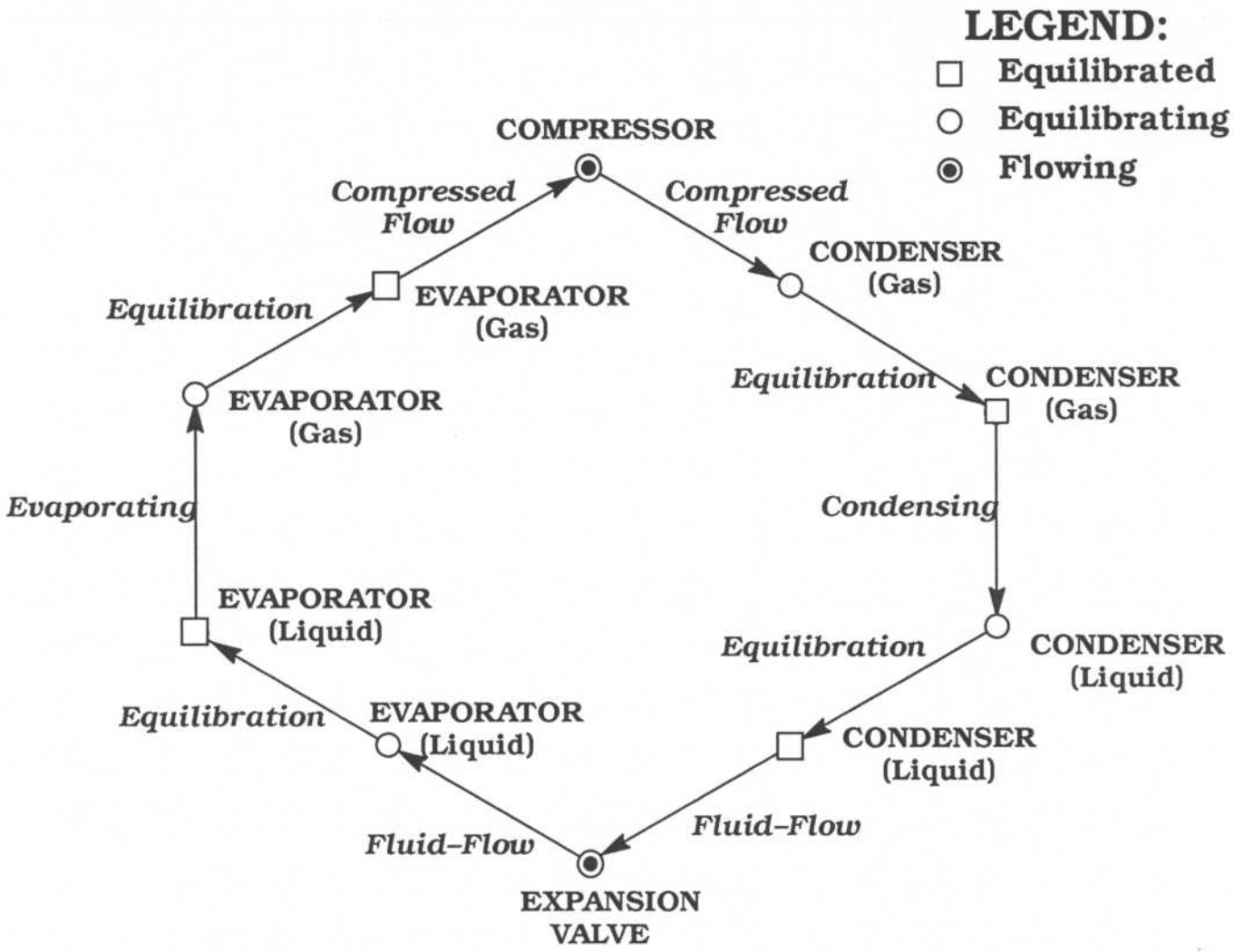
Here we return to the Navy propulsion plant scenario of Figure 1. Again we select the equilibrium situation from the contained-stuff envisionment. Figure 8 shows the MC envisionment. The result of an increased feedwater temperature can in principle be calculated by a differential qualitative analysis (DQ) based on this envisionment [9]. Roughly, the increased temperature means that the boiling episode is shorter, making the steam generation rate higher. The higher steam generation rate means the steam spends less time in the superheater, hence less heat will be transferred, implying a lower temperature at the superheater outlet. This argument could not even be stated in the cs ontology, since stuff does not move from place to place.

