In *Working Papers of the Tenth International Workshop on Qualitative Reasoning about Physical Systems (QR-96)*
Menlo Park, CA: AAAI Press, 1996.

Comprehending Complex Behavior Graphs through Abstraction

Richard S. Mallory **Bruce W. Porter** **Benjamin J. Kuipers**
Department of Computer Sciences, The University of Texas at Austin, Austin TX 78712
{mallory, porter, kuipers}@cs.utexas.edu

Abstract

Qualitative simulation is often a useful tool for studying the behavior of physical systems and has promise for providing automatic explanations of their behavior. However, in some cases it can overwhelm with detail. Behavior graphs with hundreds or thousands of states may obscure the basic patterns of behavior that a qualitative model was intended to explore. This paper describes an approach to comprehending complex behavior graphs by abstracting the behavior graph according to user-specified criteria that are simple and natural to provide. We present properties that an abstraction should meet to be faithful to the original behavior graph, prove necessary and sufficient operational conditions for an abstraction method to maintain these properties, and present a simple algorithm that incorporates these conditions and works for any behavior graph. We demonstrate on several externally-provided problems that our method can greatly simplify complex behavior graphs in number of states and behaviors while remaining faithful to the original behavior graph. It enables simple graphical output that can reveal underlying patterns of behavior that have not been apparent with previous methods, and shows promise for expanding the utility of qualitative reasoning for generating explanations.

Introduction

Qualitative simulation has proved a useful tool for studying the behavior of dynamic systems. Given a differential equation model of a system, state-based qualitative simulation produces a set of states describing the system's behavior over time. See generally (Forbus 1984; de Kleer & Brown 1984; Kuipers 1986; Kuipers 1994; Weld & de Kleer 1990). The incomplete knowledge of equations and variables employed in qualitative simulation is useful when a sketch of a system's behavior is sufficient or more precise information is not available. However, imprecise knowledge leads to ambiguity; behaviors branch whenever a state has more than one possible successor. A few branches may be comprehensible, but state-based qualitative models of modest size can produce several tens or hundreds of behaviors. This often renders the results incomprehensible. Appropriate methods for achieving abstraction and summary are required to reveal the fundamental characteristics of the behaviors that the user may most wish to explore. Abstraction and summary are also important to providing succinct descriptions and automatic explanations.

Quantitative information can sometimes be used to reduce or eliminate branching. However, quantitative information may be difficult to find or inconvenient to invent and its use may inadvertently eliminate branching that the user would like to reveal. In domains such as the life sciences, numeric information is unavailable or too incomplete to be useful. Our work in particular has been pursued in the context of a large knowledge base on plant physiology (Porter et al. 1988; Lester 1994), where many useful conclusions can be drawn without quantitative information (Rickel & Porter 1992; Rickel & Porter 1994). Thus our simulations, like many others (Ironi & Stefanelli 1994), are purely qualitative.

The complexity of behaviors is often due to uninteresting distinctions among qualitative states, and a number of methods have been developed to abstract away these distinctions. For Qsim simulations (Kuipers 1986; Kuipers 1994), these methods include chatter elimination (Clancy & Kuipers 1993), behavior aggregation (Fouché & Kuipers 1991; Clancy & Kuipers 1993), and model decomposition (Clancy & Kuipers 1994). Each identifies and reduces or eliminates a specific class of uninteresting distinctions. These methods make important or essential contributions to the tractability of qualitative simulation. Chatter elimination, in particular, is essential for many Qsim simulations. However, even with these techniques and others a qualitative model of modest size may still produce an incomprehensibly large behavior graph. There often remains much to be done to make the results of simulation clear to the human user.

This paper approaches behavior abstraction from a different perspective. Previous approaches have identified particular sources of distinctions likely to be uninteresting in any context and abstracted them away. In contrast, our algorithm abstracts away *all* distinctions *except* those selected by the user to be preserved in an abstracted behavior graph. As will be clear from an example, the criteria for such distinctions are natural and simple to supply. Typically, the user will want several different abstractions that highlight the behavior of selected individual variables or pairs of variables.

The simulation must first be completed,[1] so other techniques that render simulation tractable are essential for our algorithm. Our algorithm builds an abstract behavior graph from an envisionment graph by abstracting together adjacent states that, by the user's criteria, need not be distinguished. This will greatly simplify a complex behavior graph and often reveals fundamental patterns of behavior that have heretofore been difficult to discern. In the example we present in detail, it abstracts a behavior graph with 188 states and 63 behaviors to graphs with nine states and three or five behaviors which plainly show the oscillatory behavior of the variables of interest and their phase relationship. Moreover, abstractions are provably faithful to the original behavior graph in that every path through the abstract graph corresponds to some path through the original graph and vice versa. High-level abstractions such as these are essential to automatically providing concise, natural language explanations from qualitative models, which is our ultimate objective.

In the following sections, we discuss the utility of behavior abstraction, including criteria required from the user; the details of the abstraction algorithm, including a formal definition and proof of faithfulness; tests of the algorithm on several examples; and future research, including automatically deriving the abstraction criteria now required of the user.

The Utility of Behavior Abstraction

For our purposes, a good abstraction will reveal important underlying patterns in the behavior graph. However, our algorithm does not look for such patterns directly. Instead, it relies on the user to provide criteria that describe the information to be presented in an abstraction. These criteria are often quite simple and easy to provide. The utility of this approach is easier to see if we start with an example and then generalize the results.

