Divide and Conquer: A component–based qualitative simulation algorithm

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

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

Abstract

Traditionally, qualitative simulation uses a global, state–based representation to describe the behavior of an imprecisely defined dynamical system. This representation, however, is inherently limited in its ability to scale to larger systems since it provides a complete temporal ordering of all unrelated events thus resulting in combinatoric branching in the behavioral description. The DecSIM qualitative simulation algorithm addresses this problem using a divide and conquer approach. 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 along with a more focused behavioral description. In this paper, we present the DecSIM algorithm along with both theoretical and empirical results demonstrating the benefits provided over traditional techniques. In addition, we formally characterize the problem addressed by qualitative simulation as the composition of *state–based* and *temporal–based* constraint satisfaction problems (CSPs) allowing us to discuss how DecSIM can be viewed as a general constraint satisfaction algorithm and how the results can be extended to other classes of problems.

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 *composite* constraint satisfaction problems.

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. Each qualitative behavior generated during simulation corresponds to a solution to the CSP defined by the composition of the state–based CSP with the temporal CSP.

Viewing qualitative simulation as a composite 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. The DecSIM qualitative simulation algorithm[1] provides a solution to this class of CSPs by building upon and extending existing research within the constraint satisfaction literature. DecSIM provides a sound, but potentially incomplete algorithm[2] that solves

[1] This paper provides a more extensive discussion of the algorithm than what is provided in (Clancy & Kuipers 1998).

[2] As discussed later in the paper, the incompleteness of the algorithm only occurs under certain conditions and, in fact, has not been manifested in any of the

this class of CSP 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.

DecSIM uses a divide and conquer approach to reduce the complexity of a simulation by exploiting structure within the qualitative model. Qualitative simulation explicitly computes all possible solutions to the CSP. As with any CSP, if two variables are completely unconstrained with respect to each other, then the set of all possible solutions will contain 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. An *event* occurs whenever a variable crosses a qualitatively distinct landmark value or its derivative becomes zero. DecSIM avoids explicitly computing this cross–product by breaking the state–based CSP P_M into a set of smaller sub–problems $\{p_1, p_2, \ldots, p_n\}$ called *components*. Each component contains a subset of the variables in P_M while shared variables represent the constraints between components.

DecSIM explicitly computes all solutions (*i.e.* all qualitative behaviors) for each component. Each solution to a sub–problem p_i provides a partial solution to P_M since it provides an assignment of values to a subset of the variables. Solutions for related sub–problems (*i.e.* for components that share variables) are consistent if they assign the same value to all shared variables. Thus, the interaction between components defines a separate, *global* CSP. For a solution to a sub–problem to be *globally consistent*, it must participate in a solution to this global CSP. DecSIM addresses two problems when determining if a solution to a component is globally consistent:

1. When performing a simulation, a *solution* to a sub–problem is actually a sequence of qualitative states. To ensure that a solution for sub–problem p_i is consistent with a solution for a related sub–problem p_j, each state within the solution for p_i must be matched against a state in the solution for p_j in a manner that satisfies the composite CSP (see figure 1 for an example).

2. Global consistency must be computed in an incremental fashion as each solution to a component is incrementally extended via simulation.

models tested.

By decomposing the model into smaller problems, DecSIM is able to exploit structure within the model in two ways. First, it avoids explicitly computing all possible solutions to the original CSP. Instead, it computes all solutions for each sub–problem and then for each of these solutions it computes a single solution to the global CSP. Second, DecSIM uses causality along with a tree-clustering algorithm to identify an ordering between the sub–problems allowing solutions to the causally upstream sub–problems to be identified as globally consistent without finding a solution to the global CSP.

