In *The Eleventh International Workshop on Qualitative Reasoning (QR-97)*
Cortona, Italy, June 1997.

Control of Hybrid Systems using Qualitative Simulation

Giorgio Brajnik
Dip. di Matematica e Informatica
Università di Udine, Udine, Italy
giorgio@dimi.uniud.it

Daniel J. Clancy
Dept. of Computer Sciences
University of Texas at Austin, Texas 78712
clancy@cs.utexas.edu

Abstract

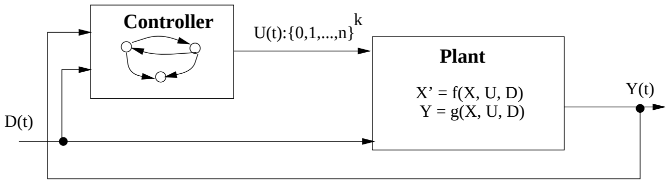
This paper presents a methodology for synthesizing, under uncertainty, a sequence of robust, discrete control actions to drive a continuous dynamical plant through admissible trajectories specified via temporal logic expressions. An action is defined as a modification to the value of a controllable exogenous variable. The TeQsim algorithm combines qualitative simulation with temporal logic model checking to validate or refine a proposed plan. Qualitative simulation is used to infer a branching–time description of the system behaviors that potentially follow from a sequence of control actions. This description is queried using temporal logic based goal constraints to perform plan validation. Plan refinement infers bounds on the sequence of actions within a plan to guarantee that the specified goal constraints are satisfied.

A framework for plan generation is presented that uses a phase portrait representation to reason about the effect of each action within a particular system configuration. This methodology is able to prune the plan search space by eliminating actions that are unable to resolve a detected goal violation.

Introduction

"Hybrid systems" defines a broad class of dynamical systems comprised of both discrete and continuous components. The interaction of discrete and continuous phenomena give rise to different classes of hybrid systems (Branicky *et al.* 1994). In this paper we focus on *controlled switching* hybrid systems where the continuous component (henceforth referred to as the *plant*) described using ordinary differential equations is controlled by discrete numeric valued inputs. Such inputs are computed by a discrete controller whose task is to drive the plant to satisfy a set of trajectory requirements (figure 1).

The aim of this paper is to propose preliminary ideas about a methodology for synthesizing, under uncertainty, a sequence of robust, discrete control actions to drive a plant through admissible trajectories. Qualitative simulation is used to reason about the ability of



Block diagram of hybrid systems made of a discrete controller sending discrete control signals $U(t)$ to a continuous plant. $D(t)$ represents uncontrollable continuous input disturbances. A feedback loop may be present as well.

Figure 1: Hybrid systems discussed in this paper

a sequence of actions to satisfy goal constraints specified via temporal logic expressions. This approach supports reasoning under both parametric and functional uncertainty within a structural model of the system. Qualitative simulation generates a behavioral description that provides guaranteed coverage of all potential real–valued trajectories of the system thus supporting robustness analysis. In addition, the qualitative representation provides an effective intermediate language that bridges the gap between the representations used for the continuous plant, the discrete controller and the declarative requirement specification (figure 2).

Some steps of the methodology have been implemented using the TeQsim (*TEmporally constrained QSIM*, pronounced *tek'sim*) simulation algorithm (Brajnik and Clancy 1996a; 1996b; 1996c). It uses Qsim (Kuipers 1986; 1994) to derive a branching–time behavioral description that is an abstract discretization of the set of real–valued solutions to the system of differential equations. TeQsim supports the specification of trajectory constraints through a combination of temporal logic and discontinuous change expressions. Discontinuous change expressions are used to specify discontinuities in exogenous variables and the resulting effects. In addition, the modeler is able to specify numerically bounded, external events that are incorporated into the simulation. Trajectory constraints can be used to specify time varying exogenous inputs, constrain the occurrence of internal or external events or specify boundary conditions for dependent and independent variables.

Synthesis of robust control actions

Synthesis of robust control actions requires a mechanism for deriving a sequence of discrete actions to satisfy the goal specification. For tracking or regulatory problems, a more traditional control approach can often provide an acceptable solution. As the complexity of the trajectory requirements increase, however, a more sophisticated planning approach is often appropriate to reason about the effects of the actions across time. Examples of such problems include controlling startup and shutdown operations for complex, time–dependent devices (*e.g.* a chemical reactor) or planning manual control operations on plants involved in risky situations (*e.g.* dam floodgate control.)

