QPC: A Compiler from Physical Models into Qualitative Differential Equations*

James Crawford, Adam Farquhar, and **Benjamin Kuipers**
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
jc@cs.utexas.edu, farquhar@cs.utexas.edu, and kuipers@cs.utexas.edu

Abstract

Qualitative reasoning can, and should, be decomposed into a *model-building* task, which creates a qualitative differential equation (QDE) as a model of a physical situation, and a *qualitative simulation* task, which starts with a QDE, and predicts the possible behaviors following from the model.

In support of this claim, we present QPC, a model builder that takes the general approach of Qualitative Process Theory [Forbus, 1984], describing a scenario in terms of views, processes, and influences. However, QPC builds QDEs for simulation by QSIM, which gives it access to a variety of mathematical advances in qualitative simulation incorporated in QSIM.

We present QPC and its approach to Qualitative Process Theory, provide an example of building and simulating a model of a non-trivial mechanism, and compare the representation and implementation decisions underlying QPC with those of QPE [Falkenhainer and Forbus, 1988; Forbus, 1990].

Introduction

There have been a variety of productive approaches to qualitative reasoning about physical systems [Bobrow, 1985; Weld and de Kleer, 1990]. Alternate approaches (e.g. [de Kleer and Brown, 1984; Forbus, 1984; Kuipers, 1984; Kuipers, 1986; Williams, 1989; Williams, 1988]) frequently differ in emphasis and content, and can seem incompatible. However, we believe that the common themes underlying these different approaches can be clarified by decomposing qualitative reasoning into two tasks:

	Model Building		Qualitative Simulation	
Physical Situation	$\rightarrow$	QDE	$\rightarrow$	Behaviors

- *Model-building* creates a qualitative differential equation (QDE) as a model of a physical situation.
- *Qualitative simulation* starts with a QDE, and predicts the possible behaviors following from the model.

The QSIM research effort (surveyed in [Kuipers, 1989]) has focussed primarily on the qualitative simulation task: predicting the possible qualitative behaviors consistent with a given QDE and initial state:

$$\text{QSIM} \vdash (\text{QDE} \;\&\; \text{State}(t_0) \rightarrow \text{or}(\text{Beh}_1, \ldots \text{Beh}_n))$$

Research into the mathematics underlying the simulation of qualitative differential equations has been very fruitful, yielding higher-order derivative constraints, phase space representations, integral representations, energy constraints, algebraic and quantitative reasoning methods, and more [Weld and de Kleer, 1990].

Qualitative reasoning methods based on component-connection descriptions [de Kleer and Brown, 1984] or view-process descriptions [Forbus, 1984] also lead to behavioral predictions, but in ways that mix elements of the model-building and model-simulation tasks, obscuring their relationship.

This paper describes QPC, which assembles a QDE model of a physical situation by drawing on a library of model-fragments (e.g. views and processes); QSIM is then used to predict the behaviors consistent with the model.[1] QPC is based on the model-building aspects of Qualitative Process Theory [Forbus, 1984; Forbus, 1990; Falkenhainer and Forbus, 1988], a major approach to the creation and simulation of qualitative models. (Franke and Dvorak have previously reported on *CC*, a compiler from component-connection models into QSIM QDEs [Franke and Dvorak, 1989].)

*This paper appeared in AAAI-90. This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8602665, IRI-8905494, and IRI-8904454, by NASA grants NAG 2-507 and NAG 9-200, and by the Texas Advanced Research Program under grant no. 003658–175.

[1] This approach was originally proposed by Kuipers in his 1986 AAAI Tutorial on Qualitative Reasoning, and was explored in [Vincent, 1988].

Algernon, our implementation of Access-Limited Logic [Crawford and Kuipers, 1989], serves as the knowledge representation language for implementing QPC. It combines the clarity, rigor, and expressive power of predicate logic with the efficiency and intuitive appeal of a frame-based semantic network. Such a foundation will be necessary for the application of qualitative reasoning to non-trivial scenarios and large knowledge bases with a realistic library of views and processes.