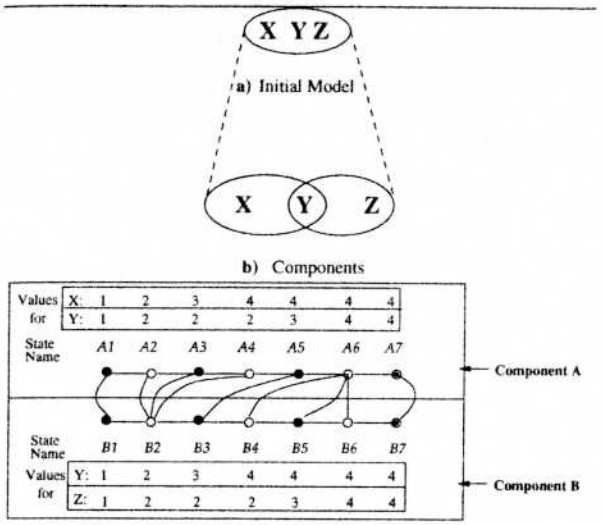
The primary contribution of this paper is the application of the DecSIM algorithm to the problem of qualitative simulation. We present both theoretical and empirical results demonstrating the benefits provided by DecSIM when compared to techniques currently used to perform a simulation. In addition, however, the characterization of qualitative simulation as a general class of a composite CSP allows us to explore how the concepts used within the DecSIM algorithm can be applied to other similar problems. In particular, this characterization highlights many of the similarities between qualitative simulation and planning.

Qualitative Simulation as a Composite CSP

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.[3] The following description of qualitative simulation will focus on issues that are relevant to its characterization as a constraint satisfaction problem. 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 alternating time–point and time–interval qualitative

[3] 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.



(c) Component state mapping

DecSIM decomposes a single model (a) containing three variables, X,Y, and Z, into a pair of components (b). Constraints between components are represented by a single shared variable Y.

- DecSIM maintains a mapping between states in components that share a variable. Each qualitative state must be mapped to a corresponding state within the neighboring component. Two states are mapped together if they are equivalent with respect to the shared variables and either their predecessors map to each other or one of the states maps to the predecessor of the other state. This mapping ensures that each component behavior is globally consistent.

- Figure (c) displays a single behavior from each component. Filled in circles correspond to time–point states and unfilled circles to time-interval states. Integer valued variables are used for this example. Variable values are displayed beneath each state. States that map to each other are connected by an arc.

- Note that a time–interval state can be mapped to multiple time–point states as other variables in the neighboring component change value. Thus, state B_2 is mapped to A_2, A_3 and A_4.

Figure 1: Example mapping between states

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 composed with a temporal CSP in which the "variables" correspond to model variables within successive qualitative states. Since a qualitative behavior can conceptually be of infinite length, the composite CSP contains an arbitrarily large number of variables defined by the set $\bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{V_i^j\}$ where V_i^j corresponds to the variable V_i from the original model in the jth state of a qualitative behavior and m corresponds to the maximum number of states within a given behavior. Similarly, the composite CSP is also composed of an arbitrarily large number of constraints.

The DecSIM Algorithm

Most applications of constraint satisfaction are concerned with identifying a single solution to the CSP. Qualitative simulation, however, must provide a characterization of all possible solutions to the CSP. Thus, the complexity of the problem is completely determined by the size of the solution space and in fact, can be shown to be at least PSPACE-complete.[4] 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.

To tractably solve the class of problems addressed by qualitative simulation, the problem space must be modified to provide a more compact representation. 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 (Simon 1969). 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.

[4]Qualitative simulation can be shown to be at least as hard as a STRIPS style planning problems which Bylander (1994) showed to be PSPACE-complete.

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.

The classification of a variable as a boundary variable or a within–partition variable is relative with respect to an individual component. Thus, a variable v_i might be a within–partition variable in component C_i, but a boundary variable in component C_j. Each variable within the original model is classified as a within–partition variable in one and only one component.

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* (see figure 2 for an example). 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 directed hyperedge extending from v_i and terminating in v_j.

Each component C_i defines a qualitative model, called a *sub-QDE*, defined by the tuple $<V_i, Q, C_{V_i}, I_i>$ where

- V_i is the union of the set of within–partition variables and boundary variables for the component,
- Q is a set of quantity spaces, one for each of the variables,
- C_{V_i} is the set of constraints within M that only contain the variables in V_i, and
- I_i is a projection of the initial state onto the set of variables in V_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 1 (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,* i.e. *consistent with the model defined by the component.*

QSIM ensures that each component behavior is locally consistent. In addition, however, each component behavior must be consistent with respect to the rest of the model. In fact, the boundary variables are effectively unconstrained within a component and thus their behavior is only constrained by the interaction between components.[6] The constraints between components are represented by shared boundary variables. These constraints define a separate, global CSP in which the "variables" correspond to components and the "variable values" to component behaviors. A component behavior is *globally consistent* if and only if there exists a solution to this global CSP.

