BRICS RS-04-4 P. Bouyer & F. Cassez & E. Fleury & K. G. Larsen: Optimal Strategies in PTGA

BRICS

Basic Research in Computer Science

Optimal Strategies in Priced Timed Game Automata

Patricia Bouyer
Franck Cassez
Emmanuel Fleury
Kim G. Larsen

BRICS Report Series **RS-04-4**

ISSN 0909-0878 **February 2004**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/04/4/`

Optimal Strategies in Priced Timed Game Automata

Patricia Bouyer
LSV – UMR 8643 CNRS & ENS de Cachan, France
Email: <bouyer@lsv.ens-cachan.fr>

Franck Cassez
IRCCyN – UMR CNRS 6597, Nantes, France
Email: <Franck.Cassez@irccyn.ec-nantes.fr>

Emmanuel Fleury
BRICS, Computer Science Dept., Aalborg University, Denmark
Email: <fleury@cs.auc.dk>

Kim G. Larsen
BRICS, Computer Science Dept., Aalborg University, Denmark
Email: <kgl@cs.auc.dk>

February 18, 2004

Abstract

Priced timed (game) automata extends timed (game) automata with costs on both locations and transitions. In this paper we focus on reachability games for priced timed game automata and prove that the optimal cost for winning such a game is computable under conditions concerning the non-zenoness of cost. Under stronger conditions (strictness of constraints) we prove in addition that it is decidable whether there is an optimal strategy in which case an optimal strategy can be computed. Our results extend previous decidability result which requires the underlying game automata to be acyclic. Finally, our results are encoded in a first prototype in HyTech which is applied on a small case-study.

Keywords: Optimality, Control, Controller Synthesis, Strategy, Priced Timed Game.

1 Introduction

In recent years the application of model-checking techniques to scheduling problems has become an established line of research. Static scheduling problems with timing constraints may often be formulated as reachability problems or timed automata, viz. as the possibility of reaching a given winning state. Real-time model checking tools such as KRONOS and UPPAAL have been applied on a number of industrial and benchmark scheduling problems [Feh99, HLP00, NY01, BMF02, Abd02, Lar03].

Often the scheduling strategy needs to take into account uncertainty with respect to the behavior of an environmental context. In such situations the scheduling problem becomes a dynamic (timed) game between the controller and the environment, where the objective for the controller is to find a *dynamic* strategy that will guarantee the game to end in a winning state [MPS95, AMPS98, DAHM01].

Optimality of schedules may be obtained within the framework of timed automata by associating with each run a performance measure. Thus it is possible to compare runs and search for the optimal run from an initial configuration to a final (winning) target. The most obvious performance measure for timed automata is clearly that of time itself. Time-optimality for timed automata was first considered in [CY92] and proved computable in [NTY00]. The related problem of synthesizing time-optimal winning strategies for timed game automata was shown computable in [AM99].

More recently, the ability to consider more general performance measures has been given. Priced extensions of timed automata have been introduced where a cost c is associated with each location ℓ giving the cost of a unit of time spent in ℓ. In [ACH93] cost-bound reachability has been shown decidable. [BFH$^+$01] and [ALTP01] independently solve the cost-optimal reachability problem for priced timed automata [1]. Efficient incorporation in UPPAAL is provided by use of so-called priced zones as a main data structure [LBB$^+$01]. In [RLS04] the implementation of cost-optimal reachability is improved considerably by exploiting duality between linear programming problems over zones with min-cost flow problems. More recently [BBL04], the problem of computing optimal *infinite* schedules (in terms of minimal limit-ratios) is solved for the model of priced timed automata.

In this paper we combine the notions of game and price and solve the problem of cost-optimal winning strategies for priced timed game automata under conditions concerning the strictness of constraints and non-zenoness of cost. Our results extend the previous result of [LTMM02] which requires the underlying game automata to be acyclic. The existing results mentioned above related to timed game automata and priced timed automata respectively, are all based on various extensions of the so-called classical region- and zone-techniques. In the combined setting the solution is obtained in a radically different way.

Consider the priced timed game automata in Fig. 1. Here the cost-rates in locations ℓ_0, ℓ_2 and ℓ_3 are 5, 10 and 1 respectively. In ℓ_1 the environment may choose to move to either ℓ_2 or ℓ_3 (dashed arrows are uncontrollable). However, due to the invariant $y = 0$ this choice must be made instantaneous. Obviously, once ℓ_2 or ℓ_3 has been reached the optimal strategy for the controller is to move to Win immediately. The crucial (and only remaining) question is how long the controller should wait in ℓ_0 before taking the transition to ℓ_1. Obviously, in order for the controller to win this duration must be no more than two time units. However, what is the optimal choice for the duration in the sense that the overall cost of reaching Win is minimal? Denote by t the chosen delay in ℓ_0. Then $5t + 10(2 - t) + 1$ is the minimal cost through ℓ_2 and $5t + (2 - t) + 7$ is the minimal cost through ℓ_3. As the environment chooses between these two transitions the best choice for the controller is to delay $t \leq 2$ such

[1] In [BFH$^+$01] the name *linearly priced timed automata* is used and in [ALTP01] the same model is named *weighted timed automata*.

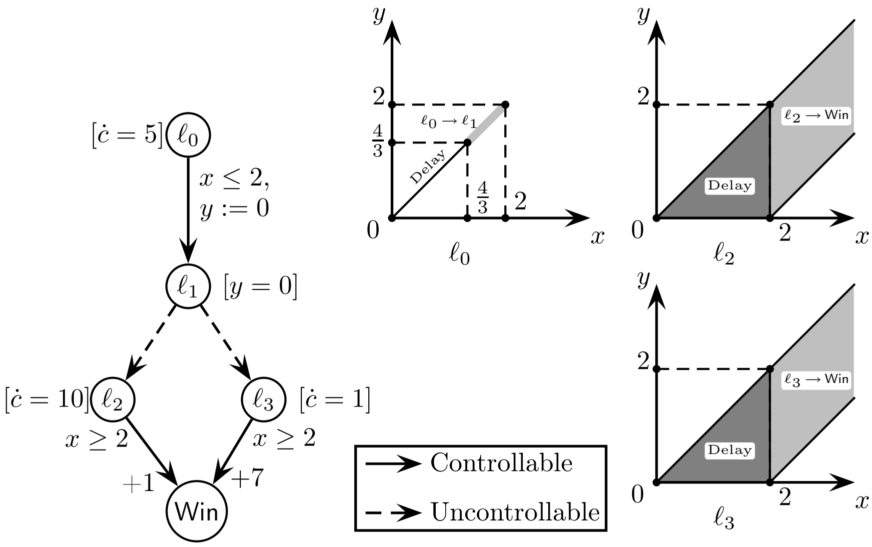

Figure 1: *A small Priced Time Game Automata. Optimal strategy has cost* $14\frac{1}{3}$.

that $\max(21-5t, 9+4t)$ is minimum, which is $t = \frac{4}{3}$ giving a minimal cost of $14\frac{1}{3}$. In Fig. 1 we illustrate the optimal strategy for all states reachable from the initial state.

The outline of the paper is as follows: in section 2 we recall some basics about *reachability timed games*. Section 3 introduces *priced timed games (PTG)* where we give a run-based definition of optimality. We also relate our run-based definition of optimality to the recursive one previously given in [LTMM02]. Section 4 is the core of the paper where we show how to compute the optimal cost of a PTG and optimal strategies. Finally section 5 reports on preliminary implementation experiments of our work in HyTech.

2 Reachability Timed Games (RTG)

In this paper we focus on *reachability games*, where the control objective is to enforce that the system eventually evolves into a particular state. It is classical in the literature to define *reachability timed games (RTG)* [MPS95, AMPS98, DAHM01] to model control problems. In this section we recall some known general results about RTG and we finally give an additional result about the controller (strategy) synthesis for RTG. Indeed controller synthesis is well defined for *safety* games but some additional care is needed for RTG as shown later in the section.

2.1 Timed Transition Systems and Games

Definition 1 (Timed Transition Systems). *A* timed transition system *(TTS for short) is a tuple* $S = (Q, Q_0, \mathsf{Act}, \longrightarrow)$ *with:*

- Q *is a set of states*
- $Q_0 \subseteq Q$ *is the set of initial states*

- Act *is a finite set of actions, disjoint from* $\mathbb{R}_{\geq 0}$. *We denote* $\Sigma = \mathsf{Act} \cup \mathbb{R}_{\geq 0}$
- $\longrightarrow \subseteq Q \times \Sigma \times Q$ *is a set of edges. If* $(q, e, q') \in \longrightarrow$, *we also write* $q \xrightarrow{e} q'$.

We make the following common assumptions about TTSs:

- 0-DELAY: $q \xrightarrow{0} q'$ if and only if $q = q'$,
- ADDITIVITY: if $q \xrightarrow{d} q'$ and $q' \xrightarrow{d'} q''$ with $d,\ d' \in \mathbb{R}_{\geq 0}$, then $q \xrightarrow{d+d'} q''$,
- CONTINUITY: if $q \xrightarrow{d} q'$, then for every d' and d'' in $\mathbb{R}_{\geq 0}$ such that $d = d' + d''$, there exists q'' such that $q \xrightarrow{d'} q'' \xrightarrow{d''} q'$,
- DETERMINISM[2]: if $q \xrightarrow{e} q'$ and $q \xrightarrow{e} q''$ with $e \in \Sigma$, then $q' = q''$.

A *run* in S is a finite or infinite sequence $\rho = q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} q_n \cdots$. $\mathsf{States}(\rho) = \{q_0, q_1, \cdots, q_n, \cdots\}$ is the set of states encountered on ρ. We denote by $\mathit{first}(\rho) = q_0$ and $\mathit{last}(\rho) = q_n$ if ρ is finite and ends in q_n. $\mathsf{Runs}(q, S)$ is the set of (finite and infinite) runs in S starting from q. The set of runs of S is $\mathsf{Runs}(S) = \cup_{q \in Q} \mathsf{Runs}(q, S)$. We use $q \xrightarrow{e}$ as a shorthand for "$\exists q'$ s.t. $q \xrightarrow{e} q'$" and extends this notation to finite runs $\rho \xrightarrow{e}$ whenever $\mathit{last}(\rho) \xrightarrow{e}$.

Definition 2 (Timed Games (Adapted from [DAHM01])). *A* timed game *(TG for short)* $G = (Q, Q_0, \mathsf{Act}, \longrightarrow)$ *is a TTS such that* Act *is partitioned into* controllable *actions* Act_c *and* uncontrollable actions Act_u.

2.2 Strategies, Reachability Games

A strategy [MPS95] is a function that during the cause of the game constantly gives information as to what the controller should do in order to win the game. In a given situation the strategy could suggest the controller to either i) "do a particular controllable action" or ii) "delay for some specific amount of time" which will be denoted by the special symbol λ. For instance if one wants to delay until some clock value x reaches $\frac{4}{3}$ (as would be a good strategy in the location ℓ_0 of Fig. 1) then the strategy would be: for $x < \frac{4}{3}$ do λ and for $x = \frac{4}{3}$ do the control action from ℓ_0 to ℓ_1.

Definition 3 (Strategy). *Let* $G = (Q, Q_0, \mathsf{Act}, \longrightarrow)$ *be a TG. A* strategy f *over* G *is a partial function from* $\mathsf{Runs}(G)$ *to* $\mathsf{Act}_c \cup \{\lambda\}$.

We denote $\mathsf{Strat}(G)$ the set of strategies over G. A strategy f is *state-based* whenever $\forall \rho, \rho' \in \mathsf{Runs}(G)$, $\mathit{last}(\rho) = \mathit{last}(\rho')$ implies that $f(\rho) = f(\rho')$. State-based strategies are also called *memory-free* strategies in game theory [Tho95, DAHM01]. The possible runs that may be realized when the controller follows a particular strategy is defined by the following notion of $\mathsf{Outcome}$ ([DAHM01]):

Definition 4 (Outcome). *Let* $G = (Q, Q_0, \mathsf{Act}, \longrightarrow)$ *be a TG and* f *a strategy over* G. *The* outcome $\mathsf{Outcome}(q, f)$ *of* f *from* q *in* G *is the subset of* $\mathsf{Runs}(q, G)$ *defined inductively by:*

- $q \in \mathsf{Outcome}(q, f)$,

[2]Determinism is not essential in our work but it simplifies proofs in the sequel.

- *if* $\rho \in \mathsf{Outcome}(q, f)$ *then* $\rho' = \rho \xrightarrow{e} q' \in \mathsf{Outcome}(q, f)$ *if* $\rho' \in \mathsf{Runs}(q, G)$ *and one of the following three conditions hold:*

  1. $e \in \mathsf{Act}_u$,
  2. $e \in \mathsf{Act}_c$ *and* $e = f(\rho)$,
  3. $e \in \mathbb{R}_{\geq 0}$ *and* $\forall 0 \leq e' < e, \exists q' \in Q$ *s.t.* $last(\rho) \xrightarrow{e'} q'' \wedge f(\rho \xrightarrow{e'} q'') = \lambda$.

