In *Proceedings of the Eighth International Workshop on Qualitative Physics about Physical Systems (QR-94)*
Nara, Japan, 1994.

Model Decomposition and Simulation

Daniel J. Clancy and Benjamin Kuipers
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
clancy@cs.utexas.edu and kuipers@cs.utexas.edu

Abstract

Qualitative reasoning uses incomplete knowledge to compute a description of the possible behaviors for dynamic systems. For complex systems containing a large number of variables and constraints, the simulation frequently is intractable or results in a large, incomprehensible behavioral description. Abstraction and aggregation techniques are required during the simulation to eliminate irrelevant details and highlight the important characteristics of the behavior. The total temporal ordering of unrelated events provided by a traditional state-based qualitative representation is one such irrelevant distinction. Model decomposition and simulation addresses this problem.

Model decomposition uses a causal analysis of the model to partition the variables into tightly connected components. The components are simulated separately in the order dictated by the causal analysis beginning with causally upstream components. Information from the simulation of causally upstream components is used to constrain the behavior of downstream components. If a feedback loop exists between components or a set of components are acausally related, then a concurrent simulation is performed for these components. A truth maintenance system is used to record and retract assumptions made during this concurrent simulation.

Model decomposition provides a general architecture which separates the method of simulation from the model decomposition algorithm. This architecture can be used to introduce alternative abstraction techniques to eliminate other irrelevant distinctions.

1 Introduction

Qualitative simulation derives a description of the possible behaviors of a dynamic system from a structural representation. The structural representation details the relationship between the variables within the system through constraints. Traditionally, a state-based approach [Forbus, 1984, Kuipers, 1986, De Kleer and Brown, 1984] is used to describe the device behavior by a set of qualitative states of the system and the transitions between these states. Each qualitative state contains a complete description of the system at either a time point or over a time interval. This level of description provides a complete temporal ordering of events within each behavior. An event occurs when a variable crosses a landmark value or alters its direction of change. The combinatoric complexity of providing a total ordering of events is a general problem encountered within artificial intelligence when trying to reason about the physical world [Hayes, 1985].

For many qualitative reasoning tasks, this level of detail is irrelevant. Branching on unrelated temporal distinctions significantly increases the complexity of the simulation. In addition, this branching makes it more difficult to interpret the behavioral description thus obscuring the relevant behavior.

Williams [1986] was the first to investigate methods of eliminating this complete temporal ordering of events within a simulation with the Temporal Constraint Propagator (TCP). TCP forsakes the traditional state-based approach and independently describes the behavior of each variable over time as a set of variable *histories*. The relevant temporal relations between events are described separately. A temporal ordering of events is only provided when their interaction affects the value of other quantities. Using TCP to reason about physical systems, however, requires a significant amount of work by the modeler. Furthermore, it is unclear how this system can be extended to incorporate advances in other qualitative reasoning paradigms.

Other techniques [Clancy and Kuipers, 1993, Fouché and Kuipers, 1991] have eliminated the irrelevant temporal correlation of events by combining behaviors and states into a more abstract representation. These abstraction techniques, however, determine the relevance of a correlation between events after the distinction is made in the simulation (i.e. the abstraction is performed in a post-processing fashion), and thus do not sufficiently address the complexity problem. Coiera [1992] eliminates these distinctions by superimposing qualitative predictions from two causally-unrelated processes on a single downstream variable. He does not address how these techniques can be applied to more complicated causal interactions.

Model decomposition bridges the gap between a state-based simulation and a history-based approach. A causal analysis is used to partition the variables into tightly connected components. Each component is simulated separately using a standard state-based simulation, providing a total ordering only for the events within each component. Temporal correlation between events in different components is only provided when necessary to constrain the resulting set of behaviors.

2 Model Decomposition - An Overview

Model decomposition uses a divide and conquer approach to the simulation of a qualitative model. The variables within the model are partitioned into components so that closely related variables are contained within the same partition. Each component is simulated independently. The interaction between components is modeled using shared, or *boundary*, variables.

The variables are partitioned using a causal analysis [Iwasaki and Simon, 1986, Iwasaki, 1988] of the constraint network. A sub-model is created for each partition containing the constraints between these *within-partition variables. Boundary variables* are causally upstream or acausally related variables directly connected to a within-partition variable through a constraint but not included in the partition. These variables are used to relate the behavior of connected components and are handled specially during the simulation of a sub-model. Constraints connecting within-partition variables and boundary variables are included in the sub-model.

