IQE: An Incremental Qualitative Envisioner

Dennis DeCoste and **John W. Collins**

Qualitative Reasoning Group
Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, Illinois 60201
email: decoste@aristotle.ils.nwu.edu

Qualitative Reasoning Group
Beckman Institute
University of Illinois
605 N. Mathews St.
Urbana, Illinois 61801
email: jcollins@cs.uiuc.edu

Abstract

The finite representation of infinite behaviors provided by total envisionments has been shown to be useful for tasks such as model development, explanation, and monitoring. Unfortunately, generating total envisionments of complex systems can be intractable, or at least excessive for highly-focused tasks. Such shortcomings have encouraged some researchers to favor alternative schemes, such as attainable envisioning and history generation. However, we argue in this paper for an incremental means of envisioning that can realize many of the practical advantages of total envisioning while supporting more-focused search to address issues of tractability and relevance. In this paper, we describe our theory of incremental envisioning and compare it with QPC [3] and QPE [9]. We emphasize how IQE attempts to reason about states at low (i.e., abstract) levels of detail when that is sufficient for the overall task. Such reasoning helps avoid the inefficiency of making state distinctions that are irrelevant for the overall task – a problem common among most other current qualitative simulators, including QPC and QPE.

1 Introduction

A *total envisionment* [5,10], indicating all possible transitions among all possible qualitative states, provides a finite representation of infinite behaviors that can be useful, for example, in explanation and monitoring tasks [8,6]. However, total envisioners, such as QPE [9], suffer the up-front cost of generating all possible states before any transitions can be generated. Such costs can make total envisioning intractable for complex systems.

QPC [3] provides an alternative to QPE. Like QPE, QPC takes advantage of the modeling facilities of Qualitative Process Theory (QPT) [10] to automate the task of applying a library of domain theories to a particular scenario. QPC compiles applicable qualitative process descriptions into qualitative differential equations (QDE's) suitable for simulation by QSIM [11]. QPC employs *incremental model-building* to adjust the current set of QDE's when the system changes operating regions during simulation. Thus, QPC avoids QPE's up-front cost of total envisioning by computing attainable envisionments – generating a history tree from an initial state, considering only those states which arise during simulation.

In this paper, we argue for another alternative, called *incremental envisioning*, which incrementally refines partial envisionment representations towards more-complete ones. This approach is embodied in the LISP program IQE. Like QPE, IQE employs an assumption-based truth maintenance system (ATMS) [4,1,7] to efficiently cache inferences across multiple states. Yet, like QPC, IQE avoids the burden of total envisioning; that is, it avoids ATMS interpretation construction to compute the total set of possible states (as QPE does). The way in which IQE uses an ATMS also facilitates efficient context-switching to reason about alternative structural abstractions and modelling assumptions.

IQE serves the role of a sort of *envisionment maintenance system* that is invoked by a problem solver performing some higher-level qualitative reasoning task, such as a diagnostic engine. Some of the facilities provided by IQE are qualitative reasoning analogs of those of an ATMS. For example, whereas an ATMS answers queries such as "Is this proposition true in this context?", IQE answers queries such as "Is this proposition true in this (partial) state?". However, IQE can also answer more general forms of such queries, like: "Is this proposition true in some refinement of this (partial) state?". IQE can also answer temporal queries, such as "Are these propositions all true in any or all of the possible next (or previous) states of this (partial) state?" and "Are there any consistent paths of states between these initial and goal states?". Furthermore, IQE can incrementally adjust its simulation results to consider alternative goal states (which problem solvers often provide), whereas simulators like QPC and QPE typically do not.

This paper describes the design of IQE and discusses the facilities it provides for the problem solver.

2 The design of IQE

There are three key characteristics of IQE's design, all of which facilitate its efficient reasoning over all states at all levels of detail: 1) extensive use of declarative qualitative knowledge, 2) reasoning about states in terms of ATMS environments, and 3) incremental

temporal reasoning.

2.1 Declarative Representation of Qualitative Knowledge

One of the goals driving the development of IQE has been to develop and utilize a declarative knowledge base of the constraints underlying qualitative reasoning. In some cases, efficiency constraints have mandated a more procedural implementation – for example, enforcing transitivity efficiently requires special purpose code which avoids the introduction of uninteresting inequality relations. Still, most of IQE's qualitative reasoning is implemented by a single file of antecedent rules. These rules take the form of logical implications augmented with closed world assumptions, which are discussed further in Section 2.1.3.

