CONTROLLER SYNTHESIS FOR TIMED AUTOMATA [1]

Eugene Asarin * **Oded Maler** ** **Amir Pnueli** ***
Joseph Sifakis **

* *Institute for Information Transmission Problems, 19 Bol. Karetnyi per. 101447 Moscow, Russia.* asarin@ippi.ras.ru
** VERIMAG, *2, av. de Vignate, 38610, Gières, France.* {Oded.Maler,Joseph.Sifakis}@imag.fr
*** *Department of Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel.* amir@wisdom.weizmann.ac.il

Abstract: In this work we tackle the following problem: given a timed automaton, restrict its transition relation in a systematic way so that all the remaining behaviors satisfy certain properties. This is an extension of the problem of controller synthesis for discrete event dynamical systems, where in addition to choosing among actions, the controller have the option of doing nothing and let the time pass. The problem is formulated using the notion of a real-time game, and a winning strategy is constructed as a fixed-point of an operator on the space of states and clock configurations.

1. INTRODUCTION

The problem of synthesizing controllers for discrete event systems has been studied extensively, under different titles, both by the computer science (e.g. [BL69], [TB73], [PR89], and the control (e.g. [RW87], [KG95]) communities. In this paper we extend the basic synthesis algorithm to treat systems with *quantitative* timing information, modeled using the powerful model of *Timed Automata* [AD94]. In the rest of this section we give a short tutorial to the game-theoretic formulation of the synthesis problem. In section 2 we define formally and solve the problem for discrete systems. In essence, this is just a state-based (rather than language-based) reformulation of the Ramadge-Wonham theory. Section 3 is devoted to introducing the model of *Timed Game Automaton*, the formulation of the synthesis problem, the solution and a proof of its correctness. Finally we discuss some implications of the results. Preliminary work on this topic has been reported by the authors in [MPS95], and in [AMP95] where more extensive introduction and bibliography appear.

1.1 *Game-Theoretic view of Controller Synthesis*

The interaction between a controller and the plant it is supposed to supervise can be seen as some variant of the two-person antagonistic games introduced already by von-Neumann and Morgenstern [NM44]. A *strategy* for a given game is a rule that tells the controller how to choose between several possible actions in any game position. A strategy is winning if the controller, by following these rules, always wins (according to a given definition of winning) no matter what the environment does.

Strategy extraction for finite-state games is done using the max-min principle of [NM44], disguised sometimes as searching AND-OR trees or as the elimination of an alternating pair of the logical quantifiers $\exists$ and $\forall$. This principle is illustrated using the game depicted at figure 1. The game starts from position 0. The controller can choose between actions a_1 and a_2 while the environment can choose between b_1 and b_2. The winning condition is specified via some subset F of $\{1,2,3,4\}$. A run of the game is winning if it ends up in an element of F. Suppose $F = \{1,4\}$ – in this case the controller has no winning strategy at state 0 because if it chooses a_1, the adversary can take b_2 and reach state 2. If it chooses a_2 the adversary can reach state 3 by taking b_1. Hence, 0 is not a winning position. If, however, we consider a game with the same transition structure but with $F = \{1,2\}$ then there is a winning strategy as the controller can, by making a_1, "force" the environment into F.

[1] This research was supported in part by the European Community projects HYBRID EC-US-043, INTAS-94-697, VHS-26270 and by Research Grants 97-01-00692 and 96-15-96048 of Russian Foundation of Basic Research.

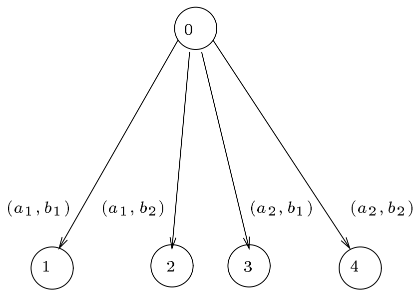

Fig. 1. A simple game.

Trivial as it might seem, this is the *essence* of any synthesis algorithm, a fact which is sometimes obscured by fancy technicalities. The mathematical formulation of this notion for a game with a state-space Q is via an operator $\pi : 2^Q \rightarrow 2^Q$ assigning for every $F \subseteq Q$ the set $\pi(F)$ denoting its *controllable predecessors*, that is, the set of states from which the controller can *force* its adversary into F. In the example above $\{0\} \notin \pi(\{1,4\})$ and $\{0\} \in \pi(\{1,2\})$. Calculating this operator, together with some set-theoretical operations constitutes the core of any synthesis algorithm. After formalizing and solving the discrete problem in the next section, we will turn to the timed case where the state-space contains continuous clocks, and see how the passage of time adds to the problem complexity.