Weld [25] describes a set of DQ rules which, combined with this representation, may be powerful enough to draw this conclusion.

6 Discussion

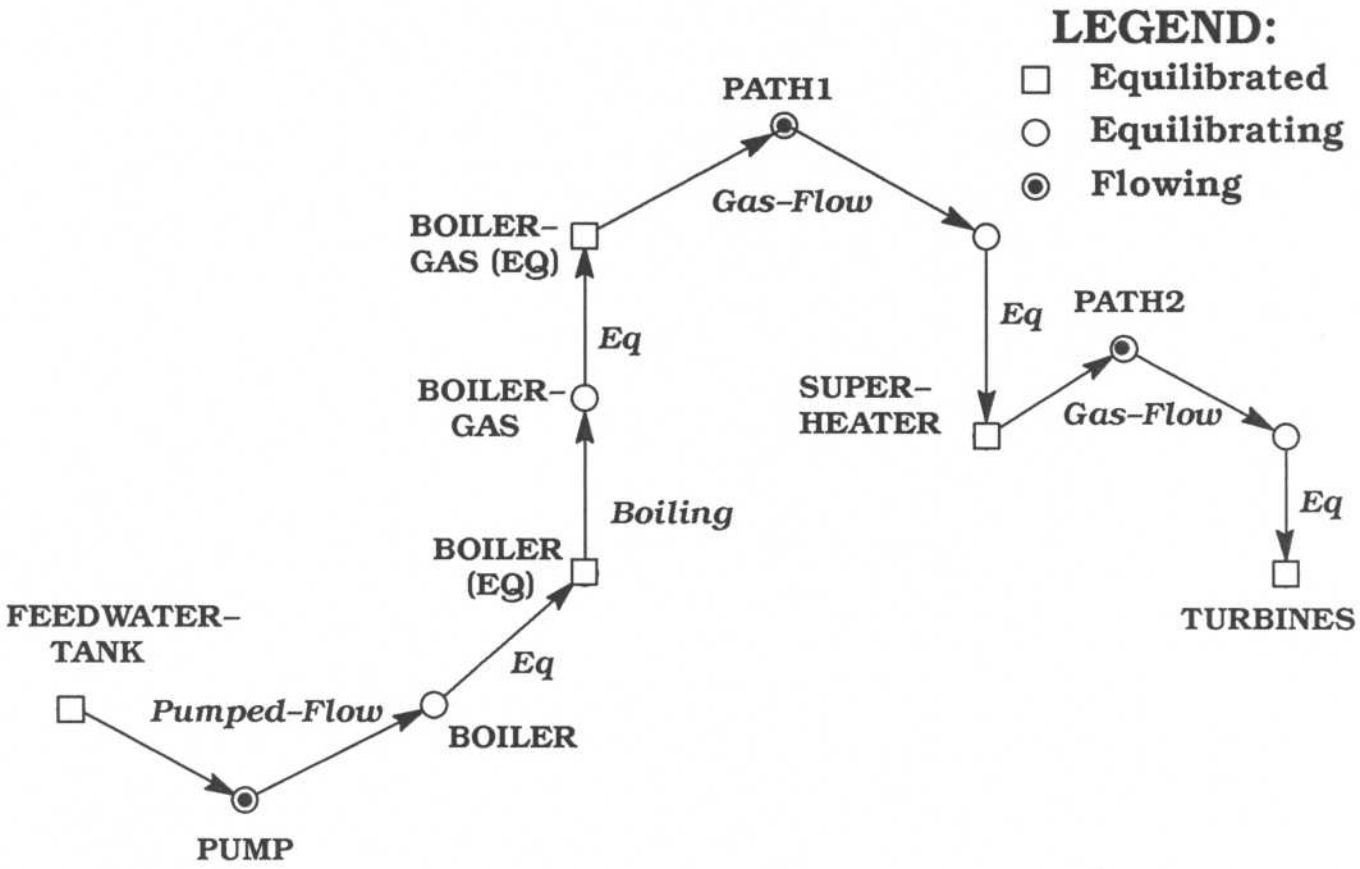
The ability to reason with multiple views of a situation provides significant advantages over using a single ontology. This paper formulates the molecular collection ontology, which complements the contained-stuff ontology most commonly used to model fluids in

