A Framework for Reasoning about Animation Systems

Eric Aaron, Dimitris Metaxas, Franjo Ivančić, and Oleg Sokolsky

Department of Computer and Information Science
University Of Pennsylvania
200 South 33rd Street
Philadelphia, PA USA 19104-6389
{eaaron, dnm}@graphics.cis.upenn.edu, {ivancic, sokolsky}@saul.cis.upenn.edu

Abstract. In this paper, we consider the potential for reasoning about animations in the language of hybrid dynamical systems (i.e., systems with both continuous and discrete dynamics). We begin by directly applying hybrid systems theory to animation, using a general-purpose hybrid system specification tool to generate multi-agent animations; this application also illustrates that hybrid system models can provide systematic, modular ways to incorporate low-level behavior into a design for higher-level behavioral modeling. We then apply the logical framework of hybrid systems to animation: we formally state properties of animation systems that may not be readily expressed in other frameworks; and we mechanically check a collision-avoidance property for a simple race-like game. This hybrid systems-oriented approach could improve our ability to reason about virtual worlds, thus improving our ability to create intelligent virtual agents.

Some properties of animation systems cannot be readily verified merely by watching a few graphical displays. As part of a motivating example, consider an animation like the following one, in which an agent's locomotion consists of three sequential segments. In the first segment, the agent tries to avoid obstacles; it is steered by a complex non-linear system of differential equations that has parameters to represent low-level attributes like aggression (how near an agent will get to an obstacle before starting to swerve around it). In the second segment, the agent is in no danger of colliding with obstacles, so it instantaneously switches to a simpler, linear course that takes it to a final target. The third segment begins upon it reaching its target: No matter where in time and space that goal is achieved, the agent waits precisely one second before switching to another behavior, perhaps literally swelling up with pride for successfully completing its task. Given a set of parameter values for the non-linear steering in the first segment, that animation system is fully determined. For such a fully determined system, an absence of agent-obstacle collisions could be verified by viewing one animation.

Consider now the task of verifying collision-avoidance for *all* the animation systems that could result from different sets of parameter values. It might not be

possible to check that property simply by viewing a small number of graphical displays, and we could certainly not watch every resulting animation. In cases such as this, where we want to reason about properties that we cannot easily see for ourselves, we might hope to ask for mechanical assistance. The first step toward that goal, however, is a major one: We must know how to model animation systems and state relevant properties formally and precisely. Once a linguistic framework for modeling and specification has been selected, we can investigate methods to assist us in reasoning.

To identify a candidate framework for reasoning about animation systems, we return to the animation described above. It is a *hybrid dynamical system* (*hybrid system*, for short), a combination of continuous and discrete dynamics; the agent's behavior is continuous in each segment, punctuated by instantaneous, discrete changes in behavior as it makes transitions between segments. There is a formal theory of hybrid systems [2, 10, 27], and although that theory has been employed in diverse application domains, animation is not typically considered in the hybrid systems literature. Nonetheless, some animation systems *are* hybrid systems, and hybrid systems theory can be applied to animation.

In this paper, we explore the potential for reasoning about animations in the language of hybrid systems. We begin by directly applying hybrid systems theory to animation, using the hybrid system specification and simulation tool CHARON [7, 8] to generate several animations (including one similar to the one described in English above). This is a necessary preliminary; without demonstrating that animation systems could be modeled in the theoretical framework of hybrid systems, we could not use logics for hybrid systems to reason about animation. As part of this demonstration, we show that a hybrid system model can provide a systematic, modular way to incorporate sophisticated low-level behavior into a design for higher-level behavioral modeling.

We then consider how logics for hybrid systems might be applied to animations. We begin by expressing several properties that may interest animators, explicitly representing properties of time and reachability in space. Although many properties of complex hybrid systems are theoretically undecidable, there are many significant decidable cases, and we use the verification tool HYTECH [21] to mechanically check a result about a simple game-like animation.

We demonstrate our approach through a series of applications and experiments involving multi-agent animations with actors, targets, and obstacles [3].

1 Applying Hybrid Systems Theory to Multi-agent Animations

The presence of systems with both continuous and discrete dynamics is not new in animation, but it is not always clear how these systems relate to well-understood hybrid system models. In contrast, we make a strong connection to existing hybrid systems theory by using the hybrid system tool CHARON [7, 8] to implement multi-agent animation systems. Because of that connection, we

are able to investigate the use of logics for hybrid systems (see section 3.1) as a framework for reasoning about animation.

