Specification and Verification of Hybrid Dynamic Systems with Timed ∀-automata

Ying Zhang[1] and Alan K. Mackworth*[2]

[1] Wilson Center for Research, Xerox Corporation, M/S 128-51E, Webster, N.Y., USA 14580, zhang@wrc.xerox.com
[2] Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada V6T 1Z4, mack@cs.ubc.ca

Abstract. The advent of computer-controlled embedded systems coupled to physical environments requires the development of new theories of dynamic system modeling, specification and verification. We present Timed ∀-automata, a generalization of ∀-automata [10], for the specification and verification of dynamic systems that can be discrete, continuous or hybrid. Timed ∀-automata are finite state and serve as a formal requirements specification language for dynamic systems so that (1) timed as well as temporal properties can be specified or recognized, and (2) global properties of either discrete or continuous behaviors can be characterized. In addition, we propose a formal model-checking method for behavior verification of dynamic systems. This method generalizes stability analysis of dynamic systems and can be completely automated for discrete-time finite-domain systems.

1 Motivation and Introduction

A robot is, typically, a real-time embedded system, consisting of a controller coupled to its plant. In general, all the computer-controlled systems in our daily lives, such as cars, elevators and copiers, can be considered to be robots. With the growing demand for robots, we face a major challenge: the development of intelligent robots that are reliable, robust and safe in their working environments [7]. Computer-controlled systems are discrete and physical plants or environments are, in general, continuous. Therefore, the coupling of a controller, a plant and its environment constitutes a complex dynamic system that is, in general, hybrid.

A robotic system is the symmetrical coupling of a robot to its environment. We have decomposed the development of a robotic system into three phases: system modeling, requirements specification and behavior verification [17, 20]. System modeling represents a complex dynamic system in terms of its compositions and interconnections, so that the overall behavior of the system is precisely defined. Requirements specification expresses global properties such as safety, reachability, liveness and real-time response. Behavior verification ensures

* Fellow, Canadian Institute for Advanced Research

that the behavior of the modeled system satisfies the specified requirements. We replace the vague question "Is the robot intelligent?" with the question "Will the robot do the right thing?" [18]. The answer to that question follows if we can:

1. model the coupled robotic system at a suitable level of abstraction,
2. specify the required global properties of the system, and
3. verify that the model satisfies the specification.

Most robot design methodologies use hybrid models of hybrid systems, awkwardly combining off-line computational models of high-level perception, reasoning and planning with on-line models of low-level sensing and control. We have developed Constraint Nets (CN) as a semantic model for hybrid dynamic systems [16, 19]. CN introduces an abstraction and a unitary framework to model discrete/continuous hybrid systems; therefore, the robot and its environment can be modeled symmetrically in a uniform model.

In this paper, we present Timed $\forall$-automata for the specification and verification of dynamic systems that can be discrete, continuous or hybrid. Timed $\forall$-automata are a generalization of $\forall$-automata [10] that have been developed for the specification and verification of concurrent programs. $\forall$-automata are powerful enough to specify global properties such as safety, reachability and liveness, and simple enough to have a formal verification procedure. Timed $\forall$-automata are finite state and serve as a formal requirements specification language for dynamic systems so that (1) timed as well as temporal properties can be specified, and (2) global properties of either discrete or continuous behaviors can be characterized. The link between a constraint net model and a timed $\forall$-automata specification is the behavior of the system, which can be represented as a generalized Kripke structure. We propose a formal model-checking method for behavior verification of hybrid dynamic systems. This method generalizes stability analysis of dynamic systems and can be completely automated for discrete-time finite-domain systems.

The rest of this paper is organized as follows. Section 2 defines the basic concepts of dynamic systems: time, traces, transductions, behaviors, and generalized Kripke structures. Section 3 develops Timed $\forall$-automata, giving the syntax and semantics. Section 4 proposes a formal method for behavior verification. Section 5 concludes this paper and points out future research directions. Theorems and lemmas are proved in Appendix A. A typical cat-mouse [4] example is used throughout the paper to illustrate the ideas. However, the method is applied to various robot testbeds in our Laboratory for Computational Intelligence at UBC, including robot soccer players [13, 21].

2 Dynamic Systems and Behaviors

In this section, we define the basic concepts of dynamic systems: time, traces, transductions, behaviors and generalized Kripke structures.

2.1 Time, Traces and Transductions

The key to understanding dynamic systems is understanding time. For our purpose, time is a linearly ordered set with a least element, the start time point. A measure is defined on some subsets of time points so that the duration of an interval of time can be captured. Formally, a time structure is defined as follows.

Definition 2.1 (Time structure) *A* time structure *is a pair* $\langle \mathcal{T}, \mu \rangle$ *where*

- $\mathcal{T}$ *is a linearly ordered set* $\langle \mathcal{T}, \leq \rangle$ *with* $\mathbf{0}$ *as the least element;*
- *let* σ *be a family of subsets of* $\mathcal{T}$ *including* $[\mathbf{0}, t)$[3] *for all* $t \in \mathcal{T}$, $\langle \mathcal{T}, \sigma \rangle$ *be a measurable space, and* $\mathcal{R}^+$ *be the set of non-negative real numbers; then,* $\mu : \sigma \to \mathcal{R}^+ \cup \{\infty\}$ *is a measure and* $\langle \mathcal{T}, \sigma, \mu \rangle$ *forms a measure space*[4]*.*

In this paper, we assume that (1) time is *complete*, i.e., for any subset of time points, if there is an upper (lower) bound, there is a least upper (greatest lower) bound, and (2) time is *infinite*, i.e., $\mu(\mathcal{T}) = \infty$. Condition (1) is important for our purpose; it guarantees that the time structure is well-defined: both Zeno sequence and rational time are excluded. Any time structure can be augmented to satisfy condition (2) by adding infinite number of time points. A time structure is *discrete* iff $\forall t > \mathbf{0}, [\mathbf{0}, t)$ is finite; it is *continuous* iff $\forall t_1, t_2, t_1 < t_2 \Rightarrow \exists t, t_1 < t < t_2$. For example, the set of natural numbers $\mathcal{N}$ and the set of non-negative real numbers $\mathcal{R}^+$, with $\mu([0, t)) = t$, are both time structures; $\mathcal{N}$ is discrete and $\mathcal{R}^+$ is continuous. If $\mathcal{T}$ is discrete, for any $t > \mathbf{0}$, let $pre(t)$ denote the least upper bound of $[\mathbf{0}, t)$. Furthermore, let $\mu([t_1, t_2)) = \mu([\mathbf{0}, t_2)) - \mu([\mathbf{0}, t_1))$.

The study of dynamic systems is the study of changes over time. Changes over time can be captured by traces. Formally, let $\mathcal{T}$ be a time structure and A be a domain of values. A *trace* v is a mapping from time to a domain, i.e., $v : \mathcal{T} \to A$. For example, $v = \lambda t.e^{-t}$ is a trace. The set of all traces from time $\mathcal{T}$ to domain A forms a *trace space*, denoted $A^{\mathcal{T}}$.