These rules enforce various inequality constraints, resolve influences on quantities, and enforce temporal constraints. Because these rules are set up to infer consequences given the minimal required antecedents, they can be applied to even very partial state descriptions. Although the current set of rules was developed with QPT in mind, IQE should also be able to handle other ontologies if this small set of rules is modified appropriately.

2.1.1 Inequalities

Inequalities are central to qualitative reasoning in IQE. An inequality exists between two numbers whose ordering is relevant to the reasoning process. In IQE, a number may be either a quantity `Q` or its time derivative `(D Q)`. IQE does not distinguish between variable and constant quantities; in particular, `ZERO` is treated as an ordinary quantity. Inequalities are limited to involve either two quantities or two derivatives; mixed inequalities are not allowed. We have found this sufficient for our current QPT models. This greatly simplifies the rules for reasoning about inequalities – particularly rules for temporal reasoning.

Inequalities are represented in IQE using a single relation: $\geq$. This representation scheme was chosen for its simplicity and expressive power. It avoids the canonical ordering problem of other schemes, since both orderings of two related numbers are represented. It simplifies the rules used to implement inequality reasoning, since there are fewer cases to handle. It also reduces the number of ATMS nodes needed per inequality from four pairs (in QPE) down to two pairs. Finally, the justifications required to enforce taxonomy among the nodes are replaced by a single clause, requiring $\geq$ to be true for one of the two possible orderings of two related numbers, whenever both numbers exist. Table 1 demonstrates how this scheme represents the possible inequality relations.

2.1.2 Qualitative Algebra

Like QPE, IQE supports various algebraic expressions, including equivalence, binary sums, differences, products and ratios. Quantities in IQE can be constrained by the following predicates: `(positive ?Q)`, `(negative ?Q)`, `(zerop ?Q)`, `(increasing ?Q)`, `(decreasing ?Q)`, and `(constant ?Q)`. These are represented internally by the appropriate inequality with `ZERO`.

Relation	Representation using $\geq$
$Q_1 > Q_2$	$Q_2 \not\geq Q_1$
$Q_1 = Q_2$	$Q_1 \geq Q_2 \wedge Q_2 \geq Q_1$
$Q_1 < Q_2$	$Q_1 \not\geq Q_2$
$Q_1 \geq Q_2$	$Q_1 \geq Q_2$
$Q_1 \leq Q_2$	$Q_2 \geq Q_1$

Table 1: Concise inequality representations using $\geq$

Just as two quantities may be related by an inequality, two inequalities may be related. The predicate `Same-Relation` takes two pairs of numbers as arguments, and enforces that the inequality relations for the two pairs are always the same. That is, knowledge of one inequality relation is imposed on the other pair. This predicate is used to implement the *Correspondences* of QPT, as well as sums and differences, as shown in Figure 1.

2.1.3 Causal Influences

QPT defines two types of causal influences: direct influences and qualitative proportionalities (or `Qprops`). They are similar in that both allow for an arbitrary number of influences to affect a quantity, and both require some mechanism for disambiguating multiple competing influences. They differ in that for a direct influence, the mere presence of a non-zero influencer (e.g. flow-rate) causes the influenced quantity to change; while for a `Qprop`, the rate of change of the influencer contributes to the rate of change of the influenced quantity. The direct influencer is also more tightly constrained, in that the derivative of the influenced quantity is equal to the net sum of its influences. These influences are implemented in `IQE` using the rules shown in Figure 2. In general, if a quantity is influenced in one direction and not the other, then it will change in that direction. If there are no influences on a quantity, then it must be constant. Influences in both directions result in ambiguity concerning direction of change.

Inferring that a quantity is influenced in one direction or the other simply requires finding an instance of an influence in that direction. But inferring that a quantity is not influenced is not so easy; it requires making a *closed-world assumption* about the possible influencers for the quantity, and then showing that none of the known influencers are in effect. This is done by specifying that all influence-related predicates are to be *closed*, which enforces a kind of "negation by failure" for the predicates. By including an explicit closed-world assumption in the resulting justification, it is possible to add to the set of possible influencers without having to start from scratch.

