In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*
Cambridge, MA: AAAI/MIT Press, 1998.

Qualitative simulation as a temporally-extended constraint satisfaction problem *

Daniel J. Clancy
Caellum/NASA Ames Research Center, MS 269-3
Moffett Field, CA 94035 USA
clancy@ptolemy.arc.nasa.gov

Benjamin J. Kuipers
Computer Science Department
University of Texas at Austin
Austin, Texas 78712
kuipers@cs.utexas.edu

Abstract

Traditionally, constraint satisfaction problems (CSPs) are characterized using a finite set of constraints expressed within a common, shared constraint language. When reasoning across time, however, it is possible to express both temporal and state–based constraints represented within multiple constraint languages. Qualitative simulation provides an instance of this class of CSP in which, traditionally, all solutions to the CSP are computed. In this paper, we formally describe this class of *temporally–extended* CSPs and situate qualitative simulation within this description. This is followed by a description of the DecSIM algorithm which is used to incrementally generate all possible solutions to a temporally–extended CSP. DecSIM combines problem decomposition, a tree-clustering algorithm and ideas similar to directed arc–consistency to exploit structure and causality within a qualitative model resulting in an exponential speed-up in simulation time when compared to existing techniques.

Introduction

Traditionally, constraint satisfaction problems (CSPs) are characterized using a finite set of constraints expressed within a common, shared constraint language (Tsang 1993). When reasoning across time, however, it is possible to express both temporal and state–based constraints represented within multiple constraint languages. A *state–based constraint* specifies restrictions between variables that must hold at any given point in time while a *temporal constraint* specifies restrictions that occur across time. Qualitative simulation provides an instance of this class of *temporally–extended* constraint satisfaction problems.

This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin and was supported in part by NSF grants IRI-9504138 and CDA 9617327, by NASA grants NAG 2-994 and NAG 9-898, and by the Texas Advanced Research Program under grant no. 003658-242.

Qualitative simulation (Forbus 1984; Kuipers 1994) reasons about the behavior of a class of dynamical systems using a branching time description of alternating time–point and time–interval states. The model specifies a finite set of variables and constraints. Each constraint specifies valid combinations of variable values for any given point in time (*i.e.* state–based constraints). Continuity constraints are then used to restrict the valid transitions between states. For example, if in a time–point state S_1 the variable X is increasing and has a value $X*$, then immediately following S_1, X must be greater than $X*$ and increasing. The continuity constraints correspond to temporal restrictions. The use of temporal constraints within qualitative simulation has been further generalized within the TeQSIM algorithm to allow the specification of an arbitrary temporal constraint (Brajnik & Clancy 1996; 1997). TeQSIM uses a propositional linear-time temporal logic (PLTL) (Emerson 1990) that combines propositional state-formulae specifying either qualitative or quantitative information about a state with temporal operators such as *next*, *eventually*, *always*, and *until*. Each qualitative behavior generated during simulation corresponds to a solution to the CSP defined by the composition of the state–based constraints with the temporal constraints.

Viewing qualitative simulation as a temporally–extended CSP is beneficial because it allows us to explore how advances within the CSP literature can be used to improve the techniques applied during simulation. In addition, it describes a new class of constraint satisfaction problems that has not been extensively explored within the CSP literature. While conceptually a traditional temporal CSP can be used to express state–based constraints, by separating the two sets of constraints we are able to exploit inherent structure that exists within the state–based constraints to reduce the overall complexity of finding a solution.

The DecSIM qualitative simulation algorithm efficiently computes all possible solutions to this class of

CSPs by building upon and extending existing research within the constraint satisfaction literature.[1] DecSIM provides a sound, but potentially incomplete algorithm that solves instances within this class of CSPs exponentially faster than the techniques currently used within the qualitative reasoning literature. This speed-up facilitates the application of qualitative simulation techniques to larger, more realistic problems.

Qualitative simulation explicitly computes all possible solutions to the temporally–extended CSP defined by the model. If two variables are completely unconstrained with respect to each other, then the set of all possible solutions contains the cross–product of the possible values for each variable. For QSIM this results in combinatoric branching when the temporal ordering of a set of events is unconstrained by the model.