We will use the model of the glucose-insulin regulatory system in the human body presented in Figure 1. See (Ironi & Stefanelli 1994; Clancy & Kuipers 1994). The model contains the amounts of glucose and insulin in the blood and the interactions between them that affect their respective rates of production and elimination. It was simulated with Qsim by perturbing it from an equilibrium state so that glucose was above the equilibrium value but decreasing and insulin was

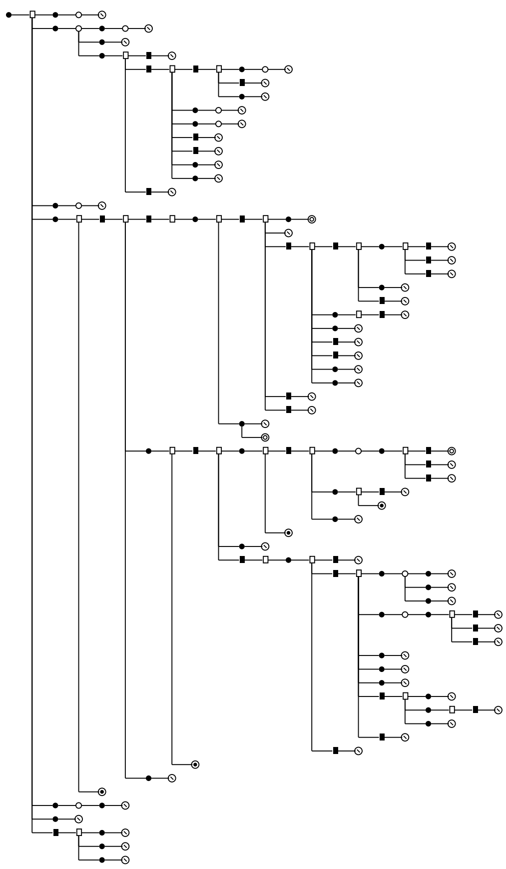

Figure 2: Qsim behavior graph for the glucose-insulin model in the previous figure, perturbed from an equilibrium state as mentioned in the text.

at the equilibrium value but increasing. Chatter elimination and other techniques were used to reduce the number of states. The resulting behavior graph contained 188 states in 63 behaviors with a large number of cycles and four identical final states. See Figure 2. The values of variables can be graphed for individual behaviors, but with this many behaviors it is difficult to get a global view of what is happening.

However, it is possible to ask focused questions about the behaviors. There are usually a few variables whose behavior is of more interest than others. These are typically the state variables (those whose derivatives appear in the model) and that is the case in the glucose-insulin model, where the amounts of insulin and glucose were of primary interest. Given this focus, we might ask for a graph of the behavior of just the amount of glucose, or just the amount of insulin. These graphs may be derived from the behavior graph for the whole model by merging adjacent states that do not differ in the qualitative value of the variable of interest. Two states are "adjacent" if one is the successor of the other or if they share common immediate predecessors or successors; we merge them when they

[1] The algorithm is currently implemented as a post processor. However, if the simulation cannot be completed using other techniques, abstraction during simulation plus disposal of unneeded states might be useful.

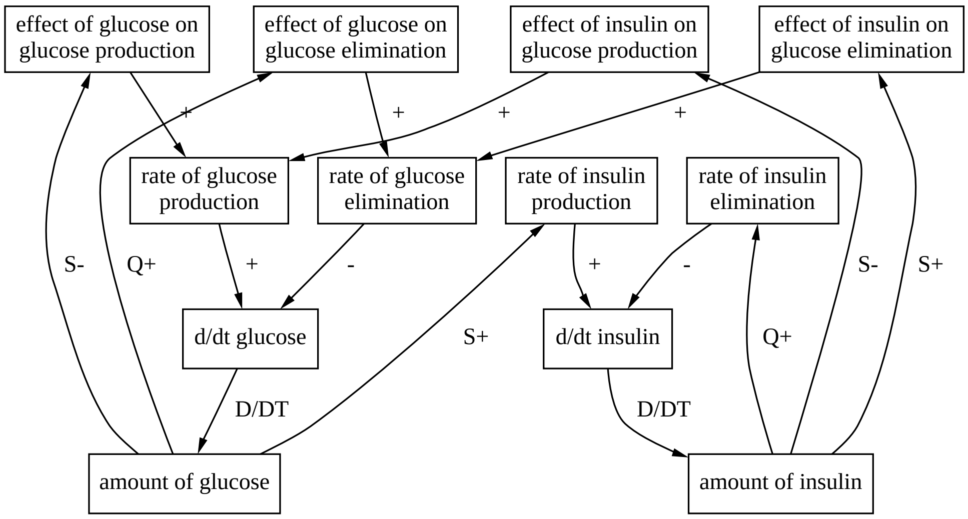

Figure 1: Model of the glucose-insulin regulatory system in the human body. Arrows indicate constraints between variables. D/DT indicates a derivative constraint, Q+ and Q- indicate monotonic functional constraints, S+ and S- indicate functional constraints that are monotonic in the center and flat on the ends, and + and − indicate that the variables at the tails of the arrows sum (with the indicated sign) to the variable at the head of the arrows.

