# Introducing Actions into Qualitative Simulation

Kenneth D. Forbus
Qualitative Reasoning Group
Beckman Institute, University of Illinois
1304 W. Springfield Avenue, Urbana, Illinois, 61801, USA

## Abstract

Many potential uses of qualitative physics, such as robot planning and intelligent computer-aided engineering, require integrating pnysics with actions taken by agents. Here we show how to extend qualitative simulation to include the effects of actions, resulting in *action-augmented envisionments.* The action-augmented envisionment incorporates both the effects of an agent's actions and what may happen in the physical world whether or not an agent does something. Consequently, it provides a richer basis for planning and for reasoning about procedures than any previous representation. This paper defines action-augmented envisionments and presents an algorithm for directly computing tnem. The properties of the algorithm are analyzed along with its suitability for robot planning and reasoning about engineering procedures. We describe results generated by a working implementation and discuss potential extensions, including incremental algorithms.

## 1 Introduction

Plans involving the physical world must take into account the indirect consequences of actions. A robot making tea, for instance, must exploit physical processes such as liquid flow and boiling to carry out its plan. An intelligent CAD tool analyzing the safety of a power plant must be able to ascertain how the processes in the plant are affected by the actions of the plant's operators. Yet little has been done to integrate qualitative physics with actions. For example, [16] illustrates how some simple qualitative physics notions might be used in robot plans, out provides neither a domain-indepenent theory nor algorithms.

Moving qualitative physics into a planner is one approach. Hogge developed a *domain compiler* [10] that takes domain models expressed in Qualitative Process (QP) theory [5] and prod uces rules and operators for a temporal planner. Given a goal like "Increase the water level in this container", the planner can use its knowledge of actions, combined with physics knowledge derived from the domain model, to figure out that it should place the container under a faucet and turn on the tap, thus allowing liquid to flow.

So far, tractable compilation has required undesirable simplifications [1]. Additional run-time inference could in principle overcome this (and other) problems, but efficiency plummets[II]. For example, Hogge's planner could figure out how to get water into an empty pot and how to make water in a full pot boil, but without run-time transitivity rules it could not compose these plans to boil water starting with an empty pot. Adding those rules caused it to exceed resource limitations without finding a solution. While still promising, the difficulty of reconstructing the entire framework ol qualitative simulation into rules suitable for efficient planning makes exploring alternatives worthwhile.

Here we explore the dual approach: Moving actions into the physics. The next section defines a new representation, the *action-augmented envisionment* (or *A€),* which includes the effects of actions in the qualitative simulation. Section 3 shows how to compute *A€s,* and Section 4 examines the correctness and complexity of the algorithm, including its potential suitability for two tasks: robot planning and procedure generation in engineered systems. Section 5 describes an implementation and some examples. Finally, we describe our plans for future work.

## 2 Action-Augmented Envisionments

Informally, a *qualitative state* describes a class of particular behaviors for a physical system. Consider a pot of water sitting on a stove. If the stove is switched on, in one state the water might be heating up, and in another state it might be boiling. Qualitative states are linked by *transitions* which describe now these gross behaviors can change. The two states above, for instance, are linked by a transition corresponding to the water reaching its boiling temperature. Qualitative simulation consists of computing these states and transitions.

Every qualitative simulation leaves some "background information" unchanged. We do not, for example, consider what the world would look like if the stove suddenly vanished. In fact, qualitative simulations focus on just those changes predictable by the physics used.

The complete set of states and transitions for some fixed set of background assumptions is the *envisionment* for that scenario. (We use the conventions of [8] as needed to describe envisionments and their aspects.) To capture the effects of actions, we must allow at least

[1] For instance, the compiler assumes that if you influence a quantity in a particular direction it will actually change that way. Thus the planner could propose that one bail out a sinking ocean liner with a teaspoon.

**Figure 1: Actions can interact with processes**
**Heat flow requires a path capable of transmitting heat. These axioms for HEAT-ALIGNED indicate how changes in location indirectly affect whether or not heat flow occurs.**

```
Process Heat-Flow(?src,?dst,?path)
     Individuals:  ?src, Quantity(heat(?src))
                   ?dst, Quantity(heat(?dst))
                   ?path, Heat-Path(?path,?src,?dst)
     Preconditions:  Heat-Aligned(?path)
     Quantity Conditions:  A[T(?src)] > A[T(?dst)]
     Relations:  Quantity(flow-rate)
                 flow-rate = T(?src) - T(?dst)
     Influences:  I+(heat(?dst),A[flow-rate])
                  I-(heat(?src),A[flow-rate])

∀ ?c,?s Contained-Stuff(?c)
  ∧ Location(Container(?c))=On(?s)
    ⇒ Simple-Heat-Path(HPath(?s,?c),?s,?c)
∀ ?c Contained-Stuff(?c) ⇒ Heat-Aligned(?c)
Knob(Stove)=ON ⇒ Heat-Aligned(Stove)
∀ ?x,?y,?path Simple-Heat-Path(?path,?x,?y)
              ∧ Heat-Aligned(?x) ∧ Heat-Aligned(?y)
              ⇒ Heat-Aligned(?path)
```