There are several benefits which we hope QPC will provide. First, comparison and contrast between QPC and QPE will shed additional light on the model-building ideas in Qualitative Process Theory. Second, a clear decomposition at the QDE representation allows qualitative reasoning generally to benefit from independent advances in model-building and qualitative simulation. Third, the incremental model-building capability provided by QPC avoids the use of total envisionments, which can be intractable in some cases. This is especially important in tasks such as monitoring and control where many "possible" situations need never be examined because knowledge about the state of the system is available [Dvorak and Kuipers, 1989].

After describing the model-building methods in QPC, its relationship with QSIM, and presenting a detailed example, we discuss the differences in philosophy and implementation between QPC and QPE, Forbus' [1989] implementation of Qualitative Process Theory.

Overview of QPC

The basic QPC algorithm consists of four steps:

1. Assemble a view-process structure from a description of the scenario.
2. Apply the closed world assumption and build the QDE.
3. Form an initial state.
4. Simulate using QSIM.

Two kinds of complexity add iterative paths to this simple sequence (figure 1). First, when the initial state is formed, additional variable values are learned which may activate additional views and processes. This may necessitate re-building the QDE. Second, when simulation reaches a boundary of the QDE being simulated, control is returned to QPC so that a new model can be created.

Representing Views and Processes

The QPC knowledge-base has three components. The first consists of background knowledge about scenarios, models, views, and processes, as well as basic information about the physical world (e.g. that materials can be in three possible states: solid, gas, or liquid). The second component is a domain library of processes and views. The third contains instantiated processes and views for specific entities in the world.

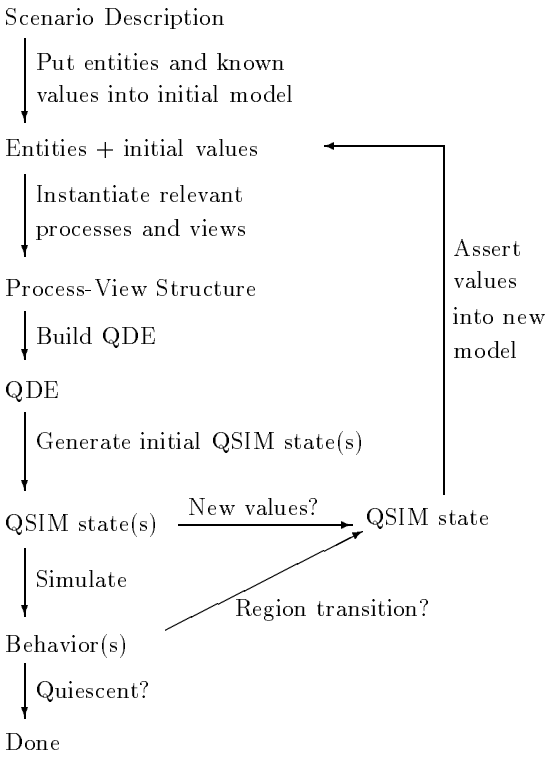

Figure 1: Flow of control in QPC

In QPC, both processes and views are represented by rules which create their instances. A user syntax like that of QPE could easily be provided, but in this paper we focus on the underlying representation used by QPC. We refer to both views and processes by the general term *model fragment*. A model fragment is created only once, and can then be included in a variety of models. Figures 2 and 3 shows rules representing the physical view of a physical object and the fluid-flow process, respectively.

Building the View-Process Structure

Model-building starts with a *scenario* which identifies the entities in the world one is interesting in modeling, and specifies their initial conditions. The entities in the scenario become part of the initial model of the scenario (but may or may not be part of subsequent models of the scenario, as entities can be created or destroyed by region transitions). QPC builds the view-process structure for the initial model by first adding, to the initial model, any entities needed to complete it, and then determining which instances of views and processes are relevant [Forbus, 1990; Falkenhainer and Forbus, 1988].