have the same qualitative value for the variable of interest because, from the perspective of this variable, they are identical.[2]

The graphs of the behaviors of glucose and of insulin are shown in Figures 3 and 4. These graphs were produced automatically by our program. The behaviors are far simpler than the behavior of the whole model shown in Figure 2. In particular, they are simple enough that the graph of states has room to display the information that characterizes each state — the qualitative value of the variable in that state, which comprises its qualitative magnitude and direction of change. With this information an understanding of the behavior of each variable can be gleaned. We can see by inspection of Figure 3 that the amount of glucose starts in the initial state (state A-0 in Figure 3) above its equilibrium value (its magnitude in state A-0 is High), and returns to its equilibrium value (Norm) either directly (A-0 → A-8) or after some number of half cycles of oscillation (A-0 → A-1 → A-2 → A-3 → A-4, during which it moves from high to low and back toward normal, and A-4 → A-5 → A-6 → A-7 → A-0, during which it moves from low to high and back toward normal). Similarly, Figure 4 shows that the amount of insulin departs from its equilibrium value, also as specified in the initial state, and then returns to it after some number of half cycles of oscillation.

We can also see how the behaviors of these two variables are related. Figure 5 shows the behavior of the qualitative magnitudes of the amounts of glucose and insulin. We can see that they oscillate in tandem, with the amount of glucose leading the amount of insulin by less than half a cycle, and return to equilibrium at the same time.

In each case, the amount of information required of the user is modest. For Figure 3, the program was instructed to label each state in the original simulation with the magnitude of the amount of glucose, in the terms shown in the figure (that is, whether it was at, above, or below its final equilibrium value), and the direction of change of the amount of glucose. All adjacent states with the same label (magnitude and direction) could then be merged into a single abstract state. What is primarily required of the user is focus — a point of view on what is important to glean from the simulation. We believe that users will typically have a view as to what variables and relationships are most important to understand in a complex simulation and thus will be able to define appropriate abstractions. Since abstractions are simple to specify and the resulting graphs are produced automatically, several abstractions may be run to get a proper overview of the simulation.

These abstractions also provide the kind of overall view of a simulation that is required for the automatic generation of explanations. Our ultimate objective has been to provide natural language explanations of qualitative behaviors to users knowledgeable in the domain of the model but familiar only with the fundamentals of qualitative reasoning. One element of such explanations is the kind of high-level summaries that we gave above in the text, describing the conclusions that can be drawn from the behavior abstractions in Figures 3, 4, and 5. We believe that a good explanation will provide such summary descriptions of a simulation's

[2] This notion is refined below. Technical conditions will restrict certain adjacent states from being merged.

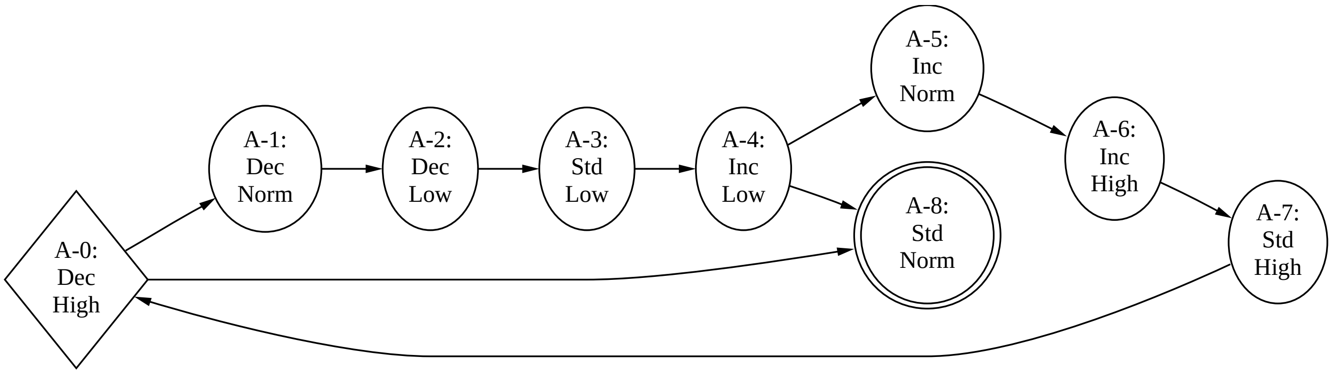

Figure 3: Behavior of the amount of glucose in the glucose-insulin model. The diamond is the initial state and the double circle is the final state. Arrows indicate successor relationships. The items within each state are its state number, the qualitative magnitude of the amount of glucose, and its qualitative direction of change. The magnitudes of "Norm", "High", and "Low" mean that the variable is at, above, or below its eventual equilibrium value, respectively. "Inc", "Dec", and "Std" mean increasing, decreasing, and steady, respectively.

