A Qualitative Physics Compiler

Adam Farquhar*
Knowledge Systems Laboratory
701 Welch Road, Bldg. C
Palo Alto, CA 94303
Adam_Farquhar@ksl.stanford.edu

Abstract

Predicting the behavior of physical systems is essential to both common sense and engineering tasks. It is made especially challenging by the lack of complete precise knowledge of the phenomena in the domain and the system being modelled. We present an implemented approach to automatically building and simulating qualitative models of physical systems. Imprecise knowledge of phenomena is expressed by qualitative representations of monotonic functions and variable values. Incomplete knowledge about the system is either inferred or alternative complete descriptions that will affect behavior are explored. The architecture and algorithms used support both effective implementation and formal analysis. The expressiveness of the modelling language and strength of the resulting predictions are demonstrated by substantial applications to complex systems.

Introduction

Predicting and reasoning about the behavior of physical systems is a difficult and important task that is as essential to everyday commonsense reasoning as it is to complex engineering tasks such as design, monitoring, control, or diagnosis.

This paper describes QPC (a Qualitative Physics Compiler), an implemented approach to reasoning about physical systems that builds on the expressiveness of Qualitative Process Theory (Forbus 1984) and the mathematical rigor of the QSIM qualitative simulation algorithm (Kuipers 1986). The QPC work was first reported in (Crawford, Farquhar, & Kuipers 1990) and is more fully developed in (Farquhar 1993).

*This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin and at the Stanford University Knowledge Systems Laboratory. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8904454, IRI-9017047, and IRI-9216584, and by NASA contracts NCC 2-760 and NAG 9-665. Research at KSL is sponsored by the Advanced Research Projects Agency, ARPA Order 8607, monitored by NASA Ames Research Center under grant NAG 2-581; and by NASA Ames Research Center under grant NCC 2-537.

The critical intuition is that reasoning about a physical system can and should be decomposed into a *model building* task which builds a model to describe the system, and a *simulation* task which uses the model to generate a description of the possible behaviors of the system. QPC's architecture, algorithms, and semantics follow from this decomposition.

Experience has shown that the QPC modeling language is expressive and the algorithms are powerful enough to provide useful behavioral predictions for a wide variety of systems. Catino and Ungar have constructed a substantial chemical engineering domain theory and use QPC to build models of a nitric acid processing plant with several hundred variables and over a hundred model fragments (Catino 1993; Catino & Ungar 1994). Rickel has solved problems of more moderate size in the domain of plant physiology (Rickel & Porter 1994). Rajagopalan has constructed models for geometric and spatial reasoning (Rajagopalan 1994). Brajnik has constructed models of lakes, damns, and turbines in the domain of water supply control (Farquhar & Brajnik 1994).

Modeling Language

The input to QPC is a *domain theory* and *scenario* specified in the QPC modeling language. A *domain theory* consists of a set of quantified definitions, called *model fragments*, each of which describes some aspect of the domain, such as physical laws (e.g. mass conservation), processes (e.g. liquid flows), or entities (e.g. containers). The specific system or situation being modeled is partially described by the *scenario* definition, which lists a set of entities that are of interest, some of their initial conditions, relations that hold throughout the scenario, and boundary conditions. See figure 1 for a sample model fragment and scenario definition.

The QPC modelling language is a descendent of the expressive language introduced by Forbus for his Qualitative Process theory (Forbus 1984). A model fragment definition specifies a set of typed participants, structural, operating conditions, and consequences. An instance of the fragment exists if participants of the appropriate types can be found that satisfy the structural relations. The instance is *active* roughly when the operation conditions (inequalities over variables) are true. While the instance is active, its consequences hold. These can be (differential) equations, inequalities, or arbitrary logical relations. There are also language constructs for defining relations, axioms, quantities, and the conditions under which entities must exist. The full language is documented in (Farquhar 1993).

