Counterexample-guided Synthesis of Observation Predicates

Rayna Dimitrova and Bernd Finkbeiner

Saarland University, Germany

Abstract. We present a novel approach to the safety controller synthesis problem with partial observability for real-time systems. This in general undecidable problem can be reduced to a decidable one by fixing the granularity of the controller: finite sets of clocks and constants in the guards. Current state-of-the-art methods are limited to brute-force enumeration of possible granularities or manual choice of a finite set of observations that a controller can track. We address this limitation by proposing a counterexample-guided method to successively refine a set of observations until a sufficiently precise abstraction is obtained. The size of the abstract games and strategies generated by our approach depends on the number of observation predicates and not on the size of the constants in the plant. Our experiments demonstrate that this results in better performance than the approach based on fixed granularity when fine granularity is necessary.

1 Introduction

Controller synthesis, both in the discrete and in the timed setting, has been an active field of research in the last decades. The timed controller synthesis problem asks to automatically find a controller for an open plant such that the controlled closed loop system satisfies a given property. It naturally reduces to the problem of finding a winning strategy for the controller player in a two-player *timed game* between a controller and its environment (the plant). This problem is well-understood for the case that the controller can fully observe the state and evolution of the plant. In reality, however, this assumption is usually violated due to limited sensors or the inability to observe the internal behavior of the plant. The controller must therefore win the game under *partial observability*.

The timed controller synthesis problem is undecidable under partial observability [2]. All known synthesis algorithms therefore rely on some *a-priori* restriction of the problem, such as fixing the *granularity* [2] of the controller by restricting the constants to which clocks may be compared to in the controller to integral multiples of $\frac{1}{m}$, where m is a predefined constant, or fixing a template for the controller [9]. Alternatively, one can predefine the observations of the controller [4, 3], which amounts to providing a *finite set of predicates* over the locations and clocks on which the strategy of the controller may be based. How to efficiently find these observation predicates is an important research question, the only known approach being the brute-force enumeration of *all* possible granularities ($1, \frac{1}{2}, \frac{1}{4}, \ldots$) until a sufficiently precise one is found.

In this paper, we present the first systematic method for the *automatic synthesis of observation predicates*. Before we describe the approach in more detail, let us clarify

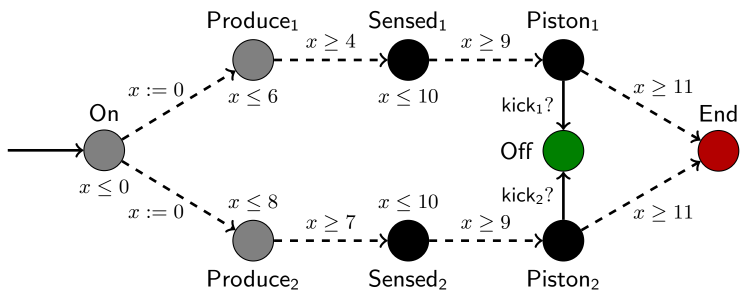


Fig. 1. Example of a partially observable plant for a production system. For readability we have omitted the kick$_1$? and kick$_2$? transitions from all other locations leading to location End.

the role of the observation predicates. Figure 1 shows, as a toy example, the model of a production system. The goal is to kick a box from a conveyor belt using a piston, before the box reaches the end of the belt. The locations On, Producing$_1$, Producing$_2$, Sensed$_1$, Sensed$_2$, Piston$_1$, Piston$_2$, End and Off of the plant indicate the position of the box on the belt. The plant produces two types of boxes, where producing a box of type 1 takes between 4 and 6 seconds and producing a box of type 2 takes between 7 and 8 seconds. However, regardless of its type, the box arrives at the respective location Piston$_1$ or Piston$_2$ between 9 and 10 seconds after the start. The goal of the controller is to avoid location End. For that, it has to execute the correct kick$_1$! or kick$_2$! action at the right time, namely when the box is in the respective location Piston$_1$ or Piston$_2$.

The challenge is that locations On, Produce$_1$ and Produce$_2$ are indistinguishable by the controller, and so are locations Sensed$_1$, Sensed$_2$, Piston$_1$ and Piston$_2$. The controller can only detect the presence of a box via a sensor (i.e, it observes the box entering locations Sensed$_1$ and Sensed$_2$) and use timing information to determine the time-frame in which the box is in location Piston$_1$ (or Piston$_2$). It cannot observe the clock x of the plant, but has its own clock y that it can test and reset. A solution to the synthesis problem is to use a clock y in the controller and activate the piston when $y = 21/2$, thus ensuring that the End is never reached as the box is guaranteed to reach location Piston$_1$ or Piston$_2$ in 9 to 10 seconds after it is sensed and remains there at least until $y = 11$. Additionally, in order to activate the correct piston, the controller needs to distinguish the type of the box. This can be done, again using timing information, by checking whether or not the box has been sensed by time $y = 7$. In order to find a correct controller, we thus need two observation predicates: $y = 21/2$ and $y >= 7$. Clearly, both predicates are necessary: if the controller only observes one of them or only some other predicate, say, only $y = 30$, then it is impossible to enforce the specification. Note also that two predicates play different roles in the control strategy. Predicate $y = 21/2$ identifies a particular point in time (out of the infinitely many) in which the controller may choose to *take an action*, predicate $y >= 7$ identifies an observation that is needed in order to be able to *decide* on the right action. In the following, we distinguish these two types of observation predicates as *action points* and *decision predicates*.

