Timed Games

Patrick Jungblut

July 3, 2008

Abstract. In this paper we will show how to solve Timed Games as a way to synthesize controller for time critical applications. Therefore we will introduce Timed Game Automata and have a look at 4 different types of winning conditions as they are described in [MPS '95]. We will then focus on reachability games and have a closer look at an on-the-fly algorithm to solve reachability games which is presented in [CDFLL '05].

1 Introduction

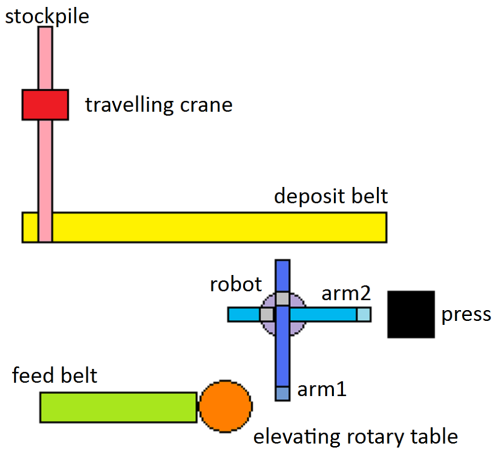

Figure 1: A production cell

Consider a production cell as shown in figure 1. We want to ensure that a controller for such a production cell can enforce wantet behavior and avoid unwantet behavior of the production cell. But a controller can just activate some actions the different components provide, like starting the press to form some plate or to pick up a plate with a robot arm. The controller can not influence the time it takes to press a plate or to move the robot arm from one position to anothe. This is

uncontrollable environment behavior. To model such conditions we must have a possibility to model time for the system.

Having such a model for the system we want to construct a controller which can ensure correct behavior, no matter how the environment behaves.

2 Timed Game Automata and Timed Games

2.1 Syntax of Timed Game Automata

A Timed Game Automaton TGA is a tuple (L, l_0, Inv, Act, X, T) where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- Inv is a function, which assigns to each location its invariant.
- $Act = Act_c \cup Act_u$ is a set of actions
- X is a set of real-valued clocks
- $T \subseteq (L \times Act \times g \times Reset \times L)$ is a set of transitions, where
  - g is a clock constraint built by: $g = x \circ c \mid x_1 - x_2 \circ c \mid g_1 \wedge g_2$
    where $x, x_1, x_2 \in X$ are clocks, $c \in \mathbb{N}$ some constant, $\circ \in \{<, \leq, =, \geq, >\}$ and g_1, g_2 clock constraints
  - $Reset \subseteq X$ is the set of clocks to reset

2.2 Playing a Timed Game (Semantics)

A Timed Game is a 2 player game taking place on a Timed Game Automaton. Player $Environment$ controls Act_u and Player $Controller$ Act_c. At a location $l \in L$ at a clock-valuation $\vec{t} \in \mathbb{R}^X_{\geq 0}$ a player P has two possibilities:

1. Using a transition $t = (l, \alpha, g, R, l')$, if $\vec{t} \models g$, $\alpha \in Act_P$, and $\vec{t}[R] \models Inv(l')$, where $\vec{t}[R]$ is the clock-valuation resulting from $\vec{t}$ by setting all clocks in R to 0.
2. Waiting

$Environment$ has priority, i.e. if $Environment$ and $Controller$ want to use a transition at the same time $Environment$ will be able to make his move. This leads us to the following statespace $S \subseteq L \times \mathbb{R}^X_{\geq 0}$.

2.3 Strategy

A memoryless (state-based) strategy $f_P : S = L \times \mathbb{R}^X_{\geq 0} \rightarrow Act_P \cup \{\lambda\}$ for a player P is a partial function s.t.

1. $f_P(s) = a$ for some $s \in S$ and $a \in Act_P$, if P has to use a
2. $f_P(s) = \lambda$, if P has to let time pass

A strategy f_P is called winning, iff P always wins the Timed Game following f_P. As soon as we know a winning strategy f_c for $Controller$ we can build a correct $Controller$ according to f_c.