A dynamic system is composed of a set of interconnected transformational processes. Transformational processes can be captured by transductions. A transduction is a causal mapping from an input trace space to an output trace space. Formally, let $\mathcal{T}$ be a time structure, A and A' be domains, which can also be products of domains. Let $A^{\mathcal{T}}$ and $A'^{\mathcal{T}}$ be trace spaces. Two traces v_1, v_2 in the same trace space are *coincident up to* t, written $v_1 \simeq_{\leq t} v_2$, iff $\forall t' \leq t$, $v_1(t') = v_2(t')$. A mapping $F : A^{\mathcal{T}} \to A'^{\mathcal{T}}$ is a *transduction* iff $\forall v_1, v_2, t, v_1 \simeq_{\leq t} v_2 \Rightarrow F(v_1) \simeq_{\leq t} F(v_2)$. For example, a state automaton with an initial state defines a transduction on discrete time; a temporal integration with a given initial value is a typical transduction on continuous time. A *transliteration* is a primitive transduction where the output at any time is a function of the input at that time. Just as nullary functions represent constants, nullary transductions represent traces.

[3] $[t_1, t_2) = \{t | t_1 \leq t < t_2\}$.

[4] The concepts of measurable space, measure and measure space follow [12].

2.2 Behaviors of Dynamic Systems

A dynamic system can be represented as a set of equations, each of which corresponds to a transduction: $v_i = F_i(v_1, ..., v_n), i = 1, ..., m, m \leq n$. A *trace of the dynamic system* is a tuple $\langle v_1, ..., v_n \rangle$ that satisfies the set of equations [16]. The *behavior of the dynamic system* is the set of traces of the dynamic system.

Consider an example of *Cat and Mouse* modified from [4]. Suppose a cat and a mouse start running from initial positions X_c and X_m, respectively, with $X_c > X_m > 0$ and with constant velocities $V_c < V_m < 0$. Both of them will stop running when the cat catches the mouse, or the mouse runs into the hole in the wall at 0. Let x_c and x_m be the position traces of the cat and the mouse, respectively. This system can be modeled by the following set of equations:

$$x_c = \int (X_c)(V_c \cdot r), \quad x_m = \int (X_m)(V_m \cdot r), \quad r = (x_c > x_m) \wedge (x_m > 0) \qquad (1)$$

where $\int(X)$ is a temporal integration with initial state X. At any time t, $r(t)$ is 1 if the running condition $(x_c(t) > x_m(t)) \wedge (x_m(t) > 0)$ is satisfied and 0 otherwise. The behavior of this system is the set of tuples $\langle x_c, x_m \rangle$, each of which satisfies the set of equations.

A useful and important type of behavior is state-based and time-invariant. Intuitively, a state-based and time-invariant behavior is a behavior whose traces after any time are totally dependent on the current snapshot. State-based and time-invariant behaviors can be defined using generalized Kripke structures. We define a *generalized Kripke structure* $\mathcal{K}$ as a triple $\langle \mathcal{S}, \rightsquigarrow, \Theta \rangle$ where $\mathcal{S}$ is a set of states, $\rightsquigarrow \subset \mathcal{S} \times \mathcal{R}^+ \times \mathcal{S}$ is a state transition relation, and $\Theta \subseteq \mathcal{S}$ is a set of initial states. We denote $\langle s_1, t, s_2 \rangle \in \rightsquigarrow$ as $s_1 \overset{t}{\rightsquigarrow} s_2$. The state transition relation $\rightsquigarrow$ satisfies the following conditions:

- *initiality*: $s \overset{0}{\rightsquigarrow} s$;
- *transitivity*: if $s_1 \overset{t_1}{\rightsquigarrow} s_2$ and $s_2 \overset{t_2}{\rightsquigarrow} s_3$, then $s_1 \overset{t_1+t_2}{\rightsquigarrow} s_3$;
- *infinity*: $\forall s \in \mathcal{S}, \exists t > 0, s' \in \mathcal{S}, s \overset{t}{\rightsquigarrow} s'$.

For example, $\langle \mathcal{R}, \rightsquigarrow, \Theta \rangle$ with $s_1 \overset{t}{\rightsquigarrow} s_2$ iff $s_2 = s_1 e^{-t}$ is a generalized Kripke structure.

A time structure $\mathcal{T}$ is a time structure of $\mathcal{K}$ iff (1) for any time point t_1 in $\mathcal{T}$ and any transition $s_1 \overset{t}{\rightsquigarrow} s_2$ in $\mathcal{K}$, there is $t_2 > t_1$ in $\mathcal{T}$ such that $t = \mu([t_1, t_2))$, and (2) for any time points t_1 and t_2 with $t_1 < t_2$ and any state s_1 in $\mathcal{S}$, there is s_2 in $\mathcal{S}$ such that $s_1 \overset{\mu([t_1,t_2))}{\rightsquigarrow} s_2$.

Lemma 2.1 *If $\mathcal{T}$ is a discrete time structure of $\mathcal{K}$, then there is $\delta > 0$, for any $t > 0, \mu([pre(t), t)) = \delta$. And $\mathcal{K}$ can be represented as the transitive closure of transitions of next relation $s_1 \overset{\delta}{\rightsquigarrow} s_2$.*

For example, $\langle \mathcal{R}, \rightsquigarrow, \Theta \rangle$ with $s_1 \overset{n\delta}{\rightsquigarrow} s_2$ iff $s_2 = f^n(s_1)$[5] is a generalized Kripke structure with discrete time.

[5] apply function f n times

Not all generalized Kripke structures have time structures. For example, the transitive closure of a two-state transition system $s_1 \overset{1}{\rightsquigarrow} s_2, s_2 \overset{2}{\rightsquigarrow} s_1$ has no time structure. However, by adding an intermediary state s_2', the transitive closure of $s_1 \overset{1}{\rightsquigarrow} s_2, s_2 \overset{1}{\rightsquigarrow} s_2', s_2' \overset{1}{\rightsquigarrow} s_1$ has a time structure. A generalized Kripke structure is *well-defined* iff it has a time structure.

A *trace of a well-defined generalized Kripke* $\mathcal{K}$ on its time structure $\mathcal{T}$ is a mapping $v : \mathcal{T} \rightarrow S$ such that (1) $v(\mathbf{0}) \in \Theta$ and (2) $\forall t_1, t_2, t_1 < t_2 \Rightarrow v(t_1) \overset{\mu([t_1,t_2))}{\rightsquigarrow} v(t_2)$. The *behavior of* $\mathcal{K}$ on $\mathcal{T}$ is the set of traces of $\mathcal{K}$ on $\mathcal{T}$. The behavior of a dynamic system is *state-based and time-invariant* iff it is equal to the behavior of some generalized Kripke structure. In the rest of this paper, we focus on state-based and time-invariant behaviors represented as generalized Kripke structures.