2. UNTIMED SYSTEMS

We assume two players A (controller) and B (environment) who play on a state-space Q. At every instant of the game, each player chooses one admissible action and the game progresses according to the mutual choice of the two players. A strategy for A is a restriction of its set of admissible actions at a given state in order that all the remaining runs of the system meet certain criteria.

1.1 *Game Automata*

Let Q be a finite set of *states* and A, B two finite sets of actions.

Definition 1. (Game Automaton).
A game automaton (GA) is a tuple $\mathcal{A} = (Q, A, B, T^A, T^B, \delta)$ where $T^A \subseteq Q \times A$ and $T^B \subseteq Q \times B$ are enabling conditions for the two types of actions, and the transition function $\delta : Q \times A \times B \rightarrow Q$ indicates which state is reached when performing a joint action.

We always assume that $\delta(q,a,b)$ is defined whenever $(q,a) \in T^A$ and $(q,b) \in T^B$.

Definition 2. (Steps and Runs). A joint *step* of $\mathcal{A}$ is

$$q \xrightarrow{(a,b)} q'$$

$(q,a) \in T^A$ and $(q,b) \in T^B$ such that $q' = \delta(q,a,b)$. A *run* of $\mathcal{A}$ is a sequence (finite or infinite) of joint steps of the form:

$$\xi = q_0 \xrightarrow{(a_1,b_1)} q_1 \xrightarrow{(a_2,b_2)} q_2 \ldots$$

We denote by $L(\mathcal{A}, P)$ the set of all runs starting from some $q \in P \subseteq Q$. The set of states reachable by a run ξ is denoted by $Reach(\xi)$ and the set of states reachable from P by some run in $L(\mathcal{A}, P)$ is denoted by $Reach(\mathcal{A}, P)$.

An automaton is *non-blocking* if for every $q \in Q$ there are $a \in A$ and $b \in B$ such that $(q,a) \in T^A$ and $(q,b) \in T^B$. In a non-blocking automaton every finite run can be extended to an infinite one. Given some $T^A \subseteq Q \times A$, we denote by $S(T^A)$ the set of states on which T^A is defined. The restriction of T^A to some $Q' \subseteq Q$ is denoted by $T^A{}_{|Q'} = T^A \cap \{(q,a) : q \in Q'\}$

The GA defines the "arena" in which the two players play. What is cosidered a winning run is defined by different *acceptance condition*. Such conditions classify a run as good or bad according to the number of times certain states are visited. A strategy for A is a restriction of T^A such that all the remaining runs are accepting. In this abstract, like in [AMP95], we focus on the most basic condition of *safety* which essentially means "avoid forbidden states". In [MPS95] and in a forthcoming paper, additional acceptance criteria are discussed along with their corresponding synthesis algorithms.

1.2 *Safety Games*

In a safety game, the goal of player A is to keep the game inside a subset G of Q. The *winning states* of the game are thus the states from which A can, by properly chosing its actions, prevent the game from going outside G. A straightforward solution would be to restrict the automaton to G and remove for every $q \in G$ all the a choices which might lead outside G. However, this procedure might lead to a blocking situation and hence an iterative procedure is needed in order to assure that the winning states are indeed the state where the automaton can proceed indefinitely without leaving G.

Definition 3. (Controller Synthesis for $\Box$-Games). Given a GA $\mathcal{A} = (Q, A, B, T^A, T^B, \delta)$ and a set $G \subseteq Q$, the controller synthesis problem $Synth(\mathcal{A}, G, \Box)$ is: find the maximal subset Q_*, $Q_* \subseteq G \subseteq Q$, and the maximal T^A_*, $T^A_* \subseteq T^A{}_{|G} \subseteq T^A$, such that the automaton $\mathcal{A}_* = (Q_*, A, B, T^A_*, T^B{}_{|Q_*}, \delta)$, is non-blocking and $Reach(\mathcal{A}_*, Q_*) \subseteq G$. (The latter fact is implied by $Q_* \subseteq G$.)