Each language form is defined by a set of axiom schema; a domain theory is defined by a set of axioms. There are several syntactic restrictions on the language constructs and the axiom schema. In particular, the resulting axioms can all be written as implications whose antecedents and consequents are conjunctions of literals. Negation is only applied to atomic formulae and embedded disjunction is not allowed. Existential quantification does not appear in the user language and is limited in the resulting axioms to a single variable in the consequent of an implication. Importantly, *time* is implicit in the user language. This ensures that the present cannot change the past and invalidate itself. The temporal scope of relations is enforced by the axiom schema (Farquhar 1993).

The language allows for systems to be described by continuous real-valued variables and time-dependent logical relations. A time-dependent relation is one whose truth value may change over time (e.g. `(on-table block1)`). The language restricts the interaction between time-dependent relations and quantities. Implications relating a time-dependent relation and a quantity can only be expressed with a `defmodelfragment` form, whereas implications among time-dependent relations alone may appear in other forms. This allows possible changes in the equations that describe the system to be isolated. A change in the equations can only result from a change in the activity of a model fragment instance. This property is essential to support QPC's hybrid architecture. The simulator is only able to predict changes in variable values and monitor relations between quantities. The above restriction ensures that this is sufficient to detect changes in the equations.

Architecture and Algorithm

The critical insight enabling QPC to model and simulate complex systems is the separation of the model building task from the simulation task. The architecture of QPC directly reflects this decomposition. The model building task is facilitated by a general purpose knowledge representation system. QPC currently uses the Algernon representation system (Crawford & Kuipers 1991). The simulation task is currently handled by the QSIM qualitative simulation system (Kuipers 1986; Farquhar *et al.* 1991).

This approach has several practical and theoretical advantages. The proof of correctness is simplified. QSIM has already proven to be sound (Kuipers 1986), so the proof reduces to showing that the correct models are generated. The sources of incompleteness can be readily identified, localized, and methods to reduce them determined. The output of QPC can be directly understood in terms of ordinary differential equations and their solutions. QPC exploits advances in simulation techniques as soon as they are incorporated into the simulator. The data structures and algorithms best suited to efficient computation in the model building and simulation tasks are quite different; a hybrid architecture reflects this difference and simplifies implementation.

We define the operation of the model building component of QPC in terms of databases. A *database* is a set of logical facts and axioms. A database is *complete* if every model fragment instance (explicit member of the set of model fragments) in the database is known to be active or inactive.

The input to the model building component is a database that is the union of the axioms induced by a domain theory and scenario description. It uses this to infer the complete set of model fragment instances that apply during the time covered by the database (this may require a case analysis that produces a disjunction of complete databases from an incomplete one). A database with a compete set of model fragment instances defines an initial value problem comprising a set of equations, initial, and boundary conditions. This initial value problem is provided to the simulator to solve. The initial value problem is defined using QSIM's representation of qualitative differential equations (QDEs). If any of the predicted behaviors cross the boundary conditions, the process is repeated: a new database is constructed to describe the system as it crosses the boundaries the current model.

The output of QPC is a rooted graph that describes the possible behaviors of the situation. A node in the graph is either a database Δ or a qualitative state S with its associated QDE. The root of the graph is the database formed from union of the axioms induced by the domain theory and scenario definition. An edge in the graph may be one of the following:

1. $\langle \Delta_1, \Delta_2 \rangle$ *refinement*
Δ_1 is an incomplete description of a situation. Δ_2 is a more complete description that results from adding some ground conditions to Δ_1.
2. $\langle \Delta, \mathrm{S} \rangle$ *initial state*
Δ is a complete description of a situation. The qualitative state S is an initial state that follows from Δ.
3. $\langle \mathrm{S}_1, \mathrm{S}_2 \rangle$ *state successor*
S_2 is a QSIM state successor of S_1. These edges represent the temporal evolution of the situation.
4. $\langle \mathrm{S}, \Delta \rangle$ *transition*
S is a transition state that is crossing the boundary of the applicability of its QDE. Δ is a description of the situation at the same time as S, after the transition.