Each sub-model is simulated independently, generating a behavioral description for the within-partition variables. Information about the behavior of the boundary variables is used to constrain the simulation and is obtained from the behavioral description generated by simulating the upstream components

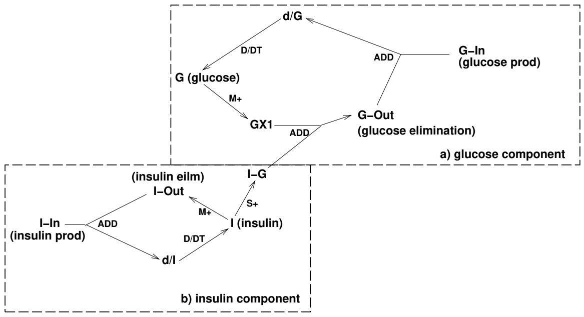


Figure 1: Simplified Glucose Insulin Regulatory System Model

A QSIM model of the human Glucose-Insulin Regulatory System (GIR) which contains two connected feedback loops corresponding to the glucose (a) and insulin (b) regulatory systems.

- The nodes in the graph are variables and the links are QSIM constraints. The arrows indicate the direction of causality derived via a causal analysis of the constraint network.
- The insulin and glucose production (I-In and G-In respectively) are assumed to be constant in this version of the model.

Model decomposition divides the model into two components (identified by the boxes) related by the intermediate variable I-G. A sub-model is created for each component. Each sub-model is simulated separately. Temporal correlation of events is only provided for variables within the same component.

- Since the insulin component is causally upstream, it is simulated prior to and independent of the glucose component. I-G is dependent in this model. I-G's behavior is completely determined during this simulation.
- The glucose sub-model includes I-G as an independent variable. Its behavior is constrained by the results from the simulation of the insulin sub-model. The behavioral description of the glucose component is prevented from branching on distinctions in the value of I-G.

containing these variables. This description guides the behavior of the boundary variables during the simulation of the downstream sub-model. The behavioral description generated during the simulation of a sub-model branches only on distinctions in the values of the within-partition variables. Distinctions in the boundary variables are eliminated through abstraction.

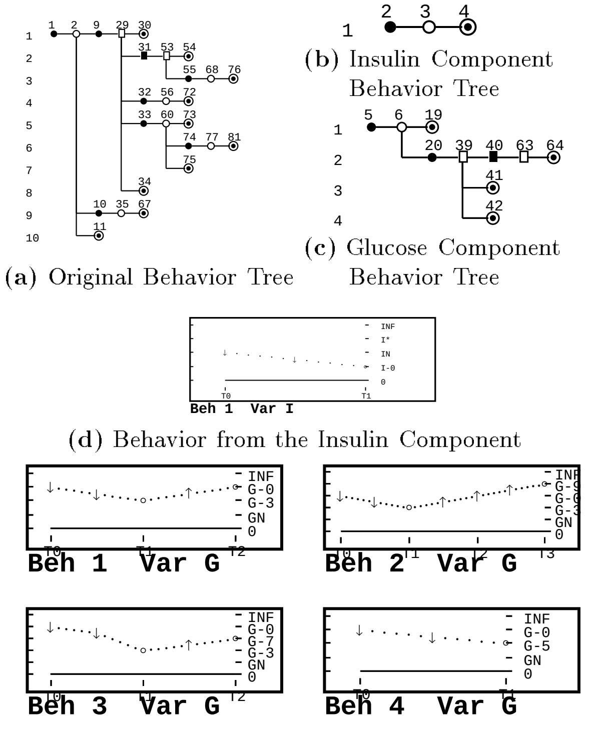
The order of simulation for the individual sub-models is determined by a causal analysis starting with the causally upstream components. If a feedback loop exists between the components or two components are acausally related, then a more complicated simulation strategy must be employed in which the components are simulated concurrently.

3 A Model of the Glucose Insulin Regulatory System

Figure 1 shows a constraint network modeling the human Glucose-Insulin Regulatory system (GIR). This model was developed by Ironi and Stefanelli [1993] using the Qualitative Compartmental Modeling Framework (QCMF) designed to assist the user in formulating models of a pathophysiological system and in analyzing their behaviors. The GIR model contains two separate feedback loops. One controls the amount of glucose (G) within the body and the other controls the amount of insulin (I). The feedback loops are connected through the intermediate variable I-G which relates the amount of insulin to the rate of glucose elimination.

Figure 2 shows the results from a simulation of the GIR model using model decomposition. The GIR model is divided into two separate sub-models corresponding to the two feedback loops related by the intermediate variable I-G. This variable is included in both sub-models; however, its behavior is completely determined by the insulin feedback loop. I-G is a dependent variable in this system and an independent variable in the glucose system.