3 Timed ∀-automata for Requirements Specification

In this section, we define Timed ∀-automata, giving the syntax and semantics. First, we introduce Discrete Timed ∀-automata where time is discrete. Then, we generalize Discrete Timed ∀-automata to Timed ∀-automata where time can be either discrete or continuous.

3.1 Discrete Timed ∀-automata

Discrete ∀-automata are non-deterministic finite state automata over infinite sequences. These automata were originally proposed as a formalism for the specification and verification of temporal properties of concurrent programs [10]. Formally, a ∀-automaton is defined as follows.

Definition 3.1 (Syntax of ∀-automata) *A* ∀-automaton $\mathcal{A}$ *is a quintuple* $\langle Q, R, S, e, c \rangle$ *where* Q *is a finite set of* automaton-states, $R \subseteq Q$ *is a set of* recurrent states *and* $S \subseteq Q$ *is a set of* stable states. *With each* $q \in Q$, *we associate an assertion* $e(q)$, *which characterizes the* entry condition *under which the automaton may start its activity in* q. *With each pair* $q, q' \in Q$, *we associate an assertion* $c(q, q')$, *which characterizes the* transition condition *under which the automaton may move from* q *to* q'.

R and S are generalizations of *accepting* states to the case of infinite inputs. We denote by $B = Q - (R \cup S)$ the set of *non-accepting (bad)* states.

A ∀-automaton is called *complete* iff the following requirements are met:

- $\bigvee_{q \in Q} e(q)$ is valid.
- For every $q \in Q$, $\bigvee_{q' \in Q} c(q, q')$ is valid.

Any automaton can be transformed to a complete automaton by introducing an additional bad (error) state q_E, with the entry condition: $e(q_E) = \neg(\bigvee_{q \in Q} e(q))$,

and the transition conditions:

$$\begin{aligned} c(q_E, q_E) &= true \\ c(q_E, q) &= false \quad \text{for each } q \in Q \\ c(q, q_E) &= \neg(\bigvee_{q' \in Q} c(q, q')) \quad \text{for each } q \in Q. \end{aligned}$$

One of the advantages of using automata as a specification language is its graphical representation. It is useful and illuminating to represent ∀-automata by diagrams. A ∀-automaton can be depicted by a labeled directed graph where automaton-states are depicted by nodes and transition relations by arcs. The basic conventions for such representations are the following:

- The automaton-states are depicted by nodes in a directed graph.
- Each initial automaton-state ($e(q) \neq false$) is marked by a small arrow, an *entry arc*, pointing to it.
- Arcs, drawn as arrows, connect some pairs of automaton-states.
- Each recurrent state is depicted by a diamond inscribed within a circle.
- Each stable state is depicted by a square inscribed within a circle.

Nodes and arcs are labeled by assertions. The labels define the entry conditions and the transition conditions of the associated automaton as follows.

- Let $q \in Q$ be a node in the diagram corresponding to an initial automaton-state. If q is labeled by ψ and the entry arc is labeled by φ, the entry condition $e(q)$ is given by $e(q) = \varphi \wedge \psi$. If there is no entry arc, $e(q) = false$.
- Let q, q' be two nodes in the diagram corresponding to automaton-states. If q' is labeled by ψ, and arcs from q to q' are labeled by $\varphi_i, i = 1 \cdots n$, the transition condition $c(q, q')$ is given by $c(q, q') = (\varphi_1 \vee \cdots \vee \varphi_n) \wedge \psi$. If there is no arc from q to q', $c(q, q') = false$.

A diagram representing an incomplete automaton can be interpreted as a complete automaton by introducing an error state and associated entry and transition conditions. Some examples of ∀-automata are shown in Fig. 1.

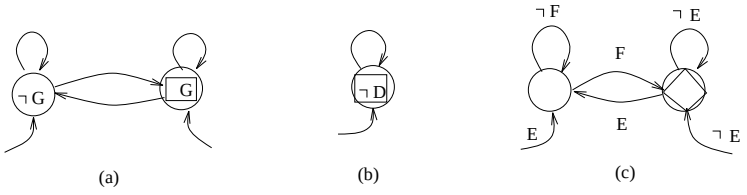

Fig. 1. ∀-automata: (a) reachability (b) safety (c) bounded response

The formal semantics of discrete ∀-automata is defined as follows. Let A be a domain of values. An assertion α on A corresponds to a subset $V(\alpha)$ of A. A value $a \in A$ satisfies an assertion α on A, written $a \models \alpha$ or $\alpha(a)$, iff $a \in V(\alpha)$. Let $\mathcal{T}$ be a discrete time structure and $v : \mathcal{T} \to A$ be a trace. A *run* of $\mathcal{A}$ over v is a mapping $r : \mathcal{T} \to Q$ such that (1) $v(\mathbf{0}) \models e(r(\mathbf{0}))$; and (2) for all $t > \mathbf{0}$, $v(t) \models c(r(pre(t)), r(t))$. A complete automaton guarantees that any discrete trace has a run over it.

If r is a run, let $Inf(r)$ be the set of automaton-states appearing infinitely many times in r, i.e., $Inf(r) = \{q | \forall t \exists t_0 \geq t, r(t_0) = q\}$. Notice that the same definition can be used for continuous as well as discrete time traces. A run r is defined to be *accepting* iff:

1. $Inf(r) \cap R \neq \emptyset$, i.e., *some* of the states appearing infinitely many times in r belong to R, or
2. $Inf(r) \subseteq S$, i.e., *all* the states appearing infinitely many times in r belong to S.

Definition 3.2 (Semantics of ∀-automata) *A ∀-automaton $\mathcal{A}$* accepts *a trace v, written $v \models \mathcal{A}$, iff* all *possible runs of $\mathcal{A}$ over v are accepting.*

For example, Fig. 1(a) accepts any trace that satisfies $\neg G$ only finitely many times, Fig. 1(b) accepts any trace that never satisfies D, and Fig. 1(c) accepts any trace that will satisfy F in the finite future whenever it satisfies E.

In order to represent timeliness, we extend ∀-automata with time. Timed ∀-automata are ∀-automata augmented with timed automaton-states and time bounds. Formally, a timed ∀-automaton is defined as follows.

Definition 3.3 (Syntax of Timed ∀-automata) *A* timed ∀-automaton $\mathcal{TA}$ *is a triple $\langle \mathcal{A}, T, \tau \rangle$ where $\mathcal{A} = \langle Q, R, S, e, c \rangle$ is a ∀-automaton, $T \subseteq Q$ is a set of* timed *automaton-states and $\tau : T \cup \{bad\} \to \mathcal{R}^+ \cup \{\infty\}$ is a* time function.

A ∀-automaton is a special timed ∀-automaton with $T = \emptyset$ and $\tau(bad) = \infty$. Graphically, a T-state is denoted by a nonnegative real number indicating its time bound. The conventions for complete ∀-automata are adopted for timed ∀-automata. Fig. 2 shows an example of a timed ∀-automaton.