Our method works by successively *refining a finite set of observation predicates* based on the *analysis of spurious counterexamples*. The key is to use timed games with fixed observations as sound abstractions of the original timed game under incomplete

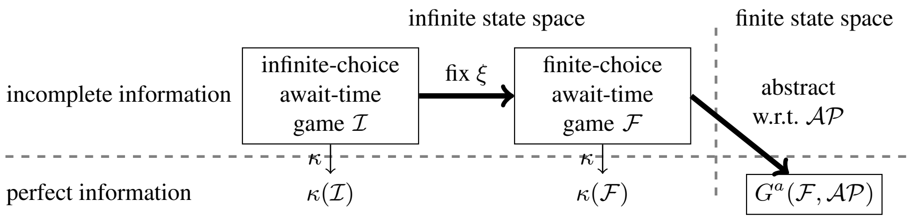

infinite state space
finite state space
incomplete information
infinite-choice await-time game $\mathcal{I}$
fix ξ
finite-choice await-time game $\mathcal{F}$
abstract w.r.t. $\mathcal{AP}$
κ
κ
perfect information
$\kappa(\mathcal{I})$
$\kappa(\mathcal{F})$
$G^a(\mathcal{F}, \mathcal{AP})$

Fig. 2. Overview of the abstraction process. An await-time game $\mathcal{I}$, representing the controller synthesis problem under partial observability, is first abstracted into a finite-choice await-time game $\mathcal{F}$, and then into a finite-state game $G^a(\mathcal{F}, \mathcal{AP})$. Games $\mathcal{I}$ and $\mathcal{F}$ have a possibly infinite state space and are played under incomplete information, game $G^a(\mathcal{F}, \mathcal{AP})$ is finite-state and is played under perfect information.

information. Our method builds on the classic CEGAR loop, where one successively refines the abstraction until either no more abstract counterexamples exist or a concrete counterexample is found. As usual in CEGAR approaches for games [10, 8], a spurious counterexample is a strategy tree for the environment player that is winning in the abstract game but not in the concrete game. For timed games, the main difficulty in the characterization of spurious counterexamples is caused by the fact that the number of moves available to the controller is infinite, corresponding to the infinite number of points in time when an action can be taken. As a result, the logical characterization of spurious counterexamples is a *quantified* formula in the theory of linear arithmetic. In the paper, we present a novel *refinement technique* for generating new observation predicates based on quantifier witnesses that eliminate the given strategy.

Figure 2 gives an overview on the abstraction process. We start with a symbolic representation of the timed safety controller synthesis problem with partial observability, which we call *await-time games*. Await-time games allow us to represent the infinite number of choices available to the controller using controllable variables. Corresponding to the two types of observation predicates, action points and decision predicates, we abstract the initial await-time game in two steps into a finite-state game with perfect information. First, we use the action points to eliminate (abstract away) the controllable variables that range over infinite domains. In the resulting *finite-choice* await-time game $\mathcal{F}$, the number of moves available to the controller is finite, but the number of states may still be infinite. In the second step we abstract $\mathcal{F}$ w.r.t. a finite set of predicates $\mathcal{AP}$ to obtain a finite-state perfect-information game $G^a(\mathcal{F}, \mathcal{AP})$. This abstraction completely fixes a finite set of observation predicates the controller can track. Since $G^a(\mathcal{F}, \mathcal{AP})$ is a finite game, we can apply standard algorithms to solve the game and find a winning strategies for the winning player. For a winning strategy for the environment player, we check if the strategy also wins in $\mathcal{F}$ and in $\mathcal{I}$. Strategies that are spurious in $\mathcal{F}$ can be eliminated with additional decision predicates, strategies that are concretizable in $\mathcal{F}$ but spurious in $\mathcal{I}$ need additional action points. We refine both sets until we find either a strategy for the controller in $G^a(\mathcal{F}, \mathcal{AP})$ or a counterexample concretizable in $\mathcal{I}$.

Related Work. The classic solution for *finite-state discrete games* under incomplete information is due to Reif [11] and is based on a determinization-like translation to perfect-information games with a *knowledge-based subset construction*.

Symbolic fixed-point algorithms based on antichains that avoid this determinization procedure were proposed in [7,5]. While these algorithms are applicable to infinite game graphs with a given finite region algebra, they require an *a priori fixed finite set of observations* that the controller is allowed to track.

Abstraction refinement methods were previously applied to games with *perfect information* [10,6] and to safety games with incomplete information [8]. Games under incomplete information are out of the scope of the first two works. The refinement procedure from [8] is based on the assumption that the controller can *choose a move from a finite set*. Unless a finite set of observations is fixed, this is not the case in real-time systems where the controller can let an arbitrary amount of time elapse.

To the best of our knowledge, prior to this work there was no approach to controller synthesis that can handle partial observability for systems that allow for infinitely many choices of the controller, without fixing a priori a finite set of available observations.

2 Timed Controller Synthesis with Partial Observability

Production system controllers are typically required to satisfy timing requirements formulated as safety properties. In a realistic setting, the information that such controllers have at their disposal is limited by their interface and sensor capabilities. Thus, all decisions in the controller's implementation are made based on (possibly) partial observations about the state and the evolution of the controller's external environment.

In this section we recall standard notions and notation and give a definition of the timed safety control problem with partial observability.

Timed automata and transition systems. Given a set X of real-valued variables, $\mathcal{G}(X)$ is the set of constraints generated by: $\varphi := x \sim c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$, where $x \in X$, $c \in \mathbb{Q}$ and $\sim \in \{<, \leq, >, \geq\}$. $\mathcal{C}(X)$ is the subset of $\mathcal{G}(X)$ that consists of *true* and conjunctions of constraints of the form $x \sim c$. We denote with $\mathbb{R}_{\geq 0}$ ($\mathbb{R}_{>0}$) the sets of non-negative (positive) reals and with $\mathbb{R}^X_{\geq 0}$ the set of total functions from X to $\mathbb{R}_{\geq 0}$. For $v \in \mathbb{R}^X_{\geq 0}$, $Z \subseteq X$ and $t \in \mathbb{R}_{>0}$ we denote with $v[0/Z]$ and with $v + t$ the valuations obtained from v by setting the values of the variables in Z to 0 or adding t to every value in v, respectively. For $v \in \mathbb{R}^X_{\geq 0}$ and $g \in \mathcal{G}(X)$ we write $v \models g$ iff v satisfies g.