We illustrate QPC with the scenario depicted in fig-

```
((physical-view ?x ?view)
 (mass ?x ?mx) (volume ?x ?vx) (pressure ?x ?px)
 ->
   ;; link a view to its variables
   (variable ?view ?mx) (variable ?view ?vx)
   (variable ?view ?px)
   ;; cd-ineqs are inequalities indexed by view
   ;; Mass not less zero
   (not (cd-ineq ?view less ?mx zero))
   ;; Volume not less zero
   (not (cd-ineq ?view less ?vx zero))
   ;; Mass=0 <-> Volume=0
   (correspondence ?view ?mx zero ?vx zero))
```

Figure 2: A rule to fill in the physical view of a physical object.

ure 4. It consists of two containers, `A` and `B`, connected by a fluid path. `B` has a *portal* located part way up one side. Initially there is fluid in container `A`.

The scenario is set up in QPC by creating (in the Algernon KB) frames for the containers `A` and `B`, the open fluid path connecting them, and then asserting that there is fluid in `A` and a portal in `B`. We also assert that `A` is an *entity* in the initial model of the scenario, but do not explicitly link `B` into the scenario (as QPC will do so automatically). The Algernon assertions to establish the scenario are shown in figure 4.

QPC then applies rules to complete the set of entities in the initial model. For example, if a container is part of a model and it is connected, via an open connection, to another container, then the second container should be considered part of the model. Instantiated for fluid-connections this rule reads:

```
((fluid-connection ?obj1 ?path1 ?obj2)
 (open ?path1 true)
 (part-of ?obj1 ?model1)
  ->
  (entity ?model1 ?obj2))
```

where the relation `entity` links a model to its objects.

QPC deduces that `B`, the portal in `B`, and the contents of `A` must be included in the initial model. Instantiation of a fluid flow from `A` to `B` implies the need for a frame for the contents of `B`, which is created and added to the model (along with a frame for its physical view). The initial model thus consists of the physical views of `A`, `B`, the portal, the contents of `A` and `B`, and the fluid flow process. The influences, relations, correspondences and inequalities of these views and processes are shown in figure 5. Notice that, as yet, no process or region transition for portal flow has been added. Neither the process nor the region transitions are set up until the relationship between the fluid level of `B` and the portal height is learned.

```
((fluid-connection ?can1 ?path ?can2)
 (part-of ?can1 ?model) (isa ?model models)
 (flow-rate ?path ?flow-rate)
 (pressure-difference ?path ?pressure-diff)
 (contents ?can1 ?liquid1)
 (isa ?liquid1 contained-liquids)
 (mass ?liquid1 ?mass1)
 (open ?path true)
 (pressure ?can1 ?pressure-can1)
 (pressure ?can2 ?pressure-can2)
 ->
 ; Find the process OR Create a new one.
 (:forc ?process
      (cd ?model ?process)
      (isa ?process fluid-flow-processes)
      (path ?process ?path))
 (variable ?process ?flow-rate)
 (variable ?process ?pressure-diff)
 (correspondence ?process ?flow-rate zero
                          ?pressure-diff zero)
 ;; pressure-diff = c1.pressure - c2.pressure
 (ADD ?process ?pressure-can2 ?pressure-diff ?pressure-can1)
 (influence ?process Q+ ?pressure-diff ?flow-rate)
 (influence ?process I- ?flow-rate ?mass1)
 (:forc ?liquid2
        (contents ?can2 ?liquid2)
        (same-material ?liquid1 ?liquid2)
        (same-state ?liquid1 ?liquid2))
 (influence ?process I+ ?flow-rate (mass ?liquid2)))
```

Figure 3: The rule to instantiate the fluid-flow process. The relation `fluid-connection` links a container, a path, and another container. The relation `cd` links a model to a view or process.

Applying the Closed-World Assumption and Building the QDE

At this point, QPC has created a view-process structure comprising a collection of influences, relations, inequalities, and correspondences. The next step is to convert to a QDE which consists of constraints, quantity spaces, landmarks, and corresponding values.

