Static and dynamic abstraction solves the problem of chatter in qualitative 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

One of the major factors hindering the use of qualitative simulation techniques to reason about the behavior of complex dynamical systems is intractable branching due to a phenomenon called chatter. This paper presents two general abstraction techniques that solve the problem of chatter. Eliminating the problem of chatter significantly extends the range of models that can be tractably simulated using qualitative simulation.

Chatter occurs when a variable's direction of change is constrained only by continuity within a region of the state space. This results in intractable, potentially infinite branching within the behavioral description due to irrelevant distinctions in the direction of change. While a number of techniques have been proposed to eliminate chatter, none of them provide a general solution that can eliminate all instances of chatter. *Chatter box abstraction* and *dynamic chatter abstraction* provide two such solutions to this problem. Both solutions eliminate chatter by abstracting the chattering region of the state space into a single qualitative state with an abstract direction of change. The algorithms differ in the manner in which they identify the chattering region of the state space.

Introduction

A variety of techniques have been developed that utilize qualitative simulation to reason about the behavior of an imprecisely defined dynamical system to perform tasks such as monitoring, diagnosis and design. Applying these techniques to complex, real-world systems, however, is often hindered by the complexity of the simulation and the behavioral representation generated. Traditionally, qualitative simulation (De Kleer and Brown, 1984; Forbus, 1984; Kuipers, 1994) is performed at a single level of detail highlighting a fixed set of distinctions. At times, some of these distinctions may be irrelevant to the current task and thus needlessly increase the complexity of the simulation. One of the major sources of irrelevant distinctions is a phenomenon called *chatter* (Kuipers, 1994). Chatter occurs when the derivative of a variable is constrained only by continuity within a region of the state space. Within the simulation, the variable is free to alternate between increasing, steady, and decreasing with individual behaviors generated for different orderings of these values. The distinctions between these behaviors provide no additional information; however, they result in intractable branching and a potentially infinite simulation. As the size of a model increases, the number of chattering variables often grows as the model becomes more loosely constrained.

While a number of solutions have been proposed for this problem, none of them provide a general solution that can handle all cases of chatter. This paper presents two such techniques that solve the problem of chatter. Both algorithms identify chattering regions of the state space and abstract them into a single state within the behavioral description. The algorithms differ in the techniques used for identifying the boundaries of the chattering region and computing the consistent successor states exiting this region. *Chatter box abstraction* explores the chattering region of the state space via a recursive call to the simulation algorithm that is restricted to the potentially chattering region. *Dynamic chatter abstraction* provides a more efficient solution that avoids the need to recursively call the simulation algorithm by identifying the chattering variables through a dynamic analysis of the model and the current state.

The elimination of chatter from a qualitative simulation increases the range of models that can be tractably simulated thus allowing qualitative simulation to be applied to tasks such as monitoring, diagnosis, design

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

and tutoring. For example, Rickel and Porter(1994) required the techniques presented here to simulate complex models automatically generated from a large scale knowledge base to answer prediction questions within the field of botany.

Chatter and Qualitative Simulation

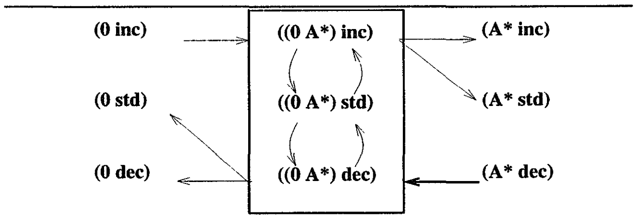
Both of the algorithms presented have been developed as extensions to the QSIM simulation algorithm (Kuipers, 1994). QSIM describes the behavior of a dynamical system via a tree of alternating time–interval and time–point states called a *behavior tree*. Each state specifies a qualitative magnitude and a direction of change (qdir) for each variable within the model. The direction of change corresponds to the sign of the variable's derivative and can be increasing (`inc`), decreasing (`dec`) or steady (`std`). The qualitative magnitude is either a landmark value or an interval between two landmarks defined upon a totally ordered quantity space of landmark values. Landmarks may be defined during the simulation to represent critical points identified within the simulation. A branch occurs within the behavioral description whenever there is insufficient information within the model to identify a unique successor to a given state. Two types of branches can occur:

