UNIVERSITÀ
DEGLI STUDI
DI UDINE

On the Synthesis of Discrete Controllers for Timed Systems

An Extended Abstract

E. Cominato 137396[1]

[1]Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli studi di Udine

2022-07-15

Automatic Verification

On the Synthesis of Discrete Controllers for Timed Systems
An Extended Abstract

E. Cominato 137396[1]

[1]Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli studi di Udine

Introduction

This paper presents algorithms for the automatic synthesis of the real time controllers by finding a winning strategy for certain games defined by the timed automata of Alur and Dill.

This paper presents algorithms for the automatic synthesis of the real time controllers by finding a winning strategy for certain games defined by the timed automata of Alur and Dill.

Introduction

Consider a dynamical system P, whose presentation describes all its possible behaviours. A subset of the plant's behaviours, satisfying some criterion is defined as good or acceptable.

A controller C is another system which can interact with P in a certain manner by observing the state of P and by issuing control actions that influence the behaviour of P.

Consider a dynamical system P, whose presentation describes all its possible behaviours. A subset of the plant's behaviours, satisfying some criterion is defined as good or acceptable.

A controller C is another system which can interact with P in a certain manner by observing the state of P and by issuing control actions that influence the behaviour of P.

Introduction

The synthesis problem is then, to find out whether, for a given P, there exists a realizable controller C such that their interaction will produce only good behaviours.

The synthesis problem is then, to find out whether, for a given P, there exists a realizable controller C such that their interaction will produce only good behaviours.

Discrete Case

Definition 1 (*Plant*)

A plant automaton is a tuple $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ where Q is a finite set of states, Σ_c is a set of controller commands, $\delta : Q \times \Sigma_c \longmapsto 2^Q$ is the transition function and $q_0 \in Q$ is an initial state.

Definition 2 (*Controllers*)

A controller (strategy) for a plant specified by $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ is a function $C : Q^+ \longmapsto \Sigma_c$. A simple controller is a controller that can be written as a function $C : Q \longmapsto \Sigma_c$.

Definition 1 (*Plant*)

A plant automaton is a tuple $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ where Q is a finite set of states, Σ_c is a set of controller commands, $\delta : Q \times \Sigma_c \longmapsto 2^Q$ is the transition function and $q_0 \in Q$ is an initial state.

Definition 2 (*Controllers*)

A controller (strategy) for a plant specified by $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ is a function $C : Q^+ \longmapsto \Sigma_c$. A simple controller is a controller that can be written as a function $C : Q \longmapsto \Sigma_c$.

We are interested in the simpler cases of controllers that base their decisions on a finite memory.

Discrete Case

Definition 3 (*Trajectories*)

Let $\mathcal{P}$ be a plant and let $C : Q^+ \longmapsto \Sigma_c$ be a controller. An infinite sequence of states $\alpha : q[0], q[1], \ldots$ such that $q[0] = q_0$ is called a trajectory of $\mathcal{P}$ if

$$q[i+1] \in \bigcup_{\sigma \in \Sigma_c} \delta(q[i], \sigma)$$

and a C-trajectory if $q[i+1] \in \delta(q[i], C[\alpha[0..i]])$ for every $i \geq 0$. The corresponding sets of trajectories are denoted by $L(\mathcal{P})$ and $L_C(\mathcal{P})$.

Definition 3 (*Trajectories*)

Let $\mathcal{P}$ be a plant and let $C : Q^+ \longmapsto \Sigma_c$ be a controller. An infinite sequence of states $\alpha : q[0], q[1], \ldots$ such that $q[0] = q_0$ is called a trajectory of $\mathcal{P}$ if

$$q[i+1] \in \bigcup_{\sigma \in \Sigma_c} \delta(q[i], \sigma)$$

and a C-trajectory if $q[i+1] \in \delta(q[i], C[\alpha[0..i]])$ for every $i \geq 0$. The corresponding sets of trajectories are denoted by $L(\mathcal{P})$ and $L_C(\mathcal{P})$.