some of the background assumptions to vary. For example, switching the stove on will initiate a heat flow, and taking the pot off the stove will break thermal contact and tnus end the heat flow. This extension requires the qualitative physics to be sensitive to changes in background assumptions. We use Qualitative Process theory because it provides two mechanisms for explicitly representing such dependencies. First, views and processes are quantified, in that they explicitly define the kinds of objects they apply to. In Figure 1, for example, heat flow can occur between two objects with thermal properties connected by a heat path. Second, explicit preconditions and consequences not involving dynamics are allowed (e.g., the stove's burner can be modeled as a heat path which operates only when Knob(Stove)=0K).

Let *V* be the set of background assumptions for a scenario, with $\mathcal{P}_f \subseteq \mathcal{P}$ the *fixed assumptions* of *V*. In previous qualitative physics systems $\mathcal{P} = \mathcal{P}_f$. Introducing actions means there will be a subset $P_m$, the *manipulable assumptions* of $P_f$ corresponding to those aspects which can be changed directly or indirectly by an action. Now we can begin to characterize the action-augmented envisionment *AS* for a scenario. Let $\mathcal{P}^\star_m$ be the set of consistent combinations of P*m*. If we denote the envisionment under a set of background assumptions P*i* as $\mathcal{E}(\mathcal{P}_i)$, then *States* $(\mathcal{AE}) = \bigcup_{p \in \mathcal{P}^\star_m}$ *States*$(\mathcal{E}(p \cup \mathcal{P}_f))$. The dynamical state transitions from each state $\mathcal{S}_i \in$ *States(AE)* are simply the union of all transitions from the component envisionments. To complete the definition of $\mathcal{AE}$ we must extend the set of transitions to include all occurrences of actions.

We define transitions due to actions by analogy with dynamical state transitions. In QP theory state transitions are represented as instances of *limit hypotheses* concerning changes in ordinal information. For instance, the hypothesis that the temperature of the water in the pot might reach its boiling temperature would be applicable to any situation where the water is being heated, regardless of the heat source involved. Similarly, we call an *action hypothesis* the conjecture that a particular action occurs. For example, the conjecture that the action Move-to(Potl,On(Stove)) is executed is an action hypothesis. This hypothesis is applicable to a variety of states, namely whenever Potl is not On(Stove). The rest of this section defines the constraints which determine valid action hypotheses and the transitions they cause.

We make two restrictions for simplicity. First, at most one action can be taken at a time *(single action assumption).* Since the vocabulary of operators could include compound operations, this assumption loses no generality. Second, actions do not coincide with dynamical state transitions *(separation assumption).* This assumption forbids actions occuring in instantaneous states, which are both unlikely and hard to manage. In essense, it requires that actions occur quickly relative to dynamical cnanges. This may restrict the expressibility of $\mathcal{AE}$s. Often this assumption is reasonable; for instance, a kettle's water doesn't cool appreciably while moving from a stove to a nearby teapot. When actions do take appreciable time, such as slowly opening a valve in a heating system, the action could be modeled as a sequence of short actions or reified as a continuous change in the physics triggered by an initial brief action.

Consider an action hypothesis *AH* as a partial function whose domain and range are $\mathcal{P}^\star_m$. Let $\mathcal{P}_{m_2} = \mathcal{A}_{\mathcal{H}}(\mathcal{P}_{m_1})$. Suppose $S_1$, $S_2$ are qualitative states such that $P_{m1}$ holds for *S*. The following restrictions must be satisfied for *S2* to be a possible outcome o $\mathcal{A}_{\mathcal{H}}$ curing at *S\:* (1) *Consistency:* $\mathcal{P}_{m_2}$ holds in *S2* **(2)** ■*Continuity:* When possible, no violations of continuity occur between *S1* and $S_2$. (3) *Closeness:* No state also considered to be a possible result of $\mathcal{A}_{\mathcal{H}}$ occuring at *S1* has more in common with *S1* than S2 does.