2.1.4 Temporal Constraints

Qualitative reasoning derives much of its power from a few temporal constraints. Change is assumed to be continuous, and must be consistent with known temporal derivatives.

```
(<== (or (>= ?Q1 ?Q2) (>= ?Q2 ?Q1)) (Inequality ?Q1 ?Q2))
;;; Same-Relation, Same-Sign, Opposite-Sign:
(<== (Same-Relation (?Q1 ZERO) (?Q2 ZERO)) (Same-Sign ?Q1 ?Q2))
(<== (Same-Relation (?Q1 ZERO) (ZERO ?Q2)) (Opposite-Sign ?Q1 ?Q2))
(<== (Same-Relation (?Q1 ?Q2) (?Q3 ?Q4))
     (and (Correspondence (?Q1 ?Q2) (?Q3 ?Q4)) (Qprop+ ?Q1 ?Q3)))
(<== (Same-Relation (?Q1 ?Q2) (?Q4 ?Q3))
     (and (Correspondence (?Q1 ?Q2) (?Q3 ?Q4)) (Qprop- ?Q1 ?Q3)))
(<== (>= ?Q1 ?Q2) (and (Same-Relation (?Q1 ?Q2) (?Q3 ?Q4) (>= ?Q3 ?Q4)))
(<== (>= ?Q2 ?Q1) (and (Same-Relation (?Q1 ?Q2) (?Q3 ?Q4) (>= ?Q4 ?Q3)))
(<== (> ?Q1 ?Q2)  (and (Same-Relation (?Q1 ?Q2) (?Q3 ?Q4) (> ?Q3 ?Q4)))
(<== (> ?Q2 ?Q1)  (and (Same-Relation (?Q1 ?Q2) (?Q3 ?Q4) (> ?Q4 ?Q3))))
;;; Q:=
(<== (and (= ?Q1 ?Q2) (= (D ?Q1) (D ?Q2)) (Qprop+ ?Q1 ?Q2)) (Q:= ?Q1 ?Q2))
;;; Sums and Differences:
(<== (=+ ?sum ?Q1 ?Q2) (or (Q= ?sum (+ ?Q1 ?Q2)) (Q= ?Q1 (- ?sum ?Q2))))
(<== (and (Qprop+ ?sum ?Q1)  (Qprop+ ?sum ?Q2))  (Q= ?sum (+ ?Q1 ?Q2)))
(<== (and (Qprop+ ?diff ?Q1) (Qprop- ?diff ?Q2)) (Q= ?diff (- ?Q1 ?Q2)))
(<== (Same-Relation (?sum ?N1) (?N2 ZERO)) (=+ ?sum ?N1 ?N2))
(<== (Same-Relation ((D ?sum) (D ?N1)) ((D ?N2) (D ZERO))) (=+ ?sum ?N1 ?N2))
(<== (Same-Relation (?sum ?N2) (?N1 ZERO)) (=+ ?sum ?N1 ?N2))
(<== (Same-Relation ((D ?sum) (D ?N2)) ((D ?N1) (D ZERO))) (=+ ?sum ?N1 ?N2))
;;; Products and Ratios:
(<== (Qprop+ ?prod ?M1) (and (Q= ?prod (* ?M1 ?M2)) (Positive ?M2)))
(<== (Qprop- ?prod ?M1) (and (Q= ?prod (* ?M1 ?M2)) (Negative ?M2)))
(<== (Qprop+ ?prod ?M2) (and (Q= ?prod (* ?M1 ?M2)) (Positive ?M1)))
(<== (Qprop- ?prod ?M2) (and (Q= ?prod (* ?M1 ?M2)) (Negative ?M1))))
(<== (Qprop+ ?ratio ?N) (and (Q= ?ratio (/ ?N ?D)) (Positive ?D)))
(<== (Qprop- ?ratio ?N) (and (Q= ?ratio (/ ?N ?D)) (Negative ?D)))
(<== (Qprop+ ?ratio ?D) (and (Q= ?ratio (/ ?N ?D)) (Negative ?N)))
(<== (Qprop- ?ratio ?D) (and (Q= ?ratio (/ ?N ?D)) (Positive ?N)))
(<== (=* ?prod ?M1 ?M2) (or (Q= ?prod (* ?M1 ?M2)) (Q= ?M1 (/ ?prod ?M2))))
(<== (Same-Sign ?prod ?m1) (and (=* ?prod ?m1 ?m2) (Positive ?m2)))
(<== (Same-Sign ?prod ?m2) (and (=* ?prod ?m1 ?m2) (Positive ?m1)))
(<== (Opposite-Sign ?prod ?m1) (and (=* ?prod ?m1 ?m2) (Negative ?m2)))
(<== (Opposite-Sign ?prod ?m2) (and (=* ?prod ?m1 ?m2) (Negative ?m1)))
(<== (Zerop ?prod) (and (=* ?prod ?m1 ?m2) (Zerop ?m2)))
(<== (Zerop ?prod) (and (=* ?prod ?m1 ?m2) (Zerop ?m1)))
```