Definition 1 (Event branch)
An event *occurs when a variable reaches a landmark or becomes steady (i.e. its derivative becomes zero). An* event branch *occurs when there are multiple events following a time–interval state whose ordering is unconstrained by the model.*

Definition 2 (Chatter branch) *A* chatter branch *occurs following time–point t_i if there exists a variable v that is currently steady whose direction of change is constrained only by continuity. A three way branch occurs depending upon whether the variable is increasing, decreasing or steady in the interval (t_i, t_{i+1}) (see figure 1).*

This paper describes two solutions to the problem of chatter. The problem of event branching has been addressed by the model decomposition and simulation (DecSIM) algorithm (Clancy and Kuipers, 1997a). DecSIM partitions the model into loosely coupled components and simulates the components independently to eliminate irrelevant distinctions between unrelated variables.

Chatter branching is problematic due to the repetitive nature of the phenomenon. Since the variable's direction of change is unconstrained, the variable is free to become steady again after the chatter branch and the process repeats itself. While some behaviors exit



- Qualitative value transitions consistent with continuity within the closed interval `(0 A*)` are identified by arcs within the figure.
- The boxed area denotes the chattering region of the state space (i.e. a chatter box). Once the system enters this region of the state space, the chattering variable can continue to cycle within the box.
- If landmarks are introduced, the simulation can remain within the boxed region for an infinite number of states since additional qualitative distinctions are introduced whenever the variable becomes steady.

Figure 1: Possible qualitative value transitions for a QSIM variable.

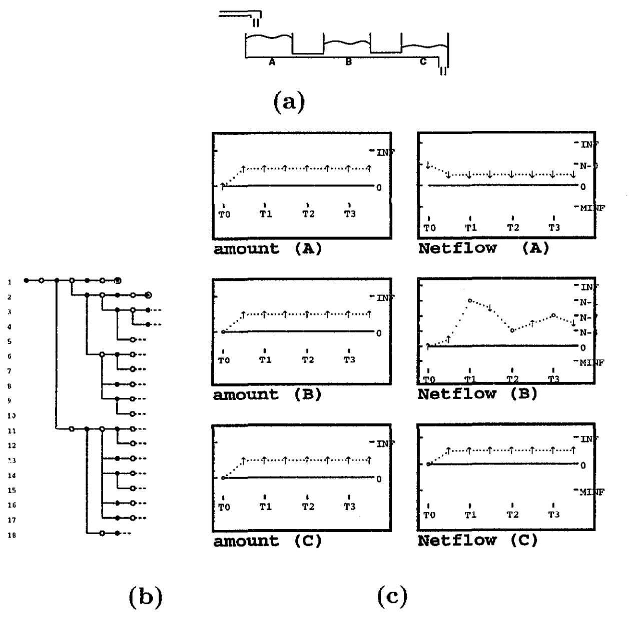
the unconstrained region of the state space, others will continue cycling between different values for the direction of change for an arbitrary number of qualitative states resulting in an infinite simulation (see figure 2).

Chatter branching is particularly difficult to eliminate due to the propagation of chatter through the model. Once one variable, v_1, begins to chatter and its derivative changes sign, it is possible that the derivative of another variable, v_2, that is related to v_1 will also become unconstrained, so v_2 begins to chatter. This process can repeat itself resulting in a large number of chattering variables within one region of the state space called a *chatter box.*

Definition 3 (Chatter box) *For a model with a set of variables V, a region of the state space is defined as a* chatter box *with respect to a set of variables V_c if the derivatives of variables in V_c are unconstrained with respect to variables in $V - V_c$.*

A state exits a chatter box when a non–chattering variable changes value or when a chattering variable changes qualitative magnitude and enters into a constrained region of the state space.