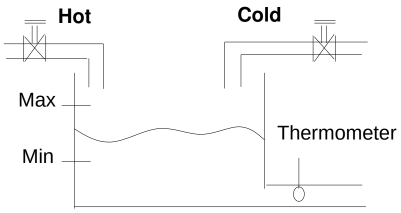
Solutions to these problems usually reason in an abstract manner using a discrete ontology supporting a declarative representation of actions, states, plans and goals. When the planning task involves a continuous plant, however, a purely discrete representation is insufficient to adequately model the dynamics of the plant. Such problems often require a hybrid planning and control viewpoint.

Synthesizing discrete control actions raises a number of complex issues:

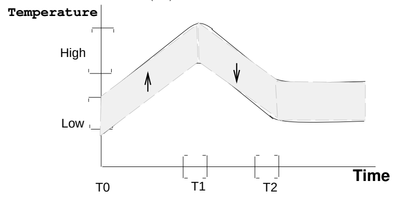
- The planner must reason about the effects of actions across time as some components of the system change autonomously. This hinders the application of the many planning techniques that assume a static environment.

- Uncertainty in the plant model must be represented in a manner that supports robustness analysis. Robustness is important since the controller is based on a discrete (hence "crisp") functioning mode, whereas the plant will always be subject to a certain degree of noise and deviation.

- An expressive language is required to describe a variety of trajectory constraints. Goal specifications developed in the area of *model predictive control*, for example, can only describe regions of the state space using linear inequalities ((Rawlings *et al.* 1994)). While appropriate for regulatory control problems, this seems inadequate for solving the control problems that require a more expressive language to specify complex, multi–level and temporally extended planning goals.



(a) Tank



(b) Desired trajectory

- The tank (a) has a hot and a cold inflow. The valves controlling these inflows are binary (*i.e.* each tap can be *on* or *off*).

- A desired trajectory for level is described by using the landmarks `Max` and `Min` to represent upper and lower bounds for the acceptable values. The desired trajectory for the temperature (b) can be easily specified with temporal logic and external events, using statements similar to those shown in (Brajnik and Clancy 1996b).

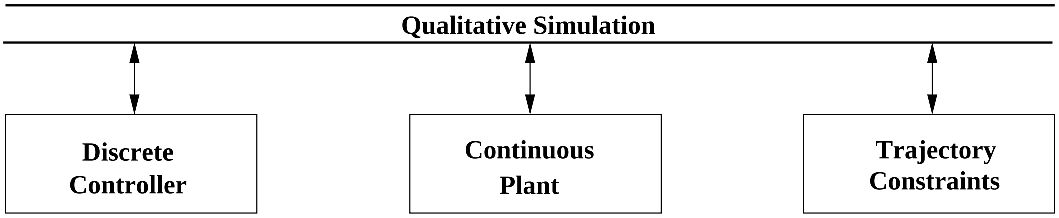
Figure 3: Hot–cold tank example

The general task is to synthesize discrete control actions driving a continuous plant while ensuring that certain trajectory constraints are met. The inputs provided include:

- a model of the plant (in terms of a semi–quantitative QDE),
- an initial state (a partially specified qualitative state),
- a description of uncontrollable disturbances (temporal logic formulæ),
- a description of the desired behavior (temporal logic formulæ), and
- the list of control input variables and their possible values.

Figure 3 describes an example a controlling the level and temperature of a tank. A time–variant trajectory specification is provided for the temperature while constraints on the level are specified in the form of time–invariant conditions.

The task is to develop an abstract plan that is a solution for the current problem specification. An *abstract plan* is a partially ordered sequence of actions. It may or may not include temporal bounds on when



The qualitative behavior representation provides a common language that bridges the gap between the representations used for all three sources of information: the continuous plant, the discrete controller, and the declarative requirement specification.

- discretization of trajectories into qualitative behaviors provides a natural identification of regions in the trajectory space that are relevant to the control task;
- the concise description of trajectories can be used to represent effects of discrete actions that form the basis for solving planning problems;
- discrete descriptions of trajectories enable logical model–checking against temporal logic formulæ.

Figure 2: The role of Qualitative Simulation

an action is performed as well as bounds on the magnitude of an action. An abstract plan is a *solution* when it can be shown that all fully instantiated instances of the plan (*i.e.* actions with precise values for time and magnitude) guarantee the desired behavior.

The general discrete control task can be decomposed into a sequence of subtasks of increasing complexity.