Figure 1: Qualitative algebra rules

```
;;; Directions of Change:
(<== (Increasing ?Q) (and (Influenced ?Q :+) (not (Influenced ?Q :-))))
(<== (Decreasing ?Q) (and (not (Influenced ?Q :+)) (Influenced ?Q :-)))
(<== (Constant ?Q)   (and (not (Influenced ?Q :+)) (not (Influenced ?Q :-))))
(<== (Ambiguous ?Q)  (and (Influenced ?Q :+) (Influenced ?Q :-)))
;;; Influenced by whom?
(<== (Influenced ?Q1 ?dir) (Influenced-by ?Q1 ?Q2 ?dir))
;;; Direct Influencers:
(<== (Influenced-by ?Q ?rate :+) (and (I+ ?Q ?rate) (Positive ?rate)))
(<== (Influenced-by ?Q ?rate :-) (and (I+ ?Q ?rate) (Negative ?rate)))
(<== (Influenced-by ?Q ?rate :+) (and (I- ?Q ?rate) (Negative ?rate)))
(<== (Influenced-by ?Q ?rate :-) (and (I- ?Q ?rate) (Positive ?rate)))
(<== (Quantity (Net-Influence ?Q)) (or (I+ ?Q ?rate) (I- ?Q ?rate)))
(<== (Qprop+ (Net-Influence ?Q) ?rate) (I+ ?Q ?rate))
(<== (Qprop- (Net-Influence ?Q) ?rate) (I- ?Q ?rate))
(<== (Same-Sign (D ?Q) (Net-Influence ?Q)) (Quantity (Net-Influence ?Q)))
;;; Indirect Influencers:
(<== (Influenced-by ?Q1 (D ?Q2) :+) (and (Qprop+ ?Q1 ?Q2) (Increasing ?Q2)))
(<== (Influenced-by ?Q1 (D ?Q2) :+) (and (Qprop- ?Q1 ?Q2) (Decreasing ?Q2)))
(<== (Influenced-by ?Q1 (D ?Q2) :-) (and (Qprop+ ?Q1 ?Q2) (Decreasing ?Q2)))
(<== (Influenced-by ?Q1 (D ?Q2) :-) (and (Qprop- ?Q1 ?Q2) (Increasing ?Q2)))
```

Figure 2: Causal influence rules

Unlike other qualitative simulators which compute transitions between complete states, IQE performs temporal reasoning incrementally, requiring only partial state descriptions.

Table 2 summarizes the temporal constraints for two temporally contiguous states, as enforced by IQE. Propositions about the next state are represented internally by wrapping a Next predicate around each proposition. This allows IQE to reason about adjacent pairs of states without requiring an explicit temporal reference for every combination of proposition and state. The use of these constraints for incremental temporal reasoning is described in Section 2.3.

2.2 Representing States with ATMS Environments

With P_M representing the set of all propositions mentioned in the qualitative model:

Definition 2.1 (Partial State) *Each* partial state S *indicates a particular subset of* P_M, *denoted* $\texttt{Props}(S)$. *The deductive closure of* $\texttt{Props}(S)$ *is denoted* $\texttt{Closure}(S)$.

In order to reason over all partial states, it is sufficient to reason only about those S_i for which $\texttt{Props}(S_i) = \texttt{Closure}(S_i)$. Thus, each partial state S_i that IQE explicitly considers is assigned a unique corresponding ATMS environment, called $\texttt{Env}(S_i)$. This allows IQE to reason efficiently over all partial states simultaneously, since the ATMS caches each inference with the minimal $\texttt{Env}(S_i)$ for which it holds.

Constraint Type	Previous State		Next State
Continuity	$Q_1 > Q_2$	$\Rightarrow$	$Q_1 \not< Q_2$
Divergence	$Q_1 \not< Q_2 \wedge \dot{Q}_1 > \dot{Q}_2$	$\Rightarrow$	$Q_1 > Q_2$
	$Q_1 > Q_2$	$\Leftarrow$	$Q_1 \not< Q_2 \wedge \dot{Q}_1 < \dot{Q}_2$
No Convergence	$Q_1 > Q_2 \wedge \dot{Q}_1 \not< \dot{Q}_2$	$\Rightarrow$	$Q_1 > Q_2$
	$Q_1 > Q_2$	$\Leftarrow$	$Q_1 > Q_2 \wedge \dot{Q}_1 \not> \dot{Q}_2$
Interval Consistency	$Q_1 = Q_2$	$\Rightarrow$	$Non-Converging(Q_1, Q_2)$
	$Converging(Q_1, Q_2)$	$\Leftarrow$	$Q_1 = Q_2$
Duration	$Interval$	$\Leftrightarrow$	$Instant$
	$Q_1 > Q_2 \wedge Instant$	$\Rightarrow$	$Q_1 > Q_2$
	$Q_1 > Q_2$	$\Leftarrow$	$Q_1 > Q_2 \wedge Instant$
	$Q_1 = Q_2 \wedge Interval$	$\Rightarrow$	$Q_1 \not> Q_2$
	$Q_1 \not> Q_2$	$\Leftarrow$	$Q_1 = Q_2 \wedge Interval$