Figure 7: The Refrigerator MC Envisionment



Location	Evap	Evap	Comp	Cond	Cond	EValve
State	Liquid	Gas	Gas	Gas	Liquid	Liquid
Ds[Heat]	1	0	1	-1	0	-1
Ds[Temperature]	-1	0	1	1	0	-1
Ds[Pressure]	0	0	1	0	0	-1
Ds[Volume]	1	0	-1	-1	0	0
Ds[Height]	1	1	0	-1	-1	0

Figure 8: The SWOS MC Envisionment



Location	Sea	Pump	Boiler	Boiler	Path1	S-H	Path2	Env
State	Liquid	Liquid	Liquid	Gas	Gas	Gas	Gas	Gas
Ds[Heat]	0	0	1	0	-1	1	-1	0
Ds[Temperature]	0	0	1	0	-1	1	-1	0
Ds[Pressure]	0	1	0	0	-1	0	-1	0
Ds[Volume]	0	0	1	0	1	1	1	0
Ds[Height]	0	0	1	1	0	0	0	0

qualitative physics. We showed that the molecular collection ontology is parasitic on the contained-stuff ontology, in that to reason about molecular collections requires an analysis in terms of contained stuffs. The contained-stuff ontology provides the conditions to determine which processes are active, and thereby determines the overall behavior of the system. The MC ontology provides the complementary ability to reason about where a piece-of-stuff came from and where it might go from here. We demonstrated that MC envisionments can be easily computed from QP models of fluids organized around contained-stuffs, and argued that this representation provides the basis for several important engineering inferences (i. e., closed-cycles, recognition of heat pumps and comparative analysis).

While we believe we have made significant progress, many avenues remain to be explored. We list some of them here.

Support for other forms of analysis: The conclusions reached using the MC ontology are crucial for supporting engineering analyses of thermodynamic systems [24]. However, we have only begun to explore how MC arguments can be used in reasoning. For example, carrying out comparative analyses [25], as Section 5.3 suggests, remains to be implemented.

Several important kinds of quantitative analyses would appear to be facilitated by the MC ontology. For example, many important system parameters, such as work output per pound of working fluid, are more naturally expressed using MCs than contained-stuffs. One could perhaps associate equations with each type of MC episode, and generate quantitative expressions for such parameters from the MC envisionment. These equations could then be analyzed to identify how these parameters could be optimized, or in general how a change in one quantity will affect the behavior of the system (c.f., [20]).

Analyzing transient behavior: Most questions that arise in quantitative engineering problems are about steady-state behavior, i.e., a single situation in the contained-stuff ontology. However, many textbooks (including [16]) feel obligated to describe what happens as a plant starts up and achieves steady-state, despite the lack of accurate mathematical models for such conditions and their irrelevance to the formal problems posed. Interestingly, their descriptions of such phenomena are invariably qualitative (albeit informal). Presumably, these descriptions are important for linking the steady-state picture of the plant's behavior to an engineer's intuition about fluids and heat. Growing an MC envisionment across transitions between situations in the contained-stuff ontology could provide the substrate necessary to reason about such situations, including explanation generation.

It is possible that interesting analyses could be made as the contained-stuff behavior hits a limit point. For instance, what happens to MC when the boiler in a steam plant ex-

plodes? Since the need for such analyses have not shown up in engineering thermodynamics problems we have examined, we have not addressed this issue thus far.)

Extended pieces of stuff It may be possible to generalize the MC ontology and envisioning algorithm to spatially extended pieces of stuff. This generalization would provide the ability to, for example, identify the spread of a contaminate through a fluid system. One way to do this might be to add dimensionality to containers and paths, so that the model could include submerged depth and length of travel along a path.

7 Acknowledgements

Brian Falkenhainer, Gordon Skorstad, and John Hogge provided valuable comments. This research was supported by the National Aeronautics and Space Administration, Contract No. NASA NAG 9137, by an IBM Faculty Development award, and by an NSF Presidential Young Investigator's award.