The relation T^A_* is called a *winning strategy* for A and the set Q_* is the set of corresponding *winning states*. Usually Q_* will be smaller than G as states from which leaving G is unavoidable are removed. Also T^A_* will be smaller than $T^A{}_{|Q_*}$ because transitions that can lead to bad consequences are removed. The calculation of $\mathcal{A}_*$ from $\mathcal{A}$ and G is performed by an iterative process which removes bad transitions (and states) from T^A, using the following operator:

Definition 4. (Controllable Predecessor). Let $\mathcal{A}$ be a GA. The operator $\pi : 2^Q \to 2^{Q\times A}$ is defined as

$$\pi(P) = \{(q,a) : \forall b \in B \ \ (q,b) \in T^B \ \Rightarrow \ q \stackrel{(a,b)}{\longrightarrow} P\}$$

In other words, $(q,a) \in \pi(P)$ iff by doing a at q, A forces the game into P no matter what B does. The algorithm for calculating T^A_* proceeds by calculating a sequence $\mathcal{A}[i]$ of automata, $\mathcal{A}[i] = (Q[i], A, B, T^A[i], T^B, \delta)$ as follows:

Algorithm 1. (Winning Strategy for $\Box$-Games).

$T^A[0] := T^A{}_{|G}$
$Q[0] := G$
for $i = 1, 2, \ldots,$ **repeat**
  $T^A[i] := T^A[i-1] \ \cap \ \pi(Q[i-1])$
  $Q[i] := S(T^A[i])$
until $Q[i] = Q[i-1]$
$T^A_* := T^A[i]$
$Q_* := Q[i]$

Claim 5. (Properties of Algorithm 1). For every i: 1) The automaton $\mathcal{A}[i]$ is well-defined and non-blocking and $Q[i] \subseteq G$. 2) If $(q,a) \notin T^A[i]$, then whatever A does after chosing a at q, B can drive the automaton outside G within at most i steps.

Proof: By induction: the base case follows trivially from the initialization of the algorithm. Suppose it is true for $i-1$. Then, by definition we have: 1) if $q \in Q[i]$ then there exists at least one a such that $(q,a) \in T^A[i]$, and 2) from $\neg Q[i]$ there is an unavoidable step leading to $\neg Q[i-1]$ and hence, by the induction hypothesis: 1) all runs can be extended to length i, and 2) from every state in $\neg Q[i+1]$ there is an unavoidable run of $\mathcal{A}$ of length not greater than i leaving G. ∎

Since 2^Q is a finite set and the sequences $Q[i]$ and $T^A[i]$ are monotone decreasing, the algorithm converges after a finite number of steps to a fixed-point $T^A_* = \pi(Q_*)$.

Corollary 6. (Correctness of Algorithm 1). Let $\mathcal{A}_*$ be the result of algorithm 1. Then: 1) $\mathcal{A}_*$ is non-blocking and $Reach(\mathcal{A}_*, Q_*) \subseteq G$. 2) Any other subset of T^A satisfying (1) is included in T^A_*.

3. TIMED SYSTEM

3.1 *Real-Time Games*

In real-time games the outcome of the players' actions depend also on their timing because performing the same action "now" or "later" might have completely different consequences. For such games we take the model of *timed automata* [AD94], in which automata are equipped with auxiliary continuous variables called *clocks* which grow uniformly when the automaton is in some state. The clocks interact with the transitions by participating in pre-conditions (guards) for certain transitions and they are possibly reset when some transitions are taken.

In this continuous-time setting, a player might choose at a given moment to wait some time t and *then* take a transition. In this case, it should consider not only what the adversary can do *after* this action but also the possibility that the latter *will not wait for t time*, and perform an action at some $t' < t$.

While synthesizing a controller for timed automata one should be careful not letting any of the players win by "Zenonism", that is, by preventing the time from progressing as does the Tortoise in its race against Achilles.