The key step is to transform a collection of influences into constraints. If X *influences* Y then Y will change as a result of a change in X, *all else being equal.* A *constraint* between X and Y is a universal law, limiting the possible joint behaviors of X and Y, *independent of context.* Thus, in order to transform influences into constraints, we require a Closed World Assumption, asserting that we know *all* the relevant influences.

Intuitively, the indirect influence or "qualitative proportionality" $Q^+(X_1, Y)$, means that an increase in X_1 will tend to increase Y. More formally:

$$Q^+(X_1, Y) \equiv Y = f(X_1, X_2, \ldots X_n) \text{ and } \frac{\partial f}{\partial X_1} > 0,$$

for some functional relationship f (with an indefinite number of arguments). The direct influence is similar:

$$I^+(X_1, Y) \equiv \frac{dY}{dt} = f(X_1, X_2, \ldots X_n) \text{ and } \frac{\partial f}{\partial X_1} > 0.$$

```
;; The U-Tube Scenario
(:create ?utube) (current-scenario global-context ?utube)
;; Create A, B, and the pipe between them.
(:create ?A) (isa ?A containers) (entity ?utube ?A)
(:create ?B) (isa ?B containers)
(:create ?pipe) (open ?pipe true)
(fluid-connection ?A ?pipe ?B)
;; The contents of A has mass greater than zero.
(:create ?A-contents) (contents ?A ?A-contents)
(state ?A-contents liquid-state)
(:create ?A*) (greater ?A* zero)
;; (tmag var model time mag) → var = mag at time in model.
(tmag (level ?A-contents) ?utube (initial-time ?utube) ?A*)
;; B has a portal.
(:create ?port) (isa ?port portals) (portal b ?port) (open ?port true)
;; ...and forms to assert that the bottoms of A and B
;; are at zero, and the top heights are positive.
```

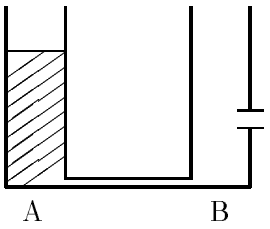

Figure 4: Scenario description for the u-tube with portal. `(:create ?x)` creates a new frame and binds `?x` to it.

A Physical View:
top-height $\geq$ fluid-level $\geq$ bottom-height

A-Contents Physical View:
mass $\geq$ zero; volume $\geq$ zero
volume $\leq$ (volume A)
mass Q^+ volume Q^+ level Q^+ pressure

mass = zero $\leftrightarrow$ volume = zero $\leftrightarrow$
level = zero $\leftrightarrow$ pressure = zero

level = (bottom-height A) $\leftrightarrow$ volume = zero
level = (top-height A) $\leftrightarrow$ volume = (volume A)

Fluid Flow Process:
pressure-diff = (pressure A) - (pressure B)
pressure-diff Q^+ flow-rate
flow-rate I^- (mass (contents A))
flow-rate I^+ (mass (contents B))
flow-rate = zero $\leftrightarrow$ pressure-diff = zero

B-Portal Physical View:
(bottom-height B) < height < (top-height B)
height $\geq$ zero

Figure 5: Highlights of the initial views and processes for the u-tube with portal. The physical views of `B` and `B-Contents` are similar to those of `A`.

Influence resolution on a variable Y identifies the sets P and N of variables that positively and negatively influence Y. Based on the CWA, this determines the number of arguments to the function f. Qualitative Process Theory makes the additional assumption that f can be approximated by a linear combination of single-variable functional relationships. This allows us to assert QSIM *constraints* to capture the set of indirect influences on Y:

$$Y = \sum_{X_i \in P} M^+(X_i) - \sum_{X_j \in N} M^+(X_j).$$

Resolution of direct influences is similar.

QPC helps clarify the role of the linear decomposition assumption in model-building. This assumption does not cause problems durring purely qualitative simulation, but as we attempt to incorporate quantitative information into the model [Kuipers and Berleant, 1988], cases where it is invalid will raise difficulties. Such cases will require a qualitative theory of multivariate functional relations.