- *an infinite run* ρ *is in* $\in \mathsf{Outcome}(q, f)$ *if all the finite prefixes of* ρ *are in* $\mathsf{Outcome}(q, f)$.

A strategy is *realizable*, whenever for all $\rho \in \mathsf{Outcome}(q, f)$ such that f is defined on ρ and $f(\rho) = \lambda$, there exists some $\delta > 0$ such that for all $0 \leq t < \delta$, there exist q' with $\rho \xrightarrow{t} q' \in \mathsf{Outcome}(q, f)$ and $f(\rho \xrightarrow{t} q') = \lambda$. Strategies which are not realizable are not interesting because they generate empty set of outcomes. Note that realizability is a weaker notion than the one of implementability considered in [CHR02, DDR04].

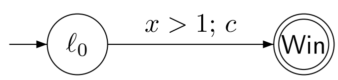

Figure 2: A timed game automaton

Consider the TTS induced by the timed automaton of Fig. 2. The game is to enforce state Win. The most natural strategy f would be to do a c when $x > 1$ and to wait until x reaches a value greater than 1. Formally this yields $f(\ell_0, x \leq 1) = \lambda$ and $f(\ell_0, x > 1) = c$. This strategy is not realizable. In the sequel, we build a strategy which is $f(\ell_0, x < 2) = \lambda$ and $f(\ell_0, x \geq 2) = c$. Now assume the constraint on the transition is $1 < x \leq 2$. In this case we start with the following strategy (not realizable): $f(\ell_0, x \leq 1) = \lambda$ and $f(\ell_0, 1 < x \leq 2) = c$. To make it realizable we will take the first half of $1 < x \leq 2$ and have a delay action on it i.e. $f(\ell_0, x < \frac{3}{2}) = \lambda$ and $f(\ell_0, \frac{3}{2} \leq x \leq 2) = c$.

In the following, we will restrict our attention to realizable strategies and simply refer to them as strategies.

Definition 5 (Reachability Timed Games (RTG)). *A* reachability timed game $G = (Q, Q_0, \mathsf{Win}, \mathsf{Act}, \longrightarrow)$ *is a timed game* $(Q, Q_0, \mathsf{Act}, \longrightarrow)$ *with a distinguished set of* winning states $\mathsf{Win} \subseteq Q$ *such that for all* $q \in \mathsf{Win}$, $q \xrightarrow{e} q'$ *implies* $q' \in \mathsf{Win}$.

If G is a RTG, a run ρ is a *winning run* if $\mathsf{States}(\rho) \cap \mathsf{Win} \neq \emptyset$. We denote $\mathsf{WinRuns}(q, G)$ the set of winning runs in G from q.

In the literature one can find (at least) two definitions of the meaning of uncontrollable actions: i) in [MPS95, AMPS98] uncontrollable actions can be used to win the game whereas ii) in [LTMM02] they cannot help to win the game. As an example, consider the game of Fig. 3. In case i) u is bound to happen before x reaches 1 and in this case we win the game from ℓ_0. In case ii) u cannot be forced to happen and thus we cannot win the game.

We follow the framework used by La Torre *et al* in [LTMM02] where uncontrollable actions cannot help to win. This choice is made for the sake of simplicity (mainly for the proof of theorem 3.) However, we can handle the semantics of [MPS95] (case i) as well but the proofs are more involved[3].

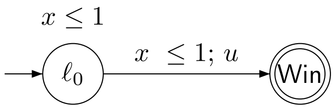

Figure 3: Winning or not winning?

We now formalize the previous notions. A *maximal run* ρ is either an infinite run or a finite run that satisfies: $\forall t \ \geq 0$, $\rho \xrightarrow{t} q' \xrightarrow{a}$ implies $a \in \mathsf{Act}_u$, thus the next discrete actions from $last(\rho)$, if any, are uncontrollable actions. A strategy f is *winning* from q if all maximal runs in $\mathsf{Outcome}(q, f)$ are in $\mathsf{WinRuns}(q, G)$. Note that f must be realizable. A state q in a RTG G is *winning* if there exists a winning strategy f from q in G. We denote by $\mathcal{W}(G)$ the set of winning states in G. We note $\mathsf{WinStrat}(q, G)$ the set of winning (and realizable) strategies from q over G.

2.3 Computation of Winning Strategies

In this section we summarize previous results obtained for particular classes of RTG: Linear Hybrid Games (LHG). Due to lack of space we will not define this model here but refer to [Hen96] for details.

The computation of the winning states is based on the definition of a *controllable predecessors* operator [MPS95, DAHM01]. Let $G = (Q, Q_0, \mathsf{Win}, \mathsf{Act}, \longrightarrow)$ be a RTG. For a subset $X \subseteq Q$ and $a \in \mathsf{Act}$ we define $\mathsf{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q', q' \ \in X\}$. Now the controllable and uncontrollable discrete predecessors of X are defined by $\mathsf{cPred}(X) = \bigcup_{c \in \mathsf{Act}_c} \mathsf{Pred}^c(X)$, respectively $\mathsf{uPred}(X) = \bigcup_{u \in \mathsf{Act}_u} \mathsf{Pred}^u(X)$. We also need a notion of *safe* timed predecessors of a set X w.r.t. a set Y. Intuitively a state q is in $\mathsf{Pred}_t(X, Y)$ if from q we can reach $q' \in X$ by time elapsing and along the path from q to q' we avoid Y. Formally this is defined by:

$$\mathsf{Pred}_t(X, Y) = \{q \in Q \mid \exists \delta \in \mathbb{R}_{\geq 0}, q \xrightarrow{\delta} q',\ q' \in X \text{ and } Post_{[0,\delta]}(q) \subseteq \overline{Y}\}$$

where $Post_{[0,\delta]}(q) = \{q' \in Q \mid \exists t \in [0, \delta] \text{ s.t. } q \xrightarrow{t} q'\}$. Now we are able to define the *controllable predecessors* operator π as follows:

$$\pi(X) = \mathsf{Pred}_t\left(X \cup \mathsf{cPred}(X), \mathsf{uPred}(\overline{X})\right) \tag{1}$$

Note that this definition of π captures the choice that uncontrollable actions cannot be used to win. We denote by $\mathsf{CompWin}$ the semi-algorithm which computes the least fixed point of $\lambda X.\{\mathsf{Win}\} \cup \pi(X)$ as the limit of an increasing sequence of sets of states (starting with the initial set Win). If G is a LHG, the result of the computation $\mu X.\{\mathsf{Win}\} \cup \pi(X)$ is denoted $\mathsf{CompWin}(G)$.

Theorem 1 (Symbolic Algorithm for LHG [HHM99, DAHM01]). $\mathcal{W}(G) = \mathsf{CompWin}(G)$ *for a LHG G and hence* $\mathsf{CompWin}$ *is a symbolic semi-algorithm for computing the winning states of a LHG. Moreover* $\mathsf{CompWin}$ *terminates for the subclass of Initialized Rectangular Games [HHM99].*

As for controller synthesis the previous algorithms allow us to compute the winning states of a game but the strategy synthesis extraction is not much detailed.

[3] The definition of π later on must be patched as well as the definition of the O function in Def. 10. Theorem 2 still holds for this case as it only depends on the winning set of states.

Although this is not the point of the paper we provide a symbolic algorithm (assuming time determinism) that synthesizes realizable strategies and thus have the following theorem:

Theorem 2 (Synthesis of Realizable Strategies). *Let G be a time deterministic LHG. If the semi-algorithm* $\mathsf{CompWin}$ *terminates for G, then we can compute a polyhedral*[4] *strategy which is: winning (and realizable) in each state of $\mathcal{W}(G)$ and state-based.*

Proof. The proof of this theorem is quite technical. A *state predicate* for G is a finite union of regions of the form (ℓ, R) where ℓ is a location of the LHG and R a convex polyhedron. We assume that Win is a state predicate. We note $W_0 = \mathsf{Win}$ and for each i, $W_{i+1} = \pi(W_i)$ (it contains W_i due to Pred_t). As the symbolic versions of $\mathsf{cPred}, \mathsf{uPred}, \mathsf{Pred}_t$ compute state predicates (if applied to a state predicate), π iteratively computes state predicates and all the W_i are state predicates.

As $\mathsf{CompWin}$ terminates we denote $W_\star = \mathsf{CompWin}(G)$ and $W_\star$ is the least fixed point of $\lambda X.\{\mathsf{Win}\} \cup \pi(X)$. We define the mapping $\iota : W_\star \longrightarrow \mathbb{N}$ by: $\iota(q) = \min\{k \mid q \in W_k\}$. Notice that if $q \in W_{i+1} \setminus W_i$ then $\iota(q) = i + 1$.

Let $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ be a finite partition of $W_\star$ viewed as a state predicate s.t. $(l, P), (l, P') \in \mathcal{P}$ implies $P \cap P' = \emptyset$.We say that a mapping $h : W_\star \longrightarrow Y$ is $\mathcal{P}$-definable if $q, q' \in P_i$ implies $h(q) = h(q')$.

Theorem 2 is a consequence of the following proposition.

Proposition 1. *If $\mathcal{P}_i$ is a finite partition of W_i viewed as a state predicate, if ι is $\mathcal{P}_i$-definable, if f_i is a $\mathcal{P}_i$-definable state-based realizable winning strategy on W_i, then we can build a $\mathcal{P}_{i+1}$-definable state-based realizable winning strategy f_{i+1} on W_{i+1} s.t. i) $\mathcal{P}_{i+1}$ is a finite partition of W_{i+1} that refines $\mathcal{P}_i$ on W_i and ii) ι is $\mathcal{P}_{i+1}$-definable.*

f_0 needs not to be defined as we have already won the game.

Proof. First we assume the set of controllable events in Act_c is totally ordered by $\sqsubseteq$. Secondly remark that π can be rewritten as $\pi(X) = \mathsf{Pred}_t(X, \mathsf{uPred}(\overline{X})) \cup (\mathsf{cPred}(X) \setminus \mathsf{uPred}(\overline{X}))$. This does not affect the result of $\mathsf{CompWin}$, but just computes intermediary sets of states W_i that are a bit different. We use this version of π in this proof. Let $\mathcal{P}_i = \{P_1, P_2, \cdots, P_n\}$ be the finite partition of W_i viewed as a state predicate (in particular we assume each P_k is convex). For $X \in \mathcal{P}_i$ denote $A(X) = \mathsf{Pred}_t(X, \mathsf{uPred}(\overline{W_i}))$ and $B_c(X) = \mathsf{Pred}^c(X) \setminus \mathsf{uPred}(\overline{W_i})$ and $B(X) = \bigcup_{c \in \mathsf{Act}_c} B_c(X)$. Now $\pi(W_i) = \bigcup_{P_i \in \mathcal{P}_i} A(P_i) \cup B(P_i)$. Let $\mathcal{P}$ be the (finite) partition of W_{i+1} defined by:

1. $\mathcal{P}_i$ (for W_i); on W_i define f_{i+1} to be f_i.
2. on $B(P_i)$ define the following partition induced by $\sqsubseteq$: let $\alpha_{i,j} = B(P_i) \cap \mathsf{Pred}^{c_j}(P_i)$ with c_j a controllable event. As the set of events is totally ordered we can order the nonempty state predicates $\alpha_{i,j}$ accordingly: $\alpha_{i,j} \sqsubseteq \alpha_{i,k}$ if $c_j \sqsubseteq c_k$. Thus we have an ordered set of state predicates $\alpha_{i,j}$. Assume those sets are ordered from n (larger one) to 1 (note they must be at least one such set if $B(P_i) \neq \emptyset$). Now we partition $B(P_i)$ into $u_{i,n} = B(P_i) \cap \alpha_{i,n}$, and

[4] A strategy f is polyhedral if for all $a \in \mathsf{Act}_c \cup \{\lambda\}$, $f^{-1}(a)$ is a finite union of convex polyhedra for each location of the LHG.

$u_{i,k-1} = (B(P_i) \setminus u_{i,k}) \cap \alpha_{i,k-1}$ for $2 \leq k \leq n$. On each non empty set $u_{i,k}$ we define $f_{i+1}(q) = c$ if c is the controllable event that corresponds to $u_{i,k}$.

3. on the set $V_{i+1} = (W_{i+1} \setminus W_i) \setminus B(P_i)$ (note that $q \in V_{i+1}$ implies $q \in A(P_j)$ for some j) define $f_{i+1}(q) = \lambda$.