The phenomenon of chatter becomes more complicated if a landmark exists within the unconstrained region of the state space and the magnitude of the variable is unconstrained around this landmark. This phenomenon is called *landmark chatter* and it results in changes in both the magnitude and the direction of change for the chattering variable. A special case of landmark chatter, called *chatter around zero*, occurs when a variable and its derivative are represented explicitly within a model and both of them exhibit chatter. In this case, the derivative variable will chatter



- In a qualitative model of three tanks arranged in sequence connected by tubes (a), *NetflowB(t)* = *InflowB(t)* - *OutflowB(t)*. The derivative of `NetflowB` is constrained only by continuity within the time-interval following the initial state.
- The simulation branches on all possible trajectories of *NetflowB(t)* while all other variables are completely uniform. The simulation is infinite in nature due to chatter and must be halted at an arbitrary state limit. A single behavior (c) from the behavior tree (b) is displayed demonstrating the unconstrained movement of *NetflowB(t)*.
- Other techniques (higher order derivatives and ignoring qdirs) are unable to eliminate chatter in `NetflowB`.

Figure 2: Intractable branching due to chatter in the simulation of a W tube.

around zero as its integral chatters.

While chattering behaviors may describe a real behavior of the system (Kuipers et. al., 1991), a disjunctive enumeration of all possible combinations of values provides no information. Furthermore, this enumeration obscures other distinctions within the description and can result in an infinite simulation. Eliminating this source of distinctions is essential if qualitative simulation is to be used to reason about complex dynamical systems.

Previous Solutions

Two previous methods have been developed for eliminating chatter within a QSIM behavior tree simulation. The *higher order derivative* (HOD) (Kuipers et. al., 1991) technique uses the second and third order derivatives of the chattering variables to determine a unique direction of change for unconstrained variables and eliminate spurious behaviors. When possible, higher–order derivatives expressions are derived via algebraic manipulation of the constraints; otherwise, the operator must specify these expressions within the model. This technique, however, is not effective when an ambiguous evaluation of the HOD expression results or when an expression cannot be derived.

Ignore qdirs (Fouché and Kuipers, 1991) eliminates chatter by ignoring distinctions in a variable's direction of change throughout the simulation. This technique requires the modeler to identify the chattering variables prior to the simulation. For many models, a variable's direction of change is ambiguous only within certain regions of the state space. In other regions of the state space, direction of change information may be relevant in constraining the behavior of the system. Thus, ignore qdirs may result in over–abstraction and generate spurious behaviors. In addition, ignore qdirs often prevents the application HOD constraints to eliminate spurious behaviors and cannot be used to eliminate landmark chatter.

DeCoste (DeCoste, 1994) addresses chatter when simulating Qualitative Process Theory models (Forbus, 1984) simply by ignoring distinctions in the direction of change when a unique value cannot be inferred via constraint propagation. He does not address the problem of landmark chatter nor does he provide an evaluation of this approach. In addition he does not discuss cases where constraint propagation is too weak of an inference mechanism to infer a unique direction of change.

Chatter Box Abstraction

The chatter box abstraction algorithm eliminates chatter by abstracting chattering regions of the state space into a single qualitative state within the behavioral description. The chatter box abstraction algorithm performs the following steps:

Step 1: Detect entry into a chatter box.
Step 2: Identify the chattering variables.
Step 3: Create and insert into the behavior tree abstract states describing the chatter box and its successors.

The algorithm determines when the simulation potentially enters a chatter box by performing an analysis of the constraints within the QDE to identify a set of potentially chattering variables. Constraints within the model are used to partition the variables into chatter equivalency classes (i.e. if one variable in the class chatters then all the variables will chatter) and identify classes containing variables that are prevented from chattering throughout the simulation. This algorithm is an extension of the algorithm used when deriving HOD expressions (Kuipers et. al., 1991).