References

[1] Allen, J. "Towards a general model of action and time" *Artificial Intelligence*, 23(2), July 1984.

[2] Collins, J. and Forbus, K. "Building qualitative models of thermodynamic processes", *submitted for publication, 1990*

[3] Collins, J. and Forbus, K. "Reasoning about Fluids via Molecular Collections", in *Proceedings of the National Conference on Artificial Intelligence*, Seattle, July, 1987.

[4] de Kleer, J. "An Assumption-Based Truth Maintenance System", *Artificial Intelligence*, **28**, 1986.

[5] de Kleer, J. and Brown, J. "A Qualitative Physics based on Confluences", *Artificial Intelligence*, **24**, 1984.

[6] de Kleer, J. and Bobrow, D. "Qualitative Reasoning with Higher Order Derivatives", in *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August, 1984.

[7] Falkenhainer, B. and Forbus, K. "Setting up large-scale qualitative models", *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306, St. Paul, MN, August 1988. Morgan Kaufmann.

[8] Forbus, K. "Qualitative Process Theory" *Artificial Intelligence*, **24**, 1984.

[9] Forbus, K. "Qualitative Process Theory", MIT AI Lab Technical report No. 789, July, 1984.

[10] Forbus, K. "The Logic of Occurrence", Proceedings of IJCAI-87, August, 1987.

[11] Forbus, K. "The Problem of Existence", in Proceedings of the Cognitive Science Society, 1985.

[12] Forbus, K. "The Qualitative Process Engine" in *Readings in Qualitative Reasoning about Physical Systems*, Weld, D. and de Kleer, J. (Eds.), Morgan Kaufmann, 1990.

[13] Gambardella, L., Gardin, F., and Meltzer, B. "Analogical representation in modeling naive physics", *Proceedings of QPW-88*, Paris, 1988.

[14] Hayes, P. "The Naive Physics Manifesto", in *Expert systems in the Micro-Electronic Age*, D. Michie (Ed.), Edinburgh University Press, 1979.

[15] Hayes, P. "Naive Physics 1: Ontology for Liquids", in Hobbs, J. and Moore, B. (Eds.), *Formal Theories of the Commonsense World*, Ablex Publishing Corporation, 1985.

[16] Haywood, R. W. *Analysis of Engineering Cycles*, Pergamon Press, 1980.

[17] Iwasaki, Y., and Simon, H. "Causality in Device Behavior", *Artificial Intelligence*, **29**, 1986.

[18] Kuipers, B. "Common Sense Causality: Deriving Behavior from Structure", *Artificial Intelligence*, **24**, 1984.

[19] Kuipers, B. "Abstraction by Time-Scale in Qualitative Simulation", in *Proceedings of the National Conference on Artificial Intelligence*, Seattle, July, 1987.

[20] Mohammed, J., and Simmons, R. "Qualitative simulation of semiconductor fabrication", Proceedings of AAAI-86, August, 1986.

[21] Reynolds, W. and Perkins, H. (1977) *Engineering Thermodynamics.* McGraw-Hill Press, New York, New York.

[22] Shearer, J., Murphy, A., and Richardson, H. *Introduction to System Dynamics* Addison-Wesley Publishing Company, Reading, Massachusetts, 1967.

[23] Simmons, R. "Representing and reasoning about change in geologic interpretation", MIT Artificial Intelligence Lab TR-749, December, 1983.

[24] Skorstad, G. and Forbus, K. "Qualitative and quantitative reasoning about thermodynamics" *Proceedings of the eleventh annual conference of the Cognitive Science Society*, Ann Arbor, MI, August, 1989.

[25] Weld, D. "Comparative Analysis", Proceedings of IJCAI-87, August, 1987.

[26] Weld, D. "Switching Between Discrete and Continuous Process Models to Predict Genetic Activity", MIT Artificial Intelligence Lab TR-793, October, 1984.

[27] Welty, J. Wicks, C.E., and Wilson, R.E., *Fundamentals of Momentum, Heat, and Mass Transfer (second edition).*John Wiley & Sons, New York, NY, 1984.

[28] Williams, B. "Qualitative Analysis of MOS Circuits", *Artificial Intelligence*, **24**, 1984.

[29] Williams, B. "The Use of Continuity in a Qualitative Physics", in *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August, 1984.