Discrete Case

For every infinite trajectory $\alpha \in L(\mathcal{P})$:

- $\mathit{Vis}(\alpha)$ denote the set of all states appearing in α
- $\mathit{Inf}(\alpha)$ denote the set of all states appearing in α infinitely many times

For every infinite trajectory $\alpha \in L(\mathcal{P})$:
- $\mathit{Vis}(\alpha)$ denote the set of all states appearing in α
- $\mathit{Inf}(\alpha)$ denote the set of all states appearing in α infinitely many times

Definition 4 (*Acceptance Condition*)

Let $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ be a plant. An acceptance condition for $\mathcal{P}$ is

$$\Omega \in \{(F, \Box), (F, \Diamond), (F, \Diamond\Box), (F, \Box\Diamond), (\mathcal{F}, \mathcal{R}_n)\}$$

where $\mathcal{F} = \{(F_i, G_i)\}_{i=1}^{n}$ and F, F_i and G_i are certain subsets of Q referred as the good states. The set of sequences of $\mathcal{P}$ that are accepted accordig to Ω is defined as follows:

$L(\mathcal{P}, F, \Box)$	$\{\alpha \in L(\mathcal{P}) : Vis(\alpha) \subseteq F\}$
$L(\mathcal{P}, F, \Diamond)$	$\{\alpha \in L(\mathcal{P}) : Vis(\alpha) \cap F \neq \emptyset\}$
$L(\mathcal{P}, F, \Diamond\Box)$	$\{\alpha \in L(\mathcal{P}) : Inf(\alpha) \subseteq F\}$
$L(\mathcal{P}, F, \Box\Diamond)$	$\{\alpha \in L(\mathcal{P}) : Inf(\alpha) \cap F \neq \emptyset\}$
$L(\mathcal{P}, \mathcal{F}, \mathcal{R}_n)$	$\{\alpha \in L(\mathcal{P}) : \exists i \alpha \in L(\mathcal{P}, F_i, \Box\Diamond) \cap L(\mathcal{P}, G_i, \Diamond\Box)\}$

Definition 4 (*Acceptance Condition*)
Let $\mathcal{P} = (Q, \Sigma_c, \delta, q_0)$ be a plant. An acceptance condition for $\mathcal{P}$ is $\Omega \in \{(F, \Box), (F, \Diamond), (F, \Diamond\Box), (F, \Box\Diamond), (\mathcal{F}, \mathcal{R}_n)\}$ *where $\mathcal{F} = \{(F_i, G_i)\}_{i=1}^{n}$ and F, F_i and G_i are certain subsets of Q referred as the good states. The set of sequences of $\mathcal{P}$ that are accepted accordig to Ω is defined as follows:*

1. α always remains in F
2. α eventually visits F
3. α eventually remains in F
4. α visits F infinitely often
5. α visits F_i infinitely often and eventually stays in G_i

Discrete Case

Definition 5 (*Controller Synthesis Problem*)

For a plant $\mathcal{P}$ and an acceptance condition Ω, the problem ***Synth***$(\mathcal{P}, \Omega)$ *is: Find a controller C such that $L_C(\mathcal{P}) \subseteq L(\mathcal{P}, \Omega)$ ot otherwise show that such a controller does not exists.*

Definition 5 (*Controller Synthesis Problem*)

For a plant $\mathcal{P}$ and an acceptance condition Ω, the problem ***Synth***$(\mathcal{P}, \Omega)$ *is: Find a controller C such that $L_C(\mathcal{P}) \subseteq L(\mathcal{P}, \Omega)$ ot otherwise show that such a controller does not exists.*

Discrete Case

Definition 6 (*Controllable Predecessors*)

Let $\mathcal{P} = (Q, \Sigma_c, \delta, q)$ be a plant and a set of states $P \subseteq Q$. The controllable predecessors of P is the set of states from which the controller can "force" the plant into P in one step:

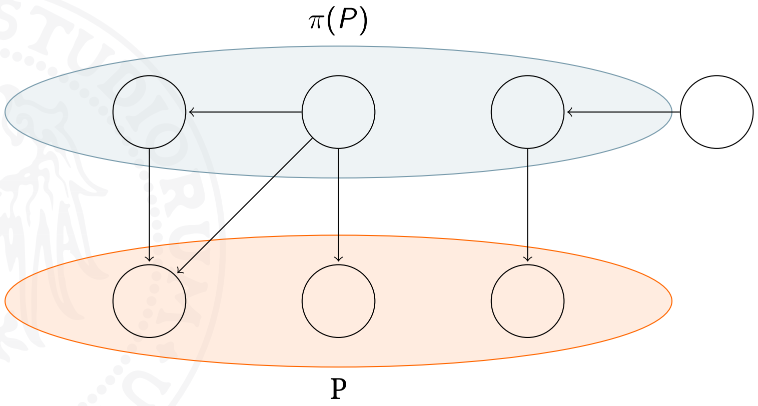
$$\{q : \exists \sigma \in \Sigma_c \,.\, \delta(q, \sigma) \subseteq P\}$$