Each path from the root to a leaf of the graph describes a possible temporal evolution of the physical situation being modeled: a sequence of models, and within each model, a sequence of qualitative states. A path from the root of the graph to a final state describes a complete qualitative behavior of the scenario.

Table 1 provides a pseudo-code description of the QPC algorithm. The main steps are:

Refine the incomplete description Δ into a set of complete descriptions Δ_r of the same situation. The refinement operation returns a set containing all complete databases that are consistent with Δ. An edge linking Δ to each of its refinements is added to the graph.

```
define QPC(Δ)
  forall Δ_r in refinements(Δ)
    collect edge(Δ, Δ_r)
    B = QSIM (qde(Δ_r), initial_values(Δ_r))
    forall b in B
      collect edge(Δ_r, initial_state(b))
      if transition(final_state(b))
        then
          Δ_t = entities(Δ_r) ∪ conditions(final_state(b)) ∪ DT
          collect edge(final_state(b), Δ_t)
          QPC(Δ_t)
```

Table 1: Pseudo-code description of the QPC algorithm.

A complete database Δ_r defines a unique QDE and set of initial conditions which are provided to QSIM. QSIM produces a set B including all possible behaviors consistent with the QDE and initial conditions.

Each behavior b in B is then processed. An edge from Δ_r to the initial state of b is added to the graph. The first state is already linked to its successor states by QSIM, so no state to state edges need be explicitly added. If the final state of b is a transition state, then a new description of the situation after the transition must be constructed. The transition database Δ_t is constructed from the entities of Δ_r, the quantity conditions derived from the transition state, and the domain theory. An edge from the final state of b to Δ_t is added to the graph.

QPC recurses on the transition database Δ_t to construct all possible behaviors of the situation following the transition.

Refinement constructs the set of all complete databases consistent with and rooted in an initial database Δ. This is accomplished by creating a set of databases in which the ambiguous instances in Δ are no longer ambiguous. These databases are referred to as the children of Δ. A child Δ_c of Δ is formed by adding conditions (ground in Δ) so that every ambiguous instance i in Δ can be proven to be either *active* or *inactive* in Δ_c. The children of Δ contain all possible partitions of the ambiguous instances in Δ into active and inactive. Note that a child need not be complete, since new entities may be introduced by the new active instances, resulting in new (ambiguous) instances. Thus, each child must be recursively refined.

In the worst case, computing the set of refinements is exponential in the number of ambiguous instances. This occurs when the ambiguous conditions are completely independent. The actual cost tends to be considerably lower, because the ambiguous conditions tend to be interdependent and often depend on certain key pieces of unknown information. Sets of conditions may be inconsistent, thus eliminating some children from further consideration.

Computing the QDE, initial, and boundary conditions from a complete database is fairly straightforward. The equations and initial conditions are simply collected from the database. The boundary conditions are the union of the operating conditions of the inactive instances (if one becomes true, then there may be new consequences) and the negated operating conditions of the active instances (if one becomes false then an instance becomes inactive, changing the equations). As we describe below, some work is necessary to represent inequalities in the equations and boundary conditions, and algebraic inference is performed to strengthen the QDE.

Each path from a complete database to a transition state describes a segment of behavior over a *closed temporal interval* of non-zero duration. Consequently, each QDE holds over a closed temporal interval. There are several reasons for this. There is intuitively pleasing that "physical processes" start and stop at a well-defined point in time. Most importantly, it allows standard analysis to be used uniformly to understand QPC's predictions. Practically, many simulators, including QSIM, require an initial time point and stop at a final time point. This approach is very simple — there is only one type of temporal interval that must be considered. The alternatives are to allow arbitrary partitions of the time line or to restrict operating conditions to be (non-)strict relations (e.g. partition the time-line into intervals closed on the left and open on the right by using only non-strict relations). Because all QPC variables are continuous, the cost is minimal.