$\mathcal{P}$ is a finite partition of W_{i+1} and refines $\mathcal{P}_i$ on W_i. It is easy to see that ι is $\mathcal{P}$-definable. f_{i+1} is state-based and $\mathcal{P}$-definable. We now show how to build a realizable winning strategy from f_{i+1}.

To be convinced there might be a problem with f_{i+1} just remind the example of Fig. 2. On this example we get the following sets W_i and f_i: on $W_1 = (\ell_0, x > 1) \cup \{\mathsf{Win}\}$, $f_1(\ell_0, x > 1) = c$ and $W_2 = (\ell_0, x \geq 0) \cup \{\mathsf{Win}\}$, $f_2(\ell_0, x > 1) = c$ and $f_2(\ell_0, x \leq 1) = \lambda$. This strategy is not realizable. To our knowledge, it is not stated in the literature how to overcome this problem and build a realizable winning strategy. To do this we need to alter f_2 a little bit on the border of $(\ell, x = 1)$. This is what is described hereafter.

Now given V_{i+1} we can compute the set of states that are on the *border* of V_{i+1} w.r.t. time-elapsing:

$$Border(V_{i+1}) = V_{i+1} \setminus \mathsf{Pred}_{t>0}(V_{i+1}, \overline{V_{i+1}})$$

If $Border(V_{i+1})$ is empty then the strategy f_{i+1} is realizable. If not we can split $Border(V_{i+1})$ into two sets: 1) the set of states from which time elapsing leads to a convex polyhedra on which f_{i+1} is a delay action 2) the set of states from which time elapsing leads to convex polyhedra on which f_{i+1} tells us to do a controllable action. The set of states that corresponds to 1) can be defined by $Z = Border(V_{i+1}) \cap \mathsf{Pred}_{t>0}(f_{i+1}^{-1}(\lambda), \overline{f_{i+1}^{-1}(\lambda)})$ and 2) by $J = V_{i+1} \setminus Z$. The states in Z lead to a state where λ is defined and can be done because those are states on W_i and f_i is realizable on W_i by induction hypothesis. It remains to update the strategy f_{i+1} on the *closest* time successor polyhedra of J. The closest time successors polyhedra D of q can be defined by "$\exists t > 0$ s.t. $Post_{]0,t]}(q) \subseteq D$". We thus get a finite set of convex polyhedra $D_1, \cdots, D_k$ of the partition $\mathcal{P}$ of W_{i+1}. By definition they all contain states that can let time $t > 0$ elapse. Take one such D_j.

- Either D_j is such that: $\forall q \in D_j,\ \forall t \geq 0,\ q \xrightarrow{t} q'$ implies $q' \in D_j$. Then we just split D_j into $D_j^1 = \{q \in D_j \mid \exists 0 < t < 1,\ \exists q' \in J \text{ s.t. } q' \xrightarrow{t} q\}$ (*i.e.* the set of states of D_j that are within 1 time unit range of J) and $D_j^2 = D_j \setminus D_j^1$. On D_j^1 define $f_{i+1} = \lambda$ and on D_j^2 f_{i+1} is unchanged.
- or D_j is bounded (in the future; as D_j is a time convex polyhedra it cannot be the case that some states of D_j can let an infinite amount of time elapse and some other cannot). In this case we split D_j into two parts.

  Define the set

  $$\frac{D_j}{2} = \{q' \in D_j \mid \exists q \in J,\ \exists \delta > 0 \text{ s.t. } q \xrightarrow{\delta} q' \text{ and } q' \xrightarrow{\delta} q'' \implies q'' \in D_j\}$$

  Note that by choice of D_j (as an adjacent part of an element in J), we have that $\frac{D_j}{2} \subseteq D_j$.

Now we change the value of f_{i+1} on

$$D'_j = \frac{D_j}{2} \cap \mathsf{Pred}_{t>0}\left(\frac{D_j}{2}, \emptyset\right)$$

by defining $f_{i+1}(q') = \lambda$ if $q' \in D'_j$. We refine the partitioning $\mathcal{P}$ of W_{i+1} by replacing the part D_j by D'_j and a convex decomposition of $D_j \setminus D'_j$. We do this for all possible D_j's. The new partitioning $\mathcal{P}$ (of W_{i+1}) refines $\mathcal{P}_i$ (on W_i) and ι is $\mathcal{P}$-definable as well as f_{i+1}.

We now prove that f_{i+1} is a winning realizable strategy over W_{i+1}, that is for every $q \in W_{i+1}$, $f_{i+1} \in \mathsf{WinStrat}(q, W_{i+1})$.

Lemma 1. *For every $q \in W_{i+1}$ such that $f_{i+1}(q) = \lambda$, there exists $\delta > 0$ such that $Post_{[0,\delta[}(q) \subseteq f_{i+1}^{-1}(\lambda)$.*

This comes from the construction above. The strategy f_{i+1} is thus realizable.

Lemma 2. *1. If $f_{i+1}(q) = c$ with c a controllable action, then $q \xrightarrow{c} q'$ implies $\iota(q') < \iota(q)$.*

2. If u is an uncontrollable action and $q \xrightarrow{u} q'$ then $\iota(q') < \iota(q)$.

3. If $f_{i+1}(q) = \lambda$, for every $t > 0$, if $\forall 0 \leq t' \leq t$, $q \xrightarrow{t'} q''$ implies $f_{i+1}(q'') = \lambda$, then if $q \xrightarrow{t} q'$ implies $\iota(q') \leq \iota(q)$.

4. If $f_{i+1}(q) = \lambda$, there exists $\Delta > 0$ such that $\forall 0 \leq t' \leq \Delta$, $q \xrightarrow{t'} q''$ implies $f_{i+1}(q'') = \lambda$ and if $q \xrightarrow{\Delta} q'$, then either $\iota(q') < \iota(q)$ or $f_{i+1}(q') \neq \lambda$, in which case $q' \xrightarrow{f_{i+1}(q')} q'''$ implies $\iota(q''') < \iota(q)$.

Proof. 1. Assume $f_{i+1}(q) = c$ with c a controllable action. There are two possible cases. Either $q \in W_{i+1} \setminus W_i$, or $q \in W_i$, in which case the value of $f_i(q)$ was already c. The second case is easy, it is by induction hypothesis. Assume then that $q \in W_{i+1} \setminus W_i$ and $f_{i+1}(q) = c$. Then $q \in \mathsf{cPred}(W_i) \setminus \mathsf{uPred}(\overline{W_i})$. There exists $q \xrightarrow{c} q'$ with $q' \in W_i$. Thus $\iota(q') \leq i < i + 1 = \iota(q)$.

2. direct from the definition of π.

3. Assume that $f_{i+1}(q) = \lambda$. There are several cases: if $q \in W_i$ and $f_i(q) = \lambda$, then we can apply the induction hypothesis. If $q \in W_i$ but $f_i(q) \neq f_{i+1}(q)$, it means that the value of the strategy has been changed during the construction for the realizability. With what precedes, there exists $\delta > 0$ such that $Post_{[0,\delta]}(q) \subseteq W_{\iota(q)}$, and if $q \xrightarrow{\delta} q'$, then $f_{i+1}(q') \neq \lambda$. Moreover, for every intermediate states q'' between q and q', $\iota(q'') = \iota(q') \leq \iota(q)$ because $q \in \mathsf{Pred}_t(W_i, \mathsf{uPred}(\overline{W_i}))$. If $q \in W_{i+1} \setminus W_i$, then $\iota(q) = i + 1$, and thus, for every q' such that $q \xrightarrow{t} q'$ is a transition allowed by f_{i+1}, then $q' \in W_{i+1}$ and thus $\iota(q') \leq i + 1 = \iota(q)$.

4. Assume $f_{i+1}(q) = \lambda$. If $q \in W_i$ and $f_i(q) = \lambda$ use the induction hypothesis. If $q \in W_i$ but $f_i(q) \neq \lambda$ (which means that the value of the strategy in q has been changed during the construction for the realizability), by some bounded delay Δ, a state $q' \in W_i$ for which $f_{i+1}(q') = f_i(q') \neq \lambda$ can be reached. We then apply point *1.* of the lemma. Otherwise, by construction of f_{i+1} we must reach a state in W_i and there is $\Delta > 0$ such that $q \xrightarrow{\Delta} q'$ and $q' \in W_i$. As $q \in W_{i+1} \setminus W_i$ we get $\iota(q) < \iota(q')$.

□

Now we can conclude that f_{i+1} is winning. Let's take a maximal run $\rho \in f_{i+1}(q, W_{i+1})$. Applying the previous lemma and as $\iota(q) = 0 \implies q \in \mathsf{Win}$, we get that ρ is a winning run, and thus f_{i+1} is a winning realizable strategy. Also f_{i+1} is polyhedral (and thus state-based).

□

This concludes the proof.

□

3 Priced Timed Games (PTG)

In this section we define *Priced Timed Games (PTG)*. We focus on *reachability PTG (RPTG)* where the aim is to reach a particular state of the game at the *lowest* possible cost. We give a run-based definition of the *optimal cost*. Then we review some previous work [LTMM02] on **acyclic** *weighted timed automata* by Salvatore La Torre *et al* where a definition of optimal cost is given as a *state-based* optimal cost function. We conclude this section by relating the two definitions and proving their equivalence.

3.1 Priced Timed Games

Definition 6 (Priced Timed Transition Systems). *A* priced timed transition system (PTTS) *is a pair* (S, Cost) *where* $S = (Q, Q_0, \mathsf{Act}, \longrightarrow)$ *is a TTS and* Cost *is a* cost function *i.e. a mapping from* $\longrightarrow$ *to* $\mathbb{R}_{\geq 0}$ *that satisfies:*

- Price Additivity: *if* $q \xrightarrow{d} q'$ *and* $q' \xrightarrow{d'} q''$ *with* $d,\ d' \in \mathbb{R}_{\geq 0}$*, then* $\mathsf{Cost}(q \xrightarrow{d+d'} q'') = \mathsf{Cost}(q \xrightarrow{d} q') + \mathsf{Cost}(q' \xrightarrow{d'} q'')$.
- Bounded Cost Rate: *there exists* $K \in \mathbb{N}$ *such that for every* $q \xrightarrow{d} q'$ *where* $d \in \mathbb{R}_{\geq 0}$, $\mathsf{Cost}(q \xrightarrow{d} q') \leq d.K$

For a transition $q \xrightarrow{e} q'$, $\mathsf{Cost}(q \xrightarrow{e} q')$ *is the* cost *of the transition and we note* $q \xrightarrow{e,p} q'$ *if* p *is the cost of* $q \xrightarrow{e} q'$.

All notions concerning runs on TTS extend straightforwardly to PTTS. Let S be a PTTS and $\rho = q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} q_n$ a finite run[5] of S. The *cost* of ρ is defined by $\mathsf{Cost}(\rho) = \sum_{i=0}^{n-1} \mathsf{Cost}(q_i \xrightarrow{e_{i+1}} q_{i+1})$.

[5] We are not interested in defining the cost of an infinite run as we will only use costs of winning runs which must be finite in the games we play.

Definition 7 (Priced Timed Games). *A* priced timed game *(PTG) (resp. Reachability PTG) is a pair* $G = (S, \mathsf{Cost})$ *such that* S *is a TG (resp. RTG) and* Cost *is a cost function.*

All the notions like strategies, outcomes, winning states are already defined for (R)TG and carry over in a natural way to (R)PTG. The *cost* $\mathsf{Cost}(q, f)$ of a winning strategy $f \in \mathsf{WinStrat}(q, G)$ is defined by:

$$\mathsf{Cost}(q, f) = \sup \{\mathsf{Cost}(\rho) \mid \rho \in \mathsf{Outcome}(q, f)\} \tag{2}$$

Definition 8 (Optimal Cost for a RPTG). *Let* G *be a RPTG and* q *be a state in* G*. The* reachable costs set $\mathsf{Cost}(q)$ *from* q *in* G *is defined by:*

$$\mathsf{Cost}(q) = \{\mathsf{Cost}(q, f) \mid f \in \mathsf{WinStrat}(q, G)\}$$

The optimal cost from q in G *is* $\mathsf{OptCost}(q) = \inf \mathsf{Cost}(q)$*. The* optimal cost *in* G *is* $\sup_{q \in Q_0} \mathsf{OptCost}(q)$ *where* Q_0 *denotes the set of initial states*[6].

Definition 9 (Optimal Strategies for a RPTG). *Let* G *be a RPTG and* q *a state in* G*. A winning strategy* $f \in \mathsf{WinStrat}(q, G)$ *is said to be* optimal *whenever* $\mathsf{Cost}(q, f) = \mathsf{OptCost}(q)$*.*

As the example below shows there are RPTG with no optimal winning strategies.