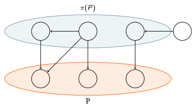
We define a function $\pi : 2^Q \longmapsto 2^Q$, mapping a set of states $P \subseteq Q$ into the set of its Controllable predecessors:

$$\pi(P) = \{q : \exists \sigma \in \Sigma_c \,.\, \delta(q, \sigma) \subseteq P\}$$

Definition 6 (Controllable Predecessors)
Let $\mathcal{P} = (Q, \Sigma_c, \delta, q)$ be a plant and a set of states $P \subseteq Q$. The controllable predecessors of P is the set of states from which the controller can "force" the plant into P in one step:
$\{q : \exists \sigma \in \Sigma_c \,.\, \delta(q, \sigma) \subseteq P\}$
We define a function $\pi : 2^Q \longmapsto 2^Q$, mapping a set of states $P \subseteq Q$ into the set of its Controllable predecessors:
$\pi(P) = \{q : \exists \sigma \in \Sigma_c \,.\, \delta(q, \sigma) \subseteq P\}$

Discrete Case





Discrete Case

Theorem 1

For every $\Omega \in \{(F,\Box),(F,\Diamond),(F,\Diamond\Box),(F,\Box\Diamond),(\mathcal{F},\mathcal{R}_n)\}$, the problem ***Synth***$(\mathcal{P},\Omega)$ *is solvable. Moreover, if $(\mathcal{P},\Omega)$ is controllable then it is controllable by a simple controller.*

For a plant $\mathcal{P} = (Q,\Sigma_c,\delta,q_0)$ and an acceptance condition Ω, we denote $W \subseteq Q$ as the set of winning states, namely, the set of states from which a controller can enforce good behaviors according to Ω.

Theorem 1

For every $\Omega \in \{(F,\Box),(F,\Diamond),(F,\Diamond\Box),(F,\Box\Diamond),(\mathcal{F},\mathcal{R}_n)\}$, the problem ***Synth***$(\mathcal{P},\Omega)$ *is solvable. Moreover, if $(\mathcal{P},\Omega)$ is controllable then it is controllable by a simple controller.*

For a plant $\mathcal{P} = (Q,\Sigma_c,\delta,q_0)$ and an acceptance condition Ω, we denote $W \subseteq Q$ as the set of winning states, namely, the set of states from which a controller can enforce good behaviors according to Ω.

Discrete Case

We can characterize this states by the following fixed-point expressions:

- $\square$ $\nu W(F \cap \pi(W))$
- $\Diamond$ $\nu W(F \cup \pi(W))$
- $\Diamond\square$ $\mu W \nu H\Big(\pi(H) \cap (F \cup \pi(W))\Big)$
- $\square\Diamond$ $\nu W \mu H\Big(\pi(H) \cup (F \cap \pi(W))\Big)$
- $\mathcal{R}_1$ $\mu W\Bigg\{\pi(W) \cap \nu Y \mu H.W \cup G \cap \Big(\pi(H) \cup (F \cap \pi(Y))\Big)\Bigg\}$

Then the plant is controllable iff $q_0 \in W$

We can characterize this states by the following fixed-point expressions:
$\square$ $\nu W(F \cap \pi(W))$
$\Diamond$ $\nu W(F \cup \pi(W))$
$\Diamond\square$ $\mu W \nu H\Big(\pi(H) \cap (F \cup \pi(W))\Big)$
$\square\Diamond$ $\nu W \mu H\Big(\pi(H) \cup (F \cap \pi(W))\Big)$
$\mathcal{R}_1$ $\mu W\Bigg\{\pi(W) \cap \nu Y \mu H.W \cup G \cap \Big(\pi(H) \cup (F \cap \pi(Y))\Big)\Bigg\}$
Then the plant is controllable iff $q_0 \in W$