3.2 *Timed Game Automata*

Let Q be a finite set of *states* and let $X = I\!R^d$ for some integer d be the *clock space*. We denote elements of X as $\mathbf{x} = (x_1, \ldots, x_d)$ and use $\mathbf{0}$ for the zero vector and $\mathbf{x}+t$ for $\mathbf{x}+(t,t,\ldots,t)$. Elements of $Q\times X$ are called *configurations*. A subset of X is called a *k-zone* if it can be obtained as a boolean combination of inequalites of the form $x_i \leq c$, $x_i < c$, $x_i - x_j \leq c$, where $c \in \{0,1,\ldots,k\}$. The set of zones is denoted by $\mathcal{Z}(X)$. These properties of subsets of X extend naturally to subsets of $Q \times X$, e.g. we say that $P \subseteq Q \times X$ is a zone if it can be written as finite union of sets of the form $\{q_i\} \times P_i$ such that all the P_i's are zones. A function $\rho : X \to X$ is a *reset* function if it sets some of the coordinates of its argument to 0 and leaves the others intact. The set of all such functions is denoted by $\mathcal{F}(X)$. We assume two finite sets of actions A and B, a special empty action ε and let $A^\varepsilon = A \cup \{\varepsilon\}$ and $B^\varepsilon = B \cup \{\varepsilon\}$.

Definition 7. (Timed Game Automaton).
A Timed game automaton (TGA) is a tuple $\mathcal{A} = (Z, A, B, T^A, T^B, \delta, \rho)$ where $Z \subseteq Q \times X$ is a zone, Q and X are the state and clock spaces, A and B are two distinct action alphabets, $T^A \subseteq Q \times X \times A^\varepsilon$ and $T^B \subseteq Q \times X \times B^\varepsilon$ are timing constraints for the two types of actions, the functions $\delta : Q \times A^\varepsilon \times B^\varepsilon \to Q$ and $\rho : Q \times A^\varepsilon \times B^\varepsilon \to \mathcal{F}(X)$ indicate which state is reached when performing a (possibly joint) action and which clocks are reset in that occasion.

Further requirements are the following: for every state q and action $a \in A^\varepsilon$, the set $T^A(q,a) = \{\mathbf{x} : (q,\mathbf{x},a) \in T^A\}$ is a k-zone. We assume k to be fixed throughout the paper — it is the largest constant in the definition of the TGA. Similar requirements hold for T^B. We assume that $\delta(q,\varepsilon,\varepsilon) = q$ and that $\rho(q,\varepsilon,\varepsilon)$ is the identity function (if both sides refrain from action nothing happens). Finally we require that the automaton is *strongly non-Zeno*, that is, in every cycle in the transition graph of the automaton (induced by δ), there is at least one transition which resets a clock variable x_i to zero, and at least one transition which can be taken only if $x_i \geq 1$. This is a very important condition as it prevents the controller and the environment to achieve their goals using unrealistic tricks that stop time. The restriction of T^A to some set of configurations Z' is denoted by $T^A{}_{|Z'}$ and the set of configurations on which T^A is defined is denoted by $S(T^A)$.

Intuitively, when the automaton is at a configuration $(q, \mathbf{x})$, time can progress as long as both players agree, that is, $(q, \mathbf{x}, \varepsilon) \in T^B \cap T^A$. As soon as one of them can take an action, i.e. $(q, \mathbf{x}, a) \in T^A$ for some $a \in A$ or $(q, \mathbf{x}, b) \in T^B$ for some $b \in B$, or both, a transition can be taken. This can be formalized as follows:

Definition 8. (Steps and Runs). A *joint step* of a TGA $\mathcal{A}$ is $(q, \mathbf{x}) \longrightarrow (q', \mathbf{x}')$ which is either

(1) a time step (of duration t):

$$(q, \mathbf{x}) \xrightarrow{t} (q, \mathbf{x} + t)$$

such that $t > 0$, and for every $t' < t$, $(q, \mathbf{x} + t', \varepsilon) \in T^A \cap T^B$.

(2) a discrete step

$$(q, \mathbf{x}) \xrightarrow{(a,b)} (q', \mathbf{x}')$$

such that $(a, b) \neq (\varepsilon, \varepsilon)$, $(q, \mathbf{x}, a) \in T^A$, $(q, \mathbf{x}, b) \in T^B$, $q' = \delta(q, a, b)$, and $\mathbf{x}' = \rho(q, a, b)(\mathbf{x})$.

A *run* of a TGA $\mathcal{A}$ starting from $(q_0, \mathbf{x}_0)$ is a sequence of joint steps

$$\xi : (q_0, \mathbf{x}_0) \longrightarrow (q_1, \mathbf{x}_1) \longrightarrow \ldots$$

such that either ξ is finite and arbitrarily large time steps are enabled after the last configuration, or ξ is infinite and the sum of the durations of the time steps diverges.