For a finite or infinite sequence π of elements of some set A we denote with $|\pi|$ its length, i.e., the number of elements of π and write $|\pi| = \infty$ when π is infinite. For $n \in \mathbb{N}$ with $n < |\pi|$, we denote with $\pi[n]$ and $\pi[0,n]$ the $n+1$-th element of π and the prefix of length $n+1$, respectively. For $\pi \in A^+$, $\mathsf{last}(\pi)$ is the last element of π.

A *timed automaton* [1] is a tuple $\mathcal{A} = (\mathsf{Loc}, X, \Sigma, \mathsf{Inv}, R, l_0)$, where Loc is a finite set of locations, X is a finite set of real-valued clocks, Σ is a finite set of actions, $\mathsf{Inv} : \mathsf{Loc} \to \mathcal{C}(X)$ is a function mapping each location to an invariant and $R \subseteq \mathsf{Loc} \times \Sigma \times \mathcal{G}(X) \times 2^X \times \mathsf{Loc}$ is a finite set of transitions.

The semantics of a timed automaton $\mathcal{A}$ is defined by a *timed transition system* $\mathcal{T} = (S, s_0, \Sigma, \rightarrow)$, where $S = \{(l,v) \in \mathsf{Loc} \times \mathbb{R}^X_{\geq 0} \mid v \models \mathsf{Inv}(l)\}$ is the set of states, $s_0 = (l_0, 0)$ is the initial state, and the transition relation $\rightarrow \subseteq S \times (\Sigma \cup \mathbb{R}_{>0}) \times S$ is such that $((l,v), \sigma, (l',v')) \in \rightarrow$ iff $v \models \mathsf{Inv}(l)$, $v' \models \mathsf{Inv}(l')$ and either $\sigma \in \Sigma$ and there exists $(l, \sigma, g, Z, l') \in R$ with $v \models g$ and $v' = v[0/Z]$, or $\sigma \in \mathbb{R}_{>0}$, $v' = v + \sigma$ and $l' = l$. We write $(l,v) \xrightarrow{\sigma} (l',v')$ as a shortcut for $((l,v), \sigma, (l',v')) \in \rightarrow$.

Timed safety control with partial observability. A *partially observable plant* is a tuple $\mathcal{P} = (\mathcal{A}, \Sigma_{\mathsf{c}}, \Sigma_{\mathsf{u}}, X_{\mathsf{o}}, X_{\mathsf{u}}, =_{\mathsf{o}}^{\mathsf{L}})$, where $\mathcal{A}$ is a timed automaton, Σ_{c} and Σ_{u} partition the set Σ of actions into a set Σ_{c} of *controllable* and a set Σ_{u} of *uncontrollable* actions, X_{o} and X_{u} partition the set X of clocks into a set X_{o} of *observable* and a set X_{u} of *unobservable* clocks, and $=_{\mathsf{o}}^{\mathsf{L}}$ is an *observation equivalence* relation on Loc. We require that $\mathcal{P}$ is *input-enabled*: each $\sigma \in \Sigma_{\mathsf{c}}$ is enabled in every state s in the transition system. We also assume that at every state a transition from Σ_{u} is enabled or time can elapse.

A controller for a partially observable plant operates under incomplete information about the location the plant is in and about the values of the plant's clocks. It knows the equivalence class of Loc w.r.t. $=_{\mathsf{o}}^{\mathsf{L}}$ in which the plant currently is. The equivalence class of a location $l \in \mathsf{Loc}$ is denoted $[l]_{=_{\mathsf{o}}^{\mathsf{L}}}$. The controller observes the values of the observable clocks X_{o} and doesn't observe those of the clocks in X_{u}.

Let us consider a partially observable plant $\mathcal{P} = (\mathcal{A}, \Sigma_{\mathsf{c}}, \Sigma_{\mathsf{u}}, X_{\mathsf{o}}, X_{\mathsf{u}}, =_{\mathsf{o}}^{\mathsf{L}})$ with $\mathcal{A} = (\mathsf{Loc}, X, \Sigma, \mathsf{Inv}, R, l_0)$ and let X_{c} be a finite set of clocks with $X_{\mathsf{c}} \cap X = \emptyset$.

We note with $X_{\mathsf{o+c}} = X_{\mathsf{o}} \dot{\cup} X_{\mathsf{c}}$ the union of the observable clocks of the plant and the clocks X_{c}, which are the clocks that belong to the controller. Let $\Sigma_{\mathsf{r}} = \{reset_Z \mid Z \in 2^{X_{\mathsf{c}}} \setminus \{\emptyset\}\}$ be a set of actions disjoint from Σ used to model resets of clocks in X_{c}. Let us define $\widetilde{S} = S \times \mathbb{R}_{\geq 0}^{X_{\mathsf{c}}}$ and $\widetilde{\Sigma} = \Sigma \cup \mathbb{R}_{>0} \cup \Sigma_{\mathsf{r}}$.