Example 2 (No optimal strategy). *Figure 4 gives a RPTG described by a priced timed automaton as introduced in [BFH⁺01]. The meaning of this automaton is the following: the cost of staying in* ℓ_0 *is* 1 *per time unit and in* ℓ_1 *it is* 2*. Also* c *is a controllable action. In this example, we can define a family of strategies* f_ε *with* $0 < \varepsilon \leq 1$ *by:* $f(\ell_0, x < 1 - \varepsilon) = \lambda$*,* $f(\ell_0, x = 1 - \varepsilon) = c$ *and* $f(\ell_1, x \leq 1) = c$*. The cost of such a strategy is* $1 + \varepsilon$*. So we can get as close as we want to* 1 *but there is no optimal winning strategy.*

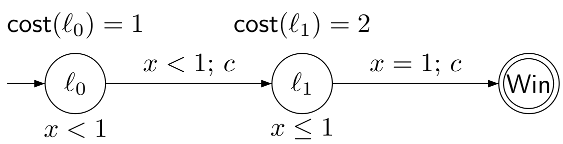

Figure 4: A PTG with no reachable optimal cost.

Our aim is many-fold. We want to 1) compute the optimal cost of winning, 2) decide whether there is an optimal strategy, and 3) in case there is an optimal strategy compute one such strategy.

3.2 Recursive Definition of the Optimal Cost

In [LTMM02] Salvatore La Torre *et al* introduced a method for computing the optimal cost in **acyclic** priced timed game. In this paper the authors define the optimal cost

[6] An alternative definition would be to take the min if we consider that the initial state is "controllable".

one can expect from a state by a function satisfying a set of recursive equations, and not using a run-based definition as we did in the last subsection. We give hereafter the definition of the function used in [LTMM02] and prove that it does correspond to our run-based definition of optimal cost.

Definition 10 (The O function (Adapted from [LTMM02])). *Let G be a RPTG. Let O be the function from Q to $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ that is the least fixed point[7] of the following functional:*

$$O(q) = \inf_{\substack{q \xrightarrow{t,p} q' \\ t \in \mathbb{R}_{\geq 0}}} \max \begin{cases} \min\left(\left(\min_{\substack{q' \xrightarrow{c,p'} q'' \\ c \in \mathsf{Act}_c}} p + p' + O(q'')\right), p + O(q')\right) & (1) \\ \sup_{\substack{q \xrightarrow{t',p'} q'' \\ t' \leq t}} \max_{\substack{q'' \xrightarrow{u,p''} q''' \\ u \in \mathsf{Act}_u}} p' + p'' + O(q''') & (2) \end{cases} \qquad (\lozenge)$$

This definition can be justified by the following arguments: item (2) of Def. 10 gives the maximum cost that an uncontrollable action can lead to if it is taken before t; note that by definition $\sup \emptyset = -\infty$ and that (2) is always defined and the outermost max is thus always defined; item (1) gives the best you can expect if a controllable can be fired; if from q' no controllable action can be taken, then either (i) there is a time step leading to some q' with $O(q')$ finite or (ii) no such state q' is reachable from q: as our semantics specify that no uncontrollable action can be used to win, we can not win from q (except if $q \in \mathsf{Win}$) and the optimal cost will be $+\infty$. We have the following theorem that relates the two definitions:

Theorem 3. *Let $G = (S, \mathsf{Cost})$ be a RPTG induced by a LHG and Q its set of states. Then $O(q) = \mathsf{OptCost}(q)$ for all $q \in Q$.*[8]

In the paper [LTMM02] by La Torre *et al* the authors use the fact that the definition of the optimal cost they give corresponds to a least fixed point of a functional and can be computed iteratively from $O^{(0)}$ defined by: if $q \in \mathsf{Win}$, then $O^{(0)}(q) = 0$ otherwise $O^{(0)}(q) = +\infty$. For $i \geq 0$ we define

$$O^{(i+1)}(q) = \inf_{\substack{q \xrightarrow{t,p} q' \\ t \in \mathbb{R}_{\geq 0}}} \max \begin{cases} \min\left(\left(\min_{\substack{q' \xrightarrow{c,p'} q'' \\ c \in \mathsf{Act}_c}} p + p' + O^{(i)}(q'')\right), p + O^{(i)}(q')\right) & (1) \\ \sup_{\substack{q \xrightarrow{t',p'} q'' \\ t' \leq t}} \max_{\substack{q'' \xrightarrow{u,p''} q''' \\ u \in \mathsf{Act}_u}} p' + p'' + O^{(i)}(q''') & (2) \end{cases} \qquad (\star)$$

Note that our definition slightly differs from the one given in [LTMM02] as we add the $p + O^{(i)}(q')$ in term (1). Indeed it can be the case that we can win only by time

[7] The righthand-sides of the equations for $O(q)$ defines a functional $\mathcal{F}$ on $(Q \longrightarrow \mathbb{R}_{\geq 0} \cup \{+\infty\})$. $(Q \longrightarrow \mathbb{R}_{\geq 0} \cup \{+\infty\})$ equipped with the natural lifting of $\leq$ on $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ constitutes a complete lattice. Also $\mathcal{F}$ can be quite easily seen to be a monotonic functional on this lattice. It follows from Tarski's fixed point theory that the least fix point of $\mathcal{F}$ exists.

[8] Note that if a state $q \in Q$ is not winning, both $O(q)$ and $\mathsf{OptCost}(q)$ are $+\infty$.

elapsing from a given state and this term is needed in our framework. In the sequel we denote[9] $C^{(i)}(q,t) = \max\left(C^{(i)}_{(1)}(q,t), C^{(i)}_{(2)}(q,t)\right)$ where we assume that $q \xrightarrow{t,p} q'$ and where:

$$\begin{cases} C^{(i)}_{(1)}(q,t) & = & \min\left(\left(\min\limits_{\substack{q' \xrightarrow{c,p'} q'' \\ c \in \mathsf{Act}_c}} p + p' + O^{(i)}(q'')\right), p + O^{(i)}(q')\right) \\ C^{(i)}_{(2)}(q,t) & = & \sup\limits_{\substack{q \xrightarrow{t',p'} q'' \\ t' \leq t}} \max\limits_{\substack{q'' \xrightarrow{u,p''} q''' \\ u \in \mathsf{Act}_u}} p' + p'' + O^{(i)}(q''') \end{cases}$$

Proof. We recall the definition of π as done in section 2:

$$\begin{array}{rcl} \pi(X) & = & \mathsf{Pred}_t\left(X \cup \mathsf{cPred}(X), \mathsf{uPred}(\overline{X})\right) \\ & = & \mathsf{Pred}_t\left(X, \mathsf{uPred}(\overline{X})\right) \cup \mathsf{Pred}_t\left(\mathsf{cPred}(X), \mathsf{uPred}(\overline{X})\right) \end{array}$$

We define inductively $W_0 = \mathsf{Win}$ and $W_{i+1} = \pi(W_i)$ and we first prove:

Lemma 3. $q \in W_i \iff 0 \leq O^{(i)}(q) < +\infty$.

Proof. If $i = 0$ then $0 \leq O^{(0)}(q) < +\infty \iff q = \mathsf{Win} \iff q \in W_0$.

Now for the induction step assume that $q \in W_i \iff 0 \leq O^{(i)}(q) < +\infty$.

First we prove that: $q \in W_{i+1} \implies 0 \leq O^{(i+1)}(q) < +\infty$**.** Take $q \in W_{i+1}$. Note that $O^{(i+1)}(q) \geq 0$ as it cannot be that all future states of q give $-\infty$ as the maximum of terms (1) and (2) in equation ($\star$): this would mean that no uncontrollable action can be taken in the future but also that no states in $\mathsf{cPred}(W_i) \cup W_i$ can be reached and contradict the fact that $q \in W_{i+1}$. Now as q is either in $\mathsf{Pred}_t(W_i, \mathsf{uPred}(\overline{W_i}))$ or $\mathsf{Pred}_t(\mathsf{cPred}(W_i), \mathsf{uPred}(\overline{W_i}))$, using the induction hypothesis we get that there exists $t \geq 0$ such that $q \xrightarrow{t,p} q'$ and term (1) of equation ($\star$) is finite. Now term (2) is either $-\infty$ (if no uncontrollable action can be taken before t) or finite as all intermediary states are outside $\mathsf{uPred}(\overline{W_i})$) and the cost due to time elapsing is bounded by some $K.t$ for some $K \in \mathbb{N}$ (this is the assumption of *bounded cost rate* of Def. 6). In any case, we get that $O^{(i+1)}(q) < +\infty$.

Now the Converse: $0 \leq O^{(i+1)}(q) < +\infty \implies q \in W_{i+1}$**.** In this case there is $t \geq 0$ such that $q \xrightarrow{t,p} q'$, and term (1) of equation ($\star$) is finite (term (1) is always defined). Then using the induction hypothesis we get that either $q' \in W_i$ or $q' \in \mathsf{cPred}(W_i)$ and term (2) is either $-\infty$ or a finite value, which means that all intermediary states are in $\mathsf{uPred}(\overline{W_i})$. Hence $q \in W_{i+1}$. □

We can now prove the following result:

Lemma 4. *For each $i \geq 0$, for each $q \in W_i$,*

$$O^{(i)}(q) = \inf\{\mathsf{Cost}(q,f) \mid f \in \mathsf{WinStrat}(q, W_i)\} \ .$$

[9] As we assume time-determinism (Cf. Def. 1) we only need (q,t) to determine the state reached after t units of time from q.

Actually we can restrict to realizable strategies as shown by the proof of the this lemma.

Proof. We will prove the result by induction on $i \geq 0$.

Base case: case $i = 0$. $O^{(0)}(\mathsf{Win}) = 0$. We can take whatever we want for a strategy from Win as any run starting from a state in Win is winning.

Induction step: Assume the proof is done for i. We want to prove the result for $i + 1$.

- proof of $O^{(i+1)}(q) \geq \inf\{\mathsf{Cost}(q, f) \mid f \in \mathsf{WinStrat}(q, W_{i+1})\}$. Let us fix $\varepsilon > 0$ and $q \in W_{i+1}$. We want to prove that there exists $f \in \mathsf{WinStrat}(q, W_{i+1})$ such that $\mathsf{Cost}(q, f) \leq O^{(i+1)}(q) + \varepsilon$. **Note that the strategy we build may be non state-based but we do not need to find a state-based strategy to prove this lemma.**

  By definition of $O^{(i+1)}(q)$ there is some $q \xrightarrow{t,p} q'$ s.t. $C^{(i+1)}(q, t) \leq O^{(i+1)}(q) + \frac{\varepsilon}{2}$. Now define the strategy f by $f(q \xrightarrow{t',p'} q'') = \lambda$ for $0 \leq t' < t$. Also in case some uncontrollable action u can be taken at some time $\delta \leq t$ it must lead to a state $q_{\delta,u} \in W_i$. Then by induction hypothesis we know that there is a strategy $f_{\delta,u} \in \mathsf{WinStrat}(q_{\delta,u}, W_i)$ s.t. $\mathsf{Cost}(q_{\delta,u}, f_{\delta,u}) \leq O^{(i)}(q_{\delta,u}) + \frac{\varepsilon}{2}$. We define $f(q \xrightarrow{\delta,p'} q'' \xrightarrow{u,p''} \rho) = f_{\delta,u}(\rho)$ for all runs ρ such that *first*$(\rho) = q_{\delta,u}$.

  Now two cases arise:

  - either the minimum of $C^{(i+1)}_{(1)}(q, t)$ is given by $p + O^{(i)}(q')$: then it must be the case that $q' \in W_i$ by lemma 3. As $q' \in W_i$ we use the induction hypothesis: there is a strategy $g_{q,t} \in \mathsf{WinStrat}(q', W_i,)$ with $\mathsf{Cost}(q', g_{q,t}) \leq O^{(i)}(q') + \frac{\varepsilon}{2}$. Also we define $f(q \xrightarrow{t,p} \rho) = g_{q,t}(\rho)$ for all runs ρ with *first*$(\rho) = q'$. Now the cost of f from q is the supremum of (i) the costs of runs beginning with $q \xrightarrow{t,p} q'$ which are bounded by $p + \mathsf{Cost}(q', g_{q,t})$ and (ii) the costs of the runs with uncontrollable actions taken before t which is bounded by $p' + p'' + \mathsf{Cost}(q_{\delta,u}, f_{\delta,u})$. This last term is bounded by the supremum of $p + O^{(i)}(q') + \frac{\varepsilon}{2}$ and $p' + p'' + O^{(i)}(q_{\delta,u}) + \frac{\varepsilon}{2}$. As $C^{(i+1)}(q, t) \leq O^{(i+1)}(q) + \frac{\varepsilon}{2}$ we get that $p + O^{(i)}(q') \leq O^{(i+1)}(q) + \varepsilon$ and $p' + p'' + O^{(i)}(q_{\delta,u}) \leq O^{(i+1)}(q) + \varepsilon$ for all possible u before t. Hence, $\mathsf{Cost}(q, f) \leq O^{(i+1)}(q) + \varepsilon$.
  - the minimum of term (1) is given by some controllable action c. We define $f(q \xrightarrow{t,p} q') = c$ and we know from lemma 3 that $q \xrightarrow{t,p} q' \xrightarrow{c,p'} q''$ with $q'' \in W_i$. Again there must be a suitable strategy g'' from q'' s.t. $\mathsf{Cost}(q'', g'') \leq O^{(i)}(q'') + \frac{\varepsilon}{2}$. We define $f(q \xrightarrow{t,p} q' \xrightarrow{c,p'} \rho) = g''(\rho)$ for all runs ρ with *first*$(\rho) = q''$. The same argument as the one given for the first case applies and can prove that $\mathsf{Cost}(q, f) \leq O^{(i+1)}(q) + \varepsilon$.