The consistency restriction is obvious. The continuity and closeness restrictions express the desiderata that only the direct and indirect consequences of an action are causally connected to it: nothing else should change as a result. Continuity expresses the constraint that, when possible, continuous parameters change smoothly. Unfortunately, continuity cannot always be satisfied. Suppose the pressure in a boiler is rising dangerously, and a safety valve is opened to bleed off excess steam. Intuitively, the discontinous action of popping the valve can result in the pressure instantly falling, a discontinuous change[2]. The definition of closeness depends on the details of the physics. The next section defines a measure of closeness for QP theory.

## 3 An algorithm for constructing $\mathcal{AE}$s

The formulation of *AEs* is based on sets of assumptions, so it is useful to describe the algorithm using the terminology of assumption-based truth maintenance [2]. The Qualitative Process Engine (QPE) [7] uses an ATMS to efficiently generate envisionments for QP theory, and we base our algorithm on it. States are defined by particular sets of assumptions $\mathcal{S}_A = \mathcal{Q}_s \cup \mathcal{P}_s$, where $\mathcal{Q}_s$ is drawn from the set of possible ordinal assumptions and $P_s$ is drawn from *V*. The set of possible ordinal assumptions and *V* are computed automatically during the instantiation of the QP model for the structural description of a scenario. $\mathcal{Q}_s$ initially consists of choices for the rela-

A more detailed model could capture the time it takes for the flow rate out to rise and thus preserve continuity, but always requiring such detail is inappropriate.

Figure 2: Operators for the kitchen domain

```
defOperator Move-To(?from,?thing,?to)
    Individuals:  ?from, Place(?from)
                  ?thing, Mobile(?thing)
                  ?to, Place(?thing)
                       ∧ ?from ≠ ?to
    Delete-List:  Location(?thing) = ?from
                  ¬ Clear(?from)
                  Clear(?to)
    Add-List:     Location(?thing) = ?to
                  ¬ Clear(?to)
                  Clear(?from)

defOperator Flip(?switch,?from,?to)
    Individuals:  ?switch, Switch(?from)
                  ?from, Has-Setting(?switch,?from)
                  ?to, Has-Setting(?switch,?to)
                       ∧ ?from ≠ ?to
    Delete-List:  ?switch = ?from
    Add-List:     ?switch = ?to
```

tionship for each pair of numbers mentioned in quantity conditions of view and process instances and other primitives of the QP language (e.g., correspondences and explicit-functions). As ambiguities and newly-discovered limit points are uncovered during simulation, $Q_s$ is augmented to include choices for them as well. P, consists of choices for the preconditions of view and process instances, as well as any choice sets explicitly defined by the domain model (e.g., declaring that any movable object has a Location which can range over the set of places in the scenario).

The temporal scoping of facts is implicit in their ATMS label, i.e., Location(Potl) = On(Stovel) holds in a state exactly when it is implied by the assumptions defining that state. This compact representation of states greatly simplifies algorithms. Actions are represented by STRIPS operators (e.g., Figure 2), since they easily satisfy the single action and separation assumptions. We require that all facts mentioned in the add and delete lists of operator instances be assumptions in

Given a domain model, which specifies the particular physical theory to be used, and a scenario, specified by $P_f$, QPE expands the scenario by applying the abstractions of the domain model. This creates instances of views, processes, and derived objects (such as "water in the pot"). It is easy to extend this process to include operator instantiation, and to automatically accumulate P*m*. Since QPE can search variations in *V*, as well as $Q_s$, *States*$(\mathcal{AE})$ is computed via the standard envisioning procedure. Furthermore, since we have the operator instances we can create the set of $\mathcal{A}_{\mathcal{H}}$'s. All that remains is (1) to ascertain when these $\mathcal{A}_{\mathcal{H}}$'s might occur and (2) to determine their effect in each case.