The consequence is that the activity of every model fragment instance is pushed to its boundaries. Thus, if an instance is active when $x > 0$, QPC considers it to be active when $x = 0$ *and is increasing*. Note that there is a distinction between strict and non-strict relations. The operating condition $x \geq 0$ holds even if x remains 0.

Soundness

QPC is sound if every possible behavior of the system being modeled is covered by one of QPC's predictions. Time is dense and modeled by the reals. For convenience, we take 0 to be the starting time for all scenarios. We define a *run*, r, to be an assignment of real values to variables for all $t > 0$, that satisfies certain restrictions: For some prefix of the run, a variable may have the value *undefined*. Each variable in the run is continuous in magnitude starting from the first time that it is not undefined. A variable must have a finite number of critical points over any finite temporal interval.

Each possible precise behavior of a scenario is a run. A run satisfies a qualitative behavior if it can be partitioned into a sequence of alternating points and closed intervals matching the states in the qualitative behavior, such that the real values assigned to variables in the run match the qualitative values in the state. A run satisfies the output of QPC if it satisfies a complete qualitative behavior of the scenario.

QPC is sound if, given axiom sets S and D induced by the scenario and domain theory,

$$\forall r\, [r \models S \cup D] \;\Rightarrow\; [r \models \; QPC(S \cup D)]$$

That is, if a run r satisfies the scenario and domain theory, then r must also satisfy the output of QPC given the same scenario and domain theory. The proof of soundness is given in (Farquhar 1993). It is by induction on the path from the root to a terminal state in the graph. We show that the QPC algorithm constructs the unique initial value problem that must follow from the initial complete database. The QSIM soundness proof can then be invoked to show that an initial prefix of the run will satisfy one of the predicted behaviors. The inductive step is similar.

The proof requires the syntactic restrictions in the QPC modeling language and the induced axioms as well as the assumption that the entities can be *stratified* such that the proof of existence of each entity is rooted in those initially specified in the scenario.

Incompleteness

Incompleteness arises in two very different ways in QPC. First, QPC allows the knowledge represented in both the domain theory and scenario specification to be incomplete. This is an important benefit of QPC and related qualitative modeling and simulation methods. Second, QPC's inference methods and, consequently, predictions are incomplete. The hybrid QPC architecture enables the sources of incomplete inference to be readily identified.

Representing Incomplete Knowledge

QPC enables the domain theory author to specify partial constraints on the relations between variables via language constructs such as monotonic function constraints (`M+`) and qualitative proportionalities (`Q+`), also known as influences. QPC also allows structural relationships to be incompletely axiomatized. The refinement procedure will ensure that the intended model is constructed even in this case.

The scenario specification may be incomplete in several ways. The initial conditions or relations may be incomplete, in which case the refinement operation will ensure that the intended model is constructed. The entities may also be incomplete, however, if the domain theory does not contain sufficient entity definitions to entail the existence of the remaining entities, then there is little that QPC can do.

Incomplete Inference

The syntactic restrictions on the QPC language ensure that it can be effectively implemented by a simple forward application of modus ponens (the proofs of correctness do not even require full transitive closure under modus ponens). This is guaranteed because there is no embedded disjunction or quantification, existential quantification appears only on the right hand side of implications, and the existential quantifications can be stratified. In fact, if it were not for order relations between quantities, closure under modus ponens would be complete for QPC. The introduction of order relations, however, destroys this property. General reasoning about order relations over functions is, unfortunately, undecidable. Nonetheless, this means that *the sole source of incompleteness in QPC's model building results from inference about order relations*.

This is an important observation. It means that we can focus our attention on strengthening QPC's ability to do inference over order relations. In fitting with QPC's hybrid architecture, the natural approach is to use a special purpose inequality reasoner such as Bounder (Sacks 1987), the quantity lattice (Simmons 1986), or even QSIM, as is done by DME (Low & Iwasaki 1992).