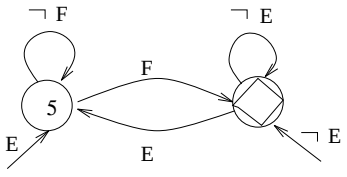

Fig. 2. Real-time response

The formal semantics of Discrete Timed ∀-automata is defined as follows. Let $r : \mathcal{T} \to Q$ be a run and $I \subseteq \mathcal{T}$ be a time interval. For any $P \subset Q$, let $Sg(P)$ be the set of consecutive P-state segments of r, i.e., $r_{|I} \in Sg(P)$ for some interval I iff $\forall t \in I$, $r(t) \in P$. A run r *satisfies the time constraints* iff

1. (local time constraint) for any $q \in T$ and any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(\{q\})$ then $\mu(I) \leq \tau(q)$ and
2. (global time constraint) let $B = Q - (R \cup S)$ and $\chi_B : Q \to \{0, 1\}$ be the characteristic function for set B; for any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(B \cup S)$ then $\int_I \chi_B(r(t))dt \leq \tau(bad)$.

Let $v : \mathcal{T} \to A$ be a trace. A *run* r of $\mathcal{TA}$ over v is a run of $\mathcal{A}$ over v; r is *accepting* for $\mathcal{TA}$ iff

1. r is accepting for $\mathcal{A}$ and
2. r satisfies the time constraints.

Definition 3.4 (Semantics of Timed ∀-automaton) *A timed ∀-automaton* $\mathcal{TA}$ accepts *a trace* v*, written* $v \models \mathcal{TA}$*, iff* all *possible runs of* $\mathcal{TA}$ *over* v *are accepting.*

For example, Fig. 2 specifies a real-time response property meaning that any event (E) will be responded to (F) within 5 time units.

3.2 Timed ∀-automata

Now we generalize Discrete Timed ∀-automata to Timed ∀-automata that can accept general traces, with discrete time traces as special cases. The syntax and semantics of Timed ∀-automata are the same as those of Discrete Timed ∀-automata, except for the definitions of runs.

The important concept of general runs is the generalization of the consecution condition. Let $\mathcal{A} = \langle Q, R, S, e, c \rangle$ be a ∀-automaton and $v : \mathcal{T} \to A$ be a trace. A *run* of $\mathcal{A}$ over v is a mapping $r : \mathcal{T} \to Q$ satisfying

1. *Initiality*: $v(\mathbf{0}) \models e(r(\mathbf{0}))$;
2. *Consecution*:
   - *Inductivity*: $\forall t > \mathbf{0}, \exists q \in Q, t' < t, \forall t'', t' \leq t'' < t, r(t'') = q$ and $v(t) \models c(r(t''), r(t))$ and
   - *Continuity*: $\forall t, \exists q \in Q, t' > t, \forall t'', t < t'' < t', r(t'') = q$ and $v(t'') \models c(r(t), r(t''))$.

When $\mathcal{T}$ is discrete, the two conditions in *Consecution* reduce to one, i.e., $\forall t > \mathbf{0}, v(t) \models c(r(pre(t)), r(t))$ and if, in addition, $\mathcal{A}$ is complete, every trace has a run. However, if $\mathcal{T}$ is not discrete, even if $\mathcal{A}$ is complete, not every trace has a run. For example, a trace with infinite transitions among automaton-states within a finite interval has no run. A trace v is *specifiable* by $\mathcal{A}$ iff there is a run of $\mathcal{A}$ over v. Any discrete trace is specifiable by a complete automaton.

The definitions of accepting runs for ∀-automata and for Timed ∀-automata are the same as those in discrete cases. For example, Fig. 1(a) accepts the traces $x = \lambda t.Ce^{-t}$ for $G \equiv |x| < \epsilon$. Fig. 1(b) accepts the traces $x = \lambda t.\sin(t)$ for $D \equiv |x| > 1$. Fig. 1(c) and Fig. 2 accept the traces $x = \lambda t.\sin(t)$ for $E \equiv x \geq 0$ and $F \equiv x < 0$.

For the *Cat and Mouse* example, a formal requirements specification is shown in Fig 3: indicating that the cat should win.

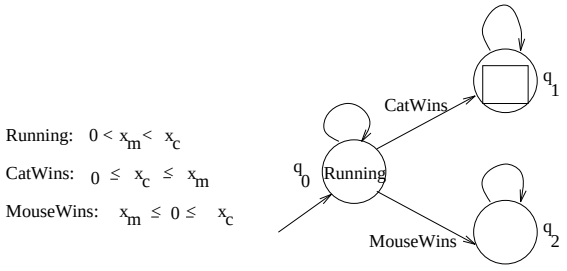

Fig. 3. Requirements specification: the cat should win

The distinguished features of timed ∀-automata are the following:

- Unlike other timed and/or hybrid automata, they are language recognizers rather than language generators. For example, they cannot generate traces like $\lambda t.Ce^{-t}$ and $\lambda t.sin(t)$, but can recognize some qualitative properties of these traces.
- They are finite state but can accept continuous-time hybrid-domain traces. Pre-sampling of the behavior of dynamic systems is not required.

Using timed ∀-automata, qualitative properties such as liveness, reachability and safety of hybrid systems can be formally specified and verified.

3.3 The Power of Timed ∀-automata

It has been shown [10] that Discrete ∀-automata have the same expressive power as Buchi Automata [14] and the Extended Temporal Logic (ETL) [15], which are strictly more powerful than (discrete) Propositional Linear Temporal Logic (PLTL) [14, 15]. Discrete Timed ∀-automata is a non-trivial generalization of Discrete ∀-automata; therefore, Discrete Timed ∀-automata is strictly more powerful than Discrete ∀-automata. However, when time is continuous, ∀-automata is no longer more powerful than PLTL, since the ability of counting in automata [11] is lost when time is dense. We have also developed Timed Linear Temporal Logics (TLTL) [21]. The relationship among Timed ∀-automata, TLTL and other automata and logics is discussed in [16].

4 A Formal Method for Behavior Verification

Let v be a trace and $\mathcal{A}$ be a $\forall$-automaton; we have defined that v is specifiable by $\mathcal{A}$ iff there is a run of $\mathcal{A}$ over v. Let $\mathcal{B}$ be the behavior of a dynamic system; $\mathcal{B}$ is *specifiable* by $\mathcal{A}$ iff $\forall v \in \mathcal{B}$, v is specifiable by $\mathcal{A}$. Let $\mathcal{B}$ be a behavior specifiable by $\mathcal{A}$ and $\mathcal{TA} = \langle \mathcal{A}, T, \tau \rangle$ be a timed $\forall$-automaton; $\mathcal{B}$ satisfies $\mathcal{TA}$, written $\mathcal{B} \models \mathcal{TA}$, iff $\forall v \in \mathcal{B}, v \models \mathcal{TA}$.