We base our animations primarily on the agent-steering method presented in [17, 18]. Below, we briefly review the tools we employed to create our animations and discuss some issues particular to implementing an animation system as a hybrid system in CHARON.

1.1 A Dynamical System for Agent Steering

There have been many approaches to guiding the behavior of autonomous agents. Logicist, artificial intelligence-based techniques have been successfully used for cognitively empowered agents [25] and animated actors [16]; perception and dynamics-based techniques [11, 29, 34] are often more readily able to adapt to dynamic environments. Our particular approach to low-level agent navigation —the continuous component of our hybrid animation system— is based on the method in [17, 18], a scalable, adaptive approach to modeling autonomous agents in dynamic virtual environments. Like treatments of similar issues in the field of behavioral robotics [24], we consider only two-dimensional motion, although the mathematical foundations for three-dimensional navigation already exist [18].

Our animated worlds consist of three kinds of agents: *actors*, *targets* that represent actors' goals, and *obstacles* that actors attempt to avoid. All are graphically drawn as spheres; for our purposes, it suffices to represent an agent by its size, location, heading angle, and velocity.[1] There may be multiple actors, obstacles, and targets in an animation system. Further, obstacles and targets may be static and/or moving. These components provide a general conceptual palette that can be used to express a broad range of behaviors. For instance, an actor performing a multi-part task could be represented by its reaching a series of targets in sequence, each target corresponding to a component subtask.

At the core of the mathematics underlying our animated worlds are non-linear *attractor* and *repeller* functions that represent the targets and obstacles (respectively) in the system. Another non-linear system combines their weighted contributions in calculating an actor's angular velocity, dynamically adapting to real-time changes in the environment. Together, these non-linear systems generate natural-appearing motion, avoiding collisions and other undesirable behaviors. The agent heading angle ϕ is computed by a non-linear dynamical system of the form:

$$\dot{\phi} = f(\phi, \mathbf{env}) = |w_{tar}| f_{tar} + |w_{obs}| f_{obs} + n, \tag{1}$$

where f_{tar} and f_{obs} are the attractor and repeller functions for the system, and w_{tar} and w_{obs} are their respective weights on the agent. (n is a noise term, which helps avoid local minima in the system.)

[1] The mathematical treatment in [18] admits a more complex representation of actors than the one we use.

The weights themselves are determined by computing the fixed points of the following non-linear system:

$$\begin{cases} \dot{w}_{tar} = \alpha_1 w_{tar}(1 - w_{tar}^2) - \gamma_{12} w_{tar} w_{obs}^2 + n \\ \dot{w}_{obs} = \alpha_2 w_{obs}(1 - w_{obs}^2) - \gamma_{21} w_{obs} w_{tar}^2 + n \end{cases}, \tag{2}$$

where the α and γ parameters are designed to reflect conditions for the stability of the system. Many other parameters are also concealed in the terms presented above. For instance, a repeller function f_{obs} depends on parameters that determine how much influence that obstacle will have on an actor.

This is only an overview of one significant part of the agent steering system. There is considerably more detail to the system, including applications to three-dimensional environments, dynamic control of forward velocity, and modeling of low-level personality attributes such as aggression and agility. The above presentation, however, gives a feel for the kind of mathematics involved, suggesting the complexity involved in implementing it. Further, it introduces the role parameters may play in agent behavior, a notion to which we return in section 3 when discussing reasoning about a parameterized class of animation systems.

1.2 Hybrid Systems and CHARON

By definition, a hybrid system is one that combines continuous and discrete dynamics. Past domains of application for hybrid system models include descriptions of biological processes [4], air-traffic management systems [26, 33], and manufacturing systems [31]. They occur frequently and naturally in many contexts and, because of this, they have received substantial attention by both computer scientists and control theorists [1, 2, 10, 27]. From a general, intuitive perspective, any system characterized by discrete transitions between modes of continuous control is a hybrid system. This includes several kinds of systems that emerge in animation, from physical modeling of objects in a dynamic environment to agent-steering.