Table 2: Temporal constraint rules

The predicates `Converging` and `Non-Converging` are true for a pair of related quantities if their absolute difference is decreasing or non-decreasing, respectively.

Each ATMS environment indicates a particular, unique set of assumptions. So, to reason about a more complete version of a partial state, `IQE` *refines* the state by making additional assumptions to create a more specific environment:

Definition 2.2 (State Refinement) *Partial state S_i is* refined *to partial state S_j by the subset p of P_M when the deductive closure of (*`Props`*$(S_i) + p$) equals* `Closure`*(S_j).*

The following subsections discuss the two major issues which arise in this scheme: 1) what to assume, and 2) when and how to refine.

2.2.1 Assumptions in `IQE`

In the course of incremental envisioning, `IQE` uses four types of assumptions to define the environments associated with states:

Scenario assumptions indicate the current belief of what entities (i.e., QPT individuals) can exist in the system and how they are structured.

Modelling assumptions represent the approximations and perspectives underlying the domain theory. Modelling assumptions (such as `CONSIDER`'s [8]) provide the flexibility in perspective required for robustness over a broad domain.

Closed-world assumptions represent the belief that the entities currently being considered are in fact the only ones to consider. By making these assumptions explicit, objects may be added to or removed from sets without having to restart simulation from scratch.

State assumptions encode the qualitative state of the system, in terms of the active views and processes, or equivalently, the inequalities on which they are conditioned. These assumptions indicate the current mode of operation of the system.

As an alternative to assuming these propositions individually, a single assumption may be used to represent a set of them, by justifying each proposition by that assumption. The tradeoff is that less information is cached in the ATMS labels, in the event that one of these propositions is no longer believed.

As shown in Figure 3, the ATMS combines these assumptions in a hierarchical fashion (ordered by environment subsumption) to define IQE's *envisionment lattice.* This space is represented by the combination of the ATMS environment lattice and the temporal relations between those environments given by IQE's temporal constraints. The set of currently believed scenario, modelling, and closed-world assumptions defines IQE's *working focus environment.* By default, all environments that IQE provides to the the problem solver will include those working assumptions.

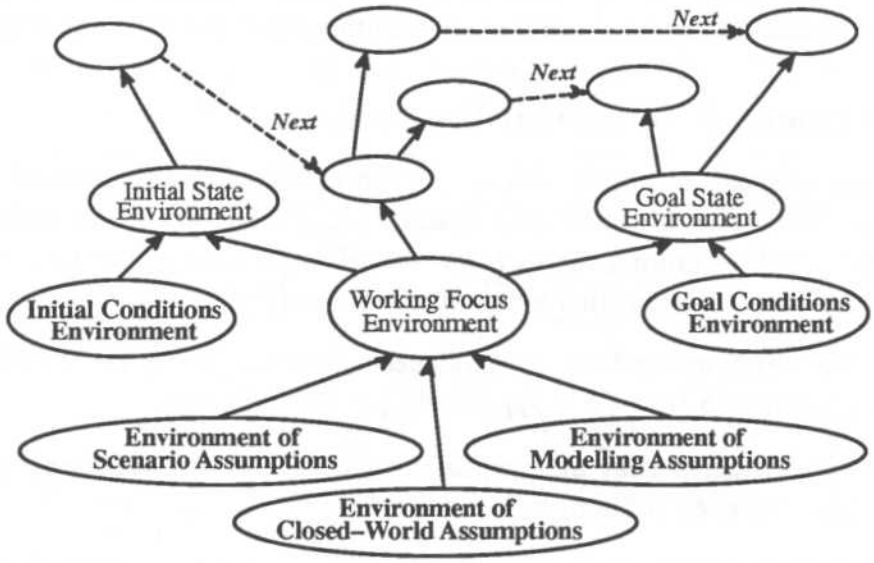

Figure 3: Example IQE envisionment lattice

Closed-world assumptions represent the belief that all members of a set are known. If a new member of a set is found after the set has been closed, then the closed-world assumption for the set is made inconsistent with the new set member. A new closed-world assumption is built to represent the augmented set which includes the new member. Each closed-world assumption for a given set thus represents a different belief about the contents of the set.