Call the assumptions corresponding to the delete list and add list of an operator instance $\mathcal{A}^d_{\mathcal{H}}$ and $\mathcal{A}^a_{\mathcal{H}}$, respectively. To determine if an operator instance Oi can apply to a situation *S1*,

1. Unless *Individual$(Oi)* are implied by S1, fail.
2. **Let** $\mathcal{P}'_s = (\mathcal{P}_{m_1} - \mathcal{A}^d_{\mathcal{H}}) + \mathcal{A}^a_{\mathcal{H}}$. **If** $\mathcal{P}'_s$ is inconsistent, **fail.**

To determine the effects of an action hypothesis we must determine if P's can be extended into a consistent situation $\mathcal{S}_2 \in States(\mathcal{AE})$[3]. Since we already have *States* $(\mathcal{AE})$, finding the results of Oi on *S1* can be viewed as a filtering problem. The set of predicted outcomes *C* can be computed by the following algorithm:

1. **Let** $\mathcal{C}_1 = \{\mathcal{S}_j \in States(\mathcal{AE}) \mid \mathcal{P}'_s \subseteq \mathcal{S}_j\}$
2. **Let** $\mathcal{C}_2$ **be the subset of** $\mathcal{C}_1$ **whose members do not violate continuity, when viewed with respect to** $\mathcal{S}_1$. **If** $\mathcal{C}_2$ **is empty,** $\mathcal{C}_2 \leftarrow \mathcal{C}_1$.
3. **Let** $\mathcal{C} = \{\mathcal{S}_j \in \mathcal{C}_2 \mid \neg\exists \mathcal{S}_k \in \mathcal{C}_2\ s.t.(\mid \mathcal{S}_k \cap \mathcal{S}_i \mid > \mid \mathcal{S}_j \cap \mathcal{S}_i \mid)\}$

The first step provides initial candidates by enforcing consistency, and the second uses the same continuity pruning used by limit analysis in QPE (see [7] for details). The final step provides a precise definition for closeness - literally, the number of assumptions shared with the previous situation. For each *Sj* Є C, the set *Transitions*$(\mathcal{AE})$ is extended to include a transition from *Si* to *Sj,* justified by Oi.

Typically an action results in a unique next state, but not always. One reason is the ambiguity of qualitative representations. Consider again the boiler with relief valve. Once the relief valve is blown there will be ambiguous influences on the pressure - the flow out through the relief valve will act to decrease it, while the generation of more steam will act to increase it. Consequently, the pressure could continue to increase, decrease, or remain constant. Unless extra knowledge can disambiguate them, each is a legitimate potential consequence of that action. Other consequences of an action can be underspecified as well. Consider throwing a pair of dice onto a table. We know that each die will stop with a particular face up, but we cannot predict in advance which face it will be.

## 4 Analysis

Our analysis addresses four questions: (1) Is the algorithm reasonable? (2) What is the complexity of explicitly generating $\mathcal{AE}$? (3) Under what circumstances would explicit generation make sense? (4) Could *AS* be generated incrementally?

### 4.1 Is the algorithm reasonable?

Proving correctness assumes the existence of a formal standard, and there isn't one for this task. Instead, we show that the algorithm is consistent with two intuitions about causality and action: (1) no extraneous changes occur, and (2) only the minimal necessary changes are predicted.

An extraneous change involves change in some part of the situation that cannot be affected by the action. For instance, if I start my car, then in the absence of any mechanism to the contrary, the stove in my kitchen remains off. In QP theory this lack of connection is expressed through *p-components:* Two individuals in distinct p-components cannot affect each other[4]. Suppose an action affects only individuals in p-components $\phi_1, \ldots, \phi_n$ **and not** $\phi_i$. Now consider the subset of $\mathcal{P}^{\star}_m$

[3]Indirect consequences of actions generally cause $S_2$ — $\mathcal{P}_{m_2} \neq \mathcal{S}_1 - \mathcal{P}_{m_1}$.

[4]P-components and how they relate to the frame problem are detailed in [5].

relevant to $\phi_i$. Any combination from that subset is consistent with the change. Because the p-components are distinct, combinations relevant to $\phi_i$ cannot interact with the subsets relevant to $\phi_1, \ldots, \phi_n$. However, any result state where $\phi_i$ changes will have fewer assumptions in common with the previous state than a result state where $\phi_i$ does not change, and hence will be filtered out by the closeness criterion.

Defining the minimal consequences of change is difficult, as the literature on counterfactuals indicates. There are two natural definitions for this representation: maximal shared *assumptions* and maximal shared *consequences.* Since the set of consequences is generally larger than the set of assumptions, it might be tempting to use it as the basis of closeness instead. However, doing so would violate our intuitions of causality. Consider a large pan of water that rests on two burners, one on and one off. Now turn off the operating burner. Maximizing shared assumptions leads to the conclusion that no heat is flowing from the stove, and the water will begin to cool. Maximizing shared consequences can lead to the conclusion that the other burner becomes on, exchanging one heat flow for another but keeping all the properties of the water intact. Yet this interpretation violates our notion of agency, where the cooling water does not. Any particular choice of actions encodes what we think are important, primary facts about the world. In causal terms, facts derived from those changed directly by an action have a secondary status. To satisfy this intuition we must maximize shared assumptions and let the consequences fall where they may.