DecSIM uses a divide and conquer approach to reduce the complexity of a simulation by exploiting structure within the qualitative model. DecSIM decomposes the state–based CSP, P_M, defined by the qualitative model M into a set of smaller sub–problems. Each sub–problem contains a subset of the variables in P_M while shared variables represent the constraints between sub–problems. DecSIM explicitly computes all solutions only for each sub–problem. In addition, however, DecSIM must also ensure that each partial solution participates in at least one solution to the original CSP. This task is characterized as a separate global CSP. Note that for each partial solution DecSIM is only required to compute a *single* solution to this global CSP as opposed to explicitly enumerating all possible solutions as is effectively done by QSIM. Furthermore, causality is used along with a tree-clustering algorithm to simplify the task of identifying this solution. The primary technical innovation provided by the DecSIM algorithm is the application of these basic concepts to a temporally–extended CSP ensuring that both the state-based and the temporal continuity constraints are satisfied. In this paper, we describe the DecSIM algorithm and present both theoretical and empirical results demonstrating the benefits provided by DecSIM when compared to techniques currently used to perform a simulation. In addition, we provide a formal characterization of a temporally–extended CSP thus laying the ground work for future research to explore how work within fields such as qualitative reasoning, planning, constraint satisfaction and reasoning about action can potentially be integrated.

[1] An earlier version of the DecSIM algorithm was published in (Clancy & Kuipers 1997) with preliminary results. Since that publication, the algorithm has been significantly generalized, resulting in stronger theoretical claims and experimental results.

Definitions and concepts

Qualitative simulation uses an imprecise, structural model describing a class of dynamical systems to derive a description of all qualitatively distinct behaviors consistent with the model.[2] In its basic form, a model, called a qualitative differential equation (QDE), is defined by the tuple $<V, Q, I, C>$ where V is a set of variables, Q a discrete set of values for each variable, I is an initial state, and C a set of state–based constraints on the variables in V.

The constraints are abstractions of mathematical relationships restricting the valid combinations of values for the variables in the constraint. The behavior of the system is described by a tree of qualitative states in which each path consists of alternating time–point and time–interval states. Each qualitative state provides a value for all of the variables within the model along with a value for a special variable representing time.

A qualitative model defines a traditional, state–based CSP. During simulation, however, this CSP is extended across time as each variable is assigned a value at successive time-point and time-interval states. These variable assignments must satisfy both the state–based constraints as well as any implicit or explicit temporal constraints. A temporally–extended CSP is defined by extending the definitions that are commonly used to define a traditional CSP.

Definition 1 (Temporally–extended CSP) *A temporally–extended CSP is defined by the tuple* $<V, D, C_s, C_t, closed>$ *where*

- V *is a set of variables,*
- D *defines the domain for each variable,*
- C_s *is the set of state–based constraints,*
- C_t *is a set of temporal constraints, and*
- closed *is a domain-specific boolean predicate that is used to determine whether or a temporal sequence can be extended further.*

A *label* is an assignment of a value to a variable at a particular point within the temporal sequence. Thus, a label is defined by the tuple $<v, d_v, t>$ where v is a variable, d_v is a value for the variable and t is a non-negative integer corresponding to the "location" of the value assignment within the temporal sequence. A label with the value of $t = 0$ corresponds to the

[2] In this presentation, we focus on the representation used by the QSIM qualitative simulation algorithm. These concepts can also be applied to alternative representations (Forbus 1984; de Kleer & Brown 1985) that have been proposed within the literature.

initial state within a temporal sequence.[3] An *extended label* (also called a *variable history*) corresponds to a sequence $(<v, d_v^1, t>, <v, d_v^2, t+1>, \ldots, <v, d_v^m, t+m>)$ of consecutive labels while a *rooted extended label* is an extended label with $t = 0$. A *compound label* is a set of labels $(<v_1, d_{v_1}, t>, <v_2, d_{v_2}, t>, \ldots <v_n, d_{v_n}, t>)$ providing values to multiple variables in the same time-state while an *extended compound label* is simply a set of extended labels for multiple variables.

A *solution* to the temporally–extended CSP corresponds to an extended compound label that satisfies both the state-based and the temporal constraints. The formal definition of a solution, however, needs to account for the potentially infinite nature of a temporal sequence. To address this issue, the domain specific boolean predicate *closed* is used.