The rest of this section provides a more detailed characterization of the global CSP that must be solved. Readers not interested in a detailed description of the algorithm can skip directly to the next section. First, we describe how global consistency is defined with respect to a component behavior. Then this concept is extended to individual states.

The constraints within this global CSP are implicitly represented by the shared boundary variables. For a set of components $\{C_1, C_2, \ldots, C_n\}$, a component behavior b_{C_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_{C_i} such that the component behaviors in B can be combined to

[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.

[6] The boundary variables within a component are causally upstream, exogenous variables with respect to the within–partition variables in the component. Thus, their behavior is not directly constrained by the within–partition variables.

form a single *composite* behavior.[7] A composite behavior combines a set of component behaviors into a single behavior describing all of the variables within each component.

Definition 2 (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 $\Pi_{V_{C_i}}(\mathcal{B}) = b_i$ *where* $\Pi_{V_{C_i}}$ *is the projection of a behavior onto the variables in* V_{C_i}.[8]

Note that the composition of a set of component behaviors may result in a large number of composite behaviors if the components are weakly connected.

Repeatedly determining if a component behavior is globally consistent each time it is extended by a single state would be prohibitively expensive. Instead, DecSIM incrementally determines the global consistency of a component behavior by maintaining a many–to–many mapping between states in components that share variables (see figure 1). Two states are mapped together if and only if the component behaviors leading up to the states are compatible with respect to each other. This mapping allows DecSIM to translate the global CSP in which the variable values correspond to component behaviors into a CSP in which the variable values correspond to component states. This allows global consistency to be computed via a local computation with respect to the temporal progression of a behavior.

The relationships between components are represented via shared variables which are used to define the links within a graph of components called the *component graph* which in turn is used to formally define the global CSP.

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 component* 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.

Thus, the component graph identifies the manner in which two components are related with respect to shared variables. The graph is directed because it incorporates information obtained from the causal graph.

If two components are related within the component graph, then DecSIM maintains a many–to–many mapping between states within the respective component trees. Two states are mapped together if and only if the component behaviors ending in these states can be combined to form a composite behavior. If components A and B are related within the component graph, then state S_A maps to component state S_B if and only if S_A and S_B are equivalent with respect to shared variables, and either

- both 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.

This mapping defines a boolean predicate, called a *component edge predicate*, for each edge within the component graph that defines how the behaviors can be combined together.

Definition 4 (Component-edge predicate) *For all edges* $C_i \xrightarrow{v_{ij}} C_j$ *within the component graph, the mapping between states in components* C_i *and* C_j *defines the predicate* $M_{C_i \xrightarrow{v_{ij}} C_j}$: $(S_{C_i}, S_{C_j}) \rightarrow \{T, F\}$ *where* S_{C_i} *and* S_{C_j} *correspond to states in components* C_i *and* C_j, *respectively.* $M_{C_i \xrightarrow{v_{ij}} C_j}(s_i, s_j)$, *called a* component edge predicate, *is true if and only if* s_i *is mapped to* s_j. $C_i \xrightarrow{v_{ij}} C_j$ *corresponds to the directed edge labeled* v_{ij} *relating component* C_i *to component* C_j. *The label identifies the set of shared variables.*

DecSIM uses the component-edge predicates coupled with the component graph to define a constraint satisfaction problem over component states that is used to determine if a component state is globally consistent.

[7] Throughout this paper, subscripts are used to identify the component to which a state, behavior or set of variables belong. Thus, b_{C_i} corresponds to a component behavior from component C_i.

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

Definition 5 (Component graph CSP) *The* component graph CSP *is defined as follows:*

- *a variable is defined for each component,*
- *the domain for each variable is the set of qualitative states defined in the component tree for the corresponding component,*
- *the constraints are defined by the set of component–edge predicates defined for the edges within the component graph.*

A solution to the component graph CSP, called a *solution to the component graph*, corresponds to a set of component states $\{s_1, s_2, \ldots, s_n\}$ such that

(1) one state comes from each component, and

(2) for all i and j such that $i, j \leq n$, if there exists an edge $C_i \xrightarrow{v_{ij}} C_j$ then $M_{C_i \xrightarrow{v_{ij}} C_j}(s_i, s_j)$ is true.

Definition 6 (Global consistency) *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.*

Defining the global CSP over component states as opposed to component behaviors significantly reduces the complexity of incrementally generating each component behavior. This translation, however, is also the source of DecSIM's incompleteness. Thus, it is possible for a component state to be marked globally consistent even though the behavior terminating in this state does not participate in a solution to the global CSP. Section 3 describes this issue in detail and discusses why it has not been a significant problem up to this point.