QSIM and other qualitative simulation algorithms are incomplete. Some of this is inherent in the accurate, but imprecise, representation (Struß 1990). But qualitative simulators are also incomplete due to their reliance on local constraint satisfaction and continuity. Recent work has substantially improved the strength of predictions by exploiting non-local constraints and more sophisticated mathematics (Lee & Kuipers 1993). QPC effortlessly takes advantage of such advances in qualitative simulation.

Information can also be lost at transitions, resulting in incompleteness. For instance, QPC currently requires variables to be everywhere continuous in magnitude. If a variable is continuous in its first or higher derivatives, some information is lost. QPC could be easily extended to allow for higher order continuities to be expressed.

QSIM and the QDE Representation

The QDE representation provides only limited support for inequalities, which are extremely common in QPC domain theories. Non-strict inequalities between variables and landmarks can be easily represented by constraining the quantity space of the variable. Strict inequalities are handled by using QSIM filters to infer that a state is inconsistent when a variable reaches its limiting values. Inequalities between variables are handled by introducing a new variable to represent their difference, and transforming the original relation to one between the difference variable and zero.

QSIM boundary conditions are relations between variables and landmarks, whereas QPC allows inequalities between variables. The difference variable technique is used here as well.

QPC pushes the activity of all model fragments to a closed interval. As a consequence, a model may start with a variable on a boundary and require that it move off of the boundary into the active region immediately. This is ensured by providing a QSIM filter that applies to the states that describe the initial open interval.

QSIM does only minimal algebraic inference. For example, given a QDE containing the constraints `(add x y z)` and `(add x y w)`, QSIM does not require `z` and `w` to be equal. When QDEs are constructed by hand, this sort of incomplete inference is rarely problematic. When the QDEs are automatically generated, however, it can be a substantial hindrance. QPC deals with this problem by performing *algebraic inference* on each QDE to add implicit constraints (such as `(equal z w)`) and eliminate redundant constraints (such as the second `add` after the `equal` constraint has been included). This step has turned out to be essential in getting QSIM to solve the large sets of equations automatically generated by QPC.

The version of QSIM used by QPC is extended to generate an *attainable envisionment*, which is a rooted graph of states, rather than the tree QSIM has traditionally used. By turning off QSIM's automatic landmark generation facilities, the attainable envisionment is guaranteed to be finite (since each variable has a finite domain of qualitative values).

The complete (perhaps infinite) set of possible behaviors is compactly represented by all paths starting in an initial state. Note that the QPC algorithm only requires the set of final *transition* states to be identified; the behavior paths need not be identified by QPC at all. Thus, the attainable envisionment is exactly the right level of description for QPC.

Examples and Applications

This section discusses a simple example and two more substantial applications of QPC. The example allows QPC to be compared to GIZMO (Forbus 1984) and QPE (Forbus 1990) and shows the advantage of QPC's hybrid architecture. In the botany application, QPC produces answers to questions that might arise in a tutoring application. The chemical engineering application shows that QPC is capable of modeling large complex systems with hundreds of variables from numerous automatically generated scenarios.

Heat Transfer

Figure 1 describes a simple heat transfer problem that appeared in (Forbus 1984) and was also used to compare GIZMO with QPE in (Forbus 1990). This is a good benchmark problem for qualitative simulators because the underconstained domain theory requires a surprisingly large number of states and behaviors, even though there is only one final state – thermal equilibrium.

Note that `heat-flow` is not conditioned on a non-zero temperature difference, as is typical in QP theory definitions. This reflects an important philosophical difference. QPC model fragment definitions reflect the underlying equations, not necessarily cognitively useful descriptions. The QP theory descriptions can be extracted from the QPC descriptions.