Since the insulin sub-model is causally upstream, it is simulated prior to the glucose component, deriving a complete behavioral description for the variables within the insulin feedback loop including the variable I-G. Next, the glucose sub-model is simulated. The behavior of I-G, determined during the simulation of the insulin component, is used to constrain the possible values of I-G during this simulation. Branches caused by distinctions in the value of I-G are eliminated via abstraction during the simulation of the glucose component.



(a) Original Behavior Tree

(b) Insulin Component Behavior Tree

(c) Glucose Component Behavior Tree

(d) Behavior from the Insulin Component

(e) Behaviors from the Glucose Component

Figure 2: Simulation of the GIR Model

A normal QSIM simulation of the GIR model using chatter box abstraction [Clancy and Kuipers, 1993] results in 10 behaviors (a). Chatter box abstraction selectively eliminates distinctions in the derivative of the chattering variables thus making the simulation tractable. Chattering regions are identified by a square within the behavior tree (e.g. states 29 and 53).

- Within the 10 behaviors generated, there are four distinct behaviors for the amount of glucose and only a single unique behavior for the amount of insulin.
- The behavior tree provides a complete temporal ordering of the glucose component and the insulin component events even though they are unrelated. These distinctions add complexity to the simulation and obscure the relevant behaviors of the individual components.

Model decomposition generates a behavior tree for each component (b & c) describing the same behaviors as the standard QSIM simulation. No temporal correlation is provided between these behaviors.

- The simulation of the insulin component is performed first and results in a single unique behavior in which the amount of insulin decreases from its initial value and then reaches steady (d).
- Simulation of the glucose component results in four unique behaviors (e). The behaviors are distinguished by the final amount of glucose with respect to the amount at the beginning of the simulation. The behavior is constrained by the results from the simulation of the insulin component.

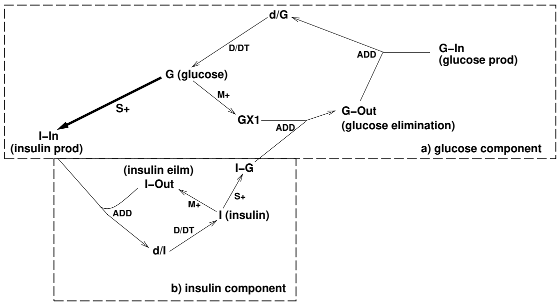


Figure 3: More Complex Model of the Glucose-Insulin Regulatory System

This version of the glucose-insulin regulatory system includes a feedback connection from the glucose component to the insulin component. An S+ constraint relates the amount of glucose to the insulin production rate. I-In is no longer constant.

- The rate of insulin production (I-In) is now included within the glucose sub-model since it is a dependent variable with respect to this component.
- The simulation of the glucose component depends upon the value of I-G while the insulin component depends upon the value of G.
- Since neither component is causally upstream, they must be simulated concurrently.
- During the concurrent simulation, the simulation of the other component is assumed to be complete. States are marked inconsistent if a state cannot be found in the other behavior tree satisfying the constraints relating the components. If a state satisfying this constraint is created later, the inconsistent state is then marked consistent and simulated.

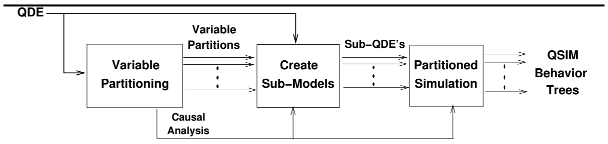


Figure 4: Overview of the Model Decomposition Algorithm

Variable Partitioning generates a causal analysis of the model and partitions the variables based upon this analysis. The partitioning identifies tightly connected components within the model to minimize irrelevant temporal correlations.

Sub-Model Creation takes the variable partitions as input and generates a sub-model for each partition. The causal analysis is used to determine which causally upstream variables affect each sub-model.

Partitioned Simulation determines the order of simulation for the sub-models from the causal analysis and performs the simulation generating a behavior tree for each sub-model.

3.1 A More Complex Model

Figure 3 contains a more complex model of the GIR system. In this model, the rate of insulin production depends upon the amount of glucose. Thus, a feedback loop exists between the two components. The rate of insulin production is a dependent variable with respect to the glucose component. No causal ordering exists between the connected components, so they must be simulated concurrently. The simulation of each sub-model depends upon the behavior of a variable in the other component. The glucose component depends upon I-G and the insulin component depends upon G.