Definition 2 (Partial and Full Solutions) *A* fully extended solution *is an extended compound label that satisfies all of the constraints as well as the* closed *predicate. Conversely, a* partially extended solution *is an extended compound label that satisfies all of the constraints but fails to satisfy the* closed *predicate.*

How the *closed* predicate is defined depends upon the domain. Within qualitative simulation, a solution is fully extended if either a cycle is detected, a transition condition occurs or the last state is quiescent (*i.e.* in a steady state). On the other hand, if planning were viewed as a temporally–extended CSP, then a solution is fully extended when the final state satisfies the goal condition. One of the reasons that it is important to define the *closed* predicate is that in qualitative simulation a partially-extended solution is eliminated from the solution set if it is determined that there does not exist a consistent extension to the solution.

Given a qualitative model and an initial state, qualitative simulation can be mapped to a temporally–extended CSP in a straight-forward manner by mapping the variables, their domains and the state–based constraints directly from one problem to the next and then by deriving the temporal constraints from the continuity constraints applied when performing a simulation.[4] Qualitative simulation then attempts to find all possible solutions to the temporally–extended CSP given some limit on either the length of a temporal sequence or the overall set of compound labels within the solution set. Thus, the complexity of the problem is completely determined by the size of the solution space. Since larger models tend to be more loosely constrained, simulation of these models often results in intractable branching and an exponential number of solutions. Thus, a state–based representation is inherently limited in its ability to represent and reason about the behavior of an imprecisely defined dynamical system.

[3] In our definition of a temporally–extended CSP, we do not restrict the semantic interpretation of a point within this sequence. In qualitative simulation, a temporal progression will alternate between time-point and time-interval states, however, other interpretations can be used for different domains.

[4] To describe the problem addressed by TeQSIM, the set of temporal constraints are extended to include any temporal logic expressions contained within the model. Currently, however, DecSIM does not handle arbitrary temporal constraints as can be expressed using TeQSIM.

The DecSIM Algorithm

DecSIM modifies the representation used to describe the solution space generated during qualitative simulation thus providing a more compact representation and a more efficient simulation algorithm. DecSIM exploits the fact that larger systems can often be decomposed into a number of loosely connected subsystems due to inherent structure that exists within the model. By decomposing the model, DecSIM is able to address one of the primary sources of complexity – combinatoric branching due to the complete temporal ordering of the behaviors of unrelated variables.

Component Generation

Given a model M with a set of variables V and an initial state I, DecSIM uses a partitioning $\{V_1, V_2, \ldots V_n\}$ of the variables in the model to generate a *component* for each partition. Currently, the partitioning of the variables is provided as an input to the DecSIM algorithm.[5] A separate behavioral description (*i.e.* set of solutions) is explicitly generated for each component. The interaction between components is represented via shared variables called *boundary variables.* For each partition V_i, DecSIM generates a component C_i containing two types of variables: **within-partition variables** are the variables specified in V_i, and **boundary variables** are variables contained in other partitions that have a *direct causal influence* on the within–partition variables.

DecSIM identifies boundary variables using an extension of Iwasaki's (1988) causal ordering algorithm to transform the model into a hybrid directed/undirected hypergraph called the *causal graph.* A variable v_i is said to have a *direct causal influence* on a variable v_j if there exists either an undirected hyperedge in the causal graph relating v_i and v_j or if there exists a di-

[5] Various techniques for automating this process have been considered; however, up to this point, we have focused on the simulation algorithm since partitioning the variables is a fairly straight–forward extension of the model–building process.

rected hyperedge extending from v_i and terminating in v_j.[6]

Each component C_i defines a qualitative model, in which the variables correspond to the union of the within–partition and boundary variables for the component and the constraints correspond to the set of constraints in the original model relating the variables contained in the component. Furthermore, the relationship between components defined by the shared boundary variables defines a labeled directed graph called the *component graph* defined as follows.

Definition 3 (Component graph) *Given a set of related components* $\{C_1, C_2, \ldots, C_n\}$ *the* component graph *is a labeled, directed graph with a node corresponding to each component. The edges are defined as follows:*

- *An edge exists from component* C_i *to node* C_j *if and only if there exists a variable* v *such that* v *is a within–partition variable in* C_i *and a boundary variable in* C_j.
- *An edge from* C_i *to* C_j *is labeled with the set of boundary variables in* C_j *that are classified as within–partition variables in* C_i.