There are several different formal models for hybrid systems. Net-based models such as Constraint Nets [35], for instance, have been acknowledged in literature on cognitive agents. We focus in particular on automata-theoretic models such as hybrid automata [5, 28]; the various kinds of automata differ in the behaviors they are capable of representing. As a brief, non-technical introduction to this perspective, we consider a hybrid automaton as having: a set of *discrete states* called *control modes*; a *continuous state space* (a subset of $\mathbb{R}^n$ for some n); and descriptions of how the system can evolve. There are constraints both on the continuous evolution of the system within a control mode and on the discrete transitions between control modes that the system might make. A state of the overall system is a pair (*control mode*, *continuous state*). (For more details on the mathematics and applications of hybrid automata, see [5, 9, 13, 19].) Research and analysis of hybrid automata underlies practical tools such

as CHARON [7,8] and the model checker HYTECH [21]. For this paper, we use CHARON to implement animation systems and HYTECH for verification.[2]

The architecture of a hybrid system in CHARON is expressed as *hierarchical agents*, a model conceptually similar to hybrid automata and hierarchical reactive modules [6]. The key features of CHARON are:

Hierarchy. The building block for describing the system architecture is an *agent* that communicates with its environment via shared variables. The building block for describing flow of control inside an atomic agent is a *mode*. A mode is basically a hierarchical state machine, i.e., it may have submodes and transitions connecting them. CHARON allows *sharing* of modes so that the same mode definition can be instantiated in multiple contexts.

Discrete updates. Discrete updates are specified by *guarded actions* labeling transitions connecting the modes. Actions may call externally defined Java functions to perform complex data manipulations.

Continuous updates. Some of the variables in CHARON can be declared *analog*, and they flow continuously during continuous updates that model passage of time. The evolution of analog variables can be constrained in three ways: *differential* constraints (e.g., by equations such as $\dot{x} = f(x,u)$), *algebraic* constraints (e.g., by equations such as $y = g(x,u)$), and *invariants* (e.g., $|x-y| \leq \varepsilon$) that limit the allowed durations of flows. Such constraints can be declared at different levels of the mode hierarchy.

Modular features of CHARON allow succinct and structured description of complex systems. (Similar features are supported by the languages SHIFT [14] and STATEFLOW (see `www.mathworks.com`).) Among other benefits, this modularity provides a natural-seeming structure for developing animation systems with multiple levels of behavior.

2 Creating Animations as Hybrid Systems

Animation systems are implemented in CHARON using the key concepts noted in section 1.2.[3] Modes are created to represent continuous behaviors; particular continuous dynamics (e.g., the non-linear system described in section 1.1) are represented as differential or algebraic constraints of a form such as `diff { d(angle) = AngleFunction(angle,...) }`. If constraints are necessary to limit the time in a particular mode, they are represented as invariants such as `inv {Cond && !Cond2 && distance(x,y)<=distance(x,z) }`. Guarded transitions between modes are presented in a straightforward `trans from Mode1 to Mode2 when Cond do Effect` syntax; when the guard `Cond` is true, the transition named `trans` is enabled, and if it is taken, statement `Effect` is executed along with the system's jump from `Mode1` to `Mode2`. The behavior of agents

[2] We used HYTECH for verification because, as of this writing, the model checking facilities for CHARON are still under development.

[3] A more detailed description of the CHARON language is presented in [32].

follows from the systems described by modes. Each atomic agent is declared to begin in some mode, and it follows the behavior described there. The behavior of a hierarchical agent is, of course, determined by the behavior of its sub-agents. In this way, the underlying continuous mathematics and relations between modes of behavior are explicitly represented in a CHARON program. Further, the modularity of CHARON code makes it easy to change one aspect of a system while leaving others intact.

CHARON also generates numerical simulations of hybrid systems, which we exploited in creating animations from CHARON system specifications. We simply simulated our animation systems in CHARON, then used a small translation routine (like a Perl script) to format the output of those simulations so that a previously developed application (developed for research outside of the context of hybrid systems) could create graphical displays. Section 4 contains more details and sample images of the animations we generated.

The CHARON model of hybrid systems as hierarchical agents corresponded neatly to the high-level abstractions we considered when designing animations. In addition, the explicit representation of high-level (discrete) and low-level (continuous) processes made it straightforward to implement different kinds of cognitive behavior or intelligence in our agents. For instance, constraints on low-level perceptual capabilities (e.g., how far can an agent "see") or underlying behavioral attributes (e.g., aggression) were explicitly represented in the code that controls continuous behavior. Higher-level decisions to switch modes of behavior could be explicitly represented by adding to the discrete dynamics; new modes could represent new "states of mind."