## 4.2 Complexity

Here we summarize the complexity analysis in [9]. Since the standard envisioning process can compute *States* $(\mathcal{AE})$, we ask (1) how costly is the additional step of generating $\mathcal{A}_{\mathcal{H}}$s and their associated transitions and (2) how costly is it to generate *States*$(\mathcal{AE})$ relative to a standard envisionment € (i.e., where $P_m$ is empty)? If $|$ *States*$(\mathcal{AE})$ $| = n$, then determining the effects of actions is $O(n^2)$ . Unfortunately, n can be exponentially larger than *States*$(\mathcal{E}(\mathcal{P}_f))$; for example, if $P_m$ consists of pairs of independent propositions *p* and $\neg p$, the number of states could increase by $2^{|\mathcal{P}_m|-1}$. In other words, the temporal inheritance algorithm itself is cheap, but generating the states in the first place is expensive.

## 4.3 When would explicit generation make sense?

Explicitly generating a problem space is generally foolhardy and typically impossible, but that is just what envisioning does. The degree of interaction between the operator vocabulary and the rest of the domain model determines how close worst-case performance is approached. If the operators are irrelevant to the domain model, then $|\mathit{States}(\mathcal{AE})| = |\mathcal{P}^{\star}_m| \times |$ *States*$(\mathcal{E})$ $|$. But generally interactions exist and thus only a small subset of the cross product may be consistent. A domain with complex dynamical behavior and only a few actions, all tightly coupled, would be the best case.

Reasoning about procedures for engineered systems appears to be just such a task. Consider a power plant (either stationary or onboard a ship). Its dynamical state can be complex, and a badly-timed action can result in disaster. But the kinds of actions an operator can take are often limited to flipping switches and opening or closing valves. Since an $\mathcal{AE}$ compactly represents the result of executing all possible plans, it could be useful in generating operating procedures and safety analyses. To deal with realistic systems will require the same decomposition strategies as traditional engineering. For instance, operating procedures are often generated by combining procedures for subsystems (i.e., a step in turning on a circulation system is aligning one leg of it, which itself is a procedure involving several steps). This suggests computing $\mathcal{AE}$s for subsystems independently and combining them, using the techniques of [4] to represent system boundaries.

## 4.4 Incremental generation

Typical "robot planning" domains are the worst kind of task for explicit $\mathcal{AE}$ generation, since $\mathcal{P}^{\star}_m$ includes each possible location for every moving object, and thus could be huge. Incremental algorithms would be better, and appear both possible and feasible. The $\mathcal{AE}$ is just a problem space, wnose operators are the actions wnich can be taken plus the set of limit hypotheses. Incremental temporal inheritance algorithms for QP theory exist [6], and could easily be extended to $\mathcal{A}_{\mathcal{H}}$s. The entire panoply of AI search strategies could then potentially be used to generate plans. However, the fact that some transitions will occur whether or not the agent desires them changes the nature of the problem somewhat. One way to view planning in the $\mathcal{AE}$ problem space is as playing a game with Nature. The agent controls actions, ana Nature controls dynamics. This view is not exact, of course, since Nature is not generally held to have goals[5] and the "players" in this game don't take strict turns. Nevertheless, the metaphor is suggestive. For example, there is a "horizon effect" in this problem space which consists of dynamical chances undoing a state achieved by the agent. To assure tnat the intended effect of the action is maintained, qualitative simulation can be used to see if either (a) no relevant dynamical transitions occur or (b) they take sufficiently long that the next action can be performed before they occur.

## 5 An implementation

We have implemented the algorithm described in Section 3, and have successfully tested it on over a half dozen examples at this writing. They include several engineering fluid systems, a kitchen scenario[6], and throwing dice. This section highlights some results from these examples.

The kitchen scenario indicates that the worst-case analysis need not be relevant to problems of interest. The system generated 244 situations, with 1054 transitions between them (78 due to dynamics, the rest due to actions). For this problem, $|\mathit{States}(\mathcal{E}(\mathcal{P}_f))|$ = 25, and P*m* includes 12 binary choice sets and one three-way choice (i.e., the location of the pot). Thus the worst case analysis predicts $|\mathit{States}(\mathcal{AE})| = 25 \times 2^{12} \times 3 = 307,$200, which is far more than 244. Since robot planning scenarios such as this are potentially the worst cases, these numbers seem reassuring.