A concurrent simulation alternates between extending the behavioral description of each component. Since the components are simulated concurrently, the behavior of the boundary variables is incomplete during the simulation of an individual sub-model. A truth maintenance system is used to address this problem. During the simulation of a sub-model, it is assumed that the simulation of related components is complete and the behavior of the boundary variables is known. If a state is marked inconsistent because the boundary variables do not satisfy the constraints relating the components, a *dependency* is recorded. A dependency records information about the inconsistent state and the condition not satisfied by the incomplete behavioral description of the related component. If this condition is met as the simulation is extended, then the state marked inconsistent is reinserted into the behavior tree and simulated. The details of this algorithm are addressed in the next section.

4 The Algorithm

The model decomposition and simulation algorithm is based on the QSIM [Kuipers, 1984, Kuipers, 1994] qualitative reasoning paradigm. It can be adapted to work with other qualitative simulators. It accepts a QSIM QDE as input and generates a set of behavior trees. Each behavior tree corresponds to one of the variable partitions. Figure 4 provides an overview of the three main components of the algorithm: variable partitioning, sub-model creation, and model simulation.

4.1 Variable Partitioning

Model decomposition requires a partitioning of the variables into components. In general, this partitioning should combine variables which are tightly connected within the constraint network and separate variables whose behavior is unrelated. The ability to perform the simulation and eliminate temporal correlations is independent of the variable partitioning selected. Model decomposition and simulation provides the same soundness guarantees as a standard QSIM simulation [Kuipers, 1986]. The partition selected only affects the complexity of the resulting simulation and the level of temporal correlation provided.

We use an adaptation of Iwasaki and Simon's [Iwasaki and Simon, 1986, Iwasaki, 1988] causal ordering algorithm to obtain a causal analysis of the model. In certain situations, the partitioning can be derived directly from the causal ordering as in the model in figure 1. In other situations (like the more complicated example in figure 3), selecting the appropriate partitioning can be more difficult. A set of heuristic rules may be required to guide the selection of the partition. A general purpose algorithm to automatically choose the best partitioning is still being developed. Currently, the partitioning is performed by hand. Automatically identifying an effective partitioning will facilitate the integration of these abstraction techniques with automatic model building and query answering systems [Rickel and Porter, 1994].

4.2 Creating Sub-Models

A sub-model is created for each variable partition describing the relationships between the variables of the partition. This sub-model, or *sub-QDE*, is used to simulate the component. There are two types of variables contained within each sub-model.

Within-partition variables are the variables belonging to the partition. These are the variables of interest for this sub-QDE.

Boundary variables are non-partition variables that are related to a within-partition variable through a constraint and are causally upstream or acausally related to that variable. These variables are independent with respect to the sub-QDE. Variables which are causally downstream from a within-partition variable are excluded because these variables act as dependent variables.

A sub-QDE is comprised of the constraints within the main QDE which contain only within-partition and boundary variables. Since the sub-model is concerned with the behavior of the within-partition variables, only those constraints which restrict these

Glucose Component	Insulin Component
Within–Partition Variables G, d/G, G–In, G–Out, Gx1, I–In	**Within–Partition Variables** I, d/I, I–Out, I–G
Boundary Variables I–G	**Boundary Variables** I–In
Constraints (M+ G Gx1) (ADD Gx1 I–G G–Out) (ADD d/G G–Out G–In) (D/DT G d/G) (Constant G–In)	**Constraints** (M+ I I–Out) (S+ I I–G) (ADD d/I I–Out I–In) (D/DT I d/I) (S+ I G)

Figure 5: Partitioning of the More Complex GIR Model

The GIR model in figure 3 is partitioned into a sub-model for the insulin feedback loop and one for the glucose feedback loop. The simulation of each component will derive the behaviors for the within-partition variables. The behavior of the boundary variable is obtained from the simulation of the other component and used to constrain the simulation.

- In the glucose sub-model, I-G is the only boundary variable. It is related to G-Out through an ADD constraint and is causally upstream from G-Out. I-In is not included as a boundary variable since it is causally downstream.
- In the insulin sub-model, I-In is the only boundary variable. It is related to I-Out and d/I through an ADD constraint.

values are included. Figure 5 shows this partitioning for the more complex example discussed above. All constraints within the main QDE are included in at least one sub-QDE. If there is any causal ordering between variables from different partitions which share a constraint, then this constraint is only included in the causally downstream sub-model. Thus, constraints are included in multiple sub-QDE's only when the variables within the constraint are acausally related and belong to different components.

4.3 Partitioned Simulation

A separate QSIM simulation is performed for each sub-model. Each simulation generates a state-based behavioral description for the within-partition variables of the sub-model. The description of a boundary variable's behavior generated by an upstream sub-model is used to guide the simulation of a downstream component. Branches due to distinctions in the values of the boundary variables are eliminated through abstraction. Two or more sub-models must be simulated concurrently when a feedback loop exists between these components. The following three sections elaborate on each of these steps.