Inequality information in the view-process structure is represented in several ways in the QDE. Inequality relations between magnitudes are used to order the quantity spaces. Inequalities between variables and variables, or variables and magnitudes, are represented in the QDE as boundary conditions triggering operating region transitions. For example, if the fluid-level reaches the level of the portal and is increasing, then the portal-flow process must be made active (adding additional influences and relations to the model).

Highlights of the initial QDE for the u-tube example are shown in figure 6.

Building the Initial State

At this point, we have created a QDE which reflects the current view-process structure, but we do not have initial values for all the variables in the model. We calculate initial values in three steps:

1. Propagate known values through the QDE.
2. Apply default assumptions.
3. Generate possible completions.

```
(define-qde utube-initial-model
  (quantity-spaces
   (a-contents-level (minf 0 a* a-top inf))
   (b-contents-level (minf 0 b-top inf))
   ...)
  (constraints
   ((m+ a-contents-mass a-contents-volume) (0 0))
   ((m+ a-contents-volume a-contents-level)
        (frame23 a-top) (0 0))
   ((m+ a-contents-level a-contents-pressure)
        (0 0))
   ((= a-contents-level a-fluid-level))
   ((= a-contents-pressure a-pressure))
   ((add b-pressure pipe-ab-pressure-diff a-pressure))
   ((m+ pipe-ab-pressure-diff pipe-ab-flow-rate)
    (0 0))
   ((minus var-1 pipe-ab-flow-rate))
   ((d/dt a-contents-mass var-1))
   ((d/dt b-contents-mass pipe-ab-flow-rate))
   ((constant b-portal-height)) ...)
  ...)
```

Figure 6: Highlights from the initial QDE for the example. Constraints on `B` are similar to those on `A`.

Propagation Frequently, initial values are given for only some of the variables, but other values follow easily from the constraints and relations in the QDE. It would be possible to build rules into the knowledge-base to calculate such values, but this would unnecessarily duplicate the knowledge already in QSIM. Instead, we use QSIM itself as an efficient special purpose reasoning tool to propagate the known values through the QDE. In the u-tube example, propagation concludes, among other values, that the mass of the contents of `A` is greater than zero (but concludes nothing about the mass of the contents of `B`).

Default Assumptions During automatic model building, it may be impossible to establish values for enough variables to uniquely determine an initial state. Our solution to this problem is to make default assumptions which are appropriate for the model. E.g, in the u-tube no initial value is known for the mass of the contents of `B`, and it is not possible to determine a value through propagation. However, QPC assumes that the mass of any newly created liquid is zero. Such values are explicitly tagged as assumptions in the knowledge-base so that they can be withdrawn if they lead to a contradiction. In the examples we have looked at, propagating known values before making default assumptions has been sufficient to avoid such contradictions.

Finding All Completions Even after propagation and the default assumptions, there may be variables which do not have known values. At this point we again use QSIM as a special purpose reasoner to construct *all* possible completions of the current state. In simple cases, such as the u-tube, there is only one possible completion. If there are multiple completions, a separate model must be created for each of them.[2]

In either case it is possible for the new values (from propagation, default assumptions or state completion) to require additional region transitions, new views, or new processes. To handle this problem, the resulting completed state information is asserted back to the knowledge-base, causing the appropriate rules to fire. In the u-tube example, the default assumptions lead QPC to assume that the fluid level in `B` is zero. This sets up a region transition that will instantiate the portal flow process if the fluid level ever reaches the portal height and is increasing.[3]

Simulation and Region Transitions

Once a complete initial state has been created, QSIM is used to simulate the possible behaviors. In the u-tube example, QSIM predicts three behaviors: one in which equilibrium is reached below the portal-height, one in which equilibrium is reached exactly at the portal height, and a third in which the fluid level in tank `B` reaches the portal and continues to increase. The first two behaviors can be simulated using only the initial model. The third behavior, however, triggers a region transition and the building of a new model.