As an example, consider the discovery of a new process influencing quantity Q, after the influences on Q have been closed. The existing closed-world assumption representing the set of influences on Q is made inconsistent with the new influence, and a new closed-world assumption is built and used to re-close the set. Closed-world assumption violations are detected by building a rule as each set is closed, which watches for new members of the set and performs the necessary cleanup operations.

2.2.2 Refining a State by Adding State Assumptions

To answer a query by the problem solver IQE may have to refine a state to consider alternative completions of that state to some level of detail. One important class of refinements is that of determining the status of process and view (P/V) instances. Such refinement involves first determining which P/V instances are relevant to the query. IQE considers a P/V instance to be relevant if it could influence the system towards or away from the behavior being queried about, Next, IQE generates a *choice set* of size two for each relevant P/V instance, containing assumptions that the instance is active or not active. Finally, IQE invokes ATMS interpretation construction upon the refined state's environment using the choice sets. The result is the set of maximal consistent environments representing all alternative *P/V refinement* environments. In each of those environments, each relevant P/V instance will either be assumed to be active or assumed to be not active. Those assumptions represent the additional state assumptions used for that refinement.

2.3 Incremental Temporal Reasoning

All of the Next propositions derived from the temporal constraint rules of Table 2 for some state together define the minimal state description of what *must* be true in any next state. The temporal relations among states have several important properties. Each state S has exactly one *successor state*, which we denote as $\mathtt{Succ}(S)$:

Definition 2.3 (Successor State) *The successor state of a state S is the state* $\mathtt{Succ}(S)$ *such that* $\mathtt{Props}(\mathtt{Succ}(S)) = \{f \in \mathcal{P}_M \mid Next\{f\} \in \mathtt{Closure}(S)\}$

Monotonicity requires that all propositions that are true in $\mathtt{Succ}(S)$ are necessarily true in $\mathtt{Succ}(S_i)$, for each refinement S_i of S:

Property 2.1 (Successor-Refinement Confluence) *Given state S and some refinement (of S) S_i, then* $\mathtt{Succ}(S_i)$ *must be a (possibly null) refinement of* $\mathtt{Succ}(S)$.

Thus $\mathtt{Succ}(S)$ represents the intersection of the successor states of all possible refinements of S. There will generally be fewer propositions which are true in $\mathtt{Succ}(S)$ than in S itself. This is because the Next propositions inferred by the temporal constraint rules are only those which *must* be true in the successor state of every possible refinement of S, not all those which *may* be true in some refinement. If the problem solver desires a more detailed description of the successor state of S, IQE must either refine $\mathtt{Succ}(S)$, or refine S and then compute the successor states of the refinements of S.

While the successor state is uniquely defined for each state, it is possible for two states to share a common successor state:

Property 2.2 (Shared Successors) *Given two different states S_i and S_j, it is possible that* $\mathtt{Succ}(S_i) = \mathtt{Succ}(S_j)$.

For example, S_i and one of its refinements S_j might both have the same successor, if the refinement adds no useful information for inferring Next propositions. Shared successors can also occur due to equilibrium states or cycles of states:

Definition 2.4 (Equilibrium State) *An* equilibrium state *is a state S such that* $\text{Succ}(S) = S$.

Property 2.3 (Equilibrium Non-Inheritance) *Refinements of an equilibrium state need not be equilibrium states; however, no refinement of a non-equilibrium state can be an equilibrium state.*

Definition 2.5 (Cycle of States) *A* cycle *is a chain of successor states such that the first and last states are identical.*

Property 2.4 (Cycle Non-Inheritance) *Refinements of a cycle of states need not form cycles; however, no cycle may result from refinement of a non-cycle.*

Figure 4 shows the *successor* states (S_5, S_6, and S_7) that should exist for example state S_1 and existing refinements of it (S_2, S_3 and S_4). If such successor states are not created during the normal ATMS computations, they must be created by IQE if the problem solver wishes to reason about them.