A X_{c}-*control strategy for the partially observable plant* $\mathcal{P}$ is a total function $f : \widetilde{S} \cdot (\widetilde{\Sigma} \times \widetilde{S})^* \to (\Sigma_{\mathsf{c}} \cup (\Sigma_{\mathsf{c}} \times X_{\mathsf{o+c}} \times \mathbb{Q}_{>0}) \cup \{\bot\} \cup \Sigma_{\mathsf{r}})$, which maps finite execution histories to decisions of the controller, which can either be to execute a controllable action σ immediately ($f(\pi) = \sigma$) or when an observable clock x reaches the future time c ($f(\pi) = (\sigma, x, c)$), to remain idle ($f(\pi) = \bot$), or to immediately reset a set of controllable clocks Z ($f(\pi) = reset_Z$). We require the following:

- if the controller decides to execute $\sigma \in \Sigma_{\mathsf{c}}$ when x reaches c: $f(\pi) = (\sigma, x, c)$, then c is greater than the current value of x, i.e., $c > v(x)$, where $\mathsf{last}(\pi) = (l, v, v_c)$;
- if the controller decides to reset clocks $Z \subseteq X_{\mathsf{c}}$: $f(\pi) = reset_Z$, then these clocks have positive values, i.e., $v_c(x) > 0$ for every $x \in Z$, where $\mathsf{last}(\pi) = (l, v, v_c)$;
- the value of f changes only when the observation changes or an action in $\Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{r}}$ is executed, i.e., $f(\pi(l, v, v_c)\sigma(l', v', v_c')) = f(\pi(l, v, v_c))$ whenever $\sigma \in \mathbb{R}_{>0}$ or $\sigma \in \Sigma_{\mathsf{u}}$ and $l =_{\mathsf{o}}^{\mathsf{L}} l'$ and for every $x \in X_{\mathsf{o}}$, $v'(x) = 0$ only if $v(x) = 0$;
- f is *consistent with the observations of the controller*, that is $f(\pi_1) = f(\pi_2)$ for every $\pi_1 \equiv \pi_2$, where the equivalence relation $\equiv$ is defined below.

We first define a function $\mathsf{obs} : \widetilde{S} \cdot (\widetilde{\Sigma} \cdot \widetilde{S})^* \to \widetilde{S} \cdot (\widetilde{S} \times \widetilde{S})^*$ that maps a sequence π to the sequence $\mathsf{obs}(\pi)$ that consists of exactly those transitions of π where a controllable action is taken or a discrete change of the state-based observation occurs. If $\pi = \widetilde{s}$, then $\mathsf{obs}(\pi) = \widetilde{s}$. Otherwise let $\pi = \pi'(l, v, v_c)\sigma(l', v', v_c')$. If either $\sigma \in \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{r}}$, or $\sigma \in \Sigma_{\mathsf{u}}$ and $l \neq_{\mathsf{o}}^{\mathsf{L}} l'$ or for some $x \in X_{\mathsf{o}}$, $v(x) > 0$ and $v'(x) = 0$, then $\mathsf{obs}(\pi) = \mathsf{obs}(\pi'(l, v, v_c))((l, v, v_c), (l', v', v_c'))$. Otherwise, $\mathsf{obs}(\pi) = \mathsf{obs}(\pi'(l, v, v_c))$.

For $\pi_1, \pi_2 \in \widetilde{S} \cdot (\widetilde{S} \times \widetilde{S})^*$ we define $\pi_1 \equiv \pi_2$ iff $|\mathsf{obs}(\pi_1)| = |\mathsf{obs}(\pi_2)|$ and:

- if $\mathsf{obs}(\pi_1)[0] = (l_1^0, v_1^0, v_{c1}^0)$ and $\mathsf{obs}(\pi_2)[0] = (l_2^0, v_2^0, v_{c2}^0)$, then we have $l_1^0 =_{\mathsf{o}}^{\mathsf{L}} l_2^0$, $v_{c1}^0 = v_{c1}^0$ and for every $x \in X_{\mathsf{o}}$ it holds that $v_1^0(x) = v_2^0(x)$, and
- for every $0 < i < |\mathsf{obs}(\pi_1)|$ with $\mathsf{obs}(\pi_1)[i] = ((l_1, v_1, v_{c1}), (l_1', v_1', v_{c1}'))$ and $\mathsf{obs}(\pi_2)[i] = ((l_2, v_2, v_{c2}), (l_2', v_2', v_{c2}'))$, we have $l_1 =_{\mathsf{o}}^{\mathsf{L}} l_2$, $l_1' =_{\mathsf{o}}^{\mathsf{L}} l_2'$, $v_{c1} = v_{c2}$, $v_{c1}' = v_{c2}'$, and for every $x \in X_{\mathsf{o}}$, $v_1(x) = v_2(x)$ and $v_1'(x) = v_2'(x)$.

A control strategy f for the plant $\mathcal{P}$ defines a set of *controlled paths* $\mathcal{CP}(f,\mathcal{P}) \subseteq \widetilde{S} \cdot ((\widetilde{\Sigma} \cdot \widetilde{S})^* \cup (\widetilde{\Sigma} \cdot \widetilde{S})^\omega)$, where $\pi \in \mathcal{CP}(f,\mathcal{P})$ iff $\pi[0] = (l_0, 0, 0)$ and for every $0 < i < |\pi| - 1$ with $\pi[i-1] = (l, v, v_c)$, $\pi[i] = \sigma$ and $\pi[i+1] = (l', v', v_c')$ we have:

- if $\sigma \in \mathbb{R}_{>0}$, then $(l,v) \xrightarrow{\sigma} (l', v')$, $v_c' = v_c + \sigma$ and either $f(\pi[0, i-1]) = \bot$ or $f(\pi[0, i-1]) = (a, x, c)$ and $v'(x) \leq c$ (time can elapse only until x reaches c);
- if $\sigma \in \Sigma_\mathsf{c}$, then $(l,v) \xrightarrow{\sigma} (l', v')$, $v_c' = v_c$ and either $f(\pi[0, i-1]) = \sigma$ or $f(\pi[0, i-1]) = (\sigma, x, c)$ and $v(x) = c$ (σ is taken immediately or x has reached c);
- if $\sigma \in \Sigma_\mathsf{u}$, then $f(\pi[0, i-1]) \notin \Sigma_\mathsf{r}$, $(l,v) \xrightarrow{\sigma} (l', v')$ and $v_c' = v_c$;
- if $\sigma = \mathit{reset}_Z$, then $f(\pi[0, i-1]) = \mathit{reset}_Z$, $v_c' = v_c[0/Z]$, $v' = v$, $l' = l$.