4.3.1 Boundary Variable Behavior Guide

During the simulation of a component, boundary variables are viewed as exogenous variables whose behavior is completely determined by the upstream components to which they belong. The constraints which restrict the behavior of these variables are contained within these upstream sub-models. The constraints which exist between the boundary variables and the within-partition variables in the current sub-model serve to restrict the behavior of the within-partition variables. By restricting the behavior of the boundary variables to match the description provided by the upstream sub-models, no constraining power is lost with regards to the within-partition variables. (i.e. The behaviors generated for the within-partition variables are the same ones produced by the standard QSIM algorithm.)

The boundary variables of a sub-model are grouped into sets according to the components to which they belong. A *guide tree* is provided for each set describing the behavior of the boundary variables within the set. These trees are derived during the simulation of the upstream components.

The guide trees are used to determine the valid successor values for the boundary variables during the simulation. A mapping is maintained between states in the current simulation and matching states within the guide trees. A state S_g within a guide tree *matches* a state S in the current simulation if and only if:

- S_g *includes* S (defined below) with respect to the boundary variables,[1] and
- the predecessor of S matches either S_g or the closest ancestor of S_g that differs from S_g in the value of at least one current boundary variable.[2]

S_g *includes* S with respect to a set of boundary variables if the state space described by S for the boundary variables is a subset of the space described by S_g. The following requirements must be met.

- S_g must be a time-interval state or S must be a time-point state. A time-interval state within the current simulation cannot be matched against a time-point state in the guide tree.
- For each boundary variable b guided by S_g, the region of the state space described by $qval(b, S)$ must be equal to or a subset of the region described by $qval(b, S_g)$. $Qval(b, s)$ is the qualitative value of variable b in state s.

A new state is marked inconsistent if a valid matching state cannot be identified within each guide tree. This algorithm ensures that every behavior for a boundary variable within the current behavior tree matches a behavior in the appropriate guide tree.

This technique for restricting the behavior of a set of variables based upon a predefined behavioral description has been generalized to allow a modeler to control the behavior of any exogenous variable within a QSIM simulation.

4.3.2 Abstracting Boundary Variable Distinctions

A standard QSIM simulation provides a total ordering of events for all variables contained within the model. In a partitioned simulation, each sub-model contains both within-partition and boundary variables. Since each sub-model is only required to provide a behavioral description for the within-partition variables, abstraction is used to eliminate branches caused by distinctions in the values of the boundary variables. This eliminates the temporal correlations between within-partition variables and boundary variables during the simulation of a sub-model.

The abstraction process is performed during the simulation after the successors of a state are computed. Equivalent successor states with respect to the within-partition variables are combined to form a single abstract state. There are two main steps in the creation and simulation of abstract states:

- creating an abstract state from a set of detailed (i.e. non-abstracted) successor states
- computing the successors of an abstract state.

Creating an Abstract State An abstract state contains a unique value for each within-partition variable. Qualitative value information for the boundary variables is maintained in the form of a disjunctive list of *sub-states*. Each sub-state contains a complete set of values for the boundary variables. A sub-state is created for each detailed state used to create an abstract state. The information in the sub-states is used to ensure that no constraining power

[1] These are the boundary variables for the sub-model currently being simulated. They are within-partition variables from the perspective of the upstream models which were used to generate the guide trees.

[2] The guide trees provide a behavioral description for a number of variables besides the current boundary variables. Branches may exist which do not reflect changes in these boundary variables. Thus, the closest ancestor that differs from S_g in the value of at least one current boundary variable is the predecessor of S_g from the perspective of these variables.

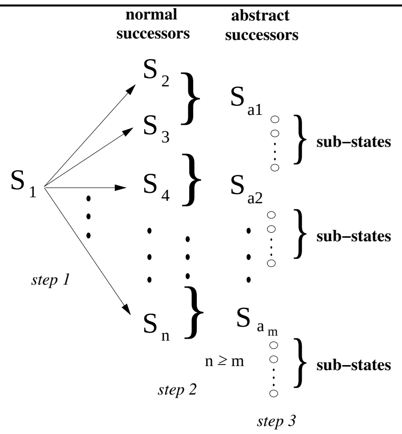


Figure 6: Eliminating Branches through Abstraction

Branches due to distinctions in the boundary variable values are eliminated by combining successors which are equivalent with respect to the within-partition variables. Abstract successors states are created as follows:

1. Create the standard QSIM successor states. The consistency of these successors is determined using the QSIM consistency filters.
2. Using only the consistent successors, create abstract successor states by combining states with equivalent values for the within-partition variables.
3. For each abstract state, create a set of sub-states containing possible qualitative values for the boundary variables. Each sub-state provides a unique set of values for the boundary variables.
4. Replace the original successors with the abstract successors within the behavior tree.

is lost when the successors of an abstract state are created. Figure 6 provides additional information about the state abstraction algorithm.