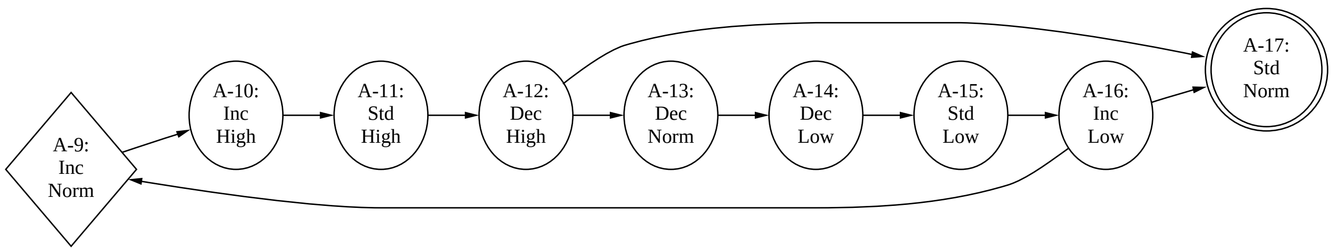

Figure 4: Behavior of the amount of insulin in the glucose-insulin model.

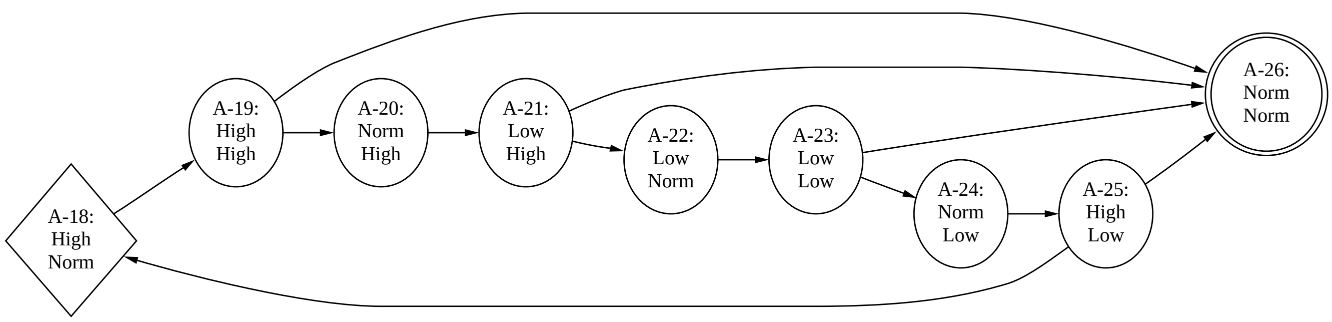

Figure 5: Behaviors of the magnitudes of glucose and insulin in the glucose-insulin model. The items in each state are the state number and the qualitative magnitudes of the amounts of glucose and insulin, respectively.

behaviors, and providing the behavior abstractions on which such summaries can be based is an essential first step. We discuss our research on explanation further below.

In summary, then, behavior abstraction employing user-specified criteria has permitted us to obtain an overview of the behavior of the system that has been difficult to achieve with other tools on a behavior graph of this complexity. In the following section we will formalize the abstraction process, including what it means for an abstraction to be faithful to the original behaviors.

Details of the Abstraction Algorithm

The algorithm requires labeling each state to reflect the distinctions of interest to the user, such as the magnitude and direction of change of a selected variable, and then grouping appropriate sets of adjacent states with the same label into abstract states, which are themselves linked with successor edges to form the abstract behavior. Labeling will be formalized below and finding abstract successors is straightforward. Only the matter of determining which states to collect in an abstract state is subtle. In the following, we will call the states produced by simulation the "base" states, to distinguish them from the abstract states.

States are labeled using one or more methods supplied by the user. A labeling method assigns a label to each state in the simulation. By doing so, it defines a set of distinctions among states — namely distinct labels — that are to be preserved in the abstraction process. The abstractions behind Figures 3, 4, and 5 each had two labeling methods. For Figures 3 and 4, the first method labeled each state with the qualitative magnitude of the amount of glucose or insulin in that state, with a simplification of the magnitude to "High", "Norm", or "Low", depending on whether it was above, at, or below the final equilibrium value. The second method labeled each state with the qualitative direction of change of the amount of glucose or insulin. The combination labels each state with the qualitative value (magnitude and direction of change) of the amount of glucose or insulin, with a simplification of the magnitude. For Figure 5, the two methods labeled each state with the qualitative magnitudes of the amounts of glucose and insulin, respectively, as simplified above. In general, given a set of labeling methods, the program finds each method's label for each state and assigns the state a composite label which is an ordered list of the labels from individual methods. Our program provides a simple language for specifying labels of the kinds mentioned above and a few others. The criteria for Figures 3, 4, and 5 were each specified in three lines of straightforward code. The language may easily be extended to accommodate arbitrary labeling schemes.

Faithful abstraction

An abstraction of the behavior graph groups appropriate sets of base states into abstract states and joins two abstract states with a successor edge when there is a successor edge between two of their respective base states. A path in the base graph corresponds to a path in the abstract graph when each state in the base graph is abstracted by some state in the abstract graph and each state in the abstract graph abstracts at least one state in the base graph.

The question remains, What groupings of base states are appropriate? All the base states in one abstract state should have the same label since we wish the abstraction to preserve differences reflected in different labels. Beyond that, we require an abstract behavior graph to reflect exactly the behaviors present in the base graph — that is, it should exhibit both of the following properties:

Completeness. Any path in the base graph should correspond to some path in the abstract graph.

Soundness. Any path in the abstract graph should correspond to at least one path in the base graph.