Local and global consistency

The core QSIM algorithm is used to derive a separate behavioral description, called a *component tree*, for each component. The terms *component behavior* and *component state* are used to refer to a behavior and a state within a component tree respectively.

Definition 4 (Local consistency) *A component behavior for component* C_i *is* locally consistent *if and only if it is a solution to the CSP defined by* C_i.

In addition, however, each component behavior must be consistent with respect to the rest of the model. To determine if a component behavior participates in a complete solution to the original CSP, DecSIM must solve a separate *global* CSP in which the "variables" correspond to components, the "variable values" to component behaviors, and the "constraints" to the restrictions represented by the variables that are shared between components.

A component behavior b_i is *globally consistent* if and only if there exists a set of component behaviors $B = \{b_{C_1}, b_{C_2}, \ldots, b_{C_n}\}$ containing b_i such that B is a solution to the global CSP.[7] Thus, DecSIM must determine if a set of component behaviors are compatible with respect to the variables that are shared between components.

Definition 5 (Compatible behaviors) *A set of component behaviors* $B = \{b_{C_1}, b_{C_2}, \ldots, b_{C_n}\}$ *are compatible if and only if the behaviors can be combined to form at least one* composite *behavior describing all of the variables within each component behavior.*

Conceptually, behavior composition is equivalent to a multi-way join as it is used in the relational database literature.

Definition 6 (Behavior composition) *Given a set of component behaviors* $B = \{b_{C_1}, b_{C_2}, \ldots, b_{C_n}\}$, $\mathcal{B}$ *is a* composite *behavior for* B *if and only if* $\mathcal{B}$ *describes all of the variables in* $\bigcup_{i=1}^{n} V_{C_i}$ *and for* i *from* 1 *to* n *and* $\Pi_{V_{C_i}}(\mathcal{B}) = b_i$ *where* $\Pi_{V_{C_i}}$ *is the projection of a behavior onto the variables in* V_{C_i}.[8]

The set of all composite behaviors is essentially the cross-product of the component behaviors, minus the combinations filtered out by the global consistency constraints. Thus, if the components are weakly connected, this set can be quite large.

Since each behavior is actually a temporal sequence, determining if a set of component behaviors is compatible is not simply a straight–forward comparison of the variable values. DecSIM addresses this issue by maintaining a many-to-many mapping between states within components related in the component graph. Two states are mapped to each other if and only if the behaviors terminating in these two states are compatible. This mapping allows DecSIM to evaluate the global consistency of a set of behaviors by comparing the final states within the behaviors with respect to this mapping.

The mapping is generated as each component behavior is incrementally extended. If components A and B are related via shared variables, then state S_A maps to component state S_B if and only if S_A and S_B are equivalent with respect to any shared variables, and either S_A and S_B are initial states, the predecessor of S_A maps to the predecessor of S_B, the predecessor of S_A maps to S_B, or S_A maps to the predecessor of S_B.

By maintaining the many-to-many mapping between compatible states in related components, DecSIM translates the global CSP in which the variable values correspond to component *behaviors* into a CSP, called the *component graph CSP*, in which the variable values correspond to qualitative *states*. This translation enables DecSIM to evaluate the global consistency

[6] The algorithm used to generate the causal graph is potentially incomplete in which case the causal relationship between any two variables may be incomplete thus resulting in an undirected hyperedge in the causal graph.

[7] Throughout this paper, subscripts are used to identify the component to which a state, behavior or set of variables belong.

[8] Please refer to (Clancy 1997) for a formal definition of the projection operator as it applies to a qualitative behavior.

of a state based upon local inferences with respect to the temporal progression of the behaviors.

Definition 7 (Component graph CSP) *The* component graph CSP *is defined by the tuple* $<V, D, C>$ *such that*

- *the variables* V *correspond to components,*
- *the domain for each variable is the set of qualitative states defined in the component tree for the corresponding component (*D *is the set of domains for each variable),*
- *the constraints* C *are defined by the many-to-many mapping maintained between related components. If components* C_i *and* C_j *are related in the component graph, then a constraint exists between the corresponding variables within the component graph CSP that is satisfied for two states* s_{C_i} *and* s_{C_j} *if and only* s_{C_i} *and* s_{C_j} *are linked by the many-to-many mapping relating these components.*

Specification of this global CSP on qualitative states allows us to define global consistency with respect to a qualitative state as opposed to an entire behavior.