When a behavior ends in a region transition, QPC attempts to construct a new set of models. This is done by creating an empty model and asserting the quantity spaces and variable values of the final state of the behavior into it. The new model is then linked to its predecessor. QPC checks the previously active process and view instances to determine which remain active in the new model. QPC then determines whether new entities need to be included, and whether new views or processes need to be activated. Finally, the QDE and initial state(s) are built as before.

In the u-tube example, after the region transition, QPC is able to retain pointers to the old model fragments for `A`, `B`, `B-portal`, and `Fluid-flow-AB`. A new portal flow process is created, since its precondition, that the fluid level in `B` is greater than or equal to the portal height, is now satisfied. This results in an additional influence on `B-contents-mass`.

The expected behavior is, of course, that the level of liquid in `B` will increase until the flow in from `A` and the flow out of the portal equalize, and then the liquid

[2] This is the first "choice point" in QPC (the second being the case in which a simulation produces several behaviors ending in region transitions). In such cases the possibilities are queued, and we use a simple search strategy to select the one to follow next.

[3] In complex models, the additional views and processes activated at this point may invalidate the closed world assumption; new views and processes may add influences on some variable v previously assumed to be constant. In such cases, we must return to the original view-process structure and assert the new influence on v, rebuild the QDE, and recalculate the initial values.

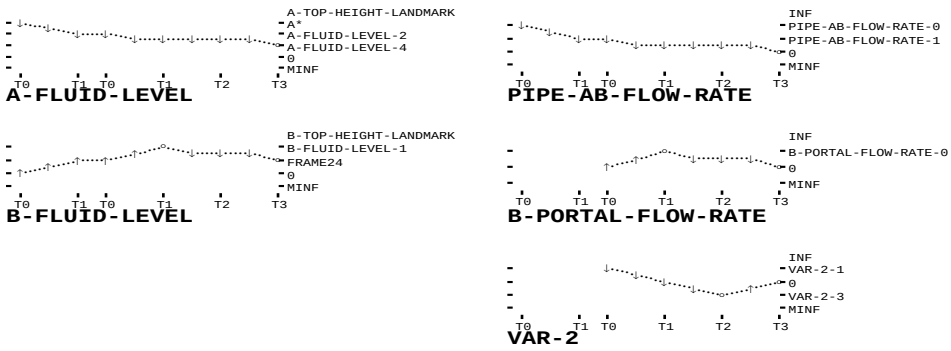

Figure 7: A QSIM behavior for the u-tube example which spans two models. The time step begins again at zero for the second model. `B-portal-flow-rate` and `var-2` are not present in the initial model, but are defined by the portal-flow process. `Var-2` is the netflow into `B`, i.e. `pipe-ab-flow-rate` - `B-portal-flow-rate`.

will drain out of the portal until a final equilibrium is reached, with the level of `B` even with the portal. This type of behavior is difficult for qualitative simulators to reason about because `B-net-flow` is the difference between two positive and decreasing values. Simple qualitative subtraction is ambiguous, and the result can become negative, zero, or positive any number of times. This behavior is known as *chatter*, and results in an infinite number of qualitatively distinct behaviors.

Fortunately, there is a solution. QSIM automatically derives constraints based on the second derivatives of the variables [Kuipers and Chiu, 1987]. It is this sort of advance in qualitative mathematics which we were hoping to take advantage of! Instead of producing an infinite tree of behaviors, QSIM produces a small number: `B-contents-level` reaches a maximum somewhere above the portal, or it reaches a maximum at the top-height, or B overflows, triggering a region transition. The first two behaviors drop down to our expected equilibrium state; the third causes a new model to be constructed. Figure 7 shows a QSIM plot for a behavior spanning two models and ending in the final equilibrium state in which the level in `B` is at the height of the portal.

Comparison with QPE

While we are following the Qualitative Process Theory approach to model-building, QPC differs in numerous ways from QPE [Forbus, 1990] and its predecessor, GIZMO [Forbus, 1984]. For simplicity, we will use the term QPE for both versions.