  Now note that f is in $\mathsf{WinStrat}(q, W_{i+1})$ as applying f will force the state in W_i within less than t time units. We have thus proved that for all $\varepsilon > 0$ there is winning strategy $f \in \mathsf{WinStrat}(q, W_{i+1})$ s.t. $\mathsf{Cost}(q, f) \leq O^{(i+1)}(q) + \varepsilon$. It is also easy to see that by construction f is realizable.

- proof of $O^{(i+1)}(q) \leq \inf\{\mathsf{Cost}(q, f) \mid f \in \mathsf{WinStrat}(q, W_{i+1})\}$. We prove that for all $f \in \mathsf{WinStrat}(q, W_{i+1})$, $O^{(i+1)}(q) \leq \mathsf{Cost}(q, f)$. Take some $q \in W_{i+1}$ and $f \in \mathsf{WinStrat}(q, W_{i+1})$. Then applying f we must reach W_i within a finite amount of time (and avoid bad states on the way):
  - either f is "delay until reaching W_i": $q \xrightarrow{t,p} q'$ with $q' \in W_i$. In this case $\mathsf{Cost}(q, f) \geq p + \mathsf{Cost}(q', f)$ (they may be uncontrollable actions before we reach q' leading a to larger cost). Now f is also a winning strategy[10] from $q' \in W_i$ and we can apply the induction hypothesis: $\mathsf{Cost}(q', f,) \geq O^{(i)}(q')$. So term $C^{(i+1)}_{(1)}(q, t)$ is lower than $\mathsf{Cost}(q, f)$, *i.e.* $C^{(i+1)}_{(1)}(q, t) \leq \mathsf{Cost}(q, f)$.
  - or f is "delay for t time units and do a c": in this case again we have $q \xrightarrow{t,p} q' \xrightarrow{c,p'} q''$ with $q'' \in W_i$ and using the induction hypothesis we get the same result: $C^{(i+1)}_{(1)}(q, t) \leq \mathsf{Cost}(q, f)$

  Now, if there is any uncontrollable action enabled before t, *i.e.* $q \xrightarrow{t',p'} q' \xrightarrow{u,p''} q''$ for $0 \leq t' \leq t$ we also have $\mathsf{Cost}(q, f) \geq p' + p'' + \mathsf{Cost}(q'', f)$ for all such q''. Again q'' must be in W_i and applying the induction hypothesis we get that $C^{(i+1)}_{(2)}(q, t) \leq \mathsf{Cost}(q, f)$. It follows that $C^{(i+1)}(q, t) \leq \mathsf{Cost}(q, f)$ and so is $O^{(i+1)}(q)$. So $O^{(i+1)}(q)$ is a lower bound of $\{\mathsf{Cost}(q, f) \mid f \in \mathsf{WinStrat}(q, W_{i+1})\}$ and consequently $O^{(i+1}(q) \leq \inf\{\mathsf{Cost}(q, f) \mid f \in \mathsf{WinStrat}(q, W_{i+1})\}$.

This completes the proof. □

Proof of Theorem 3: As π is continuous (the RPTG G is induced by a LHG), there exists i such that $q \in W_i$ and for every $k \geq i$, $q \in W_k$ and $O^{(k)}(q)$ is defined. Thus $O(q)$ is defined. Fix $\varepsilon > 0$. By definition of $O(q)$, there exists $k \geq i$ such that $O(q) \leq O^{(k)}(q) \leq O(q) + \frac{\varepsilon}{2}$. Applying lemma 4, we get that there exists a strategy $f \in \mathsf{WinStrat}(q, W_k)$ such that $O^{(k)}(q) \leq \mathsf{Cost}(q, f) \leq O^{(k)} + \frac{\varepsilon}{2}$. Thus we get that $O(q) \leq \mathsf{Cost}(q, f) + \varepsilon$ and $f \in \mathsf{WinStrat}(q, W)$. Hence $O(q) \geq \inf_{f \in \mathsf{WinStrat}(q,W)} \mathsf{Cost}(q, f)$.

Conversely, take a strategy $f \in \mathsf{WinStrat}(q, W)$. We note $\mathcal{O}$ the functional defined by equation ($\Diamond$). We have that $O = \mathcal{O}(O)$. The function $\Omega : q \longmapsto \mathsf{Cost}(q, f)$ satisfies that $\Omega \geq \mathcal{O}(\Omega)$. Thus, as O is the least fix point of $\mathcal{O}$, we get that for every q, $O(q) \leq \Omega(q)$.

□

4 Reducing Priced Timed Games to Timed Games

In this section we show that computing the optimal cost to win a priced timed game amounts to solving a control problem (without cost). The idea is the following:

[10]Note that f is defined for runs starting in q but as q' is obtained from q by applying f it is possible to view f as a winning strategy from q'.

4.1 From Optimal Reachability Game to Reachability Game

Assume we want to compute the optimal cost to win a priced timed game $\mathcal{A}$. We define a (usual and unpriced) timed game $\mathcal{A}_C$ as follows: we use a variable *cost* to stand for the cost value. We build $\mathcal{A}_C$ with the same discrete structure as $\mathcal{A}$ and specify a rate for *cost* in each location: if the cost increases with a rate of $+k$ per unit of time in $\mathcal{A}$, then we set the derivative of *cost* to be $-k$ in $\mathcal{A}_C$. Now we solve the following control problem: can we win in $\mathcal{A}_C$ with the winning states being $\mathsf{Win} \wedge cost \geq 0$? Note that if $\mathcal{A}$ is a priced timed automaton [BFH$^+$01, ALTP01] (game) then $\mathcal{A}_C$ is a (simple) linear hybrid automaton [Hen96]. Intuitively speaking we want to solve a control game with a distinguished variable *cost* that decreases when time elapses and when a discrete transition is fired according to what it costs in the priced timed game. So we are asking the question: "what is the minimal amount of resource (*cost*) needed to win the control game $\mathcal{A}_C$?" In the case of $\mathcal{A}$ we can compute the winning states of $\mathcal{A}_C$ (with an algorithm for solving hybrid games [WT97, DAHM01]) and if it terminates we have the answer to the optimal reachability game: we intersect the set of initial states with the set of winning states, and in case it is not empty, the projection on the *cost* axis we obtain a constraint on the cost like $cost > 1$. By definition of winning set of states in reachability games, *i.e.* this is the largest set from which we can win, no cost lower than 1 is winning and we can deduce that 1 is the optimal cost. Also we can deduce there is no optimal strategy because of the strict inequality.

The rest of this section is devoted to formalizing this reduction and to proving that this reduction is correct. Then we focus on the computation of optimal strategies and we investigate conditions under which we can compute the optimal cost, (*i.e.* a termination criterion).

Definition 11 (RTG associated to a RPTG). *Let* $G = ((Q, Q_0, \mathsf{Win}, \mathsf{Act}, \longrightarrow_G), \mathsf{Cost})$ *be a RPTG. We associate to* G *the RTG* $\mathsf{Cont}(G) = (Q \times \mathbb{R}_{\geq 0}, Q_0 \times \mathbb{R}_{\geq 0}, \mathsf{Win} \times \mathbb{R}_{\geq 0}, \mathsf{Act}, \longrightarrow_{\mathsf{Cont}(G)})$ *where* $(q, c) \xrightarrow{e}_{\mathsf{Cont}(G)} (q', c') \iff q \xrightarrow{e, c-c'}_G q'$. *In the sequel we abstract away the subscript of* $\longrightarrow$ *as it will be clear from the context which transition relation is referred to.*

Note that with our reduction, the cost information becomes part of the state and that the runs in G and $\mathsf{Cont}(G)$ are closely related. Now we focus on subclasses of reachability timed games, namely those obtained by enriching timed automata [AD94] with costs (Priced or Weighted Timed Automata [BFH$^+$01, ALTP01]). This enables us to rely on symbolic algorithms and to have computability results.

4.2 Priced Timed Game Automata

Let X be a finite set of real-valued variables called clocks. We denote $\mathcal{B}(X)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z}$, $x, y \in X$ and $\sim \in \{<, \leq, =, >, \geq\}$. A *valuation* of the variables in X is a mapping from X to $\mathbb{R}_{\geq 0}$ (thus an element of $\mathbb{R}_{\geq 0}^X$). For a valuation v and a set $R \subseteq X$ we denote $v[R]$ the valuation that agrees with v on $X \setminus R$ and is zero on R. We denote $v + \delta$ for $\delta \in \mathbb{R}_{\geq 0}$ the valuation s.t. for all $x \in X$, $(v + \delta)(x) = v(x) + \delta$.

Definition 12 (PTGA). *A* Priced Timed Game Automaton $A = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, f)$ *is a tuple where:*

- L *is a finite set of* locations*,*
- $\ell_0 \in L$ *is the* initial *location,*
- $\mathsf{Act} = \mathsf{Act}_c \cup \mathsf{Act}_u$ *is the set of* actions *(partitioned into controllable and uncontrollable actions),*
- X *is a finite set of* real-valued clocks*,*
- $E \subseteq L \times \mathcal{B}(X) \times \mathsf{Act} \times 2^X \times L$ *is a finite set of* transitions*,*
- $\mathsf{inv} : L \longrightarrow \mathcal{B}(X)$ *associates to each location its* invariant*,*
- $f : L \cup E \longrightarrow \mathbb{N}$ *associates to each location a* cost rate *and to each discrete transition a* cost*.*

A reachability *PTGA (RPTGA) is a PTGA with a distinguished set of locations* $\mathsf{Win} \subseteq L$*. It defines the winning set of states* $\mathsf{Win} \times \mathbb{R}^X_{\geq 0}$.

The semantics of a PTGA $A = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, f)$ is a PTG $G_A = ((L \times \mathbb{R}^X_{\geq 0}, (\ell_0, \vec{0}), \mathsf{Act}, \longrightarrow), \mathsf{Cost})$ where $\longrightarrow$ consists of: i) *discrete steps:* $(\ell, v) \xrightarrow{e} (\ell', v')$ if there exists $(\ell, g, e, R, \ell') \in E$ s.t. $v \models g$ and $v' = v[R]$; $\mathsf{Cost}((\ell, v) \xrightarrow{e} (\ell', v')) = f(\ell, g, e, R, \ell')$; ii) *time steps:* $(\ell, v) \xrightarrow{\delta} (\ell, v')$ if $\delta \in \mathbb{R}_{\geq 0}$, $v' = v + \delta$ and $v, v' \in \mathsf{inv}(\ell)$; and $\mathsf{Cost}((\ell, v) \xrightarrow{\delta} (\ell, v')) = \delta \cdot f(\ell)$. Note that this definition of Cost gives a cost function as defined in Def. 6.

Lemma 5 (PTGA to LHG). *Let A be a PTGA. There exists a LHG H with a distinguished extra variable cost such that* $\mathsf{Cont}(G_A) = G_H$[11]*.*

Proof. Let $A = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, f)$ be a PTGA. We associate to A a LHG H with the same set of locations, the same set of actions, the same invariants and an extra variable *cost* which is not a clock but a special analog variable. The transitions of H are given by:

- if $\ell \xrightarrow{g,a,Y} \ell'$ is a transition of A and $f(\ell \xrightarrow{g,a,Y} \ell') = p$ then there is a transition $\ell \xrightarrow{g,a,Y:=0,cost:=cost-p} \ell'$ in H,
- in each location ℓ the derivative of *cost* is given by: $\dot{cost} = -f(\ell)$,
- we add a global invariant in H which is that $cost \geq 0$.

By construction of H we have $\mathsf{Cont}(G_A) = G_H$. □

The correctness of the reduction is given by the following theorem:

Theorem 4. *Let A be a RPTGA and H its corresponding LHG (as given by lemma 5). If the semi-algorithm* $\mathsf{CompWin}$ *terminates for G_H and if* $W_H = \mathsf{CompWin}(G_H)$*, then: 1)* $\mathsf{CompWin}$ *terminates for G_A and* $W_A \overset{\text{def}}{=} \mathsf{CompWin}(G_A) = \exists cost.W_H$*; and 2)* $(q, c) \in W_H \iff \exists f \in \mathsf{WinStrat}(q, W_A)$ *with* $\mathsf{Cost}(q, f) \leq c$*.*

[11] Note that G_H is the TG that gives the semantics of H.