Note that $(q, \mathbf{x}, \varepsilon) \in T^A \cap T^B$ means that both A and B agree to let time progress by a *positive* amount. On the other hand it is possible to reach a state where ε is not permitted by one or more of the two players: in this case, the only thing that can happen then is a transition. A TGA is *non-blocking* if every finite perfix of a run can be extended to a full run.

The set of all runs of $\mathcal{A}$, starting at some $(q_0, \mathbf{x}_0) \in P$ is denoted by $L(\mathcal{A}, P)$. The set of configurations reachable by a discrete step is $\{(q, \mathbf{x}), (q', \mathbf{x}')\}$ and by a time step it is $\{(q, \mathbf{x} + t') : t' \in [0, t]\}$. The configurations reachable by the run, $Reach(\xi)$ is the union of the reachable configurations of the steps. By $Reach(\mathcal{A}, P)$ we denote the configurations reachable by all runs starting from P. We use the notation $(q, \mathbf{x}) \longrightarrow P$ as before.

3.3 *Timed Safety Games*

Definition 9. (Controllers for Timed $\Box$-Games). Given a TGA $\mathcal{A} = (Z, A, B, T^A, T^B, \delta, \rho)$ and a zone $G \subseteq Q \times X$, the controller synthesis problem $Synth(\mathcal{A}, G, \Box)$ is: find the maximal subset Z_* $Z_* \subseteq G \subseteq Z$, and the maximal T^A_*, $T^A_* \subseteq T^A|G \subseteq T^A$ such that the automaton $\mathcal{A}_* = (Z_*, A, B, T^A_*, T^B{}_{|Z_*}, \delta, \rho)$, is non-blocking and $Reach(\mathcal{A}_*, Z_*) \subseteq G$.

The calculation of $\mathcal{A}_*$, its winning strategy T^A_* and its set of winning configuratuions Z_* is, in principle, similar to the untimed case. However the predecessor operator should be adapted to treat the passage of time. Instead of the single operator π we had in the untimed case we will have two operators, one indicating the active transition that A can take and force the game into P and the other indicating when it is safe to do nothing and wait. A non-trivial combination of both will be used by the iterative procedure.

Definition 10. (Timed Controllable Predecessor). Let $\mathcal{A}$ be a TGA. The operators

$$\pi^\delta : 2^{Q \times X} \rightarrow 2^{Q \times X \times A^\varepsilon}$$

(active predecessors) and

$$\pi^t : 2^{Q \times X} \rightarrow 2^{Q \times X \times A^\varepsilon}$$

(passive predecessors) are defined as follows:

$$\pi^\delta(P) = \left\{ \begin{array}{l} (q, \mathbf{x}, a) : \\ \forall b \in B^\varepsilon \;\; (q, \mathbf{x}, b) \in T^B \;\Rightarrow\; (q, \mathbf{x}) \xrightarrow{(a,b)} P \;\wedge \\ (a \in A) \;\vee\; (q, \mathbf{x}, \varepsilon) \not\in T^B) \end{array} \right\}$$

$$\pi^t(P) = \left\{ \begin{array}{l} (q, \mathbf{x}, \varepsilon) : \\ \forall b \in B^\varepsilon \;\; (q, \mathbf{x}, b) \in T^B \;\Rightarrow\; (q, \mathbf{x}) \xrightarrow{(\varepsilon,b)} P \end{array} \right\}$$

The intuition behind this definition is the following. The operator π^δ is like the untimed π except for some small technical subtlety in the third line, where ε is considered an active transition of A when B must make a transition. The π^t operator defines the configuration in which it is safe for A to do nothing because either B can do nothing and the configuration is already in P, or B can take other transitions, but they all lead to P.

However, not from every configurations $(q, \mathbf{x}) \in S(\pi^t(P))$ can A force the game to stay in P. This is true only if $(q, \mathbf{x} + t) \in S(\pi^t(P))$ for every t or at least this is true until some point $(q, \mathbf{x} + t)$ where an active transition can be taken. Configurations in which A can gain only *a bounded amount of time* are losing. This motivates the following definition.