Although Murphy's Law is tantamount to assuming that Nature is playing to win, i.e., to thwart the agent's goals if possible.

[6]The figures here provide a sample of the domain model, see [9] for details.

Figure 3: A sample plan from the $\mathcal{AE}$
**Here is a sample plan for boiling water generated by graph search on the kitchen $\mathcal{AE}$.**

```
1.  MOVE-TO(ON(COUNTER1),POT,UNDER(FAUCET1)).
2.  FLIP(KNOB(FAUCET1),OFF,ON).
3.  Wait until A[AMOUNT-OF-IN(WATER,LIQUID,POT)]
                = ZERO ⟶ >.
4.  FLIP(KNOB(FAUCET1),ON,OFF).
5.  MOVE-TO(UNDER(FAUCET1),POT,ON(STOVE1)).
6.  FLIP(KNOB(STOVE1),OFF,ON).
7.  Wait until A[TBOIL(WATER,POT)]
    > A[TEMPERATURE(C-S(WATER,LIQUID,POT))] ⟶ =.
8.  Wait until A[AMOUNT-OF-IN(WATER,GAS,POT)]
                = ZERO ⟶ >.
9.  FLIP(KNOB(STOVE1),ON,OFF).
10.  MOVE-TO(ON(STOVE1),POT,ON(COUNTER1)).
```

A simple graph-search planner was built to find plans given a start state, goal, and $\mathcal{AE}$. The kitchen $\mathcal{AE}$, for example, does indeed contain many correct plans for boiling water starting with an empty pot (see figure 3), thus solving a problem Hogge's planner could not. Many of the plans are suboptimal (e.g., turning the stove and faucet off and on without moving the pot), and additional knowledge about efficiency and safety is needed to prune them.

To ensure that extraneous actions did not occur, two independent two-tank fluid systems were considered in the same scenario. As expected, the predicted effect of every action preserved assumptions concerning p-components not affected by that action. The dice example was developed to explore partially specified actions. The model allows a die to be on a table or in a hand, with PICKUP moving the die from the table to the hand and TOSS putting it back. The FACE-UP of a die is either 1,...,6 (when on the table) or ROLLING, when in the hand. As expected, the algorithm generates six possible outcomes for each toss, each corresponding to a different value of FACE-UP. This example illustrates that the constraint satisfaction techniques used in QPE can be used for some reasoning about actions even in the absense of continuous parameters.