The boundary variable behavior guide described in the previous section is actually applied to sub-states. A sub-state is marked inconsistent if it cannot be matched against a state within each of the boundary variable guide trees. An abstract state is considered inconsistent if all of its sub-states are marked inconsistent. A mapping is maintained between a sub-state and the matching states within the guide trees.

Creating Successors of an Abstract State The abstraction process eliminates distinctions in the boundary variables which are normally used by QSIM when computing a state's successors. The algorithm used to create the successors of a state during a simulation has been modified to handle an abstract state. These modifications retain the QSIM soundness guarantee as well as the constraining power of a standard QSIM simulation. The successors of an abstract state are computed as follows:

1. Continuity is used to determine the possible successor values for each variable within the model.
   - For within-partition variables, use the unique qualitative value provided within the abstract state to compute the set of valid successor values. This is the normal algorithm used by QSIM for all of the variables.
   - For boundary variables, apply the normal QSIM algorithm to each of the possible values included in the sub-states and then take the union of these possible successor values.
2. Use the standard QSIM algorithm to create and filter potential successor states from this set of possible values. Each state will contain a single qualitative value for each variable within the sub-model (i.e. both within-partition and boundary-variables).
3. For each successor state, perform a continuity test to ensure that the boundary variable values can be reached from at least one sub-state maintained by the abstract state.[3]
4. These successor states are then used to form the abstract successor states as described in the previous section.

The basic QSIM algorithm generates all possible behaviors of the modeled system. Step 1 in this algorithm ensures that this guarantee is retained. The set of possible successor values for each variable is the union of the possible successor values of the individual non-abstracted successor states. Thus, all of the successor states which would have been computed by the standard QSIM algorithm for the non-abstracted states are included in the set of successors for the abstract state.

This abstraction technique also retains all of the constraining power of the standard QSIM algorithm. No additional behaviors are generated due to the elimination of distinctions in the boundary variables via abstraction. Figure 7 shows how each successor of

[3] Since the union of the possible successor values for the boundary variables is used (in step 1), it is possible that these successor values may combine in ways that would not have been possible with the non-abstracted states. This test ensures that a possible successor value that is valid for only one of the sub-states does not combine with a value from a different sub-state.

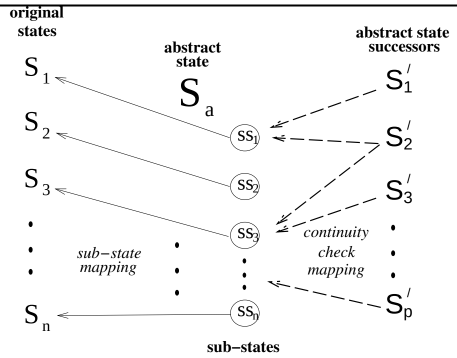


Figure 7: Relationship between an abstract state's successors and the original states.

Original states S_1 through S_n were combined to form the abstract successor S_a. A sub-state (ss_1 through ss_n) was formed for each of these original states. S_1' through S_p' are the successors of the abstract state.

- A one-to-one correspondence (the sub-state mapping) exists between the sub-states attached to an abstract state and the original successors states used to create the abstract state.
- A continuity check (the third step in the algorithm) ensures that each successor of an abstract state can be reached from at least one of the sub-states. One abstract state successor can be a successor for multiple sub-states. This mapping information is not retained during the simulation.

Each abstract state successor can be mapped back to the original states which it would have succeeded if the abstraction were not performed. Thus, no additional successors are generated due to the abstraction process.

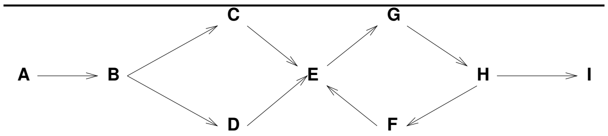


Figure 8: Idealization of Model Decomposition Applied to a Large Model

Model decomposition in conjunction with other abstraction techniques will allow larger models to be simulated producing comprehensible results. These models will be divided into a number of interrelated components. This graph shows an idealization of how the decomposition of a large model would be simulated.