Plan validation — determines if a proposed plan is a solution.

Plan refinement — refines a plan by inferring tighter bounds on either the magnitude of the actions or on the temporal occurrence of the actions. The objective is to identify a minimal refinement that still results in a solution.

Plan generation — develops an abstract plan, refines it and validates it to generate a solution.

The next section describes an example demonstrating how TeQsim can be used to perform the tasks of plan validation and plan refinement. Section will then describe the plan generation process.

Plan validation and refinement example

(Material presented in this section is based on (Brajnik and Clancy 1996c).)

A model of a lake, consisting of a reservoir, an incoming river and an outgoing river is used; lake level and outflow are regulated through a dam that includes a single floodgate[1]. Quantitative information is provided by numerical tables which are interpolated in a step–wise manner to provide lower and upper bounds for any intermediate point (representing therefore non–parametric uncertainty).

Initial values for stage (*i.e.* the level) and gate opening are provided along with a description of a forecasted step increase in the inflow (a disturbance). The outflow gate can either be opened or closed. The value of this variable can take on integer values ranging from 0 to 8. The task is to develop a sequence of control actions (*i.e.* opening or closing the gate) that prevent an overflow or underflow of the lake and ensure that the downstream flow stays within the specified range required to meet the needs of the downstream farmers. Table 1 contains the temporal logic expressions that are used to represent these goals. We assume that the optimization criterion is to minimize the number of required actions.

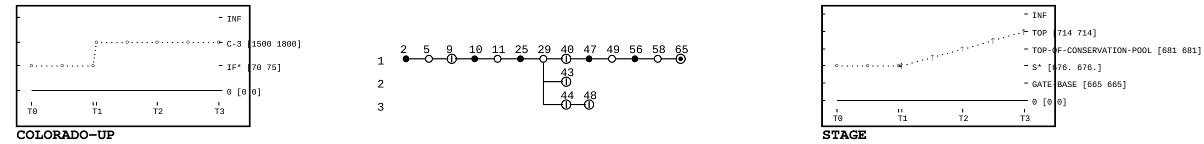
TeQsim is used to perform plan validation and refinement. Initially, a plan with no actions is tested to see if an action is required. The behavioral description generated by TeQsim is queried using the temporal logic goal constraints. The results of the query indicate that a goal violation occurs due to the possibility of an overflow (see figure 4.)

At this point, an abstract plan consisting of a single opening action is hypothesized by the user. The user specifies that the opening action must occur prior to the overflow event and plan refinement is used to infer bounds on this opening action. In this example, there are two degrees of freedom within the plan refinement process: the magnitude of the opening action, and the temporal occurrence of the action. The refinement ca-

[1] We use quantitative information concerning Lake Travis, near Austin (TX), obtained from the Lower Colorado River Authority.

	Temporal logic expression	Description
1	`(always (qvalue-< stage top))`	The qualitative value of the variable `stage` is always less than the landmark `top`.
2	`(always (qvalue-> stage gatebase))`	The value of `stage` always remains above `gatebase`.
3	`(always (value-<= colorado-dn 350))`	The outflow (*i.e.* `colorado-dn`) does not exceed 350.

Table 1: Goal specification using temporal logic



(a) Inflow Perturbation **(b) Behavior Tree** **(c) Lake Overflow**

- A TeQsim simulation of the lake model with the specified perturbation to the inflow (a) and no control action results in three behaviors (b).
- The resulting behavioral description is queried using the temporal logic goal constraints. Behaviors 2 (c) and 3 violate goal 1.

Figure 4: Lake simulation with no action

pabilities of TeQsim can only infer refined bounds for a single variable. Thus, plan refinement starts with a lower bound of 1 for the magnitude of the opening and attempts to determine if such an opening action can guarantee the desired behavior. The magnitude of the lower bound is increased until a lower bound that produces a solution is identified.

Plan refinement infers a bound on when an action must be performed by using the negation of the violated goal constraint as an input to TeQsim. This causes the simulation to be restricted to behaviors that satisfy the violated constraint resulting in a description of all and only behaviors that can potentially lead to an overflow. The results of this simulation provide a bound on when the specified action can lead to an overflow. Thus, the complement of this bound describes the region of the state space where an overflow is guaranteed not to occur.