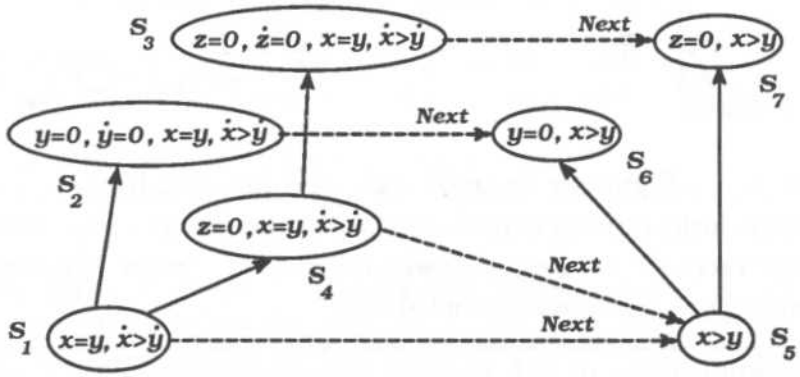

Figure 4: Example IQE temporal relations among states

All of the propositions that are true in each state are shown inside the oval representing that state.

IQE envisions by "laying down tracks" consisting of chains of successor states, at successively higher levels of detail. These tracks never branch, but may "die out" by becoming equivalent to or below the base environment. Tracks may also terminate by reaching an equilibrium state or a cycle of states.

Figure 5 illustrates an example of how IQE can lay down tracks of states at low levels of detail in an attempt to quickly realize that a partial goal state is not reachable from a partial initial state. First, the lowest track (the next successors of the initial state itself) is laid down from the initial state until the base state is noticed. This indicates that it is *consistent* that the initial state reaches the goal state. Special refinement driven by the temporal constraints from the previous state (i.e, the initial state) of $S0$ indicate a closed set of refinement states of $S0$ (see [2]). If the track of successor states of each state in that closed set shows that the goal state is not reachable from $S0$, then the goal state is not reachable from the initial state either. This process is repeated until either the unreachability is proven or such refinements are exhausted.

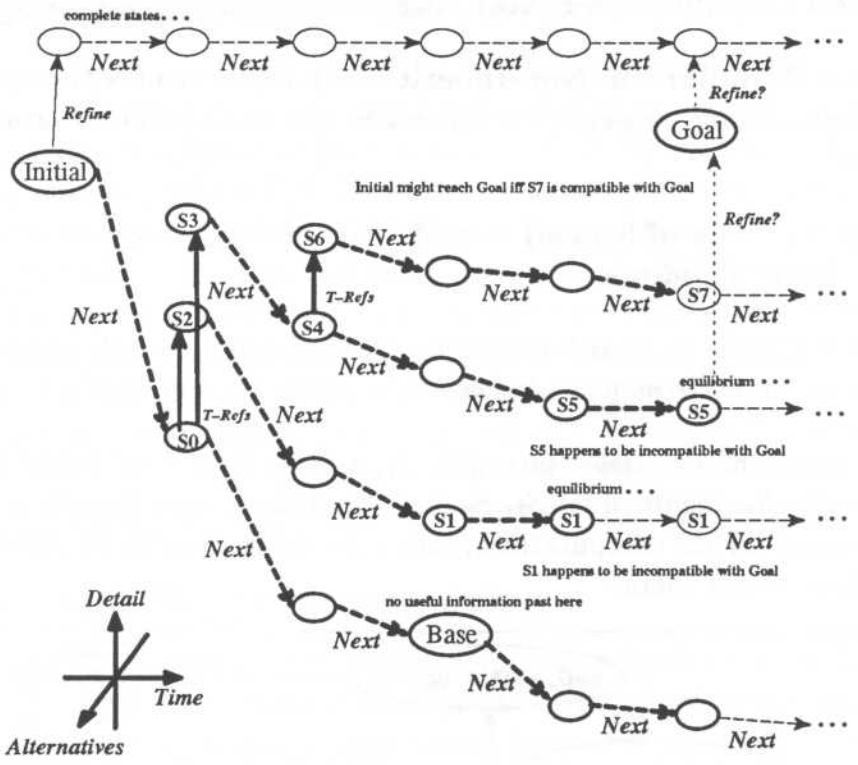

Figure 5: Example IQE envisionment lattice

Portion of an example lattice searched to realize that the partial goal state is not reachable from the partial initial state. If partial state $S7$ *was* compatible with the goal, then further refinements *would* be needed to show that the goal is not reachable from the initial state.

One possible application of IQE is doing goal-directed search. Given a partial description of an initial state and a goal state, IQE lays tracks from the initial state, checking each state along the track for compatibility with the goal. If the track hits an equilibrium state or a cycle before encountering a state consistent with the goal, then the goal state is declared *unreachable*:

Property 2.5 (State Unreachability) *A goal state G is unreachable from a state S if a cycle or equilibrium state is encountered along the track from S before encountering any states compatible with G.*

Unreachability of a goal state may be propagated back through the track (which may branch in the reverse direction) until reaching a state consistent with the goal state.