Definition 8 (Globally consistent) *A component state is* globally consistent *if and only if the state is both locally consistent and participates in a solution to the component graph CSP.*

Performing the simulation

When performing a simulation, DecSIM iterates through the components incrementally simulating the leaf states in each component tree. DecSIM uses QSIM to generate the successors of each component state, thus ensuring that each state is locally consistent. In addition, however, DecSIM must ensure that each state is also globally consistent.

Determining whether a component state is globally consistent for a fully simulated set of component behaviors is a straight–forward constraint satisfaction problem given the characterization that has been provided. DecSIM, however, incrementally generates each component behavior. Thus, it is possible that as component behaviors are extended, a solution to the component graph is generated containing a state already within a component tree that had previously not been globally consistent. In other words, a state s not being globally consistent at a given point during the simulation does not imply that s is globally *inconsistent*. DecSIM addresses this issue using two techniques: 1) successors of a component state s are only computed if s is determined to be globally consistent, and 2) the identification of a state as globally consistent must be propagated through the component graph CSP by identifying all solutions to the component graph that contain at least one state whose status with respect to global consistency had previously been undetermined.

The first condition ensures that globally inconsistent states are not simulated while the second condition ensures that all globally consistent solutions are still computed given the first restriction.

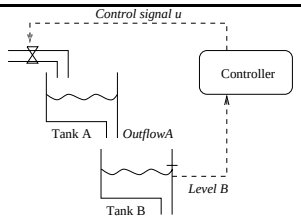
The algorithm used to test a state for global consistency exploits causality and structure within the component graph CSP to reduce the complexity of finding a solution. First, a tree-clustering algorithm (Dechter & Pearl 1988; 1989) is used to transform the component graph CSP into an acyclic *cluster graph* by grouping components that are included within a cycle into a single node within the cluster graph. This transformation significantly reduces the time required to find a solution to the component graph by allowing DecSIM to use constraint propagation between clusters as opposed to computing a complete solution to the CSP. Second, causality is used to further reduce the complexity of this process by asserting the independence of causally upstream components from those components that are strictly downstream with respect to the causal ordering.

Two primary benefits are provided by DecSIM with respect to a standard QSIM simulation. First, by transforming the component graph into an acyclic cluster graph, the worst case complexity required to find a solution to the component graph CSP is exponential in the size of the largest cluster as opposed to overall size of the component graph. Second, DecSIM is only required to find a single solution to the component graph for each component behavior as opposed to computing all possible solutions as is required by a standard QSIM simulation.

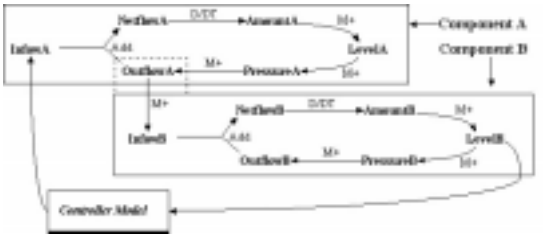
An Example

The benefits provided by DecSIM can be demonstrated using a simple model of a sequence of cascaded tanks. Two versions of this model will be used during the discussion. In the simpler version, a simple cascade is used while in the more complex version a feedback loop exists by which the inflow to the top tank is controlled by the level in the bottom tank (see figure 1).

When performing a standard QSIM simulation of a simple N-tank cascade, a total of 2^{N-1} behaviors are generated enumerating all possible solutions to the CSP. Many of these solutions, however, simply provide different temporal orderings of unrelated events. DecSIM, however, eliminates these distinctions generating a separate behavioral description, each containing a total of three behaviors, for each tank. By using causality, we are able to reason about the behavior of the upstream tanks independently of the downstream



(a) Controlled two tank cascade



(b) Causal graph and variable partitioning

The qualitative model of a controlled two tank cascade (a) is partitioned into three components: tankA, tankB and the controller. DecSIM generates a causal graph of the model (b) that is used to identify the boundary variables. The variable partitioning is identified by the solid boxes within the causal graph. Boundary variables are variables in other partitions that are causally upstream. Thus, `OutflowA` is a boundary variable for component B and therefore included in the component; however, `InflowB` is not a boundary variable for component A.