If a potentially chattering variable is changing within a time–interval state, a *focused envisionment* [1] is used to explore the current region of the state space and determine which variables, if any, exhibit chatter. A focused envisionment is a recursive call to the simulation algorithm defined as follows:

Definition 4 (Focused envisionment) *A focused envisionment is an attainable envisionment that is restricted to a specified region of the state space. States created outside the defined region are suspended from simulation (i.e. their successors are not computed).*

The focused envisionment is limited to changes in the direction of change for potentially chattering variables. The results of the focused envisionment are analyzed to determine which variables actually exhibit chatter. The states within the chattering region of the envisionment graph are then abstracted into a single abstract, time–interval state. The direction of change for chattering variables is described via a conjunctive list of the possible directions of change within the chattering region (e.g. (`inc std dec`)). The successors to this abstract state are computed from the boundary states that exit the chattering region within the focused envisionment. Boundary states that are equivalent with respect to the non–chattering variables are combined to eliminate distinctions in variables that are still chattering. This occurs when a relevant distinction occurs in a non–chattering variable while a chattering variable remains unconstrained. The algorithm used to create successor states has been extended to handle the abstract boundary states. Figure 3 provides an overview of the algorithm with respect to the W-tube example.

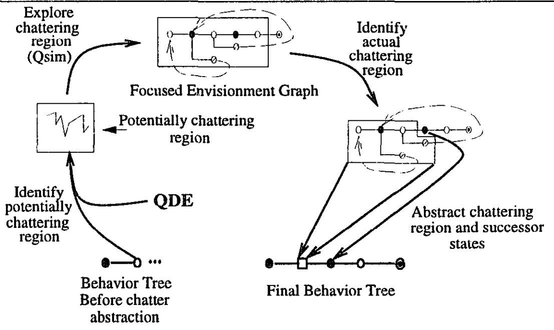
Landmark chatter is eliminated by extending the region that is explored during the focused envisionment to include the landmark around which chatter is possible. The QSIM behavior language has been extended so that an abstract state can describe an interval defined by two non-adjacent landmarks within the quantity space.

Results

Chatter box abstraction provides the following guarantees. (Proofs omitted due to space restrictions.)

Theorem 1 *Chatter box abstraction identifies a variable as chattering within an abstract state if and only if it exhibits chatter within the same region of the state space in a standard QSIM simulation.*

[1]An *envisionment* generates a finite graph of qualitative states rather than a tree and does not create new landmarks.


Explore chattering region (Qsim)
Identify actual chattering region
Focused Envisionment Graph
Potentially chattering region
Identify potentially chattering region
QDE
Abstract chattering region and successor states
Behavior Tree Before chatter abstraction
Final Behavior Tree

- The links in the figure represent processes and the objects represent data structures. Two states linked by a dotted line in the envisionment graph represent the same qualitative state.
- Using chatter box abstraction, QSIM generates a single behavior. The chatter box state is represented by a square in the behavior tree. HOD constraints are used to eliminate chatter in all of the variables except `NetflowB`.

Figure 3: Chatter box abstraction algorithm applied to the W-tube.

Theorem 2 *The set of real–valued trajectories described by a chatter box abstracted behavioral description is equal to the set described by an unabstracted tree with respect to the non–chattering variables and a super–set with respect to the chattering variables.*

Dynamic Chatter Abstraction

Chatter box abstraction explores the entire chattering region of the state space via simulation. The number of states within this region is exponential in the number of chattering variables. Thus, exploring this region can become intractable as the size of a model grows and the number of unconstrained variables increases. Dynamic chatter abstraction provides a scalable solution by avoiding the need to perform a focused envisionment. Instead, the chattering variables within a region of the state space are identified by a dynamic analysis of the model and the current state.

For each time–interval state, dynamic chatter abstraction determines whether the state is contained within a chatter box and if so identifies the set of variables that chatter within this region. It uses an understanding of the restrictions that are asserted by each constraint within the model along with the current qualitative state to perform this task.

For example, in the W-tube model if HOD constraints are not used, the direction of change for `NetflowB` is restricted only by the constraint `NetflowB` + `Flow-BC` = `Flow-AB` In the time–interval following the initial state, all three variables are increasing. Thus, in this state the qdir of `NetflowB` is unconstrained and `NetflowB` is free to chatter.