3 Reasoning about Animation Systems

Some properties of games and other animation systems may not be verifiable by viewing a single animation. For example, one might want to verify properties of all possible executions of a parameterized or non-deterministic system. Even within a fully determined system, properties about agents' relative speed and precise distance may be too difficult to judge by eye. Indeed, merely finding a formal language to express interesting properties of such systems may be non-trivial.

This touches upon a motivating observation behind our research: Well-known logics for hybrid systems are capable of expressing properties of animation systems. In addition, there are practical *model checkers* —tools that can mechanically verify some properties of simple hybrid systems— that we might apply to animation systems. These model checkers have significant limitations; many properties are theoretically undecidable, and as a practical matter, even decidable properties may only be feasibly checked in simple cases. Still, as we discuss in section 5, there are approaches to reasoning about complex systems that allow us to circumvent some undecidability barriers.

In this section, we discuss more about logics for hybrid systems and how we might apply them to animation systems.

3.1 Modal Logic and Properties of Multi-agent Animations

There are many *modal* and *temporal* logics that can be used to reason about hybrid systems, such as **CTL**, **LTL**, and the *μ-calculus*; significant research has been devoted to the theory and applications of these logics ([13] and [9] are good surveys of recent work). For readers unfamiliar with fundamental modal logic operators, we provide a brief review.[4]

Modal logics are used to reason about *possible worlds* and properties of *possibility* and *necessity* such as "Proposition P is true in all possible worlds" or "Proposition P is false in some possible world." For our present application, we consider a "world" to be a "state of a hybrid system," and we consider a "possible world" to be a state reachable (under the constraints on system evolution) from the current state of the hybrid system.

A modal logic typically contains the standard propositional logic operators (negation, implication, etc.) along with various modal or temporal operators. For this paper, we introduce two common modal operators:

- **possibility**: $\Diamond P$ (intuitively, "It is possible that P")
- **necessity**: $\Box P$ (intuitively, "It is necessary that P")

As expected, they are duals: $\neg\Box\neg P \equiv \Diamond P$. In the context of a system execution, possibility and necessity also correspond to the intuitive readings of *eventually* and *always* (respectively). That is, $\Box P$ means that P is necessarily true of every state of the entire execution; it is always true. As its dual, $\Diamond P$ means that P is not always false; it is true at some state of the execution, i.e., eventually true. (From a rigorous logical standpoint, these explanations are overly simplistic, but they convey basic intuitions necessary for this paper.)

Logics for hybrid systems are powerfully expressive. In addition to modal operators such as $\Box$ and $\Diamond$, they explicitly represent time, and we may specify that a condition be true at some particular time in an animation. We can also express properties of non-deterministic animation systems or parameterized classes of animations. We illustrate these points by presenting several example properties below. In each case, we formally express that the system execution E satisfies property P by writing $E \models P$, and we use the notation $loc(A)$ to refer to the location of an agent A.

- The velocity of actor a_1 is never greater than that of actor a_2:
  $E \models \Box(velocity(a_1) \leq velocity(a_2))$

There are several ways to reason about parameterized or non-deterministic systems. In the logical formulas below, we do so by quantifying over all possible executions of a system.

[4] By no means does this section constitute a thorough introduction. There are many noteworthy modal logic texts available, providing differently oriented introductions to the basic concepts; for example, a classic general introduction such as [23] may supply substantially different insights than more-directed texts such as [15]. The paper [13] from Davoren and Nerode may be of particular interest to readers who seek an overview of logics for hybrid systems.

Fig. 1. An image from a crowd simulation animation. The crowd of actors (light spheres) on the bottom half of the image are moving to a target (darker sphere) on the top, crossing through actors that are moving from the top of the image to a target on the bottom.