In this section, we propose a formal method for behavior verification, given a state-based and time-invariant behavior represented by a generalized Kripke structure and a requirements specification represented by a timed $\forall$-automaton.

Let φ and ψ be assertions on states and time durations. For a generalized Kripke structure $\mathcal{K} = \langle \mathcal{S}, \rightsquigarrow, \Theta \rangle$, let $\{\varphi\}\mathcal{K}\{\psi\}$ denote the validity of the following two consecution conditions:

- *Inductivity* $\{\varphi\}\mathcal{K}^{-}\{\psi\}$: $\exists \delta > 0, \forall 0 < t \leq \delta, \forall s, (\varphi(s) \wedge (s \overset{t}{\rightsquigarrow} s') \Rightarrow \psi(s', t))$.
- *Continuity* $\{\varphi\}\mathcal{K}^{+}\{\psi\}$: $\varphi(s) \Rightarrow \exists \delta > 0, \forall 0 < t < \delta, \forall s', ((s \overset{t}{\rightsquigarrow} s') \Rightarrow \psi(s', t))$.

If $\mathcal{T}$ is discrete, these two conditions reduce to one, i.e., $\varphi(s) \wedge (s \overset{\delta}{\rightsquigarrow} s') \Rightarrow \psi(s', \delta)$ where δ is the minimum time duration between two states.

The formal method for behavior verification consists of a set of model-checking rules, which is a generalization of the model-checking rules developed for concurrent programs [10].

There are three types of rules: invariance rules (I), stability or eventuality rules (L) and timeliness rules (T). Let $\mathcal{A}$ be a $\forall$-automaton $\langle Q, R, S, e, c \rangle$ and $\mathcal{K}$ be a generalized Kripke structure $\langle \mathcal{S}, \rightsquigarrow, \Theta \rangle$. The invariance rules check to see if a set of assertions $\{\alpha\}_{q \in Q}$ is a set of invariants for $\mathcal{A}$ and $\mathcal{K}$, i.e., for any trace v of $\mathcal{K}$ and any run r of $\mathcal{A}$ over v, $\forall t \in \mathcal{T}, v(t) \models \alpha_{r(t)}$. Given $B = Q - (R \cup S)$, the stability or eventuality rules check if the B-states in any run of $\mathcal{A}$ over any trace of $\mathcal{K}$ will be terminated eventually. Given $\mathcal{TA}$ as a timed $\forall$-automaton $\langle \mathcal{A}, T, \tau \rangle$, the timeliness rules check if the T-states and the B-states in any run of $\mathcal{A}$ over any trace of $\mathcal{K}$ are bounded by the time function τ. The set of model-checking rules can be represented in first-order logic, some of which are in the form of $\{\varphi\}\mathcal{K}\{\psi\}$.

Here are the model-checking rules for a behavior represented by $\mathcal{K} = \langle \mathcal{S}, \rightsquigarrow, \Theta \rangle$ and a specification represented by $\mathcal{TA} = \langle \mathcal{A}, T, \tau \rangle$ where $\mathcal{A} = \langle Q, R, S, e, c \rangle$:

Invariance Rules (I): A set of assertions $\{\alpha_q\}_{q \in Q}$ is called a set of *invariants* for $\mathcal{K}$ and $\mathcal{A}$ iff

(I1) *Initiality*: $\forall q \in Q, \Theta \wedge e(q) \Rightarrow \alpha_q$.
(I2) *Consecution*: $\forall q, q' \in Q, \{\alpha_q\}\mathcal{K}\{c(q, q') \Rightarrow \alpha_{q'}\}$.

The Invariance Rules are the same as those in [10] except that the condition for consecution is generalized.

Stability or Eventuality Rules (L): Given that $\{\alpha_q\}_{q \in Q}$ is a set of invariants for $\mathcal{K}$ and $\mathcal{A}$, a set of partial functions $\{\rho_q\}_{q \in Q} : \mathcal{S} \rightarrow \mathcal{R}^{+}$ is called a set of *Liapunov functions* for $\mathcal{K}$ and $\mathcal{A}$ iff the following conditions are satisfied:

(L1) *Definedness*: $\forall q \in Q,\ \alpha_q \Rightarrow \exists w, \rho_q = w$.

(L2) *Non-increase*: $\forall q \in S, q' \in Q,\ \{\alpha_q \wedge \rho_q = w\}\mathcal{K}^-\{c(q,q') \Rightarrow \rho_{q'} \leq w\}$ and $\forall q \in Q, q' \in S,\ \{\alpha_q \wedge \rho_q = w\}\mathcal{K}^+\{c(q,q') \Rightarrow \rho_{q'} \leq w\}$.

(L3) *Decrease*: $\exists \epsilon > 0, \forall q \in B, q' \in Q,\ \{\alpha_q \wedge \rho_q = w\}\mathcal{K}^-\{c(q,q') \Rightarrow \frac{\rho_{q'} - w}{t} \leq -\epsilon\}$ and $\forall q \in Q, q' \in B,\ \{\alpha_q \wedge \rho_q = w\}\mathcal{K}^+\{c(q,q') \Rightarrow \frac{\rho_{q'} - w}{t} \leq -\epsilon\}$.

The Stability or Eventuality Rules generalize both stability analysis of discrete or continuous dynamic systems [8] and well-foundedness for finite termination in concurrent systems [10].

Timeliness Rules (T): Corresponding to two types of time bound, we define two timing functions. Let $\{\alpha_q\}_{q\in Q}$ be invariants for $\mathcal{K}$ and $\mathcal{A}$. A set of partial functions $\{\gamma_q\}_{q\in T}$ is called a set of *local timing functions* for $\mathcal{K}$ and $\mathcal{TA}$ iff $\gamma_q : \mathcal{S} \to \mathcal{R}^+$ satisfies the following conditions:

(T1) *Boundedness*: $\forall q \in T,\ \alpha_q \Rightarrow \gamma_q \leq \tau(q)$ and $\forall q \in T, q' \in Q,\ \{\alpha_q \wedge \gamma_q = w\}\mathcal{K}^-\{c(q,q') \Rightarrow w \geq t\}$.

(T2) *Decrease*: $\forall q \in T, \{\alpha_q \wedge \gamma_q = w\}\mathcal{K}\{c(q,q) \Rightarrow \frac{\gamma_q - w}{t} \leq -1\}$.

A set of partial functions $\{\eta_q\}_{q\in Q}$ is called a set of *global timing functions* for $\mathcal{K}$ and $\mathcal{TA}$ iff $\eta_q : \mathcal{S} \to \mathcal{R}^+$ satisfies the following conditions:

(T3) *Definedness*: $\forall q \in Q, \alpha_q \Rightarrow \exists w, \eta_q = w$.

(T4) *Boundedness*: $\forall q \in B, \alpha_q \Rightarrow \eta_q \leq \tau(bad)$.