Figure 1: Controlled two tank cascade

tanks. Thus, the overall complexity of the simulation is linear in the number of tanks. In the controlled version of the N-tank cascade, a feedback loop connects the upstream and downstream tanks. DecSIM, however, is still able to provide significant improvements in the overall simulation time due to its ability to partition the problem into smaller sub-problems and reason about the interaction between sub-problems independently. For an 8 tank cascade, DecSIM generates an average of 28 behaviors for each component while QSIM generates a total of 1071 behaviors. Table 1 displays the simulation time for a number of variations on this example.

Theoretical Results

Traditional techniques for qualitative simulation, while both sound and complete with respect to the CSP defined by the model, are unable to scale to larger problems.[9] DecSIM, on the other hand, trades completeness for efficiency. The following results are established in (Clancy & Kuipers 1998).

Theorem 1 (DecSIM Soundness Guarantee) *Given a consistent qualitative model M and a decomposition of the model into components $\{C_1, C_2, \ldots C_m\}$, for all solutions B to the temporally–extended CSP defined by M, DecSIM generates the set of partial solutions $\{b_1, b_2, \ldots, b_n\}$ such that for $i : 1 \leq i \leq n ::$ $\Pi_{V_i}(B) = b_i$.*[10]

Theorem 2 (DecSIM Completeness Guarantee) *Given a consistent model M and a decomposition of the model into components $\{C_1, C_2, \ldots C_m\}$ such that there does not exist a cycle of size 3 or greater in the component graph CSP, for all partial solutions b_i generated by DecSIM describing the subset of variables V_i, there exists a corresponding solution to the temporally–extended CSP defined by M such that $b_i = \Pi_{V_i}(B)$.*

Note that the completeness theorem includes a restriction on the maximum cycle size within the component graph CSP. Except for this one limitation and the temporal ordering of behaviors in separate components, the behavioral description generated by DecSIM is identical to the description generated by QSIM. Temporal ordering information (which is intentionally omitted), however, is still available since DecSIM implicitly represents this information via the constraints represented by the mapping maintained between component states.

Incompleteness The source of the incompleteness comes from the characterization of the component graph CSP as a CSP over qualitative states as opposed to qualitative behaviors. This translation is essential if DecSIM is to efficiently determine if a component state is globally consistent; however, it may result in the introduction of component behaviors that do not have a corresponding behavior within the behavioral description generated by QSIM. The problem encountered is analogous to the distinction between constraint propagation and constraint satisfaction with respect to the temporal continuity constraints. Before allowing a state S_A to participate in a solution to the component graph CSP, DecSIM requires that its predecessor participate in a solution. However, it does not check to ensure that the solution containing S_A satisfies the continuity constraints with respect to a solution containing the predecessor of S_A. To do this, DecSIM would be required to maintain a record of solutions to the component graph CSP to ensure that a proposed solution is continuous with a solution identified for the preceding time-step. For many models, this may require DecSIM to compute *all* solutions for a cluster

[9] Note the *incompleteness* of qualitative simulation is with respect to the set of real valued trajectories described by the behavioral description and not the CSP.

[10] $\Pi_{V_i}(B)$ is the projection of the solution onto the subset of variables V_i where component C_i is assumed to describe the set of variables V_i.

within the component graph thus eliminating the computational efficiency benefits provided by DecSIM for the behavior of the system within this cluster.

In practice, the incompleteness of the algorithm has not been a problem for a number of reasons. First, the conditions under which the incompleteness of the algorithm is encountered is quite restricted and only occurs when two components are closely related. In fact, we have yet to encounter this problem in any of the models that have been tested. In addition, we have developed an algorithm that can be used to identify when this problem occurs which can be run following completion of a simulation. Finally, qualitative simulation already encounters a problem with behaviors being generated that do not correspond to a real-valued trajectory of a dynamical system described by the model. Thus, techniques using qualitative simulation already must account for possible spurious behaviors. In the end, as is often the case, a trade-off exists between completeness and the overall computational complexity of the algorithm. [11] A traditional state–based approach is inherently limited in its ability to scale to larger problems and thus at times we may be required to sacrifice completeness guarantees when reasoning about these problems.