Definition 11. (Until Operator). The operator

$$\mathcal{U} : 2^{Q \times X} \times 2^{Q \times X} \rightarrow 2^{Q \times X}$$

is defined as follows: for every Z_1, Z_2

$$\begin{array}{rl} Z_1 \mathcal{U} Z_2 = & \{(q, \mathbf{x}) : \\ & (\exists t > 0 \, (q, \mathbf{x} + t) \in Z_2 \;\wedge \\ & \;\forall t' < t \;\, (q, \mathbf{x} + t') \in Z_1) \;\vee \\ & \;\forall t > 0 \, (q, \mathbf{x} + t) \in Z_1\}. \end{array}$$

In other words, a configuration is in $Z_1 \mathcal{U} Z_2$ if by letting time pass we are guaranteed not to leave Z_1 before reaching Z_2.

Definition 12. (Timed $\Box$-Predecessors). The operator $\pi^\Box : 2^{Q\times X} \to 2^{Q\times X\times A^\varepsilon}$ is defined as

$$\begin{aligned}\pi^\Box(P) &= \pi^\delta(P) \cup \\ &\quad \{(q,\mathbf{x},\varepsilon) : (q,\mathbf{x}) \in S(\pi^t(P))\mathcal{U}S(\pi^\delta(P))\}.\end{aligned}$$

Lemma 13. (Properties of $\pi^\Box$). 1) If $(q,\mathbf{x},a) \in \pi^\Box(P)$ then every run of at most one discrete transition, starting from $(q,\mathbf{x})$ by A making a, has all its reachable condifurations in P. 2) If $(q,\mathbf{x},a) \notin \pi^\Box(P)$ there is an unavoidable run of at most one discrete step leaving P.

The algorithm for calculating T^A_* is the following:

Algorithm 2. (Strategy for Timed $\Box$-Games).

$$\begin{array}{l} T^A[0] := T^A{}_{|G} \\ Z[0] := G \\ \textbf{for } i = 1,2,\ldots, \textbf{ repeat} \\ \quad T^A[i] := T^A[i-1] \;\cap\; \pi^\Box(Z[i-1]) \\ \quad Z[i] := S(T^A[i]) \\ \textbf{until } Z[i] = Z[i-1] \\ T^A_* := T^A[i] \\ Z_* := Z[i] \end{array}$$

Claim 14. (Properties of Algorithm 2). For every i: 1) If $(q,\mathbf{x},a) \in T^A[i]$ then every run of $\mathcal{A}[i]$ having at most i steps, starting by doing a at $(q,\mathbf{x})$, will keep the automaton inside G. 2) If $(q,\mathbf{x},a) \notin T^A[i]$, then whatever A does after chosing a at $(q,\mathbf{x})$, B can drive the automaton outside G within at most i steps.

Proof: By induction: the base case follows trivially from the initialization of the algorithm. Suppose it is true for i. Then, by lemma 13 we know that: 1) from $Z[i+1]$ every step leaves the automaton inside $Z[i]$, and 2) from $\neg Z[i+1]$ there is an unavoidable step leading to $\neg Z[i]$ and hence, by the induction hypothesis: 1) all runs of length not greater than $i+1$ starting from $Z[i+1]$ stay in G, and 2) from every state in $\neg Z[i+1]$ there is an unavoidable run of length not greater than $i+1$ leaving G. ▟

Corollary 15. (Partial Correctness of Algorithm 2). Suppose the algorithm convereges to T^A_* and let $\mathcal{A}_* = (Z_*, A, B, T^A_*, T^B{}_{|Z_*}, \delta, \rho)$ be the associated TGA. Then: 1) $Reach(\mathcal{A}_*, T^A_*) \subseteq G$. 2) Any other subset of T^A satisfying (1) is included in T^A_*.

3.4 *Closure of Zones under $\pi^\Box$*

It remains to show that the algorithm converges.

Claim 16. (Properties of zones).

(1) There are finitely many k-zones.
(2) The set of zones is closed under the boolean operations.
(3) If F is a zone and $\rho \in \mathcal{F}(X)$ a reset, then the inverse image $\rho^{-1}(F) = \{x : \rho(x) \in F\}$ is also a zone.
(4) If F is a zone then its "projection to the past" $\mathrm{Past}(F) = \{x : \exists t \geq 0 \quad (x+t \in F)\}$ is also a zone.
(5) If F and G are zones and $(q,S) = (q,F)\mathcal{U}(q,G)$ then S is also a zone.