- The nodes of the graph represent components (i.e., a set of variables and constraints) within the model while the arcs are causally directed relationships between components.
- Model decomposition begins simulation with the causally upstream components and then moves through the graph. In this example, component simulation would be performed in the following order:
  - component A
  - component B
  - components C and D (but they do not have to be simulated concurrently)
  - components E, F, G, and H are all simulated concurrently since they form a feedback loop
  - component I

an abstract state would have followed from at least one of the original non-abstracted states.

Problems may be encountered when applying extensions of the QSIM algorithm to an abstract state. Some constraining power may be lost due to the inability to apply certain constraints such as the energy filter or some of the quantitative reasoning techniques to the boundary variables. In most cases, this information would have been incorporated into the upstream sub-model that originally determined the behavior of the boundary variable. Otherwise, problems such as this can be addressed by the initial variable partitioning.

4.3.3 Simulation Control and Concurrent Simulation

Simulation of a causally downstream component requires information from the upstream sub-model about the behavior of the boundary variables. Once the variables are partitioned, the causal analysis can be used to create a component graph. The nodes in this graph are the sub-models while the links show the direction of causal influence relating these nodes. Some components may be acausally related. The sub-models are simulated in the order dictated by this graph starting with the causally upstream components. Figure 8 shows an idealization of this process for a larger model.

If a feedback loop exists between components (as in figure 3) or if components are acausally related, it is not possible to sequentially order their simulation since they require boundary variable information from each other. In this case, these simulations must be performed concurrently. Since the complete behavior of the boundary variables is not available during the simulation of a sub-model, a truth maintenance system is used to record information about any assumptions which are made.

A concurrent simulation alternates among extending the simulation of each of the related components. A *dependency* is recorded when a state is marked inconsistent due to the boundary variable behavior guide and the behavior of the boundary variable is incomplete. The dependency records information about the condition required to satisfy the consistency check. As the behavior of the related component is extended, the dependencies are checked to determine if any of the conditions are satisfied by the additional behavioral information. If a dependency is satisfied, then the state which had previously been marked inconsistent is reinserted in the behavior tree and simulated. A dependency is satisfied when a state is created in a related tree fulfilling the conditions which originally caused the dependency to be created (i.e. providing the specified values for the boundary variables).

The dependencies are checked to see if any violations have occurred after each sub-model has been extended one time step. Dependencies are cross checked against each other to ensure that a deadlock condition does not occur (i.e. two states in different components are marked inconsistent because they are each waiting for the other to occur.) Some of the details of the concurrent simulation algorithm are still being developed to ensure that the desired generality is provided.

5 Alternative Abstraction Techniques

Model decomposition provides a general framework for performing abstraction during the simulation of a large model. The decomposition of a model is independent of the simulation. Furthermore, the behavior of each component is reasoned about separately from the interactions between components. This architecture allows other decomposition methods to be applied to provide alternative abstraction techniques.

Iwasaki and Bhandari [1988] build upon Simon's [1961] techniques for variable aggregation in dynamic structures. They provide a formal analysis of how variables within a system of equations can be partitioned and aggregated based upon a quantitative analysis of the equations and their roots. Some of these techniques could be extended to address the partitioning and aggregation of variables in a qualitative model. This type of analysis could be used to extend the partitioning techniques currently applied.

Raiman and Williams [1992] describe a method of identifying dominant behaviors which are used to divide the state space of a model into regions in which different behaviors dominate. Order of magnitude reasoning is used to simplify the analysis of each region. A similar technique could be used when partitioning the variables to simplify the constraint network and minimize the interactions between components. A separate simulation of the model would be performed for each of the regions identified.

Kuipers' [1987] time-scale abstraction groups variables based upon the relative speed of the mechanisms controlling their values. Variables controlled by slower mechanisms are considered constant when simulating the faster mechanisms, while the faster mechanisms are viewed as instantaneous with respect to the slower mechanisms. Time scale information could be used in partitioning the variables to automate this abstraction technique.

6 Conclusions and Future Work

State-based qualitative reasoning techniques require a total temporal ordering of events within the behavioral description. This can lead to irrelevant temporal distinctions which increase the complexity of the simulation and obscure the relevant behavior. Before developing heuristics to determine which temporal correlations are relevant to the current task, simulation techniques must be developed to allow for a range of temporal distinctions. Model decomposition and simulation provides an architecture which does not require many of the temporal orderings specified in a traditional qualitative simulation.

Model decomposition has been applied successfully to a number of examples. Two of these examples are discussed in this paper. The simulation techniques used are separate from the method used to decompose the model. Our objective is to ensure that the constraining power of a standard qualitative simulation is retained by these simulation techniques regardless of the variable partitioning selected. This would allow the decomposition algorithm to select a partitioning which highlights distinctions relevant to the current task. Due to the dependency information which must be maintained during a concurrent simulation, model decomposition may increase the computational complexity of a simulation if the granularity of the partitioning is too small. This information must also be taken into account when decomposing a model.