Proof. For item 1) of theorem 4, denote $V_H^{(i)}$ respectively $V_A^{(i)}$ the sets obtained after i iterations of the semi-algorithm $\mathsf{CompWin}$. By induction, one can prove that for each i, $q \in V_A^{(i)} \iff \exists c \geq 0$ s.t. $(q,c) \in V_H^{(i)}$ because the variable *cost* does not constrain the transitions in H. Hence, if the computation of $\mathsf{CompWin}(H)$ terminates in n iterations, the computation of $\mathsf{CompWin}(A)$ cannot take more than n iterations. In addition we get that $W_A = \exists cost.W_H$.

Now let us prove item 2) of theorem 4. For the "*if*" part, assume we have $f \in \mathsf{WinStrat}(q, W_A)$ with $\mathsf{Cost}(q,f) \leq c$. Take a maximal run $\pi \in \mathsf{Outcome}(q,f)$. Then π is winning and $\mathsf{Cost}(\pi) \leq c$. By definition of H, if there is a run from q in W_A ending in Win with cost less than c, then there is a run ρ' in G_H from (q,c) ending in $\mathsf{Win} \wedge cost \geq 0$ s.t. $\exists cost.\rho' = \rho$ (the run ρ' is in H, states along ρ' are thus of the form (q,c) ; $\exists cost.\rho'$ is the run in G_A where each state (q,c) is projected onto q). Define then $g(\rho') = f(\exists cost.\rho')$. The strategy g is well-defined and winning (because f is winning and each run ρ in $\mathsf{Outcome}((q,c),g)$ verifies that $\exists cost.\rho$ is in $\mathsf{Outcome}(q,f)$), realizable (because f is realizable). Thus (q,c) is a winning state and must be in W_H.

For the "*only if*" part. As $(q,c) \in W_H$, by theorem 2, there is a state-based, realizable and winning strategy f in $\mathsf{WinStrat}((q,c),W_H)$. We will build inductively a strategy g from f so that: g is well-defined, winning and $\mathsf{Cost}(q,g) \leq c$. Define g^0 by $g^0(q) = f(q,c)$. Now assume we have build g^i so that i) g^i is a realizable strategy ; ii) g^i is well-defined and generates only runs of length less than i ; and iii) if $\pi \in \mathsf{Outcome}(q,g^i)$ then there exists $\rho \in \mathsf{Outcome}((q,c),f)$ s.t. $\exists cost.\rho = \pi$ and $g^i(\pi) = f(last(\rho))$. For runs of length $\leq i$ [12] allowed by g^i, we define $g^{i+1} = g^i$. Now we define g^{i+1} on runs of length $i+1$ generated by g^i. Take π with $|\pi| = i$ and $\pi \in \mathsf{Outcome}(q,g^i)$. If $g^i(\pi) = a$ with $a \in \mathsf{Act}_c$, then by induction hypothesis and iii) we know that there exists $\rho \in \mathsf{Outcome}((q,c),f)$ s.t. $\exists cost.\rho = \pi$ and $g^i(\pi) = f(last(\rho)) = a$. Thus, a is enabled at $last(\rho)$ in W_H and f allows a. This entails that $\rho \xrightarrow{a} (q',c')$ is in $\mathsf{Outcome}((q,c),f)$ for some $(q',c') \in W_H$. Now define $g^{i+1}(\pi \xrightarrow{a} q')$ as $f(q',c')$. If $g^i(\pi) = \lambda$, then by induction hypothesis and iii) we know that there exists $\rho \in \mathsf{Outcome}((q,c),f)$ s.t. $\exists cost.\rho = \pi$ and $g^i(\pi) = f(last(\rho)) = \lambda$. There exists $\delta > 0$ such that if $\rho \xrightarrow{\delta} (q',c')$ with $(q',c') \in W_H$, then either (q',c') is winning or $f(q',c') \neq \lambda$. We then define for each $0 < t < \delta$, $g^{i+1}(\pi \xrightarrow{t}) = \lambda$. If (q',c') is winning, then so is q' in G_A. If not, define $g^{i+1}(\pi \xrightarrow{\delta} q') = f(q',c')$. The strategy g^{i+1} satisfies ii). It also satisfies i). Finally it satisfies iii) as we build g^{i+1} in order to satisfy iii). Now define g by: $g(q) = g^0(q)$ and if $\rho \in \mathsf{Outcome}(q,g)$ and $|\rho| = n$ then $g(\rho) = g^n(\rho)$. g is winning and $\mathsf{Cost}(q,g) \leq c$ because otherwise f would not be winning from (q,c) in G_H.

This completes the proof of theorem 4. □

4.3 Computation of the Optimal Cost and Strategy

Theorem 5. *Let A be a RPTGA and H its corresponding LHG. If the semi-algorithm* $\mathsf{CompWin}$ *terminates for G_H then for every $(\ell, v) \in W_A$, the upward closure*[13] *of the set* $\mathsf{Cost}(\ell, v)$ *is equal to $\{c \mid ((\ell,v),c) \in W_H\}$.*

[12] The length of a run is the number of strict alternations between delays and actions.

[13] The upward closure of a set of reals S is defined as $\{s' \mid \exists s \in S \text{ s.t. } s' \geq s\}$.

Proof. This is a direct consequence of theorem 4. □

Corollary 1 (Optimal Cost). *Let A be a RPTGA and H its corresponding LHG. If the semi-algorithm* CompWin *terminates for G_H then the upward closure of the set* $\mathsf{Cost}(\ell_0, \vec{0})$ *is computable and is of the form cost $\geq k$ (left-closed) or cost $> k$ (left-open) with $k \in \mathbb{Q}_{\geq 0}$. In addition we get that* $\mathsf{OptCost}(A) = k$.

Proof. The fact that it is of the form *cost* $\geq k$ or *cost* $> k$ is direct from theorem 5. Now k is a rational number because we are considering LHG and symbolic algorithms. The iterative computation of the π operator generates only polyhedra defined by rational inequations. As it terminates the result follows. □

Corollary 2 (Existence of an Optimal Strategy). *Let A be a RPTGA. If the upward closure of the set* $\mathsf{Cost}(\ell_0, \vec{0})$ *is left-open then there is no optimal strategy. Otherwise we can compute a realizable, winning, optimal strategy.*

Proof. Direct consequence of theorem 4. □

Note that in the proof of corollary 2 we build a state-based strategy for H which is no more state-based for A: indeed the strategy for H depends on the current value of the cost (which is part of the state in H). The strategy for A is thus dependent on the run and not memory-free. However it only depends on the last state (ℓ, v) of the run and on the accumulated cost along the run.

It is not straightforward to build an optimal state-based (without the accumulated cost) strategy in A as shown by the following example.

Let A be the RPTGA depicted in Fig. 5. The most natural way to define a state-

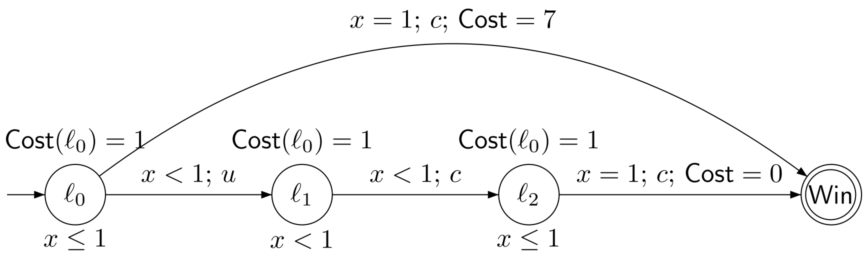

Figure 5: Life is Not Easy

based (without cost) strategy would be to take in each state (ℓ, v) the action given by the strategy in H in the state (ℓ, v, c) with some minimal c. Doing this would result in a strategy f such that $f(\ell_1, x < 1) = \lambda$. Such a strategy is however not winning. In this particular case, we can build an optimal strategy f^* the cost of which is 8: $f^*(\ell_0, x < 1) = \lambda$, $f^*(\ell_0, x = 1) = c$, $f^*(\ell_1, x < 1) = c$, $f^*(\ell_2, x < 1) = \lambda$ and $f^*(\ell_2, x = 1) = c$. This strategy is optimal in $(\ell_0, \mathbf{0})$ but is not (and needs not to be) optimal in state ℓ_1 for example. From this observation we see that it is difficult to exhibit an algorithm for building a state-based (with no cost) winning strategy.

In the next section we will exhibit a restricted class of automata for which we can synthesize optimal state-based strategies automatically. One of the challenges of future work is to enlarge this class of automata.

4.4 Termination Criterion & Optimal Strategies

Theorem 6. *Let A be a RPTGA satisfying the following hypotheses:*

- *A is bounded, i.e. all clocks in A are bounded ;*
- *the cost function of A is* strictly non-zeno*, i.e. there exists some $\kappa > 0$ such that the accumulated cost of every cycle in the region automaton associated with A is at least κ.*

Then the semi-algorithm CompWin *terminates for G_H, where H is the LHG associated with A.*

Proof. We already know that the semi-algorithm CompWin terminates for G_A ([MPS95, AMPS98]). For each i, we note $V_A^{(i)}$ respectively $V_H^{(i)}$ the set of states obtained after i iterations in the semi-algorithm described in section 2. We note $W_A = \mathsf{CompWin}(G_A)$ and $W_H = \mathsf{CompWin}(G_H)$. There exists N such that $W_A = V_A^{(N)}$. As $V_A^{(N)}$ is a polyhedral set, for each location ℓ, for each region R such that $(\ell, R) \subseteq W_A$, there exists a piecewise affine function $f_{\ell,R}$ defined on R and an operator $\succ_{\ell,R} \in \{>, \geq\}$ such that $V_H^{(N)}$ is defined by

$$\bigcup_{\ell,R \text{ region s.t.}(\ell,R)\subseteq W_A} \{((\ell, v), c) \mid v \in R \text{ and } c \succ_{\ell,R} f_{\ell,R}(v)\}$$

As each region R is bounded, there exists a constant $M_{\ell,R}$ such that for each $v \in R$, $f_{\ell,R}(v) \leq M_{\ell,R}$. Take now $M = \max_{(\ell,R)\subseteq W_A} M_{\ell,R}$. Now take an integer such that $p > \frac{M}{\kappa}$.(size of the region automaton of A). Take now $((\ell, v), c) \in V_H^{(N+p)}$. By construction there exists $((\ell', v'), c') \in V_H^{(N)}$ such that there is a path from (ℓ, v, c) to (ℓ', v', c'). Along this path there is at least a region which appears $\frac{M}{\kappa}$ times. Thus the cost c must be larger than $c' + M$ which is itself larger than M. Thus, $((\ell, v), c)$ was already in $V_H^{(N)}$. Termination follows. □

As argued before, it is not always possible to build state-based optimal strategies for A (that is without cost information). We will characterize a restricted classes of automata for which one can synthesize a state-based winning strategy.

Theorem 7. *Let A be a RPTGA satisfying the following hypotheses:*

1. *A is bounded ;*
2. *the cost function of A is strictly non-zeno ;*
3. *constraints on controllable actions are non-strict ;*
4. *constraints on uncontrollable actions are strict*

Let $W_A = \mathsf{CompWin}(G_A)$ be the set of winning states. There exists a state-based strategy f defined over W_A s.t. for each $(\ell, v) \in W_A$, $f \in \mathsf{WinStrat}((\ell, v), W_A)$ and $\mathsf{Cost}((\ell, v), f) = \mathsf{OptCost}(\ell, v)$.

Proof. The strategy f we will build is to choose in each state the optimal local choice we can do, that is it will correspond to the action given by an optimal (potentially not state-based) strategy in each state.

Lemma 6. *W_A is a closed set (with a topological meaning). In particular, we can write W_A as:*

$$W_A = \bigcup_{\ell \in L, P \in \mathcal{P}, f_{\ell,P}} \{(\ell, v, c) \mid v \in P \text{ and } c \geq \gamma_{\ell,P}(v)\}$$

where $\mathcal{P}$ is a finite set of polyhedra over $\mathbb{R}^X_{\geq 0}$ and $\gamma_{\ell,P}$ are affine functions defined on $\mathbb{R}^X_{\geq 0}$.

This is because operators cPred and Pred_t preserve closed sets whereas uPred preserves open sets. Thus the operator π preserves closed sets.