(T5) *Non-increase*: $\forall q \in S, q' \in Q,\ \{\alpha_q \wedge \eta_q = w\}\mathcal{K}^-\{c(q,q') \Rightarrow \eta_{q'} \leq w\}$ and $\forall q \in Q, q' \in S,\ \{\alpha_q \wedge \eta_q = w\}\mathcal{K}^+\{c(q,q') \Rightarrow \eta_{q'} \leq w\}$.

(T6) *Decrease*: $\forall q \in B, q' \in Q,\ \{\alpha_q \wedge \eta_q = w\}\mathcal{K}^-\{c(q,q') \Rightarrow \frac{\eta_{q'} - w}{t} \leq -1\}$ and $\forall q \in Q, q' \in B,\ \{\alpha_q \wedge \eta_q = w\}\mathcal{K}^+\{c(q,q') \Rightarrow \frac{\eta_{q'} - w}{t} \leq -1\}$.

The Timeliness Rules are modifications of the Eventuality Rules; they enforce real-time boundedness, in addition to termination.

A set of model-checking rules is *sound* if verification by the rules guarantees the correctness of the behavior against the specification; it is *complete* if the correctness of the behavior against the specification guarantees verification by the rules.

Theorem 4.1 *The set of model-checking rules (I), (L) and (T) is sound given that the behavior of $\mathcal{K}$ is specifiable by $\mathcal{A}$.*

Theorem 4.2 *The set of model-checking rules (I), (L) and (T) is complete given that time is discrete.*

These theorems are proved in Appendix A. The general condition for the completeness of the rules has been described elsewhere [16].

We illustrate this verification method with the *Cat and Mouse* example. We show that the behavior of the cat and mouse in Section 2 satisfies the requirements specification in Fig. 3, given that the constant $\Delta = \frac{X_c}{V_c} - \frac{X_m}{V_m}$ satisfies $\Delta > 0$. The generalized Kripke structure $\mathcal{K}$ for the system can be derived from the constraint net equations Eq. 1: i.e., $\langle \mathcal{R} \times \mathcal{R}, \rightsquigarrow, \Theta \rangle$ where $\langle x_c, x_m \rangle \overset{t}{\rightsquigarrow} \langle x'_c, x'_m \rangle$ if

- $x_c > x_m > 0$, $x'_c > x'_m > 0$, $x'_c = x_c + V_c t$, $x'_c = x_c + V_c t$; or
- $x_c > x_m > 0$, $x'_c = x'_m = \frac{x_c V_m - x_m V_c}{V_m - V_c} \geq 0$, $t = \frac{x_c - x_m}{V_m - V_c}$; or
- $x_c > x_m > 0$, $x'_m = 0$, $x'_c = \frac{x_c V_m - x_m V_c}{V_m} \geq 0$, $t = -\frac{x_m}{V_m}$; or
- $\neg(x_c > x_m > 0)$, $x'_c = x_c$, $x'_m = x_m$.

and $\langle X_c, X_m \rangle \in \Theta$ with $\frac{X_c}{V_c} > \frac{X_m}{V_m}$.

Let Inv denote the invariant assertion $\frac{x_c}{V_c} - \frac{x_m}{V_m} = \Delta$. Associate with automaton-states q_0, q_1 and q_2 (Fig. 3) the assertions $Running \wedge Inv$, $CatWins$ and $false$, respectively. Note that

$$\{Running \wedge Inv\}\mathcal{K}\{Running \Rightarrow Running \wedge Inv\}$$

since the derivative of $\frac{x_c}{V_c} - \frac{x_m}{V_m}$ is 0 given that $Running$ is satisfied, and

$$\{Running \wedge Inv\}\mathcal{K}\{MouseWins \Rightarrow false\}$$

since $MouseWins$ implies $\frac{x_c}{V_c} - \frac{x_m}{V_m} \leq 0$. Therefore, the set of assertions is a set of invariants.

Associate with q_0, q_1 and q_2 the same function $\rho : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}^+$, such that $\rho(x_c, x_m) = 0$ if not $Running$ and $\rho(x_c, x_m) = -(\frac{x_m}{V_m} + \frac{x_c}{V_c})$ if $Running$. Function ρ is decreasing at q_0 with rate 2. Therefore, it is a Liapunov function.

Furthermore, the behavior of the cat and mouse is specifiable by the automaton in Fig. 3. According to the soundness of the rules, the behavior satisfies the required property.

A model-checking algorithm can be deduced from the set of rules (I), (L) and (T) for discrete-time finite-domain systems. The algorithm has polynomial time complexity with respect to both the size of the system and the size of the specification [16].

5 Conclusion and Further Research

We have presented in this paper Timed ∀-automata for the specification and verification of dynamic systems. To our knowledge, Timed ∀-automata are the first proposal for recognizing or representing timed dynamic behaviors, such as safety, reachability, liveness and real-time response, of continuous as well as discrete time dynamic systems using a finite number of states. In addition, we have proposed a formal model-checking method for behavior verification of dynamic systems. This method generalizes stability analysis of dynamic systems and can be completely automated for discrete-time finite-domain systems.

Much related work has been done in the last few years. The Timed Buchi Automata (TBA) model has been proposed [2] to express constant bounds on timing delays between system events. Other developments along this line include Timed Transition Systems [5] and Time Petri Nets [3]. Hybrid Automata [1] can be viewed as a generalization of TBA, in which the behavior of variables is governed in each state by a set of differential equations. Similar work includes Hybrid Statecharts and Phase Transition Systems [9].

The major difference between our work and others is that we use Constraint Nets as language generators and Timed $\forall$-automata as language recognizers. Timed $\forall$-automata can capture qualitative properties of traces, but they are not "fine" enough to distinguish all the different continuous traces. In the near future, we intend to explore semi-automatic and automatic verification procedures for various classes of hybrid dynamic systems.

Acknowledgements

This research is supported by Wilson Center for Research and Technology of Xerox Corporation, Natural Sciences and Engineering Research Council in Canada and the Institute for Robotics and Intelligent Systems.

References

1. R. Alur, C. Courcoubetis, T. A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, number 736 in Lecture Notes on Computer Science, pages 209 – 229. Springer-Verlag, 1993.
2. R. Alur and D. Dill. Automata for modeling real-time systems. In M. S. Paterson, editor, *ICALP90: Automata, Languages and Programming*, number 443 in Lecture Notes on Computer Science, pages 322 – 335. Springer-Verlag, 1990.
3. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Transactions on Software Engineering*, 17(3):259 – 273, March 1991.
4. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*. Number 736 in Lecture Notes on Computer Science. Springer-Verlag, 1993.
5. T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 226–251. Springer-Verlag, 1991.
6. G. F. Khilmi. *Qualitative Methods in the Many Body Problem*. Science Publishers Inc. New York, 1961.
7. N. G. Leveson and P. G. Neumann, editors. *IEEE Transactions on Software Engineering*. IEEE Computer Society, January 1993. Special Issue on Software for Critical Systems.
8. D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models and Applications*. John Wiley & Sons, 1979.
9. O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 448 – 484. Springer-Verlag, 1991.
10. Z. Manna and A. Pnueli. Specification and verification of concurrent programs by $\forall$-automata. In *Proc. 14th Ann. ACM Symp. on Principles of Programming Languages*, pages 1–12, 1987.
11. R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.
12. H. L. Royden. *Real Analysis, 3rd edition*. Macmillan Publishing Company, 1988.