We say that any abstract graph exhibiting these properties is a "faithful" abstraction, in that it implies neither the presence of any paths absent from the base graph nor the absence of any paths present in the base graph. There may be occasions when departing from these rules intelligently would permit a useful and more succinct result, but we think such simplifications are likely to be useful as simplifications of faithful abstractions as we describe them. The latter are useful in large measure because they represent all and only the base behaviors of the system simulated, abstracted as requested. We will build our formalization of abstraction on these two properties and return later to the notion of adjacency, which with some restrictions follows from the requirement of faithfulness.

The abstract graph can be constructed by starting with a copy of the base graph and its labels, which exhibits both properties, and applying transformations that preserve these properties. Completeness will be preserved by transformations that merge two abstract states while retaining all the successor edges except those between the merged states. Specifically, a transformation will replace two states A and B by a new state C, delete any edges between A and B ($A \rightarrow B$ or $B \rightarrow A$), and replace A or B with C as the head or tail of any other edges incoming to or outgoing from A or B.

Preserving soundness

The question then is how to preserve soundness. Since the graphs before and after merger of two states A and B are identical except for the replacement of A and B by C and the edge changes mentioned above, the issue is whether there is some path $P \rightarrow C \rightarrow Q$ in the merged graph with no corresponding path

$P \to A \to Q$, $P \to B \to Q$, $P \to A \to B \to Q$, or $P \to B \to A \to Q$ in the unmerged graph. A merger is sound when for every such path $P \to C \to Q$ in the merged graph there is a corresponding path in the unmerged graph. The following theorem states the conditions under which the merger of two states is sound. Following the theorem we discuss how its conditions relate to the rough notion of adjacency that we have mentioned previously.

Theorem 1. Given a directed graph of states and edges as above, the merger of two states A and B into one state C is sound if and only if the following conditions hold. (succs(A) and preds(A) denote the sets of immediate successor and predecessor states of state A, respectively, and '\' denotes set difference.)

1. If $A \to B$, then at least one of the following conditions must hold (the case where $B \to A$ is analogous):
   (a) $B \to A$,
   (b) $\text{succs}(A) \setminus \{B\} \subseteq \text{succs}(B)$, or
   (c) $\text{preds}(B) \setminus \{A\} \subseteq \text{preds}(A)$
2. If $A \not\to B$ and $B \not\to A$, then at least one of the following conditions must hold:
   (a) $\text{preds}(A) = \text{preds}(B)$,
   (b) $\text{succs}(A) = \text{succs}(B)$,
   (c) $\text{preds}(A) \subseteq \text{preds}(B)$ and $\text{succs}(A) \subseteq \text{succs}(B)$, or
   (d) $\text{preds}(B) \subseteq \text{preds}(A)$ and $\text{succs}(B) \subseteq \text{succs}(A)$.

The proof has been omitted to conserve space. A version of this paper including the proof is available by anonymous ftp from host `ftp.cs.utexas.edu` in file `/pub/brewery/mallory/qr96a.ps`. The proof can be sketched briefly as follows. Each of the two cases is proved separately. The conditions for each are negated and the existence of a path in the merged states with no corresponding path in the unmerged states under these circumstances is demonstrated; this shows the necessity of the conditions. The sufficiency of each condition is then shown by assuming the existence of such a path and showing a contradiction with each condition.

Interpreting soundness

Adjacency. The conditions in the theorem break down into two cases, depending on whether or not there is a successor edge joining the two states. Each set has an intuitive interpretation. In case 1, when one state is the successor of the other, the states would seem to be candidates for merger because the simulation has not changed its state in terms of the distinctions of interest. The conditions in the theorem only exclude this merger if it is unsound. Case 2 is clearer if we start with a simplification. When two states share a predecessor, the two paths from the predecessor to one state or to the other do not differ in terms of the distinctions we have specified. They may be thought of as parallel paths to the same abstract state and thus be represented by a single path to one merged state. Similar reasoning applies to a shared successor and to shared sets of predecessors and/or successors. The conditions in the theorem exclude such mergers only when they are unsound. Together, the conditions of the theorem require that states to be merged be adjacent, as we have previously defined that term, and also that their predecessors and/or successors meet certain additional requirements. "Adjacency", then can serve as a rough summary of the requirements for sound merger.

Final states. The soundness property does *not* guarantee that if an abstract state is non-final (i.e., it has successors), then all the base states abstracted into it are also non-final. That is, a non-final abstract state may group together both final and non-final base states. In this case, a path in the base graph entering a final state has a corresponding path in the abstract graph — one that stops at the corresponding abstract state. Also, a path in the abstract graph passing through a state grouping both final and non-final base states has a corresponding path in the base states — it simply will not include any of the final base states.

An abstract state that groups both final and non-final base states may be confusing, as it is usually important to distinguish final and non-final states. However, this is easily remedied by including with other label methods one that labels each base state with "final" or "non-final" as appropriate. Other distinctions may be included in this label, such as whether the state is quiescent or unfinished (meaning that the simulation was terminated before finishing and the state's successors were not computed).