We define f formally. First we assume we have constructed as in Theorem 2 a strategy g over W_H which is winning in each point, state-based and realizable. If (ℓ, v) is a state we note $c_{(\ell,v)}$ the minimal cost c such that (ℓ, v, c) belongs to W_H. We then define $f(\ell, v) = g(\ell, v, c_{\ell,v})$. Now there can be a problem in the frontiers of polyhedra (because of realizability). For all (ℓ, v) such that $f(\ell, v) = \lambda$, but there is no $\delta > 0$ for having the realizability constraint,we define $f(\ell, v) = c$ where c is the controllable action given by f in the polyhedron just "after" (ℓ, v). Note that as all controllable constraints are closed (or non-strict), action c can be done from state (ℓ, v). Note also that by continuity (closeness of constraints, etc.), (ℓ, v, c) is a possible transition in W_H. Modified that way, f is realizable.

We then prove that f is *non-blocking* in the sense that for every state (ℓ, v), for every run $\rho \in \mathsf{Outcome}((\ell, v), f)$, either ρ is winning (*i.e.* ends in Win), or there exists $t \geq 0$ such that:

- for all $0 \leq t' < t$, $\rho \xrightarrow{t'} (\ell', v')$ implies $f(\ell', v') = \lambda$
- $\rho \xrightarrow{t} (\ell', v')$ implies $(\ell', v') \in W_A$ and $f(\ell', v') \neq \lambda$

First assume that $f(\mathit{last}(\rho)) = c$ for some controllable action c. Nothing to do in this case. Assume then that $f(\mathit{last}(\rho)) = \lambda$, we note $T = \{t \geq 0 \mid \forall 0 \leq t' \leq , f(\mathit{last}(\rho \xrightarrow{t'})) = \lambda\}$. As f is realizable (and all clocks are bounded), T is a right-open interval with t as upper bound for some t. And using the previous lemma $\mathit{last}(\rho \xrightarrow{t}) \in W_A$. In addition, as f is realizable, $f(\mathit{last}(\rho \xrightarrow{t})) \neq \lambda$. Thus $f(\mathit{last}(\rho \xrightarrow{t})) \neq c$ for some controllable action c.

We now need to prove that f is a winning strategy in all states (ℓ, v) of W_A. We do that by lifting runs in A into runs in H starting from $(\ell, v, c_{\ell,v})$. From what precedes we can not be blocked (apart when Win is reached), thus using the strongly non-zenoness assumption on the cost, we get that all runs generated by f end in Win. Thus f is a winning strategy. □

Under the previous conditions we build a strategy f which is globally optimal for all states of W_A.

5 Preliminary Experiments

A first prototype implementing the construction of optimal strategies from theorem 6 has been made in HyTech [HHWT95, HHWT97] and applied to a small example concerning energy-optimal connection to a mobile base-station.

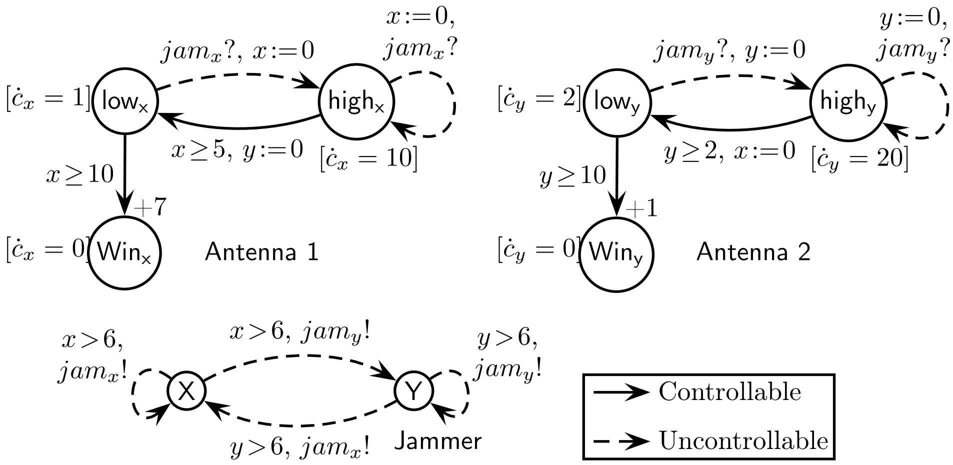

Figure 6: *Mobile Phone Example.*

In our example (see Fig. 6), we consider a mobile phone with two antenna emitting on different frequencies. Making the initial connection with the base station takes 10 time units whatever antenna is in use. Statistically, a jam of the transmission (*e.g.* collision with another phone) may appear every 6 time units in the worst case. When a collision is observed the antenna try to transmit with a higher level of energy and, depending on the latency of the antenna, can switch back to the normal mode after a while. But switching back to the low consumption mode requires more resources and force to interrupt the other transmission. Once the connection with the base station is established the message is delivered with an energy consumption varying depending on the antenna (+7 for Antenna 1 and +1 for Antenna 2). The HyTech model of the example is in the appendix A[14].

In this case the game is clearly a reachability game which is won when one of the antenna reaches the state Win. It may also be seen, from the model, that the assumptions of Theorem 7 needed to ensure termination and existence of a state-based strategy actually hold.

The graphical representation of the optimal strategy obtained from our model is given in Fig. 7. The optimal cost is 109. As the strategy is state-based, each composite location has its own partition of the clock's state-space. Starting in location low$_x$.low$_y$.X at $x = y = 0$, the strategy says that we have to delay until 10. Unfortunately, the opponent will probably produce a jam before 10 and make the system go to one of the higher energy transmission state. Note that to define the strategy we need more than zones, classically used for timed automata. Note that on the composed state high$_x$.high$_y$.Y the splitting of this optimal strategy can not be expressed with classical zones or regions.

[14]The file can also be downloaded at `http://www.lsv.ens-cachan.fr/aci-cortos/ptga`.

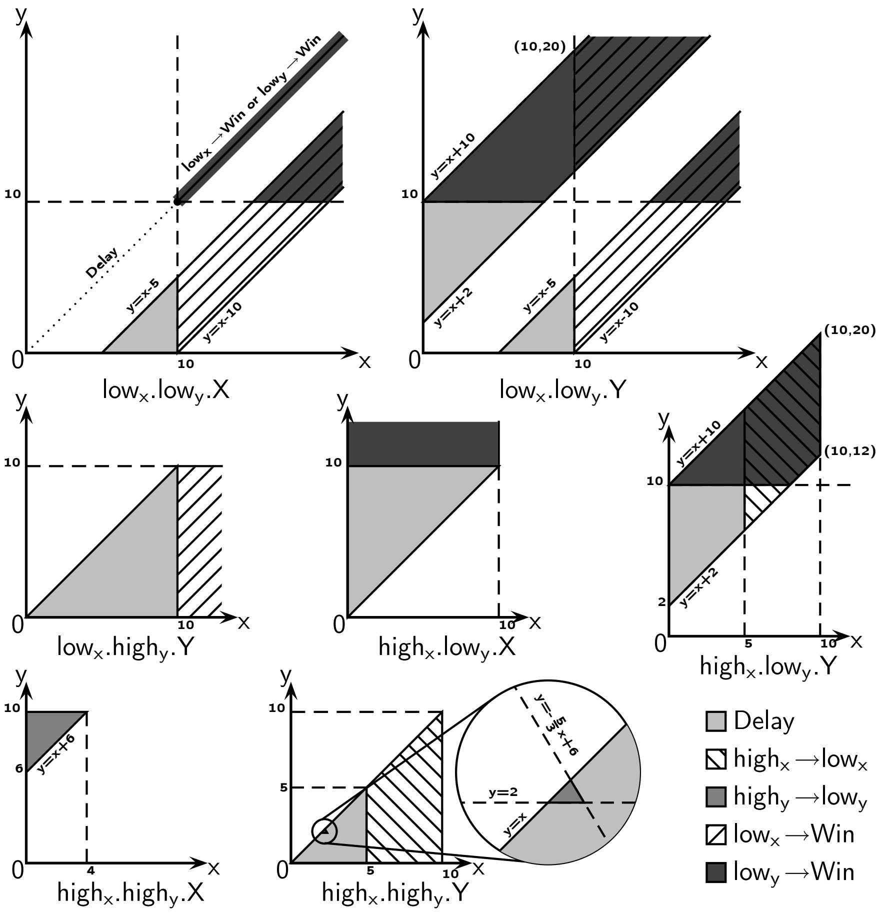

Figure 7: *Strategy of the* mobile phone *example.*

References

[Abd02] Yasmina Abdeddaim. *Modélisation et résolution de problèmes d'ordonnancement à l'aide d'automates temporisés.* PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2002.

[ACH93] Rajeev Alur, Costas Courcoubetis, and Thomas A. Henzinger. Computing accumulated delays in real-time systems. In *Proc. 5th International Conference on Computer Aided Verification (CAV'93)*, volume 697 of *Lecture Notes in Computer Science*, pages 181–193. Springer, 1993.

[AD94] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.

[ALTP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th Int. Work. Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[AM99] Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Proc. 2nd Int. Work. Hybrid Systems: Computation and Control (HSCC'99)*, volume 1569 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

[AMPS98] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

[BBL04] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Staying alive as cheaply as possible. In *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, Lecture Notes in Computer Science. Springer, 2004. To appear.

[BFH$^+$01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[BMF02] Ed Brinksma, Angelika Mader, and Ansgar Fehnker. Verification and optimization of a PLC control schedule. *Journal of Software Tools for Technology Transfer (STTT)*, 4(1):21–33, 2002.

[CHR02] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2002.

[CY92] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[DAHM01] Luca De Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[DDR04] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, Lecture Notes in Computer Science. Springer, 2004. To appear.

[Feh99] Ansgar Fehnker. Scheduling a steel plant with timed automata. In *Proc. 6th Int. Conf. Real-Time Computing Systems and Applications (RTCSA'99)*, pages 280–286. IEEE Computer Society Press, 1999.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Annual Symposim on Logic in Computer Science (LICS'96)*, pages 278–292. IEEE Computer Society Press, 1996.

[HHM99] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. Rectangular hybrid games. In *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 1999.

[HHWT95] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to HYTECH. In *Proc. 1st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer, 1995.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model-checker for hybrid systems. *Journal on Software Tools for Technology Transfer (STTT)*, 1(1–2):110–122, 1997.

[HLP00] Thomas Hune, Kim G. Larsen, and Paul Pettersson. Guided synthesis of control programs using UPPAAL. In *Proc. IEEE ICDS Int. Work. Distributed Systems Verification and Validation*, pages E15–E22. IEEE Computer Society Press, 2000.

[Lar03] Kim G. Larsen. Resource-efficient scheduling for real time systems. In *Proc. 3rd International Conference on Embedded Software (EMSOFT'03)*, volume 2855 of *Lecture Notes in Computer Science*, pages 16–19. Springer, 2003. Invited presentation.

[LBB+01] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Angskar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.

[LTMM02] Salvatore La Torre, Supratik Mukhopadhyay, and Aniello Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proc. 2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*, volume 223 of *IFIP Conference Proceedings*, pages 485–497. Kluwer, 2002.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 229–242. Springer, 1995.

[NTY00] Peter Niebert, Stavros Tripakis, and Sergio Yovine. Minimum-time reachability for timed automata. In *Proc. 8th IEEE Mediterranean Conference on Control and Automation*, 2000.

[NY01] Peter Niebert and Sergio Yovine. Computing efficient operations schemes for chemical plants in multi-batch mode. *European Journal of Control*, 7(4):440–453, 2001.

[RLS04] Jacob Rasmussen, Kim G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, Lecture Notes in Computer Science. Springer, 2004. To appear.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 1–13. Springer, 1995. Invited talk.

[WT97] Howard Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th IEEE Conference on Decision and Control*, pages 4607–4612. IEEE Computer Society Press, 1997.