In this example, if the lower bound for the opening action is less than 3 ft, then the simulation is unable to tighten the temporal bounds on the occurrence of this action. When a lower bound on the opening of 3 ft is used, the simulation generates two behaviors each with a bound of 11.2 days when the opening action must occur. This simulation proves a theorem of the form:

Overflow $\Rightarrow$ Opening occurs after 11.2 days

The contrapositive of this statement is:

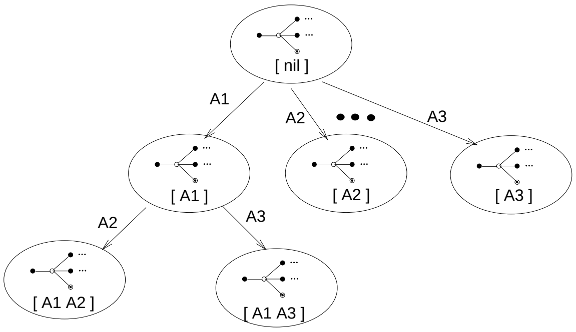
Opening occurs before 11.2 days $\Rightarrow$ No overflow

Once a partial plan is developed to deal with the overflow event, the process returns to the validation stage to determine if the abstract plan satisfies all of the goals. The abstract plan that eliminates the overflow condition may however violate the constraint on the downstream flow (goal 3). An additional refinement procedure, however, is able to infer an upper bound of 6 ft for the magnitude of the opening action to eliminate this goal violation. The iterative process of plan validation followed by plan refinement or extension continues until an abstract plan satisfying all of the constraints is developed. The solution of this example is a plan comprised of a single action, taken no later than 11.2 days from the start with magnitude of either 3, 4, or 5 feet.

Plan generation

The preceding section demonstrated how semi–quantitative simulation techniques can be applied to solve the problems of plan validation and plan refinement. Extending these techniques to the problem of plan generation requires a clear description of the search space along with the development of techniques for searching the space efficiently.

The process of plan generation involves searching for a sequence of system configurations that guarantees the desired behavior. A *system configuration* is the vector of discrete values for the controllable, exogenous variables. A single configuration may satisfy the system requirements for a period of time; however, at some point a transition to an alternative configuration may be required. Actions are used to trigger these



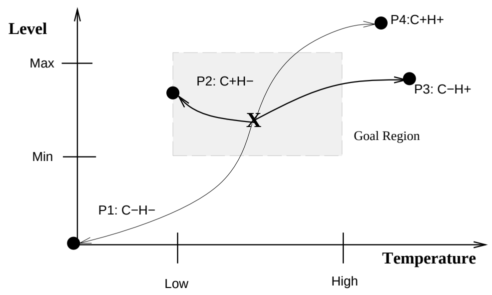
- The plan search space is defined by a directed acyclic graph where:
  - Each node represents an abstract plan and the set of qualitative behaviors that are consistent with the plan.
  - Each arc represents the extension of the abstract plan of the preceding node with an additional action.
- The plan refinement task occurs within a node of the search space by reducing the set of actions instances associated with the node (*i.e.* restricting numerical bounds of actions).
- Plan generation first attempts to refine a plan before considering plan extensions or backtracking to other candidates.

Figure 5: Plan generation search space

transitions. In the hot–cold tank example in figure 3, there are four different configurations corresponding to whether each tap is on or off. None of these configurations can independently satisfy the goal specification.

Thus, a sequence of actions defines a set of potential system behaviors for a given initial state. These behaviors transition between different system configurations as each action is performed. The planning process searches within a space of abstract plans. Objects within this search space correspond to both an abstract plan and set of behaviors associated with this plan. The requirement specification can be tested against these trajectories to determine if the state satisfies the goal constraints (*i.e.* if all behaviors satisfy all the goals). Figure 5 provides a more detailed description of this space.

Searching this space to find a plan that satisfies the goals involves identifying goal violations and then selecting an action to address these violations. Due to the complexity of semi–quantitative simulation and the potentially exponential branching within this search space, a brute force search approach is clearly inadequate. To evaluate the effectiveness of an action to resolve a violation, a compact but sufficiently expressive representation of the behaviors associated with the ac-



- P1 through P4 correspond to the four configurations of the system. The label indicates which taps are on within the configuration. The current state within the phase plane is identified by an **X**, fixed points by black dots, and a trajectory for each configuration is displayed.
- The goal specification (see figure 3) states that the temperature should increase to `High` and then decrease to `Low` without violating the constraints on the level. (*i.e.* Move to the right hand boundary and then the left hand boundary of the boxed region.)
- P1 is the initial configuration. The trajectory of this configuration does not satisfy the goal specification. An analysis of the phase diagram indicates that the only action that can satisfy the first portion of the goal specification is to turn the hot tap on. P4 fails to keep the level within the bounds, while the other two don't satisfy the requirements on temperature.