The algorithm

The algorithm for constructing abstract graphs creates the initial abstract graph as a copy of the base graph and its labels and then repeatedly makes sound mergers in the behavior graph until no more opportunities for sound mergers can be found. Specifically, it first merges all final states with the same label, since all final states have the same set of successors — the empty set. It also merges all initial states (if there is more than one) with the same label. Then, for each state A in the abstract graph, it (1) examines each successor B of A to determine whether A and B can be soundly merged under the conditions in part 1 of theorem 1, (2) it examines each pair of successors of A and each pair of predecessors of A to determine whether any of these pairs can be soundly merged under the conditions in part 2 of theorem 1, and (3) it performs any sound mergers identified. All the conditions to be checked in each of these steps are local to the states under consideration (they involve no search) so they are polynomial in the number of edges and states in the graph. The iteration itself is also polynomial in the number of states and edges in the graph. Finally, the iteration over each state is repeated, with no particular ordering of states, until no more mergers can be performed. Since each merger reduces the number of states in the graph, this outer iteration is bounded by the number of states in the base graph. For the glucose-insulin example, the algorithm runs in approximately one second on a DEC Alpha.

Evaluation

We have applied our behavior abstraction algorithm to several Qsim simulations with behavior graphs ranging in complexity from a relatively modest graph of 36 states and 10 behaviors to a very large graph that remained unfinished at 3874 states and more than 1200 behaviors. See Table 1. Our algorithm was not designed for any of these simulations — they were selected after its design and implementation simply for having fairly or very large behavior graphs. In each case we constructed abstractions similar to those we presented above for the glucose-insulin model by abstracting the qualitative values of each state variable in the simulation, and by occasionally abstracting the qualitative magnitudes of pairs of such variables. In each case, the behavior graph was significantly simplified, as Table 1 shows. The reduction factor in the numbers of states and behaviors was larger for larger graphs, as might be expected.

Of course, the number of states and behaviors can always be reduced further by abstracting further and retaining fewer distinctions among states, and thus presenting less information in the abstraction. In the limit, the behavior graph can be reduced to a single state which preserves little or no information. We have attempted to construct abstractions that presented the essential behaviors of the variables examined with a comprehensible amount of detail, preserving essential distinctions and ignoring others. Sometimes two or three trials were required before an appropriate abstraction for a particular variable was devised. But in all cases we were able with modest effort to construct abstract behavior graphs appropriate for the chosen variables. Much less effort was needed than might be required to examine each behavior in the original graph and determine a probable summary by hand, and the guarantee of faithfulness assures that no distinctions reflected in the abstraction label methods have been missed.

Since our standard for evaluating abstractions is a subjective one, we also present another example and a set of abstraction graphs to better enable the reader to evaluate our claims. The system modeled is a batch fermentation reaction which produces gluconic acid from cells, glucose, and dissolved oxygen through the intermediate product of gluconolactone. (Foss, Johansen & Sørensen 1995)

The model has the five state variables — the concentrations of cells, glucose, gluconic acid, gluconolactone, and dissolved oxygen — but no feedback. Qsim produces the envisionment graph shown in Figure 6, with 131 states (including cycle and cross-edge states)

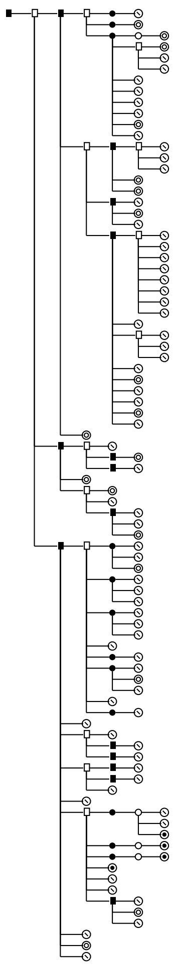

Figure 6: Qsim behavior graph for the batch fermentation model.

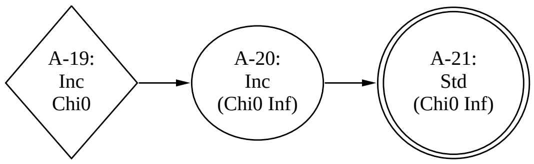

Figure 7: Number of cells in the batch fermentaion reaction.

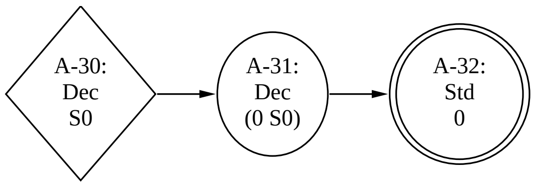

Figure 8: Concentration of glucose, the initial reactant in the batch fermentaion reaction.

	Base Graph		Abstract Graphs		
Model	States	Behaviors	Abstractions	States	Behaviors
Complete simulations					
Glucose-Insulin[1]	188	63	3	9	3–5
Batch Fermentation[2]	131	83	5	3–5	1–4
RCS[3]	53	33	4	5–7	3–5
CSTR — Ti+[4]	36	10	4	4–13	1–4
Incomplete simulations					
Controlled Tank[5]	101	49	2	7–12	2–5
CSTR — Cai+[6]	3874	>1200	3	6–60	1–25

Table 1: Results of applying behavior abstraction to the behavior graphs of five different models. The "Abstractions" column shows the number of abstract graphs constructed. The "States" and "Behaviors" columns for abstract graphs show the range of the numbers of states and behaviors for these abstractions, respectively.

[a] Discussed in the text. (Ironi & Stefanelli 1994).

[b] Discussed in the text. (Foss, Johansen & Sørensen 1995).