ν greatest

μ least

Discrete Case

Let see in more details how this works. Consider the case $\Diamond$:

$$
\begin{array}{ll}
W_0 := \emptyset & W_0 := \emptyset \\
\textbf{for } i := 0, 1, \ldots \textbf{ repeat} & W_1 := F \cup \pi(W_0) = F \cup \pi(\emptyset) = F \\
\quad W_{i+1} := F \cup \pi(W_i) & W_2 := F \cup \pi(W_1) = F \cup \pi(F) \\
\textbf{until } W_{i+1} = W_i & \ldots
\end{array}
$$

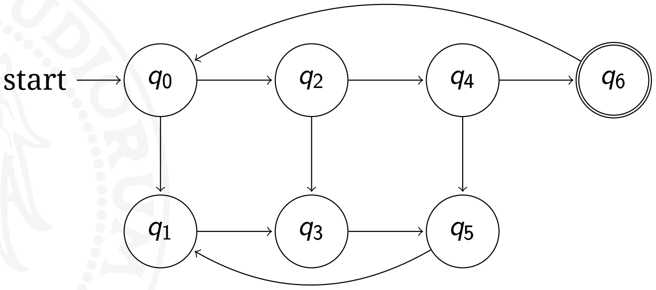
finally: $W_n := F \cup \pi(W_{n-1}) = F \cup \pi(F \cup \pi(\ldots(F \cup \pi(F))))$

Let see in more details how this works. Consider the case $\Diamond$:

$W_0 := \emptyset$ $W_0 := \emptyset$
for $i := 0, 1, \ldots$ **repeat** $W_1 := F \cup \pi(W_0) = F \cup \pi(\emptyset) = F$
$W_{i+1} := F \cup \pi(W_i)$ $W_2 := F \cup \pi(W_1) = F \cup \pi(F)$
until $W_{i+1} = W_i$ $\ldots$

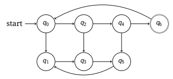
finally: $W_n := F \cup \pi(W_{n-1}) = F \cup \pi(F \cup \pi(\ldots(F \cup \pi(F))))$