A location $l \in \mathsf{Loc}$ is *reachable* in $\mathcal{CP}(f,\mathcal{P})$ iff there exists a finite path $\pi \in \mathcal{CP}(\pi)$ such that $\mathsf{last}(\pi) = (l, v, v_c)$ for some $v \in \mathbb{R}_{\geq 0}^X$ and $v_c \in \mathbb{R}_{\geq 0}^{X_\mathsf{c}}$.

We can now state the *timed safety control synthesis problem with partial observability*: Given a partially observable plant $\mathcal{P} = (\mathcal{A}, \Sigma_\mathsf{c}, \Sigma_\mathsf{u}, X_\mathsf{o}, X_\mathsf{u}, =_\mathsf{o}^\mathsf{L})$ with underlying timed automaton $\mathcal{A} = (\mathsf{Loc}, X, \Sigma, \mathsf{Inv}, R, l_0)$, an *error location* $l_\mathsf{bad} \in \mathsf{Loc}$, for which $[l_\mathsf{bad}]_{=_\mathsf{o}^\mathsf{L}} = \{l_\mathsf{bad}\}$, and a finite set X_c of clocks with $X_\mathsf{c} \cap X = \emptyset$, find a finite-state X_c-control strategy f for $\mathcal{P}$ such that l_bad is not reachable in $\mathcal{CP}(f,\mathcal{P})$ or determine that there does not exist a X_c-control strategy for $\mathcal{P}$.

3 Await-Time Games

In this section we introduce *await-time games* and show that the timed safety controller synthesis problem with partial observability reduces to the problem of finding a winning strategy for the controller in an await-time game against its environment (the plant).

Let $\mathcal{P} = (\mathcal{A}, \Sigma_\mathsf{c}, \Sigma_\mathsf{u}, X_\mathsf{o}, X_\mathsf{u}, =_\mathsf{o}^\mathsf{L})$ where $\mathcal{A} = (\mathsf{Loc}, X, \Sigma, \mathsf{Inv}, R, l_0)$ be a partially observable plant fixed for the rest of the paper, together with an error location $l_\mathsf{bad} \in \mathsf{Loc}$, for which $[l_\mathsf{bad}]_{=_\mathsf{o}^\mathsf{L}} = \{l_\mathsf{bad}\}$. Let X_c be a fixed finite set of clocks with $X_\mathsf{c} \cap X = \emptyset$.

An await-time game models the interaction between a X_c-control strategy $\mathsf{Player_c}$, and the partially observable plant $\mathcal{P}$, i.e, the controller's environment $\mathsf{Player_e}$, in a turn-based manner. Whenever it is his turn, $\mathsf{Player_c}$ has the possibility to propose what controllable action should be executed and when. Then, $\mathsf{Player_e}$ can do one or more transitions executing the actual actions of the plant, i.e., updating the location and all clocks, respecting the choice of $\mathsf{Player_c}$. Since the controller and the plant synchronize when a controllable action is executed or a discrete change in the state-based observation has occurred, the turn is back to $\mathsf{Player_c}$ as soon as this happens. More precisely, $\mathsf{Player_c}$ can choose **(C1)** an action $\sigma \in \Sigma_\mathsf{c}$ to be executed after a positive delay, or **(C2)** an action $\sigma \in \Sigma_\mathsf{c}$ to be executed without delay, or **(C3)** to remain idle, or **(C4)** a set of clocks from X_c to be reset immediately. $\mathsf{Player_e}$ can do transitions that correspond to **(E1)** the time-elapsing transitions in the plant, **(E2)** the discrete controllable and **(E3)** uncontrollable transitions in the plant, as well as transitions that let $\mathsf{Player_e}$ **(E4)** reset the controllable clocks selected by $\mathsf{Player_c}$, or **(E5)** give the turn to $\mathsf{Player_c}$.

Formally, an *await-time game* $\mathcal{I}(\mathcal{P}, l_\mathsf{bad}, X_\mathsf{c}) = (V_\mathsf{c}, V_\mathsf{o}, V_\mathsf{u}, \iota, \mathcal{T}_\mathsf{c}, \mathcal{T}_\mathsf{e}, \varphi_\mathsf{bad})$ is a tuple consisting of pairwise disjoint sets V_c, V_o and V_u of *controllable*, *observable* and *unobservable* variables respectively, and formulas ι, φ_bad, $\mathcal{T}_\mathsf{c}$ and $\mathcal{T}_\mathsf{e}$ that denote the sets of initial and error states and the transition relations for $\mathsf{Player_c}$ and $\mathsf{Player_e}$ respectively.

The states of the game are described by the finite set $V = V_{\mathsf{c}} \dot\cup V_{\mathsf{o}} \dot\cup V_{\mathsf{u}}$ of variables. We assume a designated boolean variable $t \in V_{\mathsf{c}}$ that determines which player choses a successor (i.e, updates his variables) in a given state. The transition relations of the players are given as formulas over V and the set V' of primed versions of the variables.