QPC solves the heat transfer problem with initial conditions in about a minute. There are 327 unique states and 691 branches in the behavior tree. Without initial conditions, the problem takes about 30 minutes to identify 396 unique states. Importantly, in both cases there is only one final state, in which the system is in thermal equilibrium. Note that the actual number of states generated in the two cases is not too different. This is because the initial conditions to not provide strong constraints on the possible states of the system. The difference in runtime is due entirely to simulation (the QDEs are identical for both scenarios). The constraint satisfaction algorithm currently employed by QSIM is not optimized for problems that require many initial states to be generated.

We may contrast these results with those reported in (Forbus 1990). The runtime for GIZMO was 60 times longer than QPC. Even allowing a factor of 10 in machine speeds, QPC would still be 6 times faster at generating an attainable envisionment. Although QPE is designed for performing total envisionments (and is unable to generate an attainable one), Forbus reported that this problem "had run for hours and generated over 5200 situations before the machine was halted manually"(Forbus 1990, page 233).

What happens to the stomates when K+ (potassium) increases in the guard cells? (They open)
What happens to the K+ in guard cells when ABA (abscisic acid) there increases? (It decreases)
What happens to the ABA level in the guard cells when turgor pressure drops in the leaves? (It accumulates)
What happens to the stomates when turgor pressure drops in the leaves? (They close)
What happens to turgor pressure in the leaves as the soil dries out? (It drops)

Table 2: Partial list of botany questions that have been solved by QPC.

QPC's hybrid architecture and use of a special purpose simulation engine is clearly beneficial in this benchmark problem.

Botany

In the context of the Botany Knowledge Base project at the University of Texas (Porter *et al.* 1988), Rickel has constructed a QPC domain theory for plant physiology (Rickel & Porter 1994). The textbook knowledge encoded in the Botany Knowledge Base is largely qualitative because quantitative details vary greatly among different plants and often are not known.

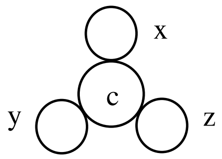
The domain theory covers the uptake, transport, and loss of water in the plant; the plant's mechanisms for dealing with water stress (synthesis, consumption and transport of the ABA hormone, and movement of potassium ions and water into and out of guard cells that regulate water loss); and the production (via photosynthesis), consumption (via respiration), and transport (via the phloem) of carbohydrate in the plant. The descriptions of these processes are relatively simple, but they are fairly standard and reasonable high-level descriptions, and they are based both on textbook knowledge and consultations with domain experts.

Over a dozen questions have been formulated and solved using QPC. Each corresponds to a scenario; the answer is determined by examining the resulting behavior predictions. Table 2 summarizes several of the running examples. For each of them, QPC predicts a single behavior which matches the correct answer. For example, in the first question, the variable representing the opening of the stomates increases from its initial value until a new equilibrium is reached. It is worth noting that the techniques originally employed by QSIM actually give rise to infinite sets of possible behaviors. It is only since the development of techniques to eliminate unnecessary distinctions (Clancy & Kuipers 1993), that they have become tractable. This shows a key advantage of QPC's architecture: advances in simulation techniques are directly exploited.

Chemical Engineering

Catino and Ungar have been using QPC to support hazard identification in chemical processing plants (Catino 1993; Catino & Ungar 1994). QPC is an appropriate analysis

```
(defScenario heat-transfer
   :entities ((c :type objects)
              (x :type objects)
              (y :type objects)
              (z :type objects))
   :relations ((connected c x)
               (connected c y)
               (connected c z))
   :initial-conditions
   ((> (temp c) (temp x))
    (> (temp c) (temp y))
    (> (temp c) (temp z))))
```




```
(defRelation connected (objects objects))
(defQuantity temp (objects))
(defQuantity heat (objects))
(defModelFragment physob
  :participants ((a :type objects))
  :consequences ((> (temp a) zero)
                 (q+0 (temp a) (heat a))))
(defModelFragment heat-flow
    :participants ((a :type objects)
                   (b :type objects))
    :structural-conditions ((connected a b))
    :quantities (delta-t flow-rate)
    :consequences
    ((add (temp b) (delta-t heat-flow) (temp a))
     (Q+0 (flow-rate heat-flow) (delta-t heat-flow))
     (I- (heat a) (delta-t heat-flow))
     (I+ (heat b) (delta-t heat-flow))))
```