Discrete Case

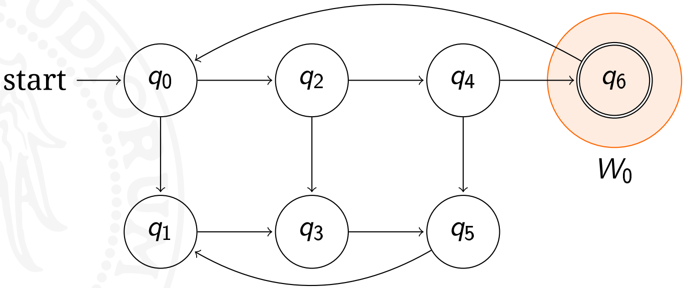


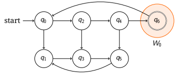
2022-07-15

Automatic Verification
└Discrete Case
  └Theorem

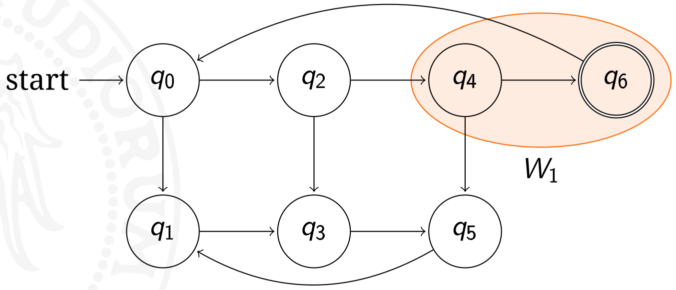


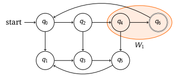
Discrete Case



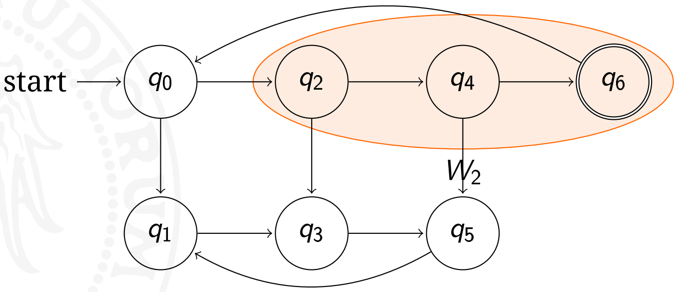


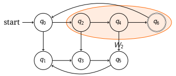
Discrete Case



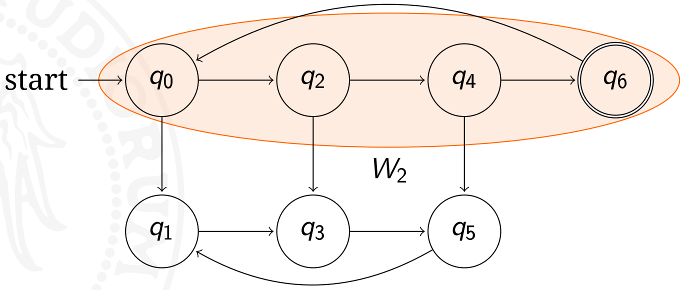


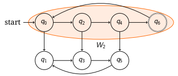
Discrete Case





Discrete Case





Discrete Case

In the process of calculating W_{i+1}, whenever we add a state q to W_i, there must be at least one action $\sigma \in \Sigma_c$ such that $\delta(q, \sigma) \subseteq W_i$.

So we define the controller at q as $C(q) = \sigma$.

When the process terminates, the controller is synthesized for all the winning states.

It can be seen that if the process fails, that is $q_0 \notin W$, then for every controller command there is a possibly bad consequence that will put the system outside F, and no controller, even an infinite state one, can prevent this.

In the process of calculating W_{i+1}, whenever we add a state q to W_i, there must be at least one action $\sigma \in \Sigma_c$ such that $\delta(q, \sigma) \subseteq W_i$.

So we define the controller at q as $C(q) = \sigma$.

When the process terminates, the controller is synthesized for all the winning states.

It can be seen that if the process fails, that is $q_0 \notin W$, then for every controller command there is a possibly bad consequence that will put the system outside F, and no controller, even an infinite state one, can prevent this.

UNIVERSITÀ
DEGLI STUDI
DI UDINE

Real Time Case

Let T denote $\mathbb{R}^+$ and let $X = T^d$.

The elements of X are $x = (x_1, \ldots, x_d)$ and the d-dimensional unit vector is $\mathbf{1} = (1, \ldots, 1)$

Definition 7 (*Reset functions*)

Let $F(X)$ denote the class of functions $f : X \mapsto X$ that can be written in the form $f(x_1, \ldots, f_d) = (f_1, \ldots, f_d)$ where each f_i is either x_i or 0.

Let T denote $\mathbb{R}^+$ and let $X = T^d$.

The elements of X are $x = (x_1, \ldots, x_d)$ and the d-dimensional unit vector is $\mathbf{1} = (1, \ldots, 1)$

Definition 7 (*Reset functions*)

Let $F(X)$ denote the class of functions $f : X \mapsto X$ that can be written in the form $f(x_1, \ldots, f_d) = (f_1, \ldots, f_d)$ where each f_i is either x_i or 0.

Real Time Case

Definition 8 (k *polyhedral sets*)

Let k be a positive integer constant. We associate with k three subsets of 2^X:

- $\mathcal{H}_k$*: the set of half-spaces consisting of all sets having one of the following forms*
  - X
  - $\emptyset$
  - $\{x \in X : x_i \# c\}$
  - $\{x \in X : x_i - x_j \# c\}$

  for some $\# \in \{<, \leq, >, \geq\}$ and $c \in \{0, \dots, k\}$
- $\mathcal{H}_k^{\cap}$*: the set of convex sets consisting of intersections of elements of* $\mathcal{H}_k$
- $\mathcal{H}_k^{*}$*: the set of k-polyhedral sets containing all sets obtained from* $\mathcal{H}_k$ *via union intersection and complementation*

Definition 8 (k *polyhedral sets*)

Let k be a positive integer constant. We associate with k three subsets of 2^X:

- $\mathcal{H}_k$*: the set of half-spaces consisting of all sets having one of the following forms*
  - X
  - $\emptyset$
  - $\{x \in X : x_i \# c\}$
  - $\{x \in X : x_i - x_j \# c\}$

  for some $\# \in \{<, \leq, >, \geq\}$ and $c \in \{0, \dots, k\}$
- $\mathcal{H}_k^{\cap}$*: the set of convex sets consisting of intersections of elements of* $\mathcal{H}_k$
- $\mathcal{H}_k^{*}$*: the set of k-polyhedral sets containing all sets obtained from* $\mathcal{H}_k$ *via union intersection and complementation*