Dynamic chatter abstraction, however, must reason not only about the qualitative values contained within the current state, but also about how these values change as variables begin to chatter. Once NetflowB chatters, its derivative can change sign and the above addition constraint no longer restricts the derivative of Flow-AB. If Flow-AB is not prevented from chattering by other constraints, it is free to chatter and must be identified as such within the current chatter box. A more detailed presentation of the algorithm is contained in (Clancy and Kuipers, 1997b).

Identifying the set of chattering variables

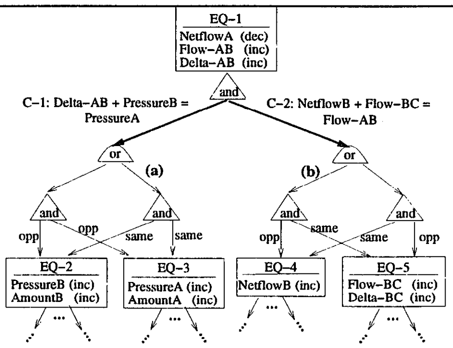
The algorithm must determine if a variable v can chatter following the current time–interval state before a non–chattering distinction occurs. If such a variable exists, then the current state is contained within a chatter box. To address this issue, two questions must be answered:

Consistency - Is there a consistent state in which v is free to chatter?

Reachability - Can this state be reached from the current state only through changes occurring in other chattering variables? Such a state is called *chatter–reachable*.

To answer these questions, dynamic chatter abstraction uses a *chatter dependency graph* to define the conditions under which a variable can chatter with respect to the current state. Two types of nodes exist within the chatter dependency graph (see figure 4). An *equivalency node* is created for each set of chatter equivalent variables within the model[2]. These nodes are connected through a directed AND-OR subgraph of intermediate *dependency nodes*. For an equivalency node EQ_{source}, the leaves of the AND-OR subgraph correspond to equivalency nodes containing variables related to the variables within EQ_{source} via a constraint within the model. The AND-OR subgraph can be viewed as a predicate that specifies the conditions under which the variables in EQ_{source} are free to chatter. A consistent, chatter–reachable state satisfying this predicate exists if and only if the variables in EQ_{source} chatter within the current region of the state space.

Evaluating the dependency graph The dependency graph evaluation algorithm categorizes chatter equivalency classes as `chattering`, `non-chattering`, or `chatter-unknown`. The algorithm iterates through the equivalency classes moving them from the



For each time–interval state, a chatter dependency graph is created describing the conditions under which each set of chatter equivalent variables can chatter with respect to the current state. In the figure, the qdir for each variable is displayed within the node. The AND-OR subgraph extending from an equivalency node EQ_{source} has the following structure:

- The root of the sub-graph is an `AND` node that has a link pointing to an `OR` node for each constraint relating a variable within EQ_{source} to other variables within the model. In the W-tube, there are two constraints that can potentially restrict the variables within EQ_1 from chattering.
- Each `OR` node corresponds to a constraint C. A child is created for each set of qualitative value assignments for the variables within C that allows the variables in EQ_{source} to chatter.
- The children of these final conjunctive nodes are the equivalency nodes for the other variables within C. The arcs are labeled with information about how the qualitative values of these variables must change if the variables in EQ_{source} are to chatter. The most common labels are `same` (i.e. the qdirs must remain the same) and `opp` (i.e. the qdirs must change sign *before* the variables in EQ_{source} can chatter.) In the W-tube, C_1 restricts the variables in EQ_1 from chattering if and only if the qdirs in EQ_2 and EQ_3 either both remain the same or both change.

Figure 4: Chatter dependency graph for the W-tube

`chatter-unknown` category into one of the other two categories.