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 new 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*. Since the structure of the component graph CSP is constant throughout a simulation, 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. Please refer to (Clancy 1997) for a detailed discussion of this portion of the algorithm.

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 2).

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. Thus, the overall number of behaviors is linear in the number of tanks. In the controlled version of the N-tank cascade, DecSIM is still able to provide significant improvements in the overall simulation time due to its ability to partition the problem into smaller sub-problems. For an 8 tank cascade, DecSIM generates an average of 28 behaviors for each component while QSIM generates a total of 1071 behaviors.

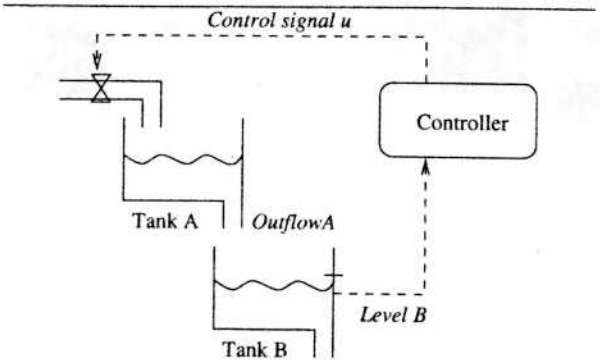
Theoretical Results

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 1997).

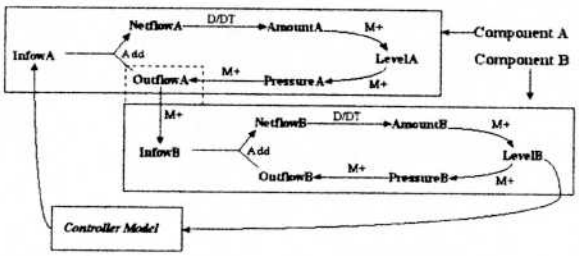
Theorem 1 (DecSIM Soundness Guarantee) *Given a consistent qualitative model M and a decomposition of the model into components $\{C_1, C_2, \ldots C_m\}$, for all solutions B to the composite CSP defined by M, DecSIM generates the set of partial solutions $\{b_1, b_2, \ldots, b_n\}$ such that for $i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

[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.

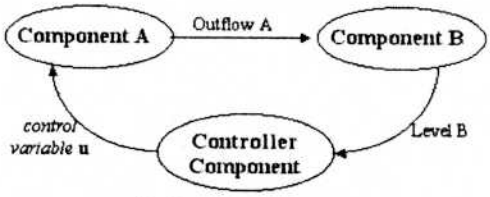
[10] $\Pi_{V_i}(B)$ is the projection of the solution onto the subset of variables V_i while component C_i is assumed to describe the set of variables V_i.



(a) Controlled two tank cascade



(b) Causal graph and variable partitioning



(c) Component graph

- 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.
- The component graph (c) has three components connected in a loop topology.

Figure 2: Controlled two tank cascade

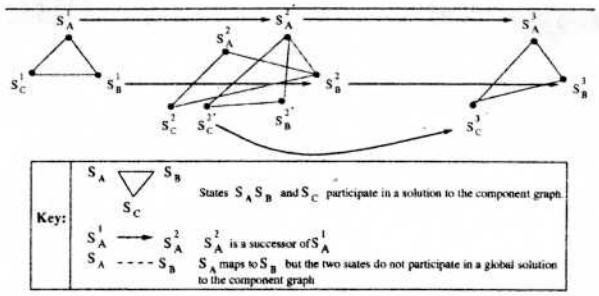
Theorem 2 (DecSIM Completeness Guarantee) *Given a consistent model M and a decomposition of the model into components $\{C_1, C_2, \ldots C_m\}$ such that there does not exist a cycle of size 3 or greater in the component graph CSP, for all partial solutions b_i generated by DecSIM describing the subset of variables V_i, there exists a corresponding solution to the composite 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.[11] Except for this one limitation, the behavioral description generated by DecSIM is identical to the description generated by QSIM except for the temporal ordering of behaviors for variables in separate components (which is intentionally omitted). Temporal ordering information, however, is still available if it is desired 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 state–based CSP via the mapping that is maintained during the simulation. 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 (see figure 3 for an example). 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.