Definition 9 (*Timed Automata*)

A timed automaton is a tuple $\mathcal{T} = (Q, X, \Sigma, I, R, q_0)$ consisting of:

- *Q a finite set of discrete states*
- *X a clock domain $X = (\mathbb{R}^+)^d$ for some $d > 0$*
- *Σ an input alphabet (including a single environment action e)*
- *$I : Q \mapsto \mathcal{H}_k^\cap$ as the state invariant function*
- *$R \subseteq Q \times \Sigma \times \mathcal{H}_k^\cap \times F(X) \times Q$ is a set of transition relations each of the form $\langle q, \sigma, g, f, q' \rangle$ where:*
  - *$q, q' inQ$*
  - *$\sigma \in \Sigma$*
  - *$g \in \mathcal{H}_k^\cap$*
  - *$f \in F(X)$*

Definition 9 (Timed Automata)
A timed automaton is a tuple $\mathcal{T} = (Q, X, \Sigma, I, R, q_0)$ consisting of:
- Q a finite set of discrete states
- X a clock domain $X = (\mathbb{R}^+)^d$ for some $d > 0$
- Σ an input alphabet (including a single environment action e)
- $I : Q \mapsto \mathcal{H}_k^\cap$ as the state invariant function
- $R \subseteq Q \times \Sigma \times \mathcal{H}_k^\cap \times F(X) \times Q$ is a set of transition relations each of the form $\langle q, \sigma, g, f, q' \rangle$ where:
  - $q, q' inQ$
  - $\sigma \in \Sigma$
  - $g \in \mathcal{H}_k^\cap$
  - $f \in F(X)$

Real Time Case

Definition 10 (*Steps and Trajectories*)

A step of (T) *is a pair of configurations* $((q,x),(q',x'))$ *such that either:*

- $q = q'$ *and for some* $t \in T, x' = x + 1t, x \in I_q$ *and* $x' \in I_q$. *In this case we say that* (q',x') *is a t-successor of* (q,x) *and that* $((q,x),(q',x'))$ *is a t-step.*
- *There is some* $r = \langle q,\sigma,g,f,q'\rangle \in R$ *such that* $x \in g$ *and* $x' = f(x)$. *In this case we say that* (q',x') *is a* σ*-successor of* (q,x) *and that* $((q,x),(q',x'))$ *is a* σ*-step*

A trajectory of $\mathcal{T}$ *is a sequence* $\beta = (q[0],x[0]),(q[1],x[1]),\dots$ *of configurations such that for every* i, $((q[i],x[i]),(q[i+1],x[i+1]))$ *is a step.*

Definition 10 (Steps and Trajectories)
A step of (T) is a pair of configurations $((q,x),(q',x'))$ such that either:
- $q = q'$ and for some $t \in T, x' = x + 1t, x \in I_q$ and $x' \in I_q$. In this case we say that (q',x') is a t-successor of (q,x) and that $((q,x),(q',x'))$ is a t-step.
- There is some $r = \langle q,\sigma,g,f,q'\rangle \in R$ such that $x \in g$ and $x' = f(x)$. In this case we say that (q',x') is a σ-successor of (q,x) and that $((q,x),(q',x'))$ is a σ-step

A trajectory of $\mathcal{T}$ is a sequence $\beta = (q[0],x[0]),(q[1],x[1]),\dots$ of configurations such that for every i, $((q[i],x[i]),(q[i+1],x[i+1]))$ is a step.

Real Time Case

Definition 11 (*Real time Controller*)

A simple real time controller is a function $C : Q \times X \mapsto \Sigma_c \cup \perp$

According to this function the controller chooses at any configuration (q, x) whether to issue some enabled transition σ or to do nothing and let time go by. We denote by $\Sigma_c^{\perp}$ the range of controller commands $\Sigma_c \cup \perp$. We also require that the controller is k-polyhedral, i.e., for every $\sigma \in \Sigma_c^{\perp}$, $C^{-1}(\sigma)$ is a k-polyhedral set.