If the first consistent state is encountered after N successions, then no path shorter than N can exist from the initial state to the goal state. When this occurs, then new tracks are laid from some refinement along the existing track. Two important control issues are how to choose which refinement to extend, and how much to refine it. It is desirable to minimize branching and maximize information content. Choosing the state to refine involves a tradeoff, since earlier states are likely to be more detailed than later ones; however, because the track never branches, the cost of re-laying from a refinement of an

early state is outweighed by the benefit of additional information. Similarly, choosing how much to refine involves a tradeoff, given that refinements contribute to both information and branching.

3 Facilities Provided by IQE

IQE's design allows it to offer several significant facilities to the problem solver, without the need to completely specify all states. Figure 6 describes some of the key ones we have identified so far.

(CHANGE-SCENARIO add-asns delete-asns) and **(CHANGE-PERSPECTIVE add-asns delete-asns)**:
Changes IQE's working focus environment by adding and deleting assumptions.

(PARTIAL-STATE propositions):
Finds or creates a partial state S in which Props(S) is the deductive closure of **propositions** and all of Props(S) are true in Env(S).
(AUGMENT-PARTIAL-STATE propositions partial-state):
Same as PARTIAL-STATE, except that it adds all of **Props(partial-state)** to **propositions**.

(TRUE-IN-SOME-REFINEMENTS? propositions partial-state):
Returns some refinements of **partial-state** for which all **propositions** are in **Props(partial-state)**.

(STEADY-STATE-IN-SOME-REFINEMENTS? partial-state):
A special case of TRUE-IN-SOME-REFINEMENTS?, where **propositions** represents all **(D Q)** values being equal to **ZERO**.

(CONSISTENT-TRANSITION? current-partial-state next-partial-state):
Predicate which is true exactly when, for each proposition **X** for which proposition **(Next X)** is in **Props(current-partial-state)**, proposition **(Not X)** is not in **Props(next-partial-state)**.

(CONSISTENT-PATH? current-partial-state next-partial-state):
Predicate which is false exactly when it is impossible for next-partial-state to occur after current-partial-state. Incremental temporal reasoning starting at low levels of detail is used to try to efficiently show this path is impossible, while incremental refinements during this process progressively move towards complete states which will represent such a path, if it exists.

Figure 6: Basic IQE facilities

---

4 Discussion

In this paper we have presented our view of incremental envisioning as providing a sort of envisionment maintenance system for use by a higher-level problem solver. We have argued that such an approach provides a significant alternative to simulators such as QPC and QPE. Its power and efficiency comes from attempting to first reason about states at low levels of detail, while being able to consider higher levels of detail when such distinctions are required to answer a query. We have outlined the design of one such system, IQE, and highlighted some of the techniques used in its implementation.

5 Acknowledgements

Thanks to Ken Forbus and Gordon Skorstad for their comments on this work. This research has been supported by NASA Langley, under contract NASA-NAG-11023.

References

[1] John Collins and Dennis DeCoste. CATMS: an ATMS which avoids label explosions. In *Proceedings of AAAI-91*, July 1991. To appear.

[2] John Collins and Dennis DeCoste. *IQE: An Incremental Qualitative Envisioner.* Technical Report, Institute for the Learning Sciences, Northwestern University, 1991. Forthcoming.

[3] James Crawford, Adam Farquhar, and Benjamin Kuipers. QPC: a compiler from physical models into qualitative differential equations. In *Proceedings of AAAI-90*, pages 365–372, July 1990.

[4] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2), March 1986.

[5] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[6] Dennis DeCoste. Dynamic across-time measurement interpretation. In *Proceedings of AAAI-90*, pages 373–379, July 1990.

[7] Dennis DeCoste and John Collins. *CATMS: An ATMS Which Avoids Label Explosions.* Technical Report, Institute for the Learning Sciences, Northwestern University, 1991.

[8] Brian Falkenhainer and Kenneth D. Forbus. Setting up large-scale qualitative models. In *Proceedings of AAAI-88*, pages 301–306, August 1988.

[9] Kenneth D. Forbus. The qualitative process engine. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 220–235, Morgan Kaufmann, 1990. (Technical Report UIUCDCS-R-86-1288, University of Illinois at Urbana-Champaign, December 1986).

[10] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[11] Ben Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.