The first four properties are standard in the context of timed automata — see [AD94]. The operation used in the last statement is the only novelty introduced by the π-operator. In order to prove statement 5 of the claim we rewrite the set S as:

$$\begin{aligned}\{\mathbf{x} :\; & d(\mathbf{x},\neg F) > d(\mathbf{x},G) \;\vee \\ & d(\mathbf{x},\neg F) = d(\mathbf{x},G) = \infty \;\vee \\ & (d(\mathbf{x},\neg F) = d(\mathbf{x},G) = d \;\wedge\; \mathbf{x}+d \in G)\}\end{aligned}$$

where $d(\mathbf{x},C) = \inf\{t : \mathbf{x}+t \in C\}$ — the "distance" from $\mathbf{x}$ to C. Note that if $\mathbf{x} \notin Past(C)$ then $d(\mathbf{x},C) = \infty$. We leave it to the reader to prove that the second and the third terms (which correspond to some boundary effects) are zones. We concentrate on the "main" first term of the formula, and in order to state that it is a zone we find a special representation of this distance function, using an auxiliary notion.

Definition 17. We call a function $f : X \to I\!R_+ \cup \{\infty\}$ *piecewise trivial* if it can be represented in the form:

$$f(\mathbf{x}) = \begin{cases} d_i - x_{k_i} & \text{when } \mathbf{x} \in D_i, i \in I \\ \infty & \text{when } \mathbf{x} \in D_\infty, \end{cases}$$

where I is a finite set, d_i stand for some integer constants and D_i for some zones.

Lemma 18. Let f and g be piecewise trivial, then 1) the functions $\min(f,g)$ and $\max(f,g)$ are piecewise trivial, and 2) the set $\{\mathbf{x} : f(\mathbf{x}) > g(\mathbf{x})\}$ is a zone.

Proof: 1) Suppose

$$f(\mathbf{x}) = \begin{cases} d_i - x_{k_i} & \text{when } \mathbf{x} \in D_i \\ \infty & \text{when } \mathbf{x} \in D_\infty, \end{cases}$$

and

$$g(\mathbf{x}) = \begin{cases} e_j - x_{l_j} & \text{when } \mathbf{x} \in E_j \\ \infty & \text{when } \mathbf{x} \in E_\infty, \end{cases}$$

then

$$\max(f,g) = \begin{cases} d_i - x_{k_i} & \text{when } \mathbf{x} \in D_i \cap E_j \cap F_{ij} \\ e_j - x_{l_j} & \text{when } \mathbf{x} \in D_i \cap E_j \cap \neg F_{ij} \\ \infty & \text{when } \mathbf{x} \in D_\infty \cup E_\infty, \end{cases}$$

Where $F_{ij} = \{\mathbf{x} : d_i - x_{k_i} > e_j - x_{l_j}\}$ is, of course, a zone. By definition and closure of zones under intersection, $\max(f,g)$ is piecewise trivial. The proof for $\min(f,g)$ is similar. 2) Consider f and g and F_{ij} as before. The desired set is the union of all sets of the form $D_i \cap E_j \cap F_{ij}$. ▟

Lemma 19. (Distance to zone is piecewise trivial). Let C be a zone. Then the function $d(\mathbf{x},C)$ is piecewise trivial.

Proof. Consider first the case when C is convex. It can be represented as an intersection of half-spaces $x_i \# c$ or $x_i - x_j \# c$, where $\#$ stands for $<, >, \leq$ or $\geq$. We proceed by induction on the number N of half-spaces (i.e. the number of faces of F). Consider first a single half-space H. There are two cases:

- $H = \{\mathbf{x} : x_i < c\}$ or $H = \{\mathbf{x} : x_i - x_j \# c\}$. In this case
$$d(\mathbf{x}, H) = \begin{cases} 0 \text{ when } \mathbf{x} \in H \\ \infty \text{ when } \mathbf{x} \notin H, \end{cases}$$
because $\text{Past}(H) = H$ in this case.
- $H = \{\mathbf{x} : x_i > c\}$. For such H
$$d(\mathbf{x}, H) = \begin{cases} 0 \text{ when } \mathbf{x} \in H \\ c - x_i \text{ when } \mathbf{x} \notin H, \end{cases}$$