[c] A model of the reaction control system for the space shuttle. (Kay 1992, simulation function `rcs-two-legs-leaks-noheline-lopull`).

[d] A model of a continuously stirred chemical tank reactor simulating an increase in the temperature of a reactant entering the reactor. (Dalle Molle 1989, simulation function `cstr-a->b-Ti+`).

[e] An unpublished model of the fluid level in a controlled tank. The simulation was unfinished due to memory requirements.

[f] A model of a continuously stirred chemical tank reactor simulating an increase in the concentration in a reactant entering the reactor. See (Dalle Molle 1989, simulation function `cstr-a->b-cai+`). The simulation was unfinished due to memory requirements. In addition, not all of the state variables were examined.

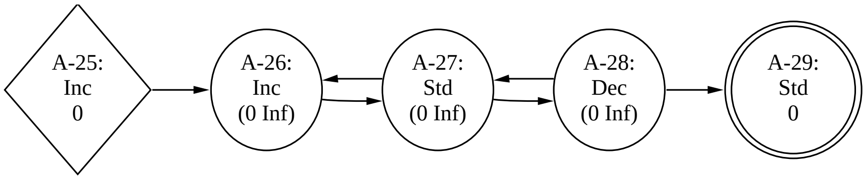

Figure 10: Concentration of gluconolactone, the intermediate produce in the batch fermentaion reaction.

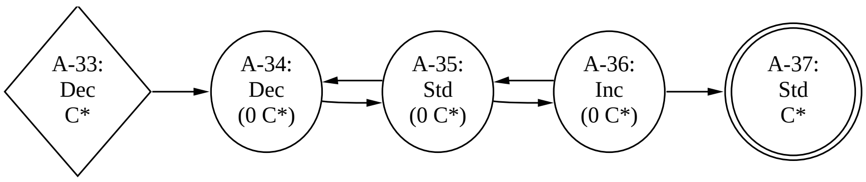

Figure 11: Concentration of dissolved oxygen, a limiting reactant in the batch fermentaion reaction.

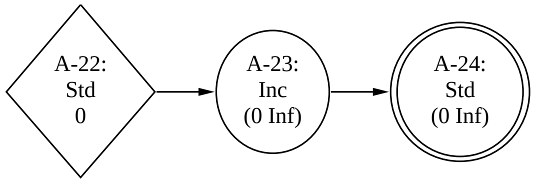

Figure 9: Concentration of gluconic acid, the final product in the batch fermentaion reaction.

and 83 behaviors, none longer than seven states before encountering a cycle or cross edge. Given criteria to label each state with the qualitative value of the selected variable, our algorithm produces abstract graphs of three states and one behavior for the first three of the above variables and graphs of five states and four behaviors (as Qsim would count them) for the last two. See Figures 7, 8, 9, 10, and 11.

Considering the quality of the abstract graphs shown here, we feel confident in claiming that our method of behavior abstraction can be a useful tool in understanding large qualitative simulations.

Future Research

Our ultimate objective has been to provide natural language explanations of qualitative behaviors. One element of such explanations is the kind of high-level summaries that we gave above in the text, describing the conclusions that can be drawn from the behavior abstractions in the glucose-insulin example. To provide such summaries automatically, we are developing methods for describing and explaining qualitative behaviors. Behavior abstraction is the most basic. Others that build on it are described below.

Using state variables as the variables of interest and qualitative values as the labeling criteria, as we have done in the examples, seems likely to be sufficient in most cases. However, the simplicity of the results in the glucose-insulin model and some others depended in part on proper selection of the ranges of qualitative magnitudes that were of interest, and this is likely to be the case for variables with several landmarks other than zero and infinity in their quantity spaces. Automatically identifying these ranges is required. This might be done by abstracting solely on the direction of change of the selected variables and then annotating each abstract state with the range of qualitative magnitudes exhibited in the base states it abstracts, but we have yet to test this hypothesis.

Identifying the type of behavior displayed by a system is the next major step in abstraction. The labels applied in the glucose-insulin example pick out the various phases of an oscillation and should be useful in recognizing the behavior as an oscillation. We are currently examining the recognition of sequences of labels so that phenomena such as oscillation can be identified. For equilibrium systems, behaviors such as being driven away from equilibrium, returning to equilibrium, and oscillation about an equilibrium point seem likely to be of interest. In general, the possible range of behaviors is extremely large, so our interest will necessarily be focused on certain classes of behaviors.

Identifying the type of behavior of a system, or a sequence of such types, provides a very high level abstraction of the behavior which we think will not only be useful in itself but will also permit us to identify other information of interest. For example, in an equilibrium system that is found to be driven away from equilibrium by an input variable constrained to increase or decrease, it would be appropriate to provide a causal description of how the changing value of the input variable causes changes in the variables of interest. But if the input variable stops changing and the system is then found to be relaxing toward a new equilibrium, the mechanisms responsible for bringing the system into equilibrium again become of primary interest.

Behavior abstraction thus seems capable of providing a base for constructing higher level abstractions and descriptions of behaviors that are essential for producing high quality natural language descriptions of qualitative behaviors.