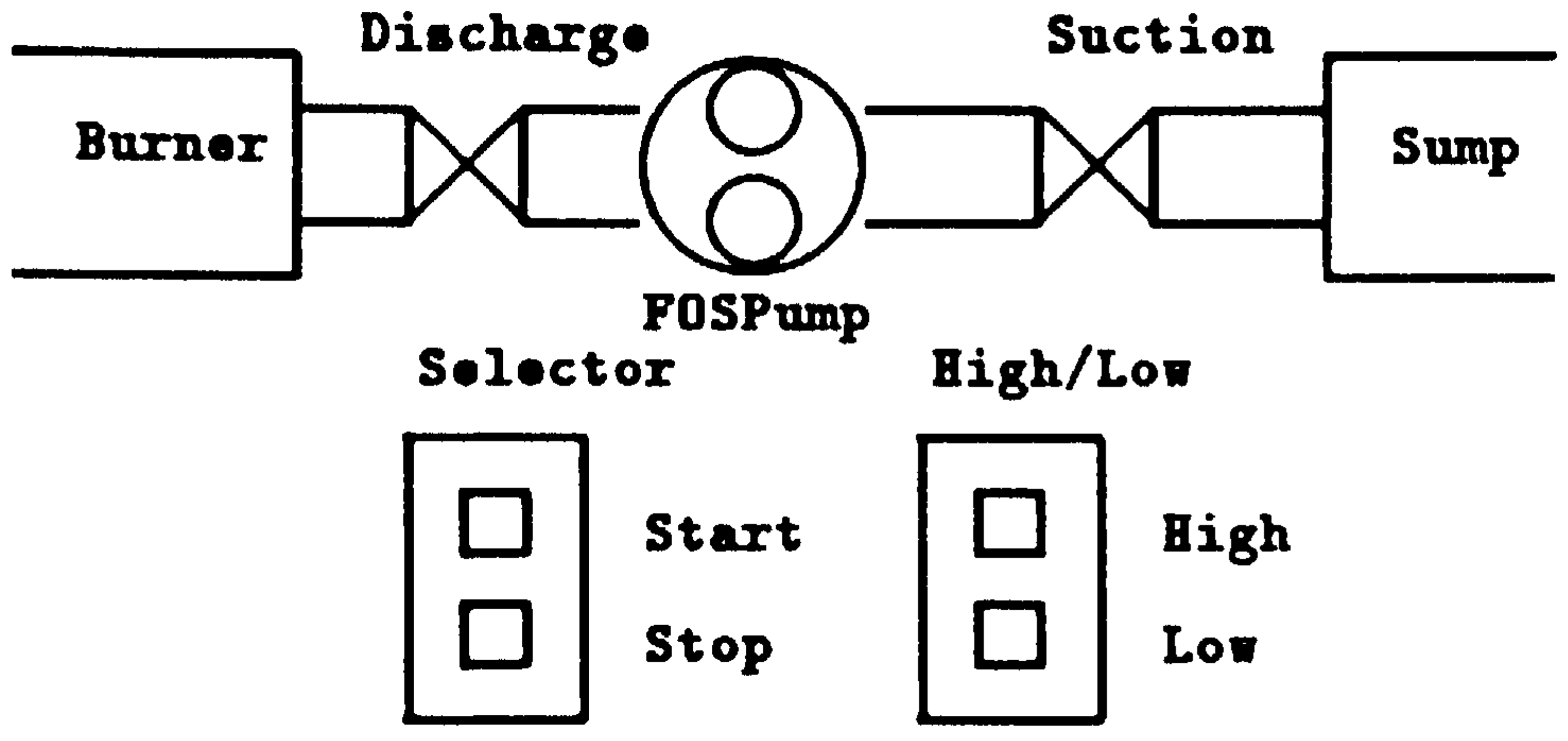
To explore reasoning about engineering procedures, a simple verifier was built which uses *A£s* to determine if a procedure can be successfully executed with the desired effects. It was tested with the simplified shipboard fuel oil service pump system (drawn from STEAMER [12]) depicted in Figure 4. The verifier confirmed that the starting and stopping procedures depicted can, if executed, lead to the desired results.

## 6 Discussion

This paper describes a method for integrating qualitative physics with models of action. It defines the *action-augmented envisionment,* which compactly represents all predicted changes due to a physics and possible actions within a scenario. The representation is domain-independent, in that it can be used for any problem that can be modeled using Qualitative Process theory. An algorithm for explicitly generating $\mathcal{AE}$**s** was presented and analyzed, and results from an implementation were summarized.

We believe this idea is an important step towards using qualitative physics in planning. Such understanding

Figure 4: Verifying engineering procedures with $\mathcal{AE}$s
**This shipboard system (drawn from STEAMER) pumps fuel oil from a storage sump to the burner in the boiler's furnace. It has been slightly simplified by ignoring gauges and a safety bypass valve.**



**Starting the pump:**

```
1.  FLIP(KNOB(SUCTION-LINE),CLOSED,OPEN).
2.  FLIP(KNOB(DISCHARGE-LINE),CLOSED,OPEN).
3.  FLIP(SELECTOR(FOSPUMP),STOP,START).
4.  FLIP(HIGH-LOW(FOSPUMP),LOW,HIGH).
5.  Wait until A[MAX-SPEED(FOSPUMP)]
        > A[SPEED(FOSPUMP)] ⟶ =.
6.  FLIP(HIGH-LOW(FOSPUMP),HIGH,LOW).
```

**Stopping the pump:**

```
1.  FLIP(SELECTOR(FOSPUMP),START,STOP).
2.  Wait until A[SPEED(FOSPUMP)] > ZERO ⟶ =.
3.  FLIP(KNOB(SUCTION-LINE),OPEN,CLOSED).
4.  FLIP(KNOB(DISCHARGE-LINE),OPEN,CLOSED).
```

could lead to important new applications, such as increased automation of procedure generation and safety analyses. For example, an important problem in training operators of complex systems is teaching them what the operating procedures are, and motivating these procedures in terms of the underlying principles of the plant. The ability to check a student's proposed procedure, for instance, could extend the ability of intelligent tutoring systems for this problem. Another problem is developin monitoring systems that can track system behavior an generate a running explanation of what is happening, to aid in fault management and operative diagnosis [14, 15, 1]. This algorithm is already being used to generate experimental knowledge-bases for measurement interpretation [3], so the explanations of system behavior can include the actions of operators. We suspect that for some engineering applications, the cost of explicit generation of *ASs* may be offset by the increased confidence in the quality of the answer, particularly as we discover how to build multi-grain domain models [4]. However, incremental algorithms are clearly possible and may be the most practical for many applications.