Computational complexity

The overall complexity of a standard QSIM simulation is determined by the size of the representation that is being computed. The worst case size of the behavioral description is exponential in the number of variables within the model. DecSIM reduces the size of the solution space by decomposing the model. For DecSIM, the worst case size is simply exponential in the number of variables in the largest component. DecSIM, however, must also reason about the global consistency of a component state by solving the component graph CSP. Thus, the overall benefits provided by DecSIM depend upon the topology of the model and the degree to which it lends itself to decomposition along with the variable partitioning selected by the modeler.

The following two conclusions (Clancy 1997) define the relationship between DecSIM and QSIM with respect to the complexity of a simulation for a model that is decomposed into k partitions: 1) as the degree of overlap between components approaches zero, the size of the total solution space is reduced by an exponential factor k where k is the number of components within the decomposed model; and 2) as the degree of overlap approaches a fully connected constraint graph for the component graph CSP, the size of the set of behaviors generated by DecSIM is within a factor of k of a standard QSIM simulation. In practice, the savings provided by a DecSIM simulation are quite pronounced as is demonstrated by our empirical results. The primary source of these savings is the fact that DecSIM is only required to compute a *single* solution to the component graph CSP (*i.e.* to ensure that each component behavior is consistent with at least one global solution) as opposed to computing all solutions (as QSIM does for the non-decomposed model).

[11] We are currently in the process of developing a proof to establish that this trade-off is an inherent limitation when computing all possible solutions to a temporally–extended CSP.

# of Tanks	Cascade		Chained		Loop	
	Qsim	DecS	Qsim	DecS	Qsim	DecS
2	0.20	0.815	3.07	6.79	0.757	5.58
3	0.62	1.6	10.9	19.90	16.14	8.14
4	2.2	3.12	37.5	25.98	89.41	12.6
5	7.09	5.49	139	36.71	493.8	23.2
6	21.9	6.32	676	62.40	2758	48.7
7	71.5	8.39	1633	70	14474	116
8	236	11.6	8101	77	nc	442

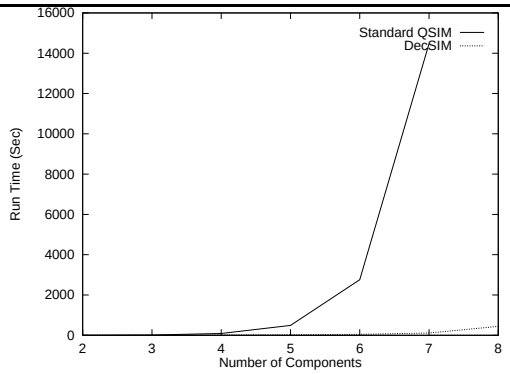
nc = Resource limitation prevented completion

Table 1: Simulation Time Results: DecSIM vs QSIM

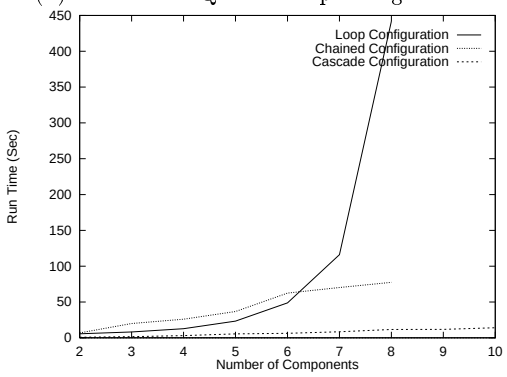
Empirical Evaluation

Empirical evaluation has been used to measure the benefits provided by a DecSIM simulation with respect to a standard QSIM simulation. Both DecSIM and QSIM were tested on a set of "extendible" models. An extendible model is a model composed of a sequence of identical components thus enabling the incremental extension of the model to facilitate an evaluation of the asymptotic behavior of the algorithm. The models used were variations on the cascaded tanks example described in figure 1. Three different versions were used; each with a different topology for the component graph CSP. A simple *cascade* topology, a *loop* topology in which the top tank is controlled by the bottom tank, and a *chain* topology in which the outflow for tank n is controlled by the level for tank $n+1$.