$\mathsf{Player_c}$ updates the variables in V_{c}, which model the decisions of the controller. A variable $act \in V_{\mathsf{c}}$ with $\mathsf{Dom}(act) = \Sigma_{\mathsf{c}} \cup \{\bot\}$ indicates the selected controllable action in cases **(C1)** and **(C2)** (and is $\bot$ in cases **(C3)** and **(C4)**). In case **(C1)**, $\mathsf{Player_c}$ also proposes for at least one clock variable $x \in X_{\mathsf{o+c}}$ a positive constant, called *await point*, indicating that he wants to execute the selected action as soon as x reaches this value, which must be strictly greater than the current value of x. For this, V_{c} contains a subset $SC = \{c_x \mid x \in X_{\mathsf{o+c}}\}$ of variables, called *symbolic constants*, ranging over $\mathbb{Q}_{\geq 0}$. V_{c} contains variables $wait$ and $reset$ with $\mathsf{Dom}(wait) = \mathbb{B}$ and $\mathsf{Dom}(reset) = 2^{X_{\mathsf{c}}}$.

$\mathsf{Player_e}$ updates the variables in $V_{\mathsf{o}} \dot\cup V_{\mathsf{u}} \dot\cup \{t\}$. The set V_{o} contains the clocks in $X_{\mathsf{o+c}}$ and a variable $oloc$ for modeling the equivalence class of the current location of the plant. The set V_{u} contains the clocks in X_{u} and a variable loc for modeling the plant's location. The auxiliary boolean variable $er \in V_{\mathsf{o}}$ indicates in which states $\mathsf{Player_c}$ can choose to reset clocks in X_{c} and the auxiliary boolean variable $et \in V_{\mathsf{u}}$ is false in states where $\mathsf{Player_e}$ has disabled further time-elapse transitions.

$\mathsf{Player_c}$ has incomplete information about the state of the game, which includes the state of the plant – location and clock valuations. He observes only the variables in $V_{\mathsf{o+c}} = V_{\mathsf{c}} \dot\cup V_{\mathsf{o}}$ and is thus oblivious to the current location and valuation of X_{u}.

Since a controller for a partially observable plant does not observe the plant continuously, but only at the points of synchronization, we need to ensure that $\mathsf{Player_c}$ cannot win the game only by basing his strategic choices on the number of unobservable steps in the play so far, i.e, that if he can win the game then he can do so with a *stuttering invariant strategy* [4]. To this end, we include in the game $\mathcal{I}$ a *skip*-transition for $\mathsf{Player_e}$, which is enabled in each state that belongs to $\mathsf{Player_e}$ and allows for making a transition without changing the values of any variables. This transformation is sound for games with safety winning conditions defined by a set of bad states. That way, we will ensure that winning strategies for $\mathsf{Player_c}$ correspond to X_{c}-control strategies for $\mathcal{P}$.

For the rest of the paper, we denote with $\mathcal{I}$ the await-time game $\mathcal{I}(\mathcal{P}, l_{\mathsf{bad}}, X_{\mathsf{c}})$.

The formulas ι and φ_{bad} assert respectively that all variables are properly initialized and that $loc = l_{\mathsf{bad}}$. The formulas $\mathcal{T}_{\mathsf{c}}$ and $\mathcal{T}_{\mathsf{e}}$ assert that the players update their variables according to the rules above. Instead of the respective formulas, we give the transition relations of the corresponding explicit game $G(\mathcal{I}) = (Q_{\mathsf{c}}, Q_{\mathsf{e}}, q_0, T_{\mathsf{c}}, T_{\mathsf{e}}, =_{\mathsf{o}}, B)$.

(C1)	$q'(act) \in \Sigma_{\mathsf{c}}$, $q'(wait) = \mathsf{true}$, $q'(reset) = \emptyset$, and $q'(c_x) > 0$ for some $x \in X_{\mathsf{o+c}}$, and for every $x \in X_{\mathsf{o+c}}$ with $q'(c_x) > 0$ we have $q'(c_x) > q(x)$
(C2)	$q'(act) \in \Sigma_{\mathsf{c}}$, $q'(wait) = \mathsf{false}$, $q'(reset) = \emptyset$ and $q'(c) = 0$ for all $c \in SC$
(C3)	$q'(act) = \bot$, $q'(wait) = \mathsf{true}$, $q'(reset) = \emptyset$ and $q'(c) = 0$ for all $c \in SC$
(C4)	$q(er) = \mathsf{true}$ (resetting controllable clocks is allowed only after a controllable action or discrete change of the observation), $q(x) > 0$ for every $x \in q'(reset)$, $q'(act) = \bot$, $q'(wait) = \mathsf{false}$, $q'(reset) \in 2^{X_{\mathsf{c}}} \setminus \{\emptyset\}$ and $q'(c) = 0$ for all $c \in SC$

Fig. 3. Transition relation T_{c} of $\mathsf{Player_c}$: for states q and q', $(q, q') \in T_{\mathsf{c}}$ iff $q(t) = \mathsf{true}$, $q'(t) = \mathsf{false}$, $q'|_{(V_{\mathsf{o}} \cup V_{\mathsf{u}})} = q|_{(V_{\mathsf{o}} \cup V_{\mathsf{u}})}$ and one of the conditions **(C1)**, **(C2)**, **(C3)**, **(C4)** holds.