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

[11] Due to use of causality, it is possible for there to be multiple links between two components and thus a cycle of size two.



The figure describes a simple example in which the incompleteness of the algorithm results in a globally *inconsistent* state being classified as consistent.

- Assume that components A, B and C participate in a cycle in the component graph CSP and that S_A^1, S_B^1 and S_C^1 participate in a consistent solution to the component graph CSP. (The subscript identifies the component while the superscript identifies the location of the state within a behavior.)
- Furthermore, suppose that the successors of S_A^1 and S_B^1 (*i.e.* $S_A^{2'}$ and S_B^2) are equivalent with respect to their shared variables; however, they do not participate in a global solution to the component graph. However, both $S_A^{2'}$ and S_B^2 participate in other solutions to the component graph and thus are marked globally consistent. Since $S_A^{2'}$ and S_B^2 are compatible, however, the link connecting these two states is still maintained even though this link does not participate in a solution.
- If this occurs, then it is possible for the successors of $S_A^{2'}$ and S_B^2 (*i.e.* S_A^3 and S_B^3) to be mapped to each other and for this mapping to participate within a global solution to the component graph.
- Thus, S_A^3 and S_B^3 are both marked globally consistent. However, there does not exist a composite behavior containing the component behavior segments terminating in these states since the predecessors of these two states do not participate in a solution together.

Figure 3: Example demonstrating the potential incompleteness

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.

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 complexity 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, established in (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; and 2) as the degree of overlap approaches a fully connected constraint graph for the component graph CSP, the size of the DecSIM solution space is within a factor of k. 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).

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 2. 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 $n+1$.

# 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

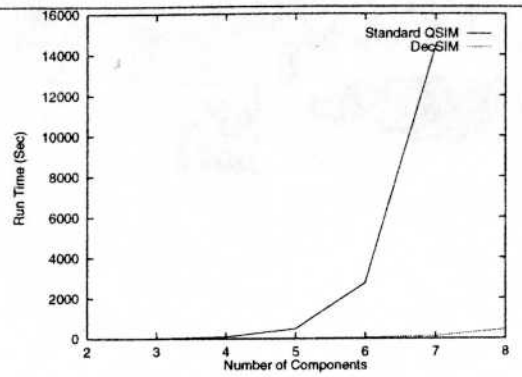
nc = Resource limitation prevented completion

Table 1: Simulation Time Results: DecSIM vs QSIM

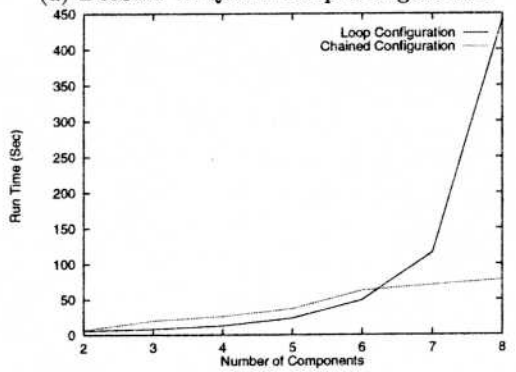
For all three models, DecSIM performed orders of magnitude better than QSIM. Table 1 shows the simulation time results as the number of tanks are varied while figure 4(a) plots the results for the loop configuration comparing QSIM to DecSIM. The benefits provided by DecSIM are even more pronounced for the other two topologies. Figure 4(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 encounter 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. Conversely, 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



(a) DecSIM vs QSIM: Loop configuration



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

Figure 4: DecSIM results

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. In addition, by characterizing qualitative simulation as a general class of CSPs, we discuss how the DecSIM algorithm might be extended to other problems besides qualitative simulation. In particular, we have shown how the problems addressed by qualitative simulation and planning are quite similar. Hopefully, this characterization will allow techniques from these two fields to be integrated thus providing a more unified representation that can be used to reason about both autonomous and non–autonomous change within the physical world.

References

Bylander, T. 1994. The computational complexity of propositional strips planning. *Artificial Intelligence* 69:165–204.

Clancy, D. J., and Kuipers, B. J. 1998. Qualitative simulation as a temporally-extended constraint satisfaction problem. In *Proc. of the Fifteenth National Conference on Artificial Intelligence.* AAAI Press.

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.

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.

Simon, H. A. 1969. *The Sciences of the Artificial.* Cambridge: MIT Press.

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