13. M. Sahota and A. K. Mackworth. Can situated robots play soccer? In *Proc. Artificial Intelligence 94*, pages 249 – 254, Banff, Alberta, May 1994.
14. W. Thomas. Automata on infinite objects. In Jan Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. MIT Press, 1990.
15. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72 – 99, 1983.
16. Y. Zhang. A foundation for the design and analysis of robotic systems and behaviors. Technical Report 94-26, Department of Computer Science, University of British Columbia, 1994. Ph.D. thesis.
17. Y. Zhang and A. K. Mackworth. Specification and verification of constraint-based dynamic systems. In A. Borning, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science 874, pages 229 – 242. Springer Verlag, 1994.
18. Y. Zhang and A. K. Mackworth. Will the robot do the right thing? In *Proc. Artificial Intelligence 94*, pages 255 – 262, Banff, Alberta, May 1994.
19. Y. Zhang and A. K. Mackworth. Constraint Nets: A semantic model for hybrid dynamic systems. *Theoretical Computer Science*, 138(1):211 – 239, 1995. Special Issue on Hybrid Systems.
20. Y. Zhang and A. K. Mackworth. Constraint programming in constraint nets. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming*, pages 49 – 68. MIT Press, 1995.
21. Y. Zhang and A. K. Mackworth. Synthesis of hybrid constraint-based controllers. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 552 – 567. Springer Verlag, 1995.

A Proofs of Theorems and Lemmas

Lemma 2.1 *If $\mathcal{T}$ is a discrete time structure of $\mathcal{K}$, then there is $\delta > 0$, for any $t > \mathbf{0}, \mu([pre(t), t)) = \delta$. And $\mathcal{K}$ can be represented as the transitive closure of transitions of next relation $s_1 \overset{\delta}{\rightsquigarrow} s_2$.*
Proof: Assume $\mu([pre(t), t)) = \delta$ and $\mu([pre(t'), t')) = \delta'$, such that $\delta < \delta'$. For time point $pre(t')$ and $s_1 \overset{\delta}{\rightsquigarrow} s_2$, there is no time t'' such that $\mu([pre(t'), t'''))) = \delta$. Therefore, $\delta = \delta'$. □

Theorem 4.1 *The set of model-checking rules (I), (L) and (T) is sound given that the behavior of $\mathcal{K}$ is specifiable by $\mathcal{A}$.*

In order to prove this theorem, we shall introduce a method of continuous induction modified from [6]. A property Γ is *inductive* on a time structure $\mathcal{T}$ iff Γ is satisfied at all $t < t_0 \in \mathcal{T}$ implies that Γ is satisfied at t_0, for all $t_0 \in \mathcal{T}$. Γ is *continuous* iff Γ is satisfied at $t_0 \in \mathcal{T}$ implies that $\exists t_1 > t, \forall t, t_0 < t < t_1$, Γ is satisfied at t. We should notice that when $\mathcal{T}$ is discrete, any property is trivially continuous. The theorem of continuous induction says:

Theorem A.1 *If a property Γ is inductive and continuous on a time structure $\mathcal{T}$ and Γ is satisfied at $\mathbf{0}$, Γ is satisfied at all $t \in \mathcal{T}$.*

Proof: We call a time point $t \in \mathcal{T}$ *regular* iff Γ is satisfied at all t', $\mathbf{0} \leq t' \leq t$. Let T denote the set of all regular time points. T is not empty since Γ is satisfied at $\mathbf{0}$. We prove the theorem by contradiction, i.e., assume that Γ is not satisfied at all $t \in \mathcal{T}$. Therefore, $T \subset \mathcal{T}$ is bounded above; let $t_0 = \bigvee T \in \mathcal{T}$ be the least upper bound of T ($\mathcal{T}$ is complete). Since t_0 is the least upper bound, it follows that Γ is satisfied at all t, $\mathbf{0} \leq t < t_0$. Since Γ is inductive, it is satisfied at time t_0. Therefore, $t_0 \in T$.

Since $T \subset \mathcal{T}$, t_0 is not the greatest element in $\mathcal{T}$. Let $T' = \{t | t > t_0\}$. There are two cases: (1) if T' has a least element t', since Γ is inductive, $t' \in T$ is a regular time point. (2) otherwise, for any $t' \in T'$, $\{t | t_0 < t < t'\} \neq \emptyset$. Since Γ is also continuous, we can find a $t' \in T'$ such that Γ is satisfied at all $T'' = \{t | t_0 < t < t'\}$. Therefore, t is a regular time point $\forall t \in T''$. Both cases contradict the fact that t_0 is the least upper bound of the set T. □

Using the method of continuous induction, we obtain the following three lemmas.

Lemma A.1 *Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{K}$ and $\mathcal{A}$. If r is a run of $\mathcal{A}$ over a trace v of $\mathcal{K}$, $\forall t \in \mathcal{T}, v(t) \models \alpha_{r(t)}$.*

Proof: We prove that the property $v(t) \models \alpha_{r(t)}$ is satisfied at $\mathbf{0}$ and is both inductive and continuous on any time structure $\mathcal{T}$.

- Initiality: Since $v(\mathbf{0}) \models \Theta$ and $v(\mathbf{0}) \models e(r(\mathbf{0}))$, we have $v(\mathbf{0}) \models \Theta \wedge e(r(\mathbf{0}))$. According to the *Initiality* condition of invariants, we have $v(\mathbf{0}) \models \alpha_{r(\mathbf{0})}$.
- Inductivity: Suppose $v(t) \models \alpha_{r(t)}$ is saisfied at $\mathbf{0} \leq t < t_0$. Since r is a run over v, according to the *Inductivity* of runs, $\exists q \in Q$ and $\exists t_1' < t_0, \forall t_1' \leq t < t_0$, $r(t) = q$ and $v(t_0) \models c(q, r(t_0))$. According to the *Inductivity* condition of the invariants, $\exists t_2' < t_0, \forall t_2' \leq t < t_0$, $v(t) \models \alpha_q$ implies $v(t_0) \models c(q, r(t_0)) \Rightarrow \alpha_{r(t_0)}$. Therefore, let $t' = \max(t_1', t_2'), \forall t' \leq t < t_0$, $r(t) = q$, $v(t) \models \alpha_q$ (assumption), $v(t_0) \models c(q, r(t_0)) \Rightarrow \alpha_{r(t_0)}$ and $v(t_0) \models c(q, r(t_0))$. Thus, $v(t_0) \models \alpha_{r(t_0)}$.
- Continuity: Suppose $v(t_0) \models \alpha_{r(t_0)}$. Since r is a run over v, according to the *Continuity* of runs, $\exists q \in Q$ and $\exists t_1' > t_0, \forall t_0 < t < t_1'$, $r(t) = q$ and $v(t) \models c(r(t_0), q)$. According to the *Continuity* condition of the invariants, $\exists t_2' > t_0, \forall t_0 < t < t_2'$, $v(t) \models c(r(t_0), q) \Rightarrow \alpha_q$. Therefore, let $t' = \min(t_1', t_2')$, $\forall t_0 < t < t'$, $r(t) = q$, $v(t_0) \models \alpha_{r(t_0)}$ (assumption), $v(t) \models c(r(t_0), q) \Rightarrow \alpha_q$ and $v(t) \models c(r(t_0), q)$. Thus, $\forall t_0 < t < t'$, $v(t) \models \alpha_{r(t)}$.