Figure 6: Hot–cold tank phase portrait description

tion (or actually the target configuration) is required.

To assist in action selection and evaluation, a phase portrait representation can be used to describe the potential trajectories of the system within a configuration. A phase portrait provides an atemporal description of the set of trajectories of a plant within an n–dimensional space where n is the number of state variables. Phase portraits for neighboring configurations are compared to determine the effect of an action on the trajectory of the state variables. When a goal violation is detected within a plan, only those actions that potentially resolve the violation are considered within the planning process.

In the hot–cold tank example, each configuration has a distinct fixed point that can be identified within the phase portrait. Figure 6 describes a trajectory for each of these configurations for a given initial state along with a description of the intuition behind how phase portraits can be used to efficiently prune the search space.

A qualitative phase portrait description for each configuration is generated prior to the plan generation process (this can be done using the QPORTRAIT program (Lee and Kuipers 1993) or through the tools described in (Bradley and Zhao 1993)). From these descriptions, a *plan schema* is generated describing the valid configuration transitions and goal violations that can potentially be resolved by each transition. The plan schema is used during the planning process to prune the search space and quickly eliminate actions that cannot resolve a goal violation. For example, in the hot–cold tank example, if both taps are on and a trajectory exceeds the maximum temperature threshold, the plan schema would be used to determine that turning the cold water tap off will not be effective in resolving this violation. The pruning of the search space by the plan schema allows the plan generation algorithm to perform a breadth first search in an attempt to identify a minimal plan.

Discussion

One of the fundamental issues encountered when using semi–quantitative simulation to reason about the behavior of a system is the breadth of potential behaviors provided. As the region of the state space described within this behavioral description increases, it often becomes more difficult to formulate a plan that can provide the desired guarantees with respect to the goal specification. This is due to uncertainty being poorly described in the model and building up very quickly during plan refinement and generation steps. There are three potential sources of this ambiguity: 1) spurious behaviors generated due to the incompleteness of the simulation algorithm; 2) certain qualitative behaviors may correspond to mathematically possible, but highly unlikely behaviors of the system; or 3) the semi–quantitative model describes a broader region of the model space than what is required for the current task. Currently, we are concerned with extensions to our approach that can accommodate the first two issues.

Both of these issues relate to the soundness guarantees that are provided by qualitative simulation. Often, however, this guarantee may be stronger than what is necessary to solve real–world problems. Currently, we are investigating additional numerical techniques that can be used to reduce the space of solutions.

At the moment, plan validation can be performed automatically, whereas plan refinement needs human intervention to provide some initial information (like lower bounds on action magnitude). Plan generation has not been implemented yet.

Related work

The approach to planning and control we presented in his paper overlaps with several theoretical frameworks proposed for hybrid systems. In particular, Tavernini (Tavernini 1987) assumes (as we do) that state variables are continuous, but their derivatives can be piecewise–continuous; others (Back *et al.* 1993; Nerode and Kohn 1993) encompass also discontinuous jumps of state variables and the one proposed by Branicky and colleagues (Branicky *et al.* 1994) subsumes all of them. These formalisms usually require appropriate maps to specify discontinuous jumps between states (Back *et al.* 1993; Branicky *et al.* 1994) or implementing digital–analogical interfaces (Nerode and Kohn 1993).

Often these frameworks don't address the problem of synthesizing control actions when there are complex trajectory requirements and/or uncertainty in models and parameters.

Several automated analysis tools exist. In particular, (Henzinger and Ho 1995) presents HyTech, a program for proving properties of hybrid systems. They are modeled using *Linear Hybrid Automata*: discontinuous state jumps are specified as a combination of a finite state machine whose states are labeled with invariant conditions and ordinary differential equations; edges connecting two states are labeled with conditions that, when satisfied by the originating state of the automaton, trigger the transition. HyTech offers also a logical language (a CTL) for providing requirement specifications system verification. Compared to what we have presented here, the formalism used in HyTech to define the continuous model(s) is rather limited, as it can deal only with constant derivatives (authors show how, by using ad–hoc methods, also linear ODEs can be formulated). Furthermore HyTech does not deal with synthesis problems, nor with uncertainty.