Conclusion

We have applied our behavior abstraction algorithm to a number of qualitative behavior graphs that are complex enough to be difficult to understand. In all tests, our algorithm reduced the number of states and behaviors sufficiently to provide a useful overview of the behavior of variables of interest over the entire behavior graph. In the glucose-insulin example, it enabled us to succinctly characterize the behaviors of the amounts of glucose and insulin and the relationship between their behaviors. The large reduction in the number of states, the selection of a few salient labels, and the display of each state's labels in a graph of the abstract behaviors all contribute to achieving an overview of the behaviors. Other methods of summarizing behaviors, such as visually comparing all the behaviors of a single variable, or of several variables, leave it to the user to see the general picture that we have produced here in a few graphs.

In addition, behavior abstraction seems likely to provide a base for building higher level abstractions and descriptions of the behavior of systems that are essential for producing high quality natural language descriptions of qualitative behaviors.

Acknowledgements

This research has taken place at the Artificial Intelligence Laboratory at the University of Texas at Austin. Support for this research is provided by a grant from Digital Equipment Corporation and a contract from the Air Force Office of Scientific Research (F49620-93-1-0239).

References

Clancy, D. J., and Kuipers, B. J. 1993. Behavior Abstraction for Tractable Simulation. Proceedings of the Seventh International Workshop on Qualitative Reasoning about Physical Systems, Orcas Islands, Washington, 57–64.

Clancy, D. J., and Kuipers, B. J. 1994. Model Decomposition and Simulation. Proceedings of the Eighth International Workshop on Qualitative Reasoning about Physical Systems, Nara, Japan, 45–54.

Dalle Molle, D. 1989. Qualitative Simulation of Dynamic Chemical Processes, Technical Report, AI89–107, Artificial Intelligence Lab., Univ. of Texas at Austin.

de Kleer, J., and Brown, J. S. 1984. A Qualitative Physics Based on Confluences. *Artificial Intelligence* 24(1):7–83. Also appears in (Weld & de Kleer 1990, 88–126).

Falkenhainer, B., and Forbus, K. D. 1990. Self-Explanatory Simulations: An Integration of Qualitative and Quantitative Knowledge. Proceedings of the Eighth National Conference on Artificial Intelligence, 380–387. Menlo Park, Calif.: American Association for Artificial Intelligence.

Falkenhainer, B., and Forbus, K. D. 1992. Self-Explanatory Simulations: Scaling up to Large Models. Proceedings of the Tenth National Conference on Artificial Intelligence, 685–690. Menlo Park, Calif.: American Association for Artificial Intelligence.

Farquhar, A. 1994. A Qualitative Physics Compiler. Proceedings of the Twelfth National Conference on Artificial Intelligence, 1168–1174. Menlo Park, Calif.: American Association for Artificial Intelligence.

Forbus, K. D. 1984. Qualitative Process Theory. *Artificial Intelligence* 24:85–168. Also appears in (Weld & de Kleer 1990, 178–219).

Foss, B. A.; Johansen, T. A.; and Sørensen, Aa. V. 1995. Nonlinear Predictive Control Using Local Models Applied to a Batch Process. *Control Engineering Practice* 3:389-396.

Fouché, P., and Kuipers, B. J. 1991. Towards a Unified Framework for Qualitative Simulation. Working Papers of the Fifth International Workshop on Qualitative Reasoning about Physical Systems, Austin, Texas, 295–301.

Ironi, L., and Stefanelli, M. 1994. A Framework for Building Qualitative Models of Compartmental Systems. *Computer Methods and Programs in Biomedicine* 42:233–254.

Kay, H. 1992. A Qualitative Model of the Space Shuttle Reaction Control System. Technical Report, AI92-188, Artificial Intelligence Lab., Univ. of Texas at Austin.

Kuipers, B. J. 1986. Qualitative Simulation. *Artificial Intelligence* 29(3):289–338.

Kuipers, B. J. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge.* Cambridge, Mass.: MIT Press.

Lee, W. W., and Kuipers, B. J. 1993. A Qualitative Method to Construct Phase Portraits. Proceedings of the Eleventh National Conference on Artificial Intelligence, 614–619. Menlo Park, Calif.: American Association for Artificial Intelligence.

Lester, J. 1994. Generating Natural Language Explanations from Large Scale Knowledge Bases. Ph.D. diss., Dept. of Computer Sciences, Univ. of Texas at Austin.

Porter, B. W.; Lester, J.; Murray, K.; Pittman, K.; Souther, A.; and Jones, T. 1988. AI Research in the Context of a Multifunctional Knowledge Base: The Knowledge Base Project. Technical Report, AI88–88, Artificial Intelligence Lab., Univ. of Texas at Austin.

Rickel, J. W., and Porter, B. W. 1992. Automated Modeling for Answering Prediction Questions: Exploiting Interaction Paths. Working Papers of the Sixth International Workshop on Qualitative Reasoning about Physical Systems, Edinburgh, Scotland, 82–95.

Rickel, J. W., and Porter, B. W. 1992. Automated Modeling for Answering Prediction Questions: Selecting the Time Scale and System Boundary. Proceedings of the Twelfth National Conference on Artificial Intelligence, 1191–1198. Menlo Park, Calif.: American Association for Artificial Intelligence.

Weld, D. S., and de Kleer, J. eds. 1990. *Readings in Qualitative Reasoning About Physical Systems.* San Mateo Calif.: Morgan Kaufmann.