A Two Antenna Mobile-phone Model

```
 -- -------------- --
 -- Mobile example --
 -- -------------- --

var x,y: clock;
    t,t1: analog;
    cost, cost_x, cost_y: analog;
    cost0, cost1, cost2, cost3, cost4: parameter;
    c,u,flag: discrete;

 -- Legend
 -- flag =-1 --> Uncontrollable action
 -- flag = 0 --> delay
 -- flag = 1 --> low_x->Win
 -- flag = 2 --> high_x->A
 -- flag = 3 --> low_y->Win
 -- flag = 4 --> high_y->low_y
 --
 -- c --> If c changed -> controllable action
 -- u --> If u changed -> uncontrollable action
 --
 -- t, t1 --> These variables are not part of the model
 --           we just need them to some existential quantification
 --           to compute the \lambda(X) of a set

automaton Antenna_1
synclabs: jam_x;
initially low_x & x=0;

loc low_x:  while x>=0 wait {dcost_x=-1}
        when x>=10 & flag=1 do {c'=1-c,u'=u,cost'=cost-7} goto Win;
        when True sync jam_x do {x'=0,c'=c,u'=1-u} goto high_x;

loc high_x:  while x>=0 & x<=10 wait {dcost_x=-10}
        when x>=5 & flag=2 do {y'=0,c'=1-c,u'=u} goto low_x;
        when True sync jam_x do {x'=0,c'=c,u'=1-u} goto high_x;

loc Win: while x>=0 wait {dcost_x=0}
end -- Antenna_1


automaton Antenna_2
synclabs: jam_y;
initially low_y & y=0;

loc low_y:  while y>=0 wait {dcost_y=-2}
        when y>=10 & flag=3 do {c'=1-c,u'=u,cost'=cost-1} goto Win;
        when True sync jam_y do {y'=0,c'=c,u'=1-u} goto high_y;

loc high_y:  while y>=0 & y<=10 wait {dcost_y=-20}
```

```
        when y>=2 & flag=4 do {x'=0,c'=1-c,u'=u} goto low_y;
        when True sync jam_y do {y'=0,c'=c,u'=1-u} goto high_y;

loc Win:
        while y>=0 wait {dcost_y=0}
end -- Antenna_2

automaton Jammer
synclabs: jam_x, jam_y;
initially X;

loc X:  while True wait {dcost=dcost_x+dcost_y,dt=-1,dt1=-1}
        when x>6 sync jam_x goto X;
        when x>6 sync jam_y goto Y;

loc Y:  while True wait {dcost=dcost_x+dcost_y,dt=-1,dt1=-1}
        when y>6 sync jam_y goto Y;
        when y>6 sync jam_x goto X;
end -- Jammer


------------------
-- Computations --
------------------

var init,           -- Initial states
    winning,        -- Winning locations
    winning_states, -- Winning states obtained by the backward computation
    winning_strat,  -- Winning states decorated with extra information in
                    -- in order to extract the strategies.
    iterator,       -- Iterator in the fix point computation
    reachable,      -- Reachable regions

    uPre_X,          -- Uncontrollable predecessors leading to winning states
    uPre_nonX,       -- Uncontrollable predecessors of the complement of X
    cPre_X,          -- Controllable predecessors of X
    strict_tPre_X,   -- The set of states from t>0 can elapse
    tPre1X_uPre_nonX,-- Time predecessors of X from which we can reach X
                     -- and avoid \bar{X} all along the time paths leading
                     -- to X.
    tPrecopy,        -- Compute the "interior" of tPre1X_uPrebarX.
                     -- Represents the states from which a time t>0 can
                     -- elapse and we can stay in X.

    r0, r1, r2, r3, r4,

    inf_0_1, inf_0_2, inf_0_3, inf_0_4,
    inf_1_0, inf_1_2, inf_1_3, inf_1_4,
    inf_2_0, inf_2_1, inf_2_3, inf_2_4,
    inf_3_0, inf_3_1, inf_3_2, inf_3_4,
    inf_4_0, inf_4_1, inf_4_2, inf_4_3 : region;
```

```
 -- Initial region --
init := x=0 & y=0 &
        loc[Antenna_1]=low_x & loc[Antenna_2]=low_y & loc[Jammer]=X;

 -- Winning region --
winning := (loc[Antenna_1]=Win | loc[Antenna_2]=Win) & cost>=0 ;

 -- Compute reachable regions --
 -- ------------------------- --
prints "";
prints "Reachable regions";
prints "-----------------";

reachable := iterate reachable from
          (hide cost,cost_x,cost_y,cost0,cost1,cost2,cost3,cost4,
                c,u,t,t1,flag in init endhide) using {
        reachable :=
          (hide cost,cost_x,cost_y,cost0,cost1,cost2,cost3,cost4,
                c,u,t,t1,flag in post(reachable) endhide);
             };


 -- Backward analysis --
 -- ----------------- --
prints "";
prints "Computing the winning states";
prints "---------------------------";

 -- Winning states
winning_states := winning & reachable;

 -- Decorated Winning states
winning_strat := winning_states;

-- -------------------------------------------- --
-- Backward analysis: look for a fix point of \pi --
-- -------------------------------------------- --
iterator := iterate iterator from winning using {

        -- Uncontrollable predecessors of the complement of X
        uPre_nonX := hide t,u in
                      t=0 & u=0 & pre(~iterator & u=1 & t=0)
                     endhide;

        -- Controllable predecessors of X
        -- Note: cPre_X is a union of region of the form flag=k (k>=1) &
        -- something the flag=k indicates which controllable action should
        -- be taken to reach a particular region of X.
        cPre_X := hide t,c in
                   t=0 & c=0  & pre(iterator & t=0 & c=1)
                  endhide ;
```

```
        -- Time predecessors of X from which we can reach X and avoid
        -- \bar{X} all along the time paths leading to X.
        tPre1X_uPre_nonX := hide t in
                            (hide c,u in
                              t>=0 & c=0 & u=0 &
                              pre(iterator & t=0 & c=0 & u=0)
                             endhide) &
                             ~(hide t1 in
                                (hide c,u in
                                  t1>=0 & t1<=t & c=0 & u=0 &
                                  pre(uPre_nonX & t1=0 & c=0 & u=0)
                                 endhide)
                               endhide)
                            endhide;

        -- Compute the interior of tPre1X_uPre_nonX --
        tPrecopy := flag=0 & tPre1X_uPre_nonX &
                 hide t,c,u in
                  t>0 & c=0 & u=0 & pre(tPre1X_uPre_nonX & t=0 & u=0 & c=0)
                 endhide;

        -- We need to hide the value of flag in cPreX because we do not need
        -- it to compute the next iteration and it contrains the state space
        iterator := tPre1X_uPre_nonX
                 | ((hide flag in cPre_X endhide) & ~uPre_nonX) ;

        -- Add this winning regions to the set of winning regions already
        -- computed with the informations of what the cost is if we take
        -- a particular action.
        winning_strat := winning_strat
                      |  tPrecopy
                      |  (cPre_X &  ~uPre_nonX) ;

        -- Add the newly computed regions to the set of computed regions
        winning_states := winning_states | iterator ;

        -- Intersect with reachable regions
        winning_strat := winning_strat & reachable;
        winning_states := winning_states & reachable;
        };

 -- -------- --
 -- End Loop --
 -- -------- --

prints "";
prints "-----------------";
prints "-- Optimal cost --";
prints "-----------------";
print hide x,y,flag in (winning_states & init) endhide;

 -- Removing the winning from the decorated winning states
```

```
winning_strat := winning_strat & ~(winning);

prints "";
prints "--------------";
prints "-- Strategy --";
prints "--------------";

r0 := hide flag,cost in cost0=cost & winning_strat & flag=0  endhide ;
r1 := hide flag,cost in cost1=cost & winning_strat & flag=1  endhide ;
r2 := hide flag,cost in cost2=cost & winning_strat & flag=2  endhide ;
r3 := hide flag,cost in cost3=cost & winning_strat & flag=3  endhide ;
r4 := hide flag,cost in cost4=cost & winning_strat & flag=4  endhide ;


-- compute inf A <= inf B
-- hide a in  A  & ~(hide b in B & b<a  endhide) endhide ;

inf_0_1 := hide cost0 in
            r0  & ~(hide cost1 in r1 & cost1<cost0  endhide)
           endhide ;
inf_0_2 := hide cost0 in
            r0  & ~(hide cost2 in r2 & cost2<cost0  endhide)
           endhide ;
inf_0_3 := hide cost0 in
            r0  & ~(hide cost3 in r3 & cost3<cost0  endhide)
           endhide ;
inf_0_4 := hide cost0 in
            r0  & ~(hide cost4 in r4 & cost4<cost0  endhide)
           endhide ;
inf_1_0 := hide cost1 in
            r1  & ~(hide cost0 in r0 & cost0<cost1  endhide)
           endhide ;
inf_1_2 := hide cost1 in
            r1  & ~(hide cost2 in r2 & cost2<cost1  endhide)
           endhide ;
inf_1_3 := hide cost1 in
            r1  & ~(hide cost3 in r3 & cost3<cost1  endhide)
           endhide ;
inf_1_4 := hide cost1 in
            r1  & ~(hide cost4 in r4 & cost4<cost1  endhide)
           endhide ;
inf_2_0 := hide cost2 in
            r2  & ~(hide cost0 in r0 & cost0<cost2  endhide)
           endhide ;
inf_2_1 := hide cost2 in
            r2  & ~(hide cost1 in r1 & cost1<cost2  endhide)
           endhide ;
inf_2_3 := hide cost2 in
            r2  & ~(hide cost3 in r3 & cost3<cost2  endhide)
           endhide ;
inf_2_4 := hide cost2 in
            r2  & ~(hide cost4 in r4 & cost4<cost2  endhide)
```

```
          endhide ;
inf_3_0 := hide cost3 in
           r3  & ~(hide cost0 in r0 & cost0<cost3  endhide)
          endhide ;
inf_3_1 := hide cost3 in
           r3  & ~(hide cost1 in r1 & cost1<cost3  endhide)
          endhide ;
inf_3_2 := hide cost3 in
           r3  & ~(hide cost2 in r2 & cost2<cost3  endhide)
          endhide ;
inf_3_4 := hide cost3 in
           r3  & ~(hide cost4 in r4 & cost4<cost3  endhide)
          endhide ;
inf_4_0 := hide cost4 in
           r4  & ~(hide cost0 in r0 & cost0<cost4  endhide)
          endhide ;
inf_4_1 := hide cost4 in
           r4  & ~(hide cost1 in r1 & cost1<cost4  endhide)
          endhide ;
inf_4_2 := hide cost4 in
           r4  & ~(hide cost2 in r2 & cost2<cost4  endhide)
          endhide ;
inf_4_3 := hide cost4 in
           r4  & ~(hide cost3 in r3 & cost3<cost4  endhide)
          endhide ;


prints "";
prints "Delay on:";
prints "---------";
print inf_0_1 & inf_0_2 & inf_0_3 & inf_0_4 &
      ~(inf_1_0 | inf_2_0 | inf_3_0 | inf_4_0);

prints "";
prints "Do low_x->Win on:";
prints "-----------------";
print inf_1_0 & inf_1_2 & inf_1_3 & inf_1_4;

prints "";
prints "Do high_x->low_x on:";
prints "--------------------";
print inf_2_0 & inf_2_1 & inf_2_3 & inf_2_4;

prints "";
prints "Do low_y->Win on:";
prints "-----------------";
print inf_3_0 & inf_3_1 & inf_3_2 & inf_3_4;

prints "";
prints "Do high_y->low_y on:";
prints "--------------------";
print inf_4_0 & inf_4_1 & inf_4_2 & inf_4_3;
```

Recent BRICS Report Series Publications

RS-04-4 **Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen.** ***Optimal Strategies in Priced Timed Game Automata*****. February 2004. 32 pp.**

RS-04-3 **Mads Sig Ager, Olivier Danvy, and Jan Midtgaard.** ***A Functional Correspondence between Call-by-Need Evaluators and Lazy Abstract Machines*****. February 2004. 17 pp. This report supersedes the earlier BRICS report RS-03-24. Extended version of an article to appear in** ***Information Processing Letters*****.**

RS-04-2 **Gerth Stølting Brodal, Rolf Fagerberg, Ulrich Meyer, and Norbert Zeh.** ***Cache-Oblivious Data Structures and Algorithms for Undirected Breadth-First Search and Shortest Paths*****. February 2004. 19 pp.**

RS-04-1 **Luca Aceto, Willem Jan Fokkink, Anna Ingólfsdóttir, and Bas Luttik.** ***Split-2 Bisimilarity has a Finite Axiomatization over CCS with Hennessy's Merge*****. January 2004. 16 pp.**

RS-03-53 **Kyung-Goo Doh and Peter D. Mosses.** ***Composing Programming Languages by Combining Action-Semantics Modules*****. December 2003. 39 pp. Appears in** ***Science of Computer Programming*****, 47(1):2–36, 2003.**

RS-03-52 **Peter D. Mosses.** ***Pragmatics of Modular SOS*****. December 2003. 22 pp. Invited paper, published in Kirchner and Ringeissen, editors,** ***Algebraic Methodology and Software Technology: 9th International Conference*****, AMAST '02 Proceedings, LNCS 2422, 2002, pages 21–40.**

RS-03-51 **Ulrich Kohlenbach and Branimir Lambov.** ***Bounds on Iterations of Asymptotically Quasi-Nonexpansive Mappings*****. December 2003. 24 pp.**

RS-03-50 **Branimir Lambov.** ***A Two-Layer Approach to the Computability and Complexity of Real Numbers*****. December 2003. 16 pp.**

RS-03-49 **Marius Mikucionis, Kim G. Larsen, and Brian Nielsen.** ***Online On-the-Fly Testing of Real-time Systems*****. December 2003. 14 pp.**

RS-03-48 **Kim G. Larsen, Ulrik Larsen, Brian Nielsen, Arne Skou, and Andrzej Wasowski.** ***Danfoss EKC Trial Project Deliverables*****. December 2003. 53 pp.**