- If the aggression parameter of actor a is set within the range $[2,4]$, actor a reaches target t:
  $(\forall E)E \models (aggression \in [2,4]) \Rightarrow \Diamond(loc(a) = loc(t))$

- No matter what non-deterministic choices are made, at 5 seconds into the animation, agent a_2 is at least 100 units from agent a_1:
  $(\forall E)E \models \Box((clock = 5) \Rightarrow distance(a_1, a_2) \geq 100)$

- No matter what non-deterministic choices are made, agent a_1 is at target t when agent a_2 is not, but agent a_2 eventually reaches target t:
  $(\forall E)E \models \Diamond(loc(a_1) = loc(t) \wedge loc(a_2) \neq loc(t) \wedge \Diamond(loc(a_2) = loc(t)))$

4 Experiments and Applications

4.1 Multi-agent Animations

Figures 1-4 contain images from our multi-agent animations. In this paper, actors are the lightest objects, obstacles are the darkest objects, and targets are an intermediate gray. In the actual animations, actors, obstacles, and targets are distinguished by color. (CHARON-generated animations, including ones from which these figures were taken, may be found at [3].)

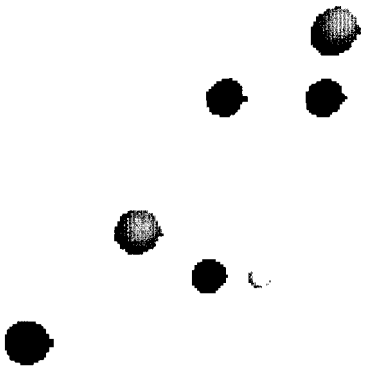

Fig. 2. An actor (lightest object) has already swerved around a stationary obstacle (darkest) and is now reacting to avoid a moving obstacle.

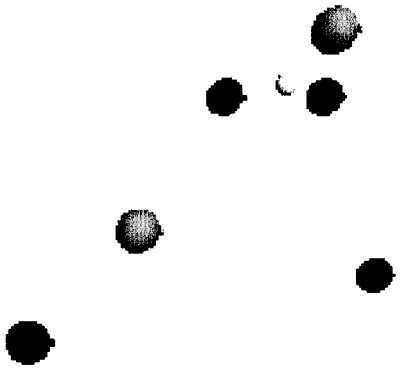

Fig. 3. The same actor as in Figure 2, later in the same animation, having switched to simpler behavior and passing between two obstacles.

Figure 1 shows a frame from a crowd simulation animation in which several actors need to navigate around other actors to reach distant targets. Figures 2 and 3 show frames from a CHARON animation similar to the one described at the beginning of this paper, in which the actor's behavior is described in three segments. In the first segment (Figure 2), the actor avoids a static obstacle and a moving obstacle on the way to a first target. It need not engage in complex evasive behavior as it progresses to the second target in the second segment (Figure 3); it will fit between the two remaining obstacles by simply traveling in a straight line, so it may intelligently switch to a simpler system for navigation. That resulting animation system thus contains a discrete transition between continuous dynamics and is straightforwardly represented as a hybrid system. Note also that if the actor did not switch to a simpler behavior for segment two, it would perform unnecessary obstacle avoidance, as shown in Figure 4. Further, the CHARON specification of the actor's behavior is on the intuitive level of "reach target one, avoiding obstacles; reach target two, simply going straight; wait one second, then celebrate." It does not require details about the location in space (or time) of targets or obstacles.

4.2 Verifying Properties

Although there are significant undecidability barriers when reasoning about hybrid systems in general, property verification is decidable for restricted classes of hybrid systems [9,22]. Consider, for example, systems in which each mode constrains every variable to constant velocity; changes in direction or velocity require transitions between modes. Some animation systems can be specified

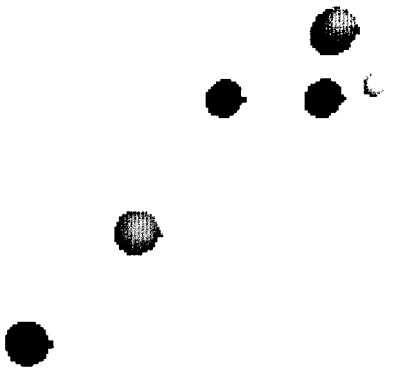

Fig. 4. Contrast with Figure 3: Here, the actor maintains the complex, obstacle-avoidant behavior, and winds up taking a longer route to the target. (Note that the moving obstacle has moved off-screen.)

within this restricted framework, and many properties expressible in the modal logic framework described in section 3.1 are decidable for such systems.