2.4 Winning Conditions

[MPS '95] presents 4 different winning conditions for a Timed Game: Let $G \subseteq L$ be a set of goal locations.

1. Controller: $\Diamond G$: Controller wins if he can enforce to reach G
2. Controller: $\Box G$: Controller wins if he can enforce to not leave G
3. Controller: $\Diamond\Box G$: Controller wins if he can enforce to finally stay in G
4. Controller: $\Box\Diamond G$: Controller wins if he can enforce to reach G infinitely often

We first focus on solving Timed Games with winning condition $\Diamond G$ as we can also solve $\Box G$ in the same way, by changing roles of $Environment$ and $Controller$ and solving $\Diamond\overline{G}$ for $Environment$. If there is no strategy for $Environment$ to reach $\overline{G}$ $Controller$ can enforce to stay in G.

3 Solving Timed Games

3.1 Backward fixpoint iteration

[MPS '95] provides the following fixpoint iteration to compute the set win of states from which we can enforce to reach G:

1. $win_0 :=$ goal $\times \mathbb{R}^X_{\geq 0}$
2. $win_{i+1} := win_i \cup Pre_{enf}(win_i)$

The set win is constructed by iteratively adding those states to win from which we can enforce to reach the current win. If after reaching the fixpoint $(l_0, \vec{0}) \in win$, we have found a solution. Which can be extractet by a search for a path in win from $(l_0, \vec{0})$ to some state in G.

3.1.1 Computation of $Pre_{enf}(win)$

A state $s = (l, x) \in S$ is in $Pre_{enf}(win)$ iff:

- $\exists s' = (l, x') \in win$ for some $x' > x$ and $\forall x \leq x'' \leq x'$ holds $\nexists t = (l, \alpha, g, R, l') \in T$ s.t. $\alpha \in Act_u$ and $x \models g$ and $(l', x''[R]) \notin win$ or
- $\exists s' = (l', x') \in win$ s.t. $\exists x'' > x$ and $t = (l, \alpha, g, R, l') \in T$ s.t. $x' = x''[R]$ and $\forall x \leq x''' \leq x''$ holds $\nexists t' = (l, \alpha', g', R', l'') \in T$ s.t. $\alpha' \in Act_u$ and $x \models g$ and $(l'', x'''[R]) \notin win$

Note: Let x, y be clock-valuations, we say $x \leq y$ if $\exists \delta \in \mathbb{R}_{\geq 0}$ s.t. $y = x + \delta\vec{1}$

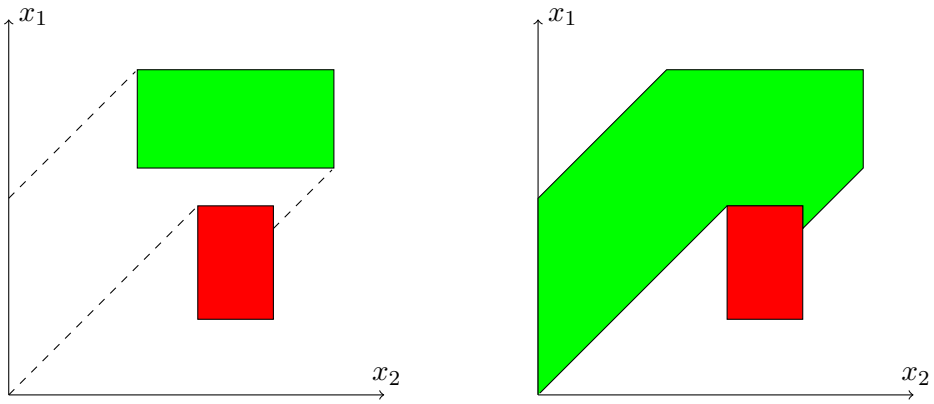

Figure 2: Pre_{enf} on a time plane

In Figure 2 on the left the green box represents a current win for some location, while the red box represents the part of time where uncontrollable transitions can lead out of win. On the right, we see in green the enlarged part of win.