Influences and Constraints

Simulation requires a CWA to assert that all influences on all variables are known. Automatic model-building, on the other hand, requires an open-world assumption, so that models can be build by composing model fragments which are stated independently of context.

In QPE, the meanings of Q^+ and I^+ are context dependent, with an open world assumption holding in the view-process library, and a closed-world assumption holding after influence resolution. We believe that this use of the same symbols for semantically distinct concepts has been a source of confusion in the literature.

In QPC, influences belong only to the model-building phase, while a QDE consists only of constraints. Since influences and constraints are semantically distinct, we make them syntactically distinct as well, using Q^+ and I^+ for influences, and M^+ and d/dt for constraints.

Total Envisionment versus Incremental Model-Building

QPE simulates the possible behaviors of a mechanism by producing a *total envisionment*: a graph of all possible states, linked by the transitions between them. The total envisionment representation has several advantages, including a finite representation for infinite behaviors, and support for certain global operations such as cycle detection and state aggregation.

On the other hand, it also raises significant problems of both semantics and efficiency. Semantically, the total envisionment representation depends on the fact that all qualitatively important landmark values are known statically when simulation begins. Dynamically created landmarks are critical to making many important distinctions among behaviors, such as the distinction between increasing and decreasing oscillations [Kuipers, 1985; Kuipers, 1986].

Efficiency can also be a problem. Creation of all possible states of the mechanism is an up-front cost of the total envisionment, required before the transition graph can be constructed. For a very complex model, or worse an "unboundedly creative" one [Forbus, 1989], creation of the set of states is intractable. We minimize this problem in QPC by building models and creating

states incrementally, as needed by the simulation. This makes a critical difference when external constraints, such as observations, can focus the simulator's attention to a tractable "beam search" within a potentially infinite behavior tree. In particular, in applications such as monitoring and control, many "possible" states of the system need never be considered and only limited look-ahead is needed [Dvorak and Kuipers, 1989].

The QPE implementation is based on an ATMS [de Kleer, 1986], whereas QPC is built in Algernon, a frame-based knowledge representation language based on Access-Limited Logic [Crawford and Kuipers, 1989]. The ATMS is used in QPE as an efficient tool for implementing several exhaustive search or generation tasks, such as creation of all possible states for the total envisionment, or search for a combination of `consider` statements capable of answering a given question. However, as discussed above, we believe that the total envisionment is often more difficult to compute, and less useful, than the set of possible behaviors. We also believe that the inference involved in model-building will require the service of a full knowledge representation language.

A Layered Representation for Model Revision

Since QPC builds models incrementally, we must deal with a version of the frame problem: what must change and what remains the same after a region transition? Rather than build the new model from scratch, we have structured the representation so that chunks of the old model may be incorporated in the new model. The representation is structured in layers, as shown below, so that each layer changes more slowly than the one below it.

Individuals and their Relationships

Views and Processes

Models (QDEs)

Variable Values

- At the lowest level are the values of variables in the model. These values generally change at every step of the simulation.
- One step up are the models (QDEs) built by QPC. Models are likely to remain valid for several simulation steps, but still change whenever a region transition occurs.
- Changing more slowly, are the views and processes. When region transitions occur, they generally cause one or more view or process instances to become invalid and one or more new ones to be activated. In general, however, most of the views and processes from the previous model are still valid. For example, in the u-tube example, when the portal flow begins, a new "portal flow" process is created, but the views of the containers, and the old fluid flow process, are unchanged.
- Finally, the set of individuals and their relationships change the most slowly. For example, initiation of a boiling process would create a new individual to represent the steam produced. Our framework handles the creation or deletion of individuals naturally.

Conclusion

We have demonstrated QPC as a model-building tool that takes the Qualitative Process Theory view of the modeling task, and compiles models into QDEs for simulation by QSIM.

This approach clarifies several aspects of the structure of qualitative reasoning. First, the tasks of model-building and qualitative simulation can be treated as essentially independent, communicating in the language of qualitative differential equations. Second, the comparison between QPC and QPE helps us distinguish between the fundamental ideas in Qualitative Process Theory and the design decisions of QPE.