□

Lemma A.2 *Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{K}$ and $\mathcal{A}$ and r be a run of $\mathcal{A}$ over a trace v of $\mathcal{K}$. If $\{\rho_q\}_{q \in Q}$ is a set of Liapunov functions for $\mathcal{K}$ and $\mathcal{A}$, then*

- $\rho_{r(t_2)}(v(t_2)) \leq \rho_{r(t_1)}(v(t_1))$ *when* $\forall t_1 \leq t \leq t_2, r(t) \in B \cup S$,
- $\frac{\rho_{r(t_2)}(v(t_2)) - \rho_{r(t_1)}(v(t_1))}{\mu([t_1, t_2))} \leq -\epsilon$ *when* $t_1 < t_2$ *and* $\forall t_1 \leq t \leq t_2, r(t) \in B$, *and*
- *for any run r and any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(B \cup S)$, $\int_I \chi_B(r(t))dt < \infty$.*

Proof: For any run r over v and for any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(B \cup S)$, ρ on I is nonincreasing, i.e., for any $t_1 < t_2 \in I$, $\rho_{r(t_1)}(v(t_1)) \geq \rho_{r(t_2)}(v(t_2))$, and the decreasing speed at intervals of the bad states is no less than ϵ. Let m be the upper bound of $\{\rho_{r(t)}(v(t))|t \in I\}$. Since $\rho_q \geq 0$, $\int_I \chi_B(r(t))dt \leq m/\epsilon < \infty$. □

Lemma A.3 *Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{K}$ and $\mathcal{A}$ and r be a run of $\mathcal{A}$ over a trace v of $\mathcal{K}$. If there exist local and global timing functions for $\mathcal{K}$ and $\mathcal{TA}$, then*

- *for any $q \in T$, any run r and any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(\{q\})$, $\mu(I) \leq \tau(q)$ and*
- *for any run r and any interval I of $\mathcal{T}$, if $r_{|I} \in Sg(B \cup S)$, $\int_I \chi_B(r(t))dt \leq \tau(bad)$.*

Proof: Similar to the proof of Lemma A.2. □

Proof of Theorem 4.1: For any trace v of $\mathcal{K}$, there is a run since v is specifiable by $\mathcal{A}$. For any run r of $\mathcal{A}$ over v, if any automaton-state in R appears infinitely many times in r, r is accepting for $\mathcal{A}$. Otherwise there is a time point t_0, such that the sub-sequence r on $I = \{t \in \mathcal{T} | t \geq t_0\}$ has only bad and stable automaton-states. If there exist a set of invariants and a set of Liapunov functions, $\int_I \chi_B(r(t))dt$ is finite. Therefore, all the automaton-states appearing infinitely many times in r belong to S; r is accepting for $\mathcal{A}$ too. If there exists a set of local and global timing functions, r satisfies the time constraints; r is accepting for $\mathcal{TA}$. Therefore, the behavior of $\mathcal{K}$ satisfies the specification $\mathcal{TA}$. □

Theorem 4.2 *The set of model-checking rules (I), (L) and (T) is complete given that time is discrete.*
Proof: If $\mathcal{TA}$ is valid over $\mathcal{K}$, then there exist a set of invariants, a set of Liapunov functions, and a set of local and global timing functions that satisfy the requirements.

Let $n(s, s')$ denote $s \overset{\delta}{\rightsquigarrow} s'$ where δ is the minimum time duration between two states. The invariants can be constructed as the fixpoint of the set of equations:

$$\alpha_{q'}(s') = (\exists q, s, \alpha_q(s) \wedge n(s, s') \wedge c(q, q')(s')) \bigvee (\Theta(s') \wedge e(q')(s')).$$

We can verify that $\{\alpha_q\}_{q \in Q}$ is a set of assertions on the states of $\mathcal{K}$ and satisfies the requirements of initiality and consecution. Furthermore, $s \models \alpha_q$ iff $\langle q, s \rangle$ is a reachable pair for $\mathcal{A}$ and $\mathcal{K}$.

Given the constructed invariants $\{\alpha_q\}_{q \in Q}$, a set of Liapunov functions $\{\rho_q\}_{q \in Q}$ and a set of global timing functions $\{\eta_q\}_{q \in Q}$ can be constructed as follows:

- $\forall q \in R, s \models \alpha_q$, let $\rho_q(s) = 0$ and $\eta_q(s) = 0$.

– $\forall q \notin R, s \models \alpha_q$, $\rho_q(s)$ and $\eta_q(s)$ are defined as follows. Construct a directed graph $G = \langle V, E \rangle$, such that $\langle q, s \rangle \in V$ iff $q \notin R, s \models \alpha_q$, and $\langle q, s \rangle \mapsto \langle q', s' \rangle$ in E iff $n(s, s') \wedge c(q, q')(s')$. For any path p starting at $\langle q, s \rangle$, let $|p|_B$ be the number of B-states in p. Let $\rho_q(s) = \sup\{|p|_B\}$ and $\eta_q(s) = \delta\rho_q(s)$.

We can verify that $\{\rho_q\}_{q \in Q}$ is a set of Liapunov functions, and that $\{\eta_q\}_{q \in Q}$ is a set of global timing functions.

Similar construction can be carried out for local timing functions. A set of local timing functions $\{\gamma_q\}_{q \in T}$ can be constructed as follows. For all $q \in T$, construct a directed graph $G = \langle V, E \rangle$, such that $s \in V$ iff $s \models \alpha_q$, and $s \mapsto s'$ in E iff $n(s, s') \wedge c(q, q)(s')$. For any path p starting at s, let $|p|$ be the number of states in the path. Let $\gamma_q(s) = \delta \sup\{|p|\}$. We can verify that $\{\gamma_q\}_{q \in T}$ is a set of local timing functions. □

This proof is the basis of the verification algorithm for discrete systems [16].

This article was processed using the LaTeX macro package with LLNCS style