Definition 11 (*Real time Controller*)

A simple real time controller is a function $C : Q \times X \mapsto \Sigma_c \cup \perp$

According to this function the controller chooses at any configuration (q, x) whether to issue some enabled transition σ or to do nothing and let time go by. We denote by $\Sigma_c^{\perp}$ the range of controller commands $\Sigma_c \cup \perp$. We also require that the controller is k-polyhedral, i.e., for every $\sigma \in \Sigma_c^{\perp}$, $C^{-1}(\sigma)$ is a k-polyhedral set.

Real Time Case

Definition 12 (*Controlled Trajectories*)

Given a simple controller C, a pair $((q,x),(q',x'))$ of configurations is a C-step if it is either:

- *an e − step*
- *a σ − step such that $C(q,x) = \sigma \in \Sigma_c$*
- *a t − step for some $t \in T$ such that for every t', $t' \in [0,t)$, $C(q, x + 1t') = \perp$*

A C-trajectory is a trajectory consisting of C-steps. We denote the set of C-trajectories of $\mathcal{T}$ by $L_C(\mathcal{T})$.

Definition 12 (Controlled Trajectories)
Given a simple controller C, a pair $((q,x),(q',x'))$ of configurations is a C-step if it is either:
an e − step
a σ − step such that $C(q,x) = \sigma \in \Sigma_c$
a t − step for some $t \in T$ such that for every t', $t' \in [0,t)$, $C(q, x + 1t') = \perp$
A C-trajectory is a trajectory consisting of C-steps. We denote the set of C-trajectories of $\mathcal{T}$ by $L_C(\mathcal{T})$.

Real Time Case

Definition 13 (*Real time Controller Synthesis*)

*Given a timed automaton $\mathcal{T}$ an a acceptance condition Ω, the problem **RT-Synth**$(\mathcal{T}, \Omega)$ is: Construct a real-time controller C such that $L_C(\mathcal{T}) \subseteq L(\mathcal{T}, \Omega)$*

Definition 13 (*Real time Controller Synthesis*)

*Given a timed automaton $\mathcal{T}$ an a acceptance condition Ω, the problem **RT-Synth**$(\mathcal{T}, \Omega)$ is: Construct a real-time controller C such that $L_C(\mathcal{T}) \subseteq L(\mathcal{T}, \Omega)$*

Real Time Case

In order to tackle the real time controller synthesis problem we introduce the following definitions:

Definition 14 ($(t, \sigma) - successor$)

For $t \in T$ and $\sigma \in \Sigma$, the configuration (q', x') is defined to be a $(t, \sigma) - successor$ of the configuration (q, x) if there exists an intermediate configuration $(\hat{q}, \hat{x})$ such that $(\hat{q}, \hat{x})$ is a $t - successor$ of (q, x) and (q', x') is a $\sigma - successor$ of $(\hat{q}, \hat{x})$.

Then we define a function $\delta : (Q \times X) \times (T \times \Sigma_c^{\perp}) \mapsto 2^{Q \times X}$ where $\delta((q, x), (t, \sigma))$ stands for all the possible consequences of the controller attempting to issue the command $\sigma \in \Sigma_c^{\perp}$ after waiting t time units starting at configuration (q, x)

In order to tackle the real time controller synthesis problem we introduce the following definitions:
Definition 14 ($(t, \sigma) - successor$)
For $t \in T$ and $\sigma \in \Sigma$, the configuration (q', x') is defined to be a $(t, \sigma) - successor$ of the configuration (q, x) if there exists an intermediate configuration $(\hat{q}, \hat{x})$ such that $(\hat{q}, \hat{x})$ is a $t - successor$ of (q, x) and (q', x') is a $\sigma - successor$ of $(\hat{q}, \hat{x})$.
Then we define a function $\delta : (Q \times X) \times (T \times \Sigma_c^{\perp}) \mapsto 2^{Q \times X}$ where $\delta((q, x), (t, \sigma))$ stands for all the possible consequences of the controller attempting to issue the command $\sigma \in \Sigma_c^{\perp}$ after waiting t time units starting at configuration (q, x)

Note that this covers the case of (q', x') being simply a $\sigma - successor$ of (q, x) by viewing it as a $(0, \sigma) - successor$ of (q, x).