For all three models, DecSIM performed exponentially better than QSIM. Table 1 shows the simulation time results as the number of tanks are varied while figure 2(a) plots the results for the loop configuration comparing QSIM to DecSIM. The benefits provided by DecSIM are even more pronounced for the other two topologies. Figure 2(b) provides a comparison of the results from the DecSIM simulation for all three topologies. Note the dependence of the computational complexity on the topology of the model. In the loop configuration, the component graph CSP is composed of a single, large cycle. Thus, it is more likely to en-



(a) DecSIM vs QSIM: Loop configuration



(b) DecSIM on different versions of the N-tank cascade

Figure 2: DecSIM results

counter backtracking when determining if a component state is globally consistent. Thus, the complexity of the simulation becomes exponential in the number of tanks. DecSIM, however, still performs significantly better than QSIM. For the simple N-tank cascade the complexity is linear in the number of tanks.

Conclusions

In this paper, we have characterized qualitative simulation as the composition of a state–based and a temporal–based CSP. Furthermore, we have shown that the state–based representation that is traditionally used when performing a simulation is inherently limited thus restricting the degree to which techniques based upon qualitative simulation can scale to larger, more realistic problems. DecSIM provides an alternative simulation algorithm that addresses this problem by decomposing the model into components. Decomposition eliminates combinatoric branching due to the complete temporal ordering of behaviors for unrelated variables contained in separate components. Thus, the complexity of the simulation is determined by the inherent complexity of the problem specification as opposed to an artifact of the inference mechanism used to perform simulation. Furthermore, characterizing qualitative simulation as a general class of CSPs allows the ideas presented here to be applied within a broader context. Hopefully, the concept of a temporally-extended CSP can help integrate ideas from fields such as qualitative simulation, planning and reasoning about action to provide a more unified representation that can be used to reason about both autonomous and non–autonomous change within the physical world.

References

Brajnik, G., and Clancy, D. J. 1996. Temporal constraints on trajectories in qualitative simulation. In Clancey, B., and Weld, D., eds., *Proc. of the Thirteenth National Conference on Artificial Intelligence*. AAAI Press. To appear.

Brajnik, G., and Clancy, D. J. 1997. Focusing qualitative simulation using temporal logic: theoretical foundations. *Annals of Mathematics and Artificial Intelligence*. To appear.

Clancy, D. J., and Kuipers, B. J. 1997. Model decomposition and simulation: A component based qualitative simulation algorithm. In Kuipers, B. J., and Webber, B., eds., *Proc. of the Fourteenth National Conference on Artificial Intelligence*. AAAI Press.

Clancy, D. J., and Kuipers, B. 1998. Divide and conquer: A component–based qualitative simulation algorithm. Technical Report forthcoming, Artificial Intelligence Laboratory, The University of Texas at Austin.

Clancy, D. J. 1997. Solving complexity and ambiguity problems in qualitative simulation. Technical Report AI-TR97-264, Artificial Intelligence Laboratory, The University of Texas at Austin.

de Kleer, J., and Brown, J. S. 1985. A qualitative physics based on confluences. In Hobbs, J. R., and Moore, R. C., eds., *Formal Theories of the Commonsense World*. Norwood, New Jersey: Ablex. chapter 4, 109–183.

Dechter, R., and Pearl, J. 1988. Tree-clustering schemes for constraint processing. In *Proceedings of the Seventh National Conference on Artificial Intelligence*. Los Altos, CA.: Morgan Kaufman.

Dechter, R., and Pearl, J. 1989. Tree–clustering for constraint networks. *Artificial Intelligence* 38:353–366.

Emerson, E. 1990. Temporal and modal logic. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science*. Elsevier Science Publishers/MIT Press. 995–1072. Chap. 16.

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

Iwasaki, Y. 1988. Causal ordering in a mixed strcuture. In *Proc. of the Seventh National Conference on Artificial Intelligence*, 313–318. AAAI Press / The MIT Press.

Kuipers, B. 1994. *Qualitative Reasoning: modeling and simulation with incomplete knowledge*. Cambridge, Massachusetts: MIT Press.

Tsang, E. 1993. *Foundations of Constraint Satisfaction*. San Diego, CA: Academic Press.