(E1)	$q(wait) = \mathsf{true}$, $q(et) = \mathsf{true}$ (time-elapse trans. enabled) and for some $\sigma \in \mathbb{R}_{>0}$: – $(q(loc), q\vert_X) \xrightarrow{\sigma} (q'(loc), q'\vert_X)$, $q'(oloc) = q(oloc)$, and $q'\vert_{X_\mathsf{c}} = q\vert_{X_\mathsf{c}} + \sigma$, – if $q(act) \neq \bot$, then for every $x \in X_{\mathsf{o+c}}$ with $q(x) < q(c_x)$, we have $q'(x) \leq q(c_x)$ (cannot let time pass beyond an await point if $\mathsf{Player}_\mathsf{c}$ chose an action in Σ_c), – if $q(act) \neq \bot$, then $q'(et) = \mathsf{false}$ iff $q'(x) \geq q(c_x)$ and $q(x) < q(c_x)$ for some $x \in X_{\mathsf{o+c}}$ with $q(c_x) > 0$ (disable time-elapse transitions upon reaching an await point), – $q'(t) = \mathsf{false}$, $q'(er) = q(er)$
(E2)	$q(wait) = \mathsf{false}$ or $q(et) = \mathsf{false}$ (time-elapse trans. disabled), $q(act) = \sigma \in \Sigma_\mathsf{c}$ and: – $(q(loc), q\vert_X) \xrightarrow{\sigma} (q'(loc), q'\vert_X)$, $q'(oloc) = [q'(loc)]_{=^{\mathsf{L}}_{\mathsf{o}}}$, and $q'\vert_{X_\mathsf{c}} = q'\vert_{X_\mathsf{c}}$, – $q'(t) = \mathsf{true}$ (the turn is back to $\mathsf{Player}_\mathsf{c}$), $q'(er) = \mathsf{true}$, $q'(et) = \mathsf{true}$
(E3)	$q(reset) = \emptyset$ and for some $\sigma \in \Sigma_\mathsf{u}$: – $(q(loc), q\vert_X) \xrightarrow{\sigma} (q'(loc), q'\vert_X)$, $q'(oloc) = [q'(loc)]_{=^{\mathsf{L}}_{\mathsf{o}}}$, and $q'\vert_{X_\mathsf{c}} = q'\vert_{X_\mathsf{c}}$, – $q'(t) = \mathsf{true}$ iff $q'(oloc) \neq q(oloc)$ or $q(x) > 0$ and $q'(x) = 0$ for some $x \in X_\mathsf{o}$ (the turn is back to $\mathsf{Player}_\mathsf{c}$ iff the observation changed), and if $q'(t) = \mathsf{true}$, then $q'(er) = \mathsf{true}$ and $q'(et) = \mathsf{true}$, otherwise $q'(er) = q(er)$ and $q'(et) = q(et)$
(E4)	$q(reset) = Z \neq \emptyset$ ($\mathsf{Player}_\mathsf{c}$ chose to reset the clocks in Z) and: – $q'(x) = 0$ for every $x \in Z$ and $q'(x) = q(x)$ for every $x \in (X \cup X_\mathsf{c}) \setminus Z$, – $q'(loc) = q(loc)$, $q'(oloc) = q(oloc)$, $q'(t) = \mathsf{true}$, $q'(er) = \mathsf{false}$, $q'(et) = \mathsf{true}$ (the turn is back to $\mathsf{Player}_\mathsf{c}$ and resetting controllable clocks is disabled)
(E5)	$q(wait) = \mathsf{true}$, $q(act) = \bot$, $q(et) = \mathsf{false}$ ($\mathsf{Player}_\mathsf{c}$ chose to remain idle and $\mathsf{Player}_\mathsf{e}$ has disabled further time-elapse transitions, after making at least one) and: – $q'(loc) = q(loc)$, $q'(oloc) = q(oloc)$ and $q'(x) = q(x)$ for every $x \in X \cup X_\mathsf{c}$, – $q'(t) = \mathsf{true}$ (give the turn back to $\mathsf{Player}_\mathsf{c}$), $q'(er) = \mathsf{false}$, $q'(et) = \mathsf{true}$

Fig. 4. Transition relation T_e of $\mathsf{Player}_\mathsf{e}$: for states q and q', $(q, q') \in T_\mathsf{e}$ iff $q(t) = \mathsf{false}$, $q'\vert_{(V_\mathsf{c} \setminus \{t\})} = q\vert_{(V_\mathsf{c} \setminus \{t\})}$ and one of the conditions **(E1)**, **(E2)**, **(E3)**, **(E4)**, **(E5)** holds, or $q' = q$.

The states of $G(\mathcal{I})$ are valuations of V, i.e., elements of the set $\mathsf{Vals}(V)$ that consists of all total functions $q : V \to \bigcup_{x \in V} \mathsf{Dom}(x)$ such that $q(x) \in \mathsf{Dom}(x)$ for every $x \in V$. For $q \in \mathsf{Vals}(V)$ and $U \subseteq V$, we denote $q\vert_U$ the projection of q onto U.

The states $Q = Q_\mathsf{c} \dot{\cup} Q_\mathsf{e} = \mathsf{Vals}(V)$ are partitioned into those $Q_\mathsf{c} = \{q \in Q \mid q(t) = \mathsf{true}\}$ that belong to $\mathsf{Player}_\mathsf{c}$ and those $Q_\mathsf{e} = \{q \in Q \mid q(t) = \mathsf{false}\}$ that belong to $\mathsf{Player}_\mathsf{e}$. The initial state $q_0 \in Q_\mathsf{c}$ is the unique state that satisfies ι and the set B of error states consists of all states in Q_e that satisfy φ_bad.

The *observation equivalence* $=_\mathsf{o}$ on Q is defined by the partitioning of the variables in V as follows: $q_1 =_\mathsf{o} q_2$ iff $q_1\vert_{V_{\mathsf{o+c}}} = q_2\vert_{V_{\mathsf{o+c}}}$.