Real Time Case

Definition 15 (*Extended Transition Function*)

For every $t \in T$ *and* $\sigma in \Sigma_c$*, the set* $\delta((q,x),(t,\sigma))$ *consists of all the configurations* (q',x') *such that:*

- (q',x') *is a* (t,σ) *– successor of* (q,x)
- (q',x') *is a* (t,e) *– successor of* (q,x) *for some* $t' \in [0,t]$

Definition 15 (Extended Transition Function)
For every $t \in T$ and $\sigma in \Sigma_c$, the set $\delta((q,x),(t,\sigma))$ consists of all the configurations (q',x') such that:
- (q',x') is a (t,σ) – successor of (q,x)
- (q',x') is a (t,e) – successor of (q,x) for some $t' \in [0,t]$

This definition covers successor configurations that are obtained in one of two possible ways:
some configurations result from the plant waiting patiently at state q for t time units, and then taking a σ-labeled transition according to the controller recommendation,
the second possibility is of configurations obtained by taking an environment transition at any time $t' \leq t$
This is in fact the crucial new feature of real-time games - there are no turns and the adversary need not wait for the player's next move.

Real Time Case

Definition 16 (*Controllable Predecessors*)

The controllable predecessors function $\pi : 2^Q \times 2^X \mapsto 2^Q \times 2^X$ *is defined for every* $K \subseteq Q \times X$ *by*

$$\pi(K) = \{(q,x) : \exists t \in T \exists \sigma \in \Sigma_c \;\; \delta((q,x),(t,\sigma)) \subseteq K\}$$

Definition 16 (*Controllable Predecessors*)

The controllable predecessors function $\pi : 2^Q \times 2^X \mapsto 2^Q \times 2^X$ *is defined for every* $K \subseteq Q \times X$ *by*

$$\pi(K) = \{(q,x) : \exists t \in T \exists \sigma \in \Sigma_c \;\; \delta((q,x),(t,\sigma)) \subseteq K\}$$

As in the discrete case, we define a predecessor function that indicates the configurations from which the controller can force the automaton into a given set of configurations

Real Time Case

Theorem 2 (*Control Synthesis for Timed ystems*)

Given a timed automaton $\mathcal{T}$ and an acceptance condition

$$\{(F,\Box),(F,\Diamond),(F,\Diamond\Box),(F,\Box\Diamond),(\mathcal{F},\mathcal{R}_n)\}$$

*the problem **RT-Synth**$(\mathcal{T},\Omega)$ is solvable*

Theorem 2 (*Control Synthesis for Timed ystems*)

Given a timed automaton $\mathcal{T}$ and an acceptance condition

$$\{(F,\Box),(F,\Diamond),(F,\Diamond\Box),(F,\Box\Diamond),(\mathcal{F},\mathcal{R}_n)\}$$

*the problem **RT-Synth**$(\mathcal{T},\Omega)$ is solvable*

Real Time Case

Sketch of Proof

We have just shown that $2^Q \times \mathcal{H}_k^*$ is closed under π.

Any of the iterative processes for the fixed point equations (1) – (5) starts with an element of $2^Q \times \mathcal{H}_k^*$.

For example, the iteration for $\Diamond$ starts with $W_0 = Q \times F$.

Each iteration consists of applying Boolean set-theoretic operations and the predecessor operation, which implies that every W_i is also an element of $2^Q \times \mathcal{H}_k^*$ - a finite set.

Thus, by monotonicity, a fixed point is eventually reached.

Sketch of Proof

We have just shown that $2^Q \times \mathcal{H}_k^*$ is closed under π.

Any of the iterative processes for the fixed point equations (1) – (5) starts with an element of $2^Q \times \mathcal{H}_k^*$.

For example, the iteration for $\Diamond$ starts with $W_0 = Q \times F$.

Each iteration consists of applying Boolean set-theoretic operations and the predecessor operation, which implies that every W_i is also an element of $2^Q \times \mathcal{H}_k^*$ - a finite set.

Thus, by monotonicity, a fixed point is eventually reached.

The strategy is extracted in a similar manner as in the discrete case. When ever a configuration (q, x) is added to W, it is due to one or more pairs of the form $([t_1, t_2], \sigma)$ indicating that within any $t, t_1 < t < t_2$ issuing σ after waiting t will lead to a winning position. Hence by letting $C(q, x) = \perp$ when $t_1 > 0$ and $C(q, x) = \sigma$ when $t_1 = 0$ we obtain a $k-$polyhedral controller.

Citations