Another simulation tool is reported in (Mosterman and Biswas 1996). The representation is based on bond–graphs and controlled junctions that can switch on or off parts of the model. These junctions are controlled by a finite state automaton whose transitions are labeled with inequality conditions. The tool is based on a method (called *Mythical Mode Algorithm*) that determines the possible valid sequences of configurations of the binary junctions leading to the "next" state of the continuous system. Discontinuities can be triggered by some external action, like closing a switch in an electrical circuit or by changes of internal variables, like a diode turning on or off when the voltage crosses a threshold. The method is notable in that it is automatic and it makes very general assumptions (compatibility and continuity laws need to hold). On the other hand, it doesn't address the uncertainty issue nor the synthesis of control actions.

Conclusion

Qualitative simulation has developed a number of techniques for using a discrete representation for reasoning about the behavior of continuous, dynamical systems. These techniques are particularly well suited to hybrid control with complex trajectory requirements. The work presented within this paper focuses on the problem of synthesizing a sequence of discrete control actions for a continuous plant under uncertainty. TeQsim supports simulation of such hybrid systems along with a declarative language for specifying goal constraints based upon temporal logic. It has been used to perform the tasks of plan validation and refinement on a non–trivial, real–world problem. A framework for performing plan generation has been proposed utilizing a phase portrait representation to reason about the affect of an action with respect to the goal specification. This approach would allow for the development of robust plans that are guaranteed to produce the desired behavior for the given model of the hybrid system. We are currently exploring ways to automate such a process.

Acknowledgements

This work has taken place while Giorgio Brajnik was visiting 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-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. Source code for TeQSIM is included in the most recent QSIM release available at **http://www.cs.utexas.edu/users/qr/**.

References

Back, A.; Guckenheimer, J.; and Meyers, M. 1993. A dynamical simulation facility for hybrid systems. In Grossman et al. (1993). 255–267.

Bradley, E. and Zhao, F. 1993. Phase-space control system design. *IEEE Control Systems* 2(13):39–47.

Brajnik, G. and Clancy, D. J. 1996a. Guiding and refining simulation using temporal logic. In *Proc. of the Third International Workshop on Temporal Representation and Reasoning (TIME'96)*, Key West, Florida. IEEE Computer Society Press. 144–151.

Brajnik, G. and Clancy, D. J. 1996b. Temporal constraints on trajectories in qualitative simulation. In *Working papers of the Tenth International Workshop for Qualitative Reasoning*, Fallen Leaf Lake, CA. 22–31. AAAI Technical Report WS-96-01.

Brajnik, G. and Clancy, D. J. 1996c. Temporal constraints on trajectories in qualitative simulation. In Clancey, B. and Weld, D., editors 1996c, *Proc. of the Tenth National Conference on Artificial Intelligence*. AAAI Press. 979–984.

Branicky, M. S.; Borkar, V. S.; and Mitter, S. K. 1994. A unified framework for hybrid control. In *Proc. 33rd IEEE Conf. Decision Control*, Lake Buena Vista, FL. 14228–4234.

Grossman, R.; Nerode, A.; Ravn, A.; and Reischel, H., editors 1993. *Hybrid Systems*, Lecture Notes in Computer Science 736. Springer Verlag.

Henzinger, T. A. and Ho, P. H. 1995. HyTech: The Cornell HYbrid TECHnology Tool. In Antsaklis, P.J.; Kohn, W.; Nerode, A.; and Sastri, S., editors 1995, *Hybrid Systems II*, Lecture Notes in Computer Science 999. Springer Verlag.

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

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

Lee, W. W. and Kuipers, B. 1993. A qualitative method to construct phase portraits. In *Proc. of the National Conference on Artificial Intelligence (AAAI-93).* AAAI/MIT Press.

Mosterman, P. J. and Biswas, G. 1996. A formal hybrid modeling scheme for handling discontinuities in physical system models. In *Proc. AAAI–96*, Portland (OR). AAAI Press. 985–990.

Nerode, A. and Kohn, W. 1993. Models for hybrid systems: automata, topologies, controllability, observability. In Grossman et al. (1993). 317–356.

Rawlings, J. B.; Meadows, E. S.; and Muske, K. R. 1994. Nonlinear model predictive control: a tutorial and survey. In *Proc. ADCHEM '94*, Kyoto, Japan.

Tavernini, L. 1987. Differential automata and their discrete simulators. *Nonlinear analysis, Theory, Methods and Applications* 11(6):665–683.