In addition to theoretical clarity, QPC provides us with several more tangible benefits. First, we believe that a history-based approach to model-building as well as simulation will be essential for qualitative reasoning about complex mechanisms that would overwhelm a total-envisionment-based approach. Second, we believe that the mathematical methods developed for use with the QSIM representation are essential to reasoning qualitatively about models of complex systems. Finally, QPC provides a bridge between Algernon, a general-purpose knowledge representation language designed for large-scale knowledge bases, and QSIM, an efficient special purpose reasoning system in the domain of qualitative simulation. We expect to exploit this combination to work in the following areas:

- Answering questions and explaining the predicted behaviors. We expect QPC to support explanations which draw on descriptions of the system at multiple levels of detail: the scenario description, the view-process structure, the QDE, and the predicted behaviors of the system.
- Resolving discrepancies between prediction and observation by considering alternative views of the objects in the model. For example, the nail does not fall (as predicted by the physical view) because it is attracted by the magnet.
- Using QPC (and thus QSIM and model-based reasoning) as a component of very large knowledge-bases [Porter *et al.*, 1988; Lenat and Guha, 1990].

References

D. G. Bobrow, editor. *Qualitative Reasoning about Physical Systems*. Bradford Books/MIT Press, Cambridge, MA, 1985.

J. M. Crawford and B. J. Kuipers. Toward a theory of access-limited logic for knowledge representation. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, Los Altos, CA, 1989. Morgan Kaufman.

J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

J. de Kleer. An assumption-based truth maintenance system, Extending the ATMS, Problem solving with the ATMS. *Artificial Intelligence*, 28(2):127–224, 1986.

D. Dvorak and B. J. Kuipers. Model-based monitoring of dynamic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI–89)*, pages 1238–1243, Los Altos, CA, 1989. Morgan Kaufman.

B. Falkenhainer and K. D. Forbus. Setting up large-scale qualitative models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-88)*, Los Altos, CA, 1988. Morgan Kaufman.

K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

K. D. Forbus. Pushing the edge of the (QP) envelope. 3rd Qualitative Physics Workshop, Stanford, CA, 1989.

K. D. Forbus. The qualitative process engine. In *Readings in Qualitative Reasoning about Physical Systems*, pages 220–235. Morgan Kaufman, 1990.

D. Franke and D. Dvorak. Component connection models. Model-Based Reasoning Workshop, Eleventh International Joint Conference on Artificial Intelligence (IJCAI–89), Detroit, Michigan, 1989.

B. J. Kuipers and D. Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 324–329, Los Altos, CA, 1988. Morgan Kaufman.

B. J. Kuipers and C. Chiu. Taming intractable branching in qualitative simulation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, 1987.

B. J. Kuipers. Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence*, 24:169–204, 1984.

B. J. Kuipers. The limits of qualitative simulation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Altos, CA, 1985. William Kaufman.

B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

B. J. Kuipers. Qualitative reasoning: modeling and simulation with incomplete knowledge. *Automatica*, 25:571–585, 1989.

D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems*. Addison-Wesley, Reading, MA, 1990.

B. W. Porter, J. Lester, K. Murray, K. Pittman, A. Souther, L. Acker, and T. Jones. AI research in the context of a multifunctional knowledge base: the botany knowledge base project. Technical Report AI TR-88-88, University of Texas at Austin, 1988.

T. C. Vincent. Model building using qualitative process theory. Master's thesis, UT Austin, May 1988.

D. S. Weld and J. de Kleer. *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufman, Los Altos, CA, 1990.

B. Williams. Minima: A symbolic approach to qualitative algebraic reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 264–269, Los Altos, CA, 1988. Morgan Kaufman.

B. Williams. Qualitative analysis of MOS circuits. *Artificial Intelligence*, 24:281–346, 1989.

Availability of Code

The code for QSIM may be obtained for research purposes from Benjamin Kuipers. We plan to have distributable versions of Algernon and QPC available shortly after AAAI-90.