Evaluation of a node EQ_1 in the dependency graph begins by instantiated a partial state description with the current qualitative magnitude and direction of change for the variables within the node and all variables currently classified as `non-chattering`. Then a backtracking algorithm is used in an attempt to identify a chatter–reachable state that satisfies the AND-OR sub-graph extending from EQ_1. Information is added to the partial state when another equivalency node is encountered. The information added depends upon the label of the link pointing to the node. If the search encounters an equivalency node EQ_2 whose direction of change is required to change (i.e. the arc is labeled with `opp`), then EQ_2 must be classified as

[2]Prior to the simulation, variables are partitioned into chatter equivalency classes as in the chatter box abstraction algorithm.

chattering before the required information can be added to the partial state description. This ensures that the state is chatter–reachable. The evaluation algorithm is called recursively if EQ_2 is still classified as **chatter-unknown**. A cycle within the recursive calling sequence is treated as a dead end condition and the algorithm backtracks. A cycle between nodes EQ_1 and EQ_2 means that along the path traversed, EQ_1 depends upon EQ_2 to chatter first while EQ_2 depends upon EQ_1 to chatter first. Thus, neither one can chatter[3]. This condition, however, does not necessarily prevent these nodes from chattering since there might be another set of qualitative value assignments (i.e. another path in the dependency graph) that allows one of these nodes to chatter. Once one chatters, the other may be free to chatter as well.

If a partial state is created satisfying the AND-OR subgraph, the QSIM constraint satisfaction algorithm is used to ensure that the variable assignments are consistent with all of the constraints within the model. If they are, then the equivalency node is classified as **chattering**. If a chatter–reachable, consistent state cannot be identified, then the equivalency node is classified as **non-chattering**.

Abstract and successor state creation

Once the set of chattering variables is identified, the algorithm creates an abstract state with an abstracted qdir of (**inc std dec**) for each of the chattering variables. The QSIM state successor algorithm has been extended to handle such abstract states.

Landmark chatter

Dynamic chatter abstraction is able to handle chatter around zero without any additional processing. The algorithm described above does not assume that the derivative constraint prevents the integral variable from chattering. Instead, information about the constraining power of the derivative constraint is represented via labels within the chatter dependency graph. If both a variable and its derivative are identified as chattering, then the derivative variable will exhibit chatter around zero.

Complexity

The size of the chatter dependency graph is polynomial in the number of chatter equivalency classes. Since the evaluation algorithm is performing constraint satisfaction, we expect that the worst case complexity of the algorithm may be exponential, however, the average case complexity is much better. In fact, we have not encountered an exponential time factor in any of the models tested (see table 1). This is because the algorithm performs a directed search of the potentially chattering region simply to identify variables as chattering or non–chattering as opposed to enumerating all possible solutions within this search space. Conflicts are often encountered well before an entire path is traversed within the search space. A detailed complexity analysis of the dynamic chatter abstraction algorithm still must be performed.

[3]Remember that the opp label requires variables in the destination node to change sign *before* the source node can chatter.

Evaluation

Both the chatter box abstraction and dynamic chatter abstraction algorithms have been empirically evaluated using a corpus of over 20 models obtained from various researchers within the field of qualitative reasoning. The results were validated against a standard QSIM simulation. Both algorithms identify variables as chattering if and only if they exhibited chatter within the same region of the state space in the standard simulation. Table 1 reports the results of this evaluation on a subset of the models used. In all of the models tested, dynamic chatter abstraction performed at least as well as chatter box abstraction and often it performed significantly better. Furthermore, a number of improvements are planned to increase the efficiency of the dynamic chatter abstraction algorithm. In particular, a propagation phase will be added prior to the dependency graph evaluation to reduce the complexity of this step.

Discussion

Both abstraction algorithms provide unique benefits with respect to the elimination of chatter. Dynamic chatter abstraction provides a scalable solution that is able to efficiently solve the problem of chatter. However, it is less modular than chatter box abstraction. Since the algorithm incorporates information about how the simulation algorithm processes the constraints within the model, extensions to the simulation algorithm may require modifications to the dynamic chatter abstraction algorithm. On the other hand, chatter box abstraction uses the basic simulation algorithm as its main inference engine. As a result, extensions to the simulation algorithm can be incorporated without modifications to the chatter abstraction algorithm.