Extensions to the model decomposition and simulation technique are still required to provide the generality that is desired. In particular, this technique has yet to be applied to large, multi-component models with complicated interaction patterns. This research is currently being extended in a number of ways.

- The details of the concurrent simulation algorithm must be developed further to ensure that the variable partitioning selected does not affect the constraining power of the simulation.
- Model decomposition requires a partitioning of the variables into components. This partitioning determines the temporal correlations which are provided by the simulation. Techniques for automatically selecting the optimal partitioning for a given model and task are being investigated.
- Model decomposition provides an architecture which reasons about the behavior of each component separately from the interactions between the components. This architecture allows other decomposition methods to be applied to provide alternative abstraction techniques. In particular, we are interested in incorporating order of magnitude and time-scale information when performing the variable partitioning.
- A complexity analysis of the simulation algorithm must be performed and compared against a standard QSIM simulation to evaluate its effectiveness at reducing the computational complexity introduced by these irrelevant temporal distinctions.

Acknowledgments

This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8904454, IRI-9017047, and IRI-9216584, and by NASA contracts NCC 2-760 and NAG 9-665.

References

[Clancy and Kuipers, 1992] D. J. Clancy and B. J. Kuipers. Aggregating Behaviors and Tractable Simulation. In *AAAI Design from Physical Principles Fall Symposium Working Notes*, pp 38-43, Cambridge, MA, 1992.

[Clancy and Kuipers, 1993] D. J. Clancy and B. J. Kuipers. Behavior Abstraction for Tractable Simulation. In *Proceedings of the Seventh International Workshop on Qualitative Reasoning about Physical Systems*, pp 57-64, 1993.

[Coiera, 1992] E. W. Coiera. Qualitative Superposition. In *Artificial Intelligence*, 56:171-196, 1992.

[De Kleer and Brown, 1984] J. De Kleer and J. S. Brown. A Qualitative Physics Based on Confluences. In *Artificial Intelligence* 24:7-83, 1984.

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

[Fouché and Kuipers, 1991] P. Fouché and B. J. Kuipers. Towards a Unified Framework for Qualitative Simulation. In *Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, 295-301, 1991.

[Hayes, 1985] P. J. Hayes. The Second Naive Physics Manifesto. In J. Hobbs and B. Moore (Ed), *Formal Theories of the Commonsense World*. Ablex Publishing Corp.

[Ironi and Stefanelli, 1993] L. Ironi and M. Stefanelli. A Framework for Building Qualitative Models of Compartmental Systems. Technical Report N. 897, Instituto Di Analisi Numerica, Pavia, Italy, 1993.

[Iwasaki, 1988] Y. Iwasaki Causal Ordering in a Mixed Structure. In *Proceedings from the Seventh National Conference on Artificial Intelligence, AAAI-88*, pp 313-318, 1988.

[Iwasaki and Simon, 1986] Y. Iwasaki and H. A. Simon. Causality in Device Behavior. In *Artificial Intelligence*, 19, 1986.

[Iwasaki and Bhandari, 1988] Y. Iwasaki and I. Bhandari. Formal Basis for Commonsense Abstraction of Dynamic Systems. In *Proceedings from the Seventh National Conference on Artificial Intelligence, AAAI-88*, pp 307-312, 1988.

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

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

[Kuipers, 1987] B. J. Kuipers. Abstraction by Time-Scale in Qualitative Simulation. In *Proceedings from the Sixth National Conference on Artificial Intelligence, AAAI-87*, 1987.

[Kuipers, 1994] B. J. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge.* Cambridge, MA: MIT Press, 1994. In press.

[Raiman and Williams, 1992] O. Raiman and B. C. Williams. Caricatures: Generating Models of Dominant Behavior. In *Proceedings of the Sixth International Workshop on Qualitative Reasoning about Physical Systems*, 1-6, 1992.

[Rickel and Porter, 1994] J. Rickel and B. Porter. Automated Modeling for Answering Prediction Questions: Selecting the Time Scale and System Boundary. To appear in *Proceedings of the Twelfth National Conference on Artificial Intelligence* (AAAI-94), 1994.

[Simon and Ando, 1961] H. A. Simon and A. Ando. Aggregation of Variables in Dynamic Systems. In *Econometrica*, Vol 29, 1961.

[Williams, 1986] B. C. Williams. Doing time: Putting qualitative reasoning on firmer ground. In *Proceedings of the American Conference on Artificial Intelligence, AAAI-86*, pp 105-113, 1986.