To demonstrate this, we specified the rudiments of a race-like game in the model checker HYTECH [21] and mechanically checked a collision-avoidance property. Our animation system contains three agents, two racing actors and one obstacle, each moving at constant speed around a square, two-lane track. The rules of the race encode that each actor (naturally) prefers to race on the inside lane, and must do so whenever possible. Because one racer is slower than the other and the obstacle is slower than both, racers may move to the outside lane to pass slower agents, moving promptly back to the inside lane when they are done passing.

Consider an infinite race, an endless execution of this animation system. Will the two racers ever collide? Although we thought we had specified collision-avoidant behavior, we were mistaken. By checking the states reachable by this animation system, HYTECH discovered a scenario in which a collision would occur: when the faster racer is changing lanes in front of the slower racer *at the corner of the track*, as graphically represented in Figure 5. Note that Figure 5 is not a frame from an actual animation; we did not use HYTECH to generate animations. A simple animation of a similar race-like game implemented in CHARON can be found in [3].

5 Conclusions and Extensions

Hybrid continuous/discrete dynamics are commonly utilized in animations without regard for underlying hybrid systems theory. We took a different approach, directly applying general-purpose hybrid systems theory to generate multi-agent

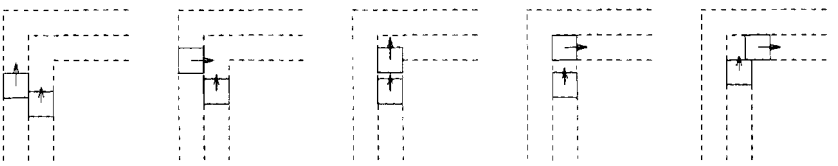

Fig. 5. A block diagram of unexpected collision behavior in a simple race game, as detected by HyTech. The faster racer has passed the slower one, but the slower one catches up on a corner.

animations, explicitly linking the normally disparate disciplines by demonstrating that some animation systems may be modeled by hybrid automata. Models of hybrid systems can simultaneously integrate and distinguish high-level and low-level behavior; thus, they may be natural tools for modeling intelligent virtual agents.

Because we were within the theoretical framework of hybrid systems, we were also able to specify and reason about properties of animation systems using expressive modal logics for hybrid systems. Although automatic verification of complex properties is infeasible in many cases, we did mechanically check a property about collisions in a race-like game animation.

Furthermore, we need not simply abandon hope of verifying complex properties of complex systems: There are *approximation* techniques for verification that might be applicable to animation systems. Properties such as those discussed in this paper are *reachability* properties of animation systems, fundamentally about whether some proposition holds in the states reachable by an animation system. It is often impossible to effectively reason about the exact set of reachable states of a complex system, but we may be able to verify properties on an approximation of that set. That is, if S is the actual set of states reachable by a system, and we cannot decide property P on S, we might instead be able to overestimate S by a computationally simpler set $S' \supset S$ on which P is decidable. Then, if we prove that P holds on all states in S', we know P also holds on all states in S. This kind of reasoning by approximation is an active area of research in the hybrid systems community [12,20,30], but it has not yet been explored in the context of animation systems.

There is an understood connection between logic and animation: Our ability to reason about virtual worlds is essential to our ability to create intelligent virtual agents. This observation underlies the groundbreaking, artificial intelligence-based approach to cognitive modeling taken by Funge [16], which permits animated characters to simulate cognitive abilities such as perception, inference, and planning. Potentially, a hybrid systems-oriented approach to cog-

nitive modeling could escape some typical AI-based restrictions. Such a cognitive model could follow the delineation of high-level and low-level cognitive behavior mentioned in section 2, with discrete hybrid system modes corresponding to "states of mind" and continuous mathematical systems representing mental activity; mental states of animated characters could vary in real time. It could also enable characters to reason directly about time, not just about endpoints of previously determined discrete events. In addition, it could allow the characters' cognitive and physical systems to be cleanly integrated and expressed in the same mathematical language. It is an open question, however, whether an implementation of this approach to cognitive modeling might enjoy the same virtues of practical applicability as Funge's system.

Despite the clear relationship between hybrid systems theory and animation systems, this natural interdisciplinary interface has not been well explored. We do not know what models of hybrid systems might reveal about the mathematical structure of animation systems, or if there is any mathematical structure to be exploited for effective reasoning with modal logics. We do believe, however, that further exploration can improve human inference about animation systems, extending our perspective on and vocabulary of animation.