Both abstraction techniques incorporate HOD information when applicable and thus complement this technique. With respect to ignore qdirs, the techniques presented here supersede the need for this capability since they are both able to automatically detect chatter and selectively eliminate it. For larger models, ig-

Model	Vars	Chat Vars	Number of Behaviors: No chatter abstraction: Envisionment	Number of Behaviors: No chatter abstraction: Behavior tree (no lms)	Number of Behaviors: Chatter abstraction: Behavior tree (w/ lms)	Simulation time (sec): Dynamic Chatter	Simulation time (sec): Chatter Box
W Tube	16	1	3	> 1845	1	0.9	3.4
Glucose-insulin Interaction	11	2	155	> 2807	41	14	52
Van der Pol Equation^A	10	4	43	> 1788	12^B	2.3	15
Controlled Hot/Cold tank^A	14	5	24	$> 601^C$	14	5.0	48.2
Turgor Stomates	19	7	509	> 2598	1	2.4	20.5
Cooling Plant^A	15	10	659	> 4095	1	7.7	418
Heart Model	42	28	689	> 2784	200	100	C

A – Exhibits chatter around zero.

B – Oscillatory behavior results in infinite behavior tree. Simulation terminated once the structure within the tree could be determined.

C – Could not be simulated to completion due to resource limitations.

- Each model was tested both with and without chatter abstraction. When chatter abstraction was not used, both an envisionment and a behavior tree simulation without landmark introduction were used. The number of behaviors generated are listed above using a state limit of 5000. Note that while an envisionment often results in a smaller number of behaviors, the total number of qualitatively distinct behaviors represented is the same as there are an infinite number of paths within the envisionment graph.
- Both types of abstraction generated the same number of behaviors; however, dynamic chatter abstraction performed significantly better than chatter box abstraction with respect to simulation time.
- HOD constraints were used whenever applicable. Ignore qdirs does not work on models exhibiting chatter around zero. On other models ignore qdirs can be used, however, it requires a significant amount of work by the modeler to identify which variables are chattering.

Table 1: Evaluation of Dynamic Chatter and Chatter Box Abstraction

nore qdirs requires the modeler to invest a significant amount of time to identify the chattering variables and cannot be applied when using automatic model building (Rickel and Porter 1994). In addition, ignore qdirs cannot be used to eliminate landmark chatter. This phenomenon tends to happen more often as the size of a model increases.

While the techniques presented here have been developed as extensions to the QSIM algorithm, they could be applied in a similar manner to eliminate chatter from Qualitative Process Theory (QPT) models (Forbus, 1984).

Conclusions

Recent discussions within the field of qualitative reasoning have questioned the usefulness of qualitative *simulation* when reasoning about the behavior of complex dynamical systems. In general, these discussions have focused on the complexity of the simulation and the difficulty of scaling up to larger, more complex systems due to irrelevant distinctions. Chatter is a major source of these distinctions. The techniques presented in this paper solve this problem. These techniques in combination with the DecSIM component simulation algorithm (Clancy and Kuipers, 1997a) significantly extend the range of models that can be tractably simulated using qualitative simulation thus supporting the application of qualitative simulation to tasks such as monitoring, diagnosis and design for complex, real-world systems.

References

D. J. Clancy and B. J. Kuipers, 1997a. Model Decomposition and Simulation: A component based qualitative simulation algorithm. In *Proceedings from the Fourteenth National Conference on Artificial Intelligence*, 1997.

D. J. Clancy and B. J. Kuipers, 1997b. Dynamic Chatter Abstraction: A scalable technique for avoiding irrelevant distinctions during qualitative simulation. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning about Physical Systems*, 1997.

D. DeCoste, 1994. Goal-directed qualitative reasoning with partial states. Technical Report 57, The Institute for the Learning Sciences, Northwestern University, 1994.

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

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

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

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

B. J. Kuipers, C. Chiu, D. Dalle Molle and D. R. Throop, 1991. Higher-order derivative constraints in qualitative simulation. *Artificial Intelligence*, 51:343-379, 1991.

J. Rickel and B. Porter, 1997. Automated Modeling of Complex Systems to Answering Prediction Questions To appear in *Artificial Intelligence*, 1997.