Suppose now that the result holds for a given N. Consider a zone C with $N+1$ faces. It can be represented as $C = C_1 \cap C_2$ where C_i have at most N faces. It is easy to see that $d(\mathbf{x}, C) =$

$$\begin{cases} 0 & \text{when } \mathbf{x} \in C \\ \infty & \text{when } \mathbf{x} \notin \text{Past}(C) \\ \max(d(\mathbf{x}, C_1), d(\mathbf{x}, C_2)) & \text{when } \mathbf{x} \in \text{Past}(C) - C \end{cases}$$

By virtue of Lemma 18 and the inductive hypothesis $\max(d(x, C_1), d(x, C_2))$ is piecewise trivial. The piecewise triviality of $d(x, F)$ is immediate.

Arbitrary non-convex F can be represented as a finite union of convex zones: $C = \bigcup_i C_i$. In this case $d(x, C) = \min_i(d(x, C_i))$ and the result is immediate from the convex case and Lemma 18. This concludes the proof of Claim 16. The closure properties of zones imply that S is a zone.

Corollary 20. (Termination). Algorithm 2 terminates after a finite number of steps.

Proof: According to the previous claim, the set of k-zones is finite and closed under the π operator. Hence all the iterations work on a finite set and a fixed point is guaranteed. ∎

Theorem 21. (Main Result). The controller synthesis for timed safety games is decidable, and is solved using algorithm 2.

Other results on timed controller synthesis were based on weaker models, which correspond roughly to a timed automaton with one clcok. The only exception we are aware of is [WH92] where the Ramadge-Wonham approach was applied to a finite-state quotient of a timed automaton (the "region graph" of [AD94]).

4. DISCUSSION

Timed automata can model a variety of phenomena, including approximations of continuous dynamical systems, digital circuits with delays, scheduling problems in manufacturing and multimedia as well as timing analysis of embedded software. The algorithm described in this paper can be useful in all these application domains. A prototype version has been implemented at VERIMAG and the experimental results will be reported elsewhere.

These techniques can be applied, in principle, to more general classes of hybrid synamical systems, where the continuous dynamics is more complicated than that of a clock. In such cases, of course, there is no hope for an algorithm which is guaranteed to terminate. Wong-Toi [W97] describes a similar procedure for "linear" hybrid systems, that is, systems where in each state, the continuous variables evolve according to a fixed derivative. In [LTS98] the authors combine our approach with concepts from optimal control in order to treat more general hybrid systems.

5. REFERENCES

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

[AMP95] E. Asarin, O. Maler and A. Pnueli, Symbolic Controller Synthesis for Discrete and Timed Systems, in *Hybrid Systems II*, LNCS 999, Springer, 1995.

[BL69] J.R. Büchi and L.H. Landweber, Solving Sequential Conditions by Finite-state Operators, *Trans. of the AMS* 138, 295-311, 1969.

[KG95] R. Kumar and V.K. Garg, *Modeling and Control of Logical Discrete Event Systems*, Kluwer, 1995.

[MPS95] A. Maler, O. Pnueli and J. Sifakis. On the Synthesis of Discrete Controllers for Timed Systems, *Proc. STACS'95,* LNCS 900, 229-242, Springer, 1995.

[NM44] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[PR89] A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module, In *Proc. 16th ACM Symp. Princ. of Prog. Lang.*, pages 179–190, 1989.

[RW87] P.J. Ramadge and W.M. Wonham, Supervisory Control of a Class of Discrete Event Processes, *SIAM J. of Control and Optimization* 25, 206-230, 1987.

[LTS98] C. Tomlin, J. Lygeros and S. Sastry, Synthesizing Controllers for Nonlinear Hybrid Systems, in *Hybrid Systems: Computation and Control*, LNCS 1386, 360-373, Springer, 1998.

[TB73] B.A. Trakhtenbrot and Y.M. Barzdin, *Finite Automata: Behavior and Synthesis*, North-Holland, Amsterdam, 1973.

[W97] H. Wong-Toi, The Synthesis of Controllers for Linear Hybrid Automata, *Proc. CDC'97*, 1997.

[WH92] H. Wong-Toi and G. Hoffmann, The Control of Dense Real-Time Discrete Event Systems, Technical report STAN-CS-92-1411, Stanford University, 1992.