Acknowledgments

We thank Siome Goldenstein for his advice and technical assistance. We also thank Thao Dang, Jan Allbeck, and Norm Badler for helpful discussions. This research was supported in part by NSF grant NSF-SBR 8920230.

References

1. *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43(4), April 1998.
2. *Proceedings of the IEEE*, 88, July 2000.
3. E. Aaron, F. Ivančić, and S. Goldenstein. CHARON-generated animations. Available at http://www.cis.upenn.edu/~eaaron/IVA01_animations.html.
4. R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes In Computer Science*. Springer Verlag, April 2001.
5. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
6. R. Alur and R Grosu. Modular refinement of hierarchic reactive machines. In *Proceedings of the 27th Annual ACM Symposium on Principles of Programming Languages*, 2000.
7. R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in CHARON. In N. Lynch and B. H. Krogh, editors, *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

8. R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional refinement for hierarchical hybrid systems. In *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 33–48. Springer Verlag, 2001.
9. R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, July 2000.
10. R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
11. David Brogan, Ronald Metoyer, and Jessica Hodgins. Dynamically simulated characters in virtual environments. *IEEE Computer Graphics and Applications*, 18(5):59–69, Sep/Oct 1998.
12. T. Dang and O. Maler. Reachability analysis via face lifting. In T. Henzinger and S. Sastry, editors, *Hybrid Systems : Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109. Springer Verlag, Berlin, 1998.
13. J. Davoren and A. Nerode. Logics for hybrid systems. *Proceedings of the IEEE*, 88:985–1010, July 2000.
14. A. Deshpande, A. Göllu, and L. Semenzato. Shift programming language and run-time systems for dynamic networks of hybrid automata. Technical report, University of California at Berkeley, 1997.
15. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
16. J. Funge. *AI for Games and Animation*. A K Peters, 1999.
17. S. Goldenstein, E. Large, and D. Metaxas. Non-linear dynamical system approach to behavior modeling. *The Visual Computer*, 15:349–369, 1999.
18. Siome Goldenstein, Menelaos Karavelas, Dimitris Metaxas, Leonidas Guibas, and Ambarish Goswami. Scalable dynamical systems for multi-agent steering and simulation. In *Proceedings of the IEEE Conference in Robotics and Automation*, May 2001. to appear.
19. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
20. T.A. Henzinger and P.-H. Ho. A note on abstract-interpretation strategies for hybrid automata. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 252–264. Springer-Verlag, 1995.
21. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science 1019*, pages 41–71. Springer-Verlag, 1995.
22. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.
23. G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Mehuen and Co., New York, 1968.
24. E. Large, H. Christensen, and R. Bajcsy. Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *International Journal of Robotics Research*, 18(1):37–58, 1999.
25. H. Levesque and F. Pirri, editors. *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Springer, 1999.
26. J. Lygeros, G. J. Pappas, and S. Sastry. An approach to the verification of the Center-TRACON Automation System. In T. Henzinger and S. Sastry, editors,

Hybrid Systems : Computation and Control, volume 1386 of *Lecture Notes in Computer Science*, pages 289–304. Springer Verlag, Berlin, 1998.
27. N. Lynch and B. H. Krogh, editors. *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.
28. N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996.
29. H. Noser, O. Renault, D. Thalmann, and N. Thalmann. Navigation for digital actors based on synthetic vision, memory and learning. *Computer and Graphics*, 1995.
30. G. J. Pappas and S. Sastry. Towards continuous abstractions of dynamical and control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 329–341. Springer Verlag, Berlin, Germany, 1997.
31. D. Pepyne and C. Cassandras. Hybrid systems in manufacturing. *Proceedings of the IEEE*, 88:1108–1123, July 2000.
32. O. Sokolsky, Y. Hur, R. Grosu, and E. Aaron. *CHARON Language Manual, Version 0.6*. University of Pennsylvania, 2000. Available at http://www.cis.upenn.edu/mobies/charon/CHARONmanual.ps.
33. C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management : A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.
34. X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. of SIGGRAPH '94*, pages 43–50, 1994.
35. Y. Zhang and A. Mackworth. Constraint nets: A semantic model for hybrid dynamic systems. *Theoretical Computer Science*, 138(1):211–239, 1995.