3.2 Clock Zones

We cannot easily handle infinite the infinite state space. [Alur '99] provides Clock Zones which enable us to represent the infinite state space by a finite symbolic state space: A Clock Zone is a convex polyhedron $z \in \mathbb{R}^X_{\geq 0}$ defined by a formula $\bigwedge x_i \circ c_i \wedge \bigwedge x_i - x_j \circ c_{ij}$, where $x_i, x_j \in X$ are clocks and $c_i, c_{ij} \in \mathbb{N} \cup \{+\infty\}$ and $\circ \in \{<, \leq, \geq, >\}$. There is an efficent matrix based data structure to represent a Clock Zone which is widely used in practival applications for Timed Automata and Timed Games: the DBM. A finite union of Clock Zones is called a Federation. Federations are not necessarily convex. In [CDFLL '05] Federations are used to compute the $Pre_{enf}(win)$ operation.

3.3 On-the-fly Timed Game solving

The backward fixpoint iteration is often not applicable in practice. Each single step of the iteration is expensive and non reachability of goal state will not be noticed until fixpoint is reached. We also have to work on the whole statespace which can be huge, which often leads to incomputability of a solution because of the lack of memory. Thus [CDFLL '05] provides an algorithm which computes the state space on-the-fly, by starting in the initial location and performing an reachability analysis cascaded with the backward propagation of winning information.

- Initialization:
  1. Start in the initial state
  2. Feed a waiting queue q with the outgoing transitions of the initial state
- Loop:
  1. As long as q is not empty: take a transition t from q and analyse target state s' of t:
  2. If we meet s' for the first time:
     - s' is the goal state? If yes, add t to q
     - Add all outgoing transitions of s' to q
  3. If we already met s' before:
     - Propagate winning information from s' back to the source s of t
     - If the winning information of s changes, add in-transitions to s to q

3.4 Two remaining winning conditions

[MPS '95] also provides fixpoint iterations for the two remaining winning conditions:

3.4.1 Controller: $\Diamond\Box G$

- $win_0 := \emptyset$, $i := 0$
- **do**
  - $help_0 := S$, $j := 0$
  - **do** $(help_{j+1} := Pre_{enf}(help_j) \cap (G \cup Pre_{enf}(win_i)), j++)$
  - **while** $help_{j+1} \neq help_j$
  - $win_{i+1} := help_j$, $i++$
- **while** $win_{i+1} \neq win_i$

3.4.2 Controller: $\Box\Diamond G$

- $win_0 := S, i := 0$
- **do**
  - $help_0 := \emptyset, j := 0$
  - **do** $(help_{j+1} := Pre_{enf}(help_j) \cup (G \cap Pre_{enf}(win_i)), j++)$
  - **while** $help_{j+1} \neq help_j$
  - $win_{i+1} := help_j, i++$
- **while** $win_{i+1} \neq win_i$

4 Summary

We have seen how we can synthesize controller by solving Timed Games. Therefore we had a closer look at Timed Game Automata, and their syntax and semantics, and provided Clock Zones as an efficient construct to represent the infinite timed state space by a finite symbolic state space. We showed how to solve Timed Games with different winning conditions, focusing on reachability games. For reachability games we presented an efficient on-the-fly algorithm from [CDFLL '05].

5 References

1. F. Cassez, A. David, E. Fleury, K.G. Larsen, D. Lime. Efficient On-The-Fly Algorithms for the Analysis of Timed Games. In CONCUR'05 2005. [CDFLL '05]
2. O. Maler, A. Pnueli and Joseph Silfakis. On The Synthesis of Discrete Controllers for Timed Systems (An Extended Abstract). [MPS '95]
3. R. Alur. Timed Automata. 11th International Conference on Computer-Aided Verification, LNCS pages 8-22, 1999. [Alur '99]