Even if *ASs* turn out to be infeasible to compute explicitly for all but the simplest systems, we expect the *AS* representation to be a useful theoretical foundation for understanding plans and procedures involving the physical world. Consider the notion of safety. Suppose a domain model characterizes particular conditions or processes as undesirable or dangerous (e.g., the pressure in a boiler reaching its maximum rating or the level of water in it reaching the bottom). We can call a state "unsafe" if one of these conditions holds in that state. An $\mathcal{AE}$ generated with this model will contain all the potential perils inherent in the scenario. Any operating (or maintenance) procedure forces the system through some subset of the envisionment, depending on the initial conditions and interleaving of actions and dynamics. Let the *dynamic closure* of a state be those states which can be reached from it via a sequence of limit hypotheses. A procedure which forces the system into an unsafe state is clearly unsafe. If the states traversed are all safe, and none of their dynamic closures contain unsafe states, then the procedure could be called safe, in the sense that arbitrary delays between executing the actions in their correct sequence cannot lead to narm. If the dynamic closure contains unsafe states, then the procedure could be considered risky, and extra constraints imposed to ensure that transitions to undesirable states are avoided. The consequences of operator error can be examined by perturbing the procedure to see if it leads to unsafe states.

Currently we are exploring several extensions. (1) We are formalizing plan evaluation constraints, including efficiency and safety, for guiding the graph search through *AS* and formulating incremental generation. (2) We are investigating more detailed duration representations to reduce ambiguity. One possibility is to use partial quantitative information, as in Kuipers and Berleant's recent work [13]. (3) Winslett [17] has independently developed a *possible models approach* for general reasoning about actions which can be viewed as a generalization of the *AS* technique. The algorithms described here, especially incremental versions, might thus be adaptable to more general problems involving reasoning about actions.

## 7 Acknowledgements

This paper benefited from discussions with Johan de Kleer, Marianne Winslett, Brian Falkenhainer, and Dennis Decoste. This research was supported by the Office of Naval Research, Contract No. N00014-85-K-0225, an NSF Presidential Young Investigator award, and an equipment grant from IBM.

## References

[l] Abbott, K. "Robust operative diagnosis as problem-solving in a hypothesis space", Proceedings of AAAI-88, August, 1988.

[2] de Kleer, J. "An assumption-based truth maintenance system" *Artificial Intelligence,* 28, 1986.

[3] Decoste, D. "Dynamic across-time measurement interpretation" *in preparation, 1989*

[4] Falkenhainer, B. and Forbus, K. "Setting up large-scale qualitative models", Proceedings of AAAI-88, August, 1988.

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

[6] Forbus, K. "The problem of existence", Proceedings of the Cognitive Science Society, 1985.

[7] Forbus, K. "The Qualitative Process Engine", Technical Report No. UIUCDCS-R-86-1288, December, 1986.

[8] Forbus, K. "The logic of occurrence", Proceedings of IJCAI-87, August, 1987.

[9] Forbus, K. "Introducing actions into qualitative simulation", Technical Report No. UIUCDCS-R-88-1452, August, 1988.

[10] Hogge, J. "Compiling plan operators from domains expressed in Qualitative Process theory", Proceedings of AAAI-87, July, 1987.

[11] Hogge, J. "The compilation of planning operators from Qualitative Process theory models", Technical Report No. UIUCDCS-R-87-1368, September, 1987.

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

[13] Kuipers, B. and Berleant, D. "Using incomplete quantitative knowledge in qualitative reasoning" Proceedings of AAAI-88, August, 1988.

[14] Malin, J. and Lance, N. "An expert system for fault management and automatic shutdown avoidance in a regenerative life support system" Proc. First Annual Workshop on Robotics and Expert Systems, Instrument Society of America, June, 1985.

[15] Malin, J. and Harris, R. "CONFIG: Qualitative simulation tool for analyzing behavior of engineered devices", First annual workshop on Space Operations Automation and Robotics, Houston, Texas, August, 1987.

[16] Schmolze, J. "Physics for robots", Proceedings of AAAI-86, August, 1986.

[17] Winslett, M. "Reasoning about action using a possible models approach", Proceedings of AAAI-88, August, 1988.