Figure 1: A simple heat transfer problem. In the `heat-transfer` scenario, there are three objects `x`, `y`, and `z` in thermal contact with a hotter center object `c`.

tool for this task and domain for several reasons. The phenomenon based compositional approach naturally describes chemical processes, which are not component-oriented, but occur whenever the requisite substances are present and conditions hold. The hazard identification task requires explicitly examining all possible outcomes to determine if undesirable ones may occur. In addition to unexpected reactions, this may also include unexpected changes in the topology of the plant (e.g. a valve incorrectly opened, or a leak in a heat exchanger). Although approximate quantitative information is available for describing the normal operation of a chemical plant, hazard identification must consider what will occur when such constraints are violated. QPC provides this ability.

The domain theory covers mass flow, heat flow, chemical reaction, vaporization, and liquid-vapor equilibrium. It is capable of modeling controllers, phases consisting of any number of chemical species, thermal and mass fraction effects of mass flow, and moving landmark values as in a changing dew point (Catino 1993).

Catino and Ungar have conducted a substantial hazard identification case study of nitric acid production plant (Catino & Ungar 1994). Their system begins with a scenario describing the design case (normal operation) of the plant and then systematically perturbs this specification to generate fault scenarios. The fault scenarios are simulated by QPC and checked for the presence of hazards. Over twenty scenarios were generated and checked for the nitric acid plant. These models are the largest ones that have currently been solved by QPC. The scenario definition for the design case specifies 41 initial entities in addition to various landmarks, initial conditions, and modeling assumptions. The model built by QPC consists of 81 active model fragments, 317 variables, and 387 constraints. It currently takes about an hour and a half to build and solve and results in a single steady state that describes how the plant operates when it is functioning as designed (It is a full dynamic model whose solution is a steady state in that the derivatives of the state variables happen to be zero. Under perturbed initial conditions, more interesting dynamic behavior results).

Related Work

QPC's architecture, algorithms, and soundness guarantee distinguish it from other qualitative modeling systems.

Both GIZMO (Forbus 1984) and QPE (Forbus 1990) compose qualitative models from a scenario description and a domain theory of model fragments, but they construct the complete set of model fragment instances, variables, inequalities, and so on, that *might ever appear in a total envisionment* of the scenario. After constructing this *scenario elaboration*, GIZMO constructs a set of initial situations, which includes a status for every entity, instance, quantity condition, and influenced variable in the elaboration, it the uses *limit analysis* to construct possible successors. QPE constructs a total envisionment; It is not able to make use of any initial conditions.

QPC does not elaborate the scenario. QPC fully exploits the initial conditions and the preconditions of model fragment and entity definitions. The sets of instances and entities are incrementally constructed. QPC decomposes the construction of initial situations into two phases. Refinement effectively assigns a status to the instances; QSIM assigns values to the variables. Both of these operations are very focused, and apply only to the instances and variables that are relevant to the situation.

The device modeling environment, DME (Low & Iwasaki 1992), is the system closest to QPC. In its qualitative prediction mode it also uses QSIM to generate state successors, but does not make full use of QSIM. In particular, only the direct successors of a state are generated and the results asserted immediately back into the knowledge base. This makes prediction much more expensive and means that qualitative simulation techniques such as chatter elimination or attainable envisionment cannot be readily used. DME typically expects more interaction from the user than QPC, which is fully automatic. This is in keeping with DME's goal as a modeling *environment*. DME also provides sophisticated explanation facilities (Gautier & Gruber 1993).

Conclusion

QPC provides an expressive language and powerful techniques for reasoning about physical systems without a precise domain theory or a complete specification of the system. QPC's hybrid architecture exploits independent advances in model building and qualitative simulation techniques and supports formal analysis. The incremental algorithm takes advantage of knowledge about a system's initial conditions when available.

This claim is supported by substantial applications. In particular, the nitric acid production plant application (Catino & Ungar 1994) shows that QPC can model and simulate complex systems requiring hundreds of variables and constraints from automatically generated scenario descriptions.

Acknowledgements

I would like to thank Ben Kuipers for providing the initial motivation to work on this problem; Jimi Crawford for taking the first steps with me; and especially Giorgio Brajnik, Cathy Catino, Raman Rajagopalan, and Jeff Rickel, who were the patient first users of QPC. Their feedback provided a strong impetus for QPC and helped to ensure that it is capable of solving diverse problems.

References

Catino, C., and Ungar, L. 1994. A model-based approach to automated hazard identification of chemical plants. Submitted to AIChE Journal.

Catino, C. 1993. *Automated Modeling of Chemical Plants with Application to Hazard and Operability Studies*. Ph.D. Dissertation, Department of Chemical Engeneering, University of Pennsylvania, Philadelphia, PA.

Clancy, D. J., and Kuipers, B. J. 1993. Behavior abstraction for tractable simulation. In *Proceedings of the Seventh International Workshop on Qualitative Reasoning about Physical Systems*.

Crawford, J. M., and Kuipers, B. J. 1991. ALL: formalizing acess-limited reasoning. In Sowa, J., ed., *Principles of Semantic Networks*. San Mateo, CA: Morgan Kaufman. 299–330.

Crawford, J.; Farquhar, A.; and Kuipers, B. 1990. QPC: A compiler from physical models into qualitative differential equations. In *Proceedings of the Eighth National Conference on Artificial Intelligence*.

Farquhar, A., and Brajnik, G. 1994. A semi-quantitiative physics compiler. In *Proceedings of the Eighth International Workshop on Qualitative Reasoning*.

Farquhar, A.; Kuipers, B.; Rickel, J.; Throop, D.; and The Qualitative Reasoning Group. 1991. QSIM: The program and its use. Technical Report AI90–123, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, Texas 78712.

Farquhar, A. 1993. *Automated Modeling of Physical Systems in the Presence of Incomplete Knowledge*. Ph.D. Dissertation, The University of Texas at Austin. Available as technical report UT-AI-93-207.

Forbus, K. 1984. *Qualitative Process Theory*. Ph.D. Dissertation, MIT AI Lab, Cambridge, MA.

Forbus, K. 1990. The qualitative process engine. In Weld, D., and de Kleer, J., eds., *Readings in Qualitative Reasoning about Physical Systems*. Los Altos, CA: Morgan Kaufman.

Gautier, P. O., and Gruber, T. R. 1993. Generating explanations of device behavior using compositional modeling and causal ordering. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*. AAAI Press/The MIT Press.

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

Lee, W. W., and Kuipers, B. 1993. A qualitative method to construct phase portraits. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*.

Low, C. M., and Iwasaki, Y. 1992. Device modelling environment: an interactive environment for modelling device behavior. *Intelligent Systems Engineering* 1(2):115–145.

Porter, B.; Lester, J.; Murray, K.; Pittman, K.; Souther, A.; Acker, L.; and Jones, T. 1988. AI research in the context of a multifunctional knowledge base: The botany knowledge base project. Technical Report Technical Report AI88-88, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, Texas 78712.

Rajagopalan, R. 1994. A model for integrated qualitative spatial and dynamic reasoning about physical systems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*.

Rickel, J., and Porter, B. 1994. Automated modeling for answering prediction questions: Selecting the time scale and system boundary. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*.

Sacks, E. 1987. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 649–654. Los Altos, CA: Morgan Kaufmann.

Simmons, R. 1986. "Commonsense" arithmetic reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 118–124.

Struß, P. 1990. Problems of interval-based qualitative reasoning. In Weld, D., and de Kleer, J., eds., *Qualitative Reasoning about Physical Systems*. Morgan Kaufmann. 288–305.