The transition relation $T = T_\mathsf{c} \dot{\cup} T_\mathsf{e}$ is partitioned into the transition relations T_c for $\mathsf{Player}_\mathsf{c}$ and T_e for $\mathsf{Player}_\mathsf{e}$, which are defined in Fig. 3 and Fig. 4, respectively.

A *path* in $\mathcal{I}$ is a finite or infinite sequence $\pi \in Q^* \cup Q^\omega$ of states such that for all $1 \leq n < \vert\pi\vert$, we have $(\pi[n-1], \pi[n]) \in T$. We call a path π maximal iff π is infinite or $\mathsf{last}(\pi) \in B$. A *play* (*prefix*) in $\mathcal{I}$ is a maximal path (finite path) π in $\mathcal{I}$ such that $\pi[0] = q_0$. The extension of $=_\mathsf{o}$ to paths is straightforward. We denote with $\mathsf{prefs}_\mathsf{c}(\mathcal{I})$ ($\mathsf{prefs}_\mathsf{e}(\mathcal{I})$) the set of prefixes π in $\mathcal{I}$ such that $\mathsf{last}(\pi) \in Q_\mathsf{c}$ ($\mathsf{last}(\pi) \in Q_\mathsf{e}$).

A *strategy for* $\mathsf{Player}_\mathsf{c}$ is a total function $f_\mathsf{c} : \mathsf{prefs}_\mathsf{c}(\mathcal{I}) \to \mathsf{Vals}(V_\mathsf{c})$ mapping prefixes to valuations of V_c such that for every $\pi \in \mathsf{prefs}_\mathsf{c}(\mathcal{I})$ there exists $q \in Q$ with $(\mathsf{last}(\pi), q) \in T_\mathsf{c}$ such that $f_\mathsf{c}(\pi) = q\vert_{V_\mathsf{c}}$, and which is consistent w.r.t. $=_\mathsf{o}$: for all $\pi_1, \pi_2 \in \mathsf{prefs}_\mathsf{c}(\mathcal{I})$ with $\pi_1 =_\mathsf{o} \pi_2$ it holds that $f_\mathsf{c}(\pi_1) = f_\mathsf{c}(\pi_2)$. A strategy for

$\mathsf{Player_e}$ is a total function $f_\mathsf{e} : \mathsf{prefs_e}(\mathcal{I}) \to \mathsf{Vals}(V_\mathsf{o} \cup V_\mathsf{u} \cup \{t\})$ such that for every $\pi \in \mathsf{prefs_e}(\mathcal{I})$ there is a $q \in Q$ with $(\mathsf{last}(\pi), q) \in T_\mathsf{e}$ and $f_\mathsf{e}(\pi) = q|_{(V_\mathsf{o} \cup V_\mathsf{u} \cup \{t\})}$.

The *outcome* of a strategy f_c for $\mathsf{Player_c}$ is the set $\mathsf{outcome}(f_\mathsf{c})$ of plays such that $\pi \in \mathsf{outcome}(f_\mathsf{c})$ iff for every $0 < n < |\pi|$ with $\pi[n-1] \in Q_\mathsf{c}$ it holds that $\pi[n]|_{V_\mathsf{c}} = f_\mathsf{c}(\pi[0, n-1])$. A strategy f_c for $\mathsf{Player_c}$ is *winning* if for every $\pi \in \mathsf{outcome}(f_\mathsf{c})$ and every $n \geq 0$, $\pi[n] \notin B$. The outcome of a strategy f_e for $\mathsf{Player_e}$ is defined analogously and f_e is *winning* if for every $\pi \in \mathsf{outcome}(f_\mathsf{e})$ there exists a $n \geq 0$ such that $\pi[n] \in B$. For a strategy f we denote with $\mathsf{prefs}(f)$ the set of all prefixes in $\mathsf{outcome}(f)$.

We can reduce the timed safety control synthesis problem with partial observability to finding a finite-state winning strategy for $\mathsf{Player_c}$ in $\mathcal{I}$.

Proposition 1. *There exists a finite-state X_c-control strategy for the partially observable plant $\mathcal{P}$ with error location l_bad, such that l_bad is not reachable in $\mathcal{CP}(f, \mathcal{P})$ iff* $\mathsf{Player_c}$ *has a finite-state winning strategy in the await-time game $\mathcal{I}(\mathcal{P}, l_\mathsf{bad}, X_\mathsf{c})$.*

4 Abstracting Await-Time Games

In this section we describe an abstraction-based approach to solving await-time games. A finite-state abstract game with perfect information is constructed in two steps. In the first step we construct an await-time game with fixed action points $\mathcal{F}$ by abstracting away the symbolic constants, and thus leaving $\mathsf{Player_c}$ with a finite state of possible choices in each of its states, and letting $\mathsf{Player_e}$ resolve the resulting nondeterminism. In the second step we do predicate abstraction of $\mathcal{F}$ w.r.t. a finite set $\mathcal{AP}$ of predicates and thus completely fix the set of (observation) predicates that the controller can track.

Step 1: Fixing the action points. A *finite-choice await-time game* $\mathcal{F}(\mathcal{I}, \xi)$ for the game $\mathcal{I}$ is defined by an *action-point function* $\xi : X_{\mathsf{o+c}} \to 2^{\mathbb{Q}_{>0}}$. For each clock $x \in X_{\mathsf{o+c}}$, the set $\xi(x) \subseteq \mathbb{Q}_{>0}$ is a *finite* set of positive rational constants called *action points for* x. The action points for a clock $x \in X_{\mathsf{o+c}}$ are used to replace the symbolic constant c_x from $\mathcal{I}$ in $\mathcal{F}(\mathcal{I}, \xi)$ as we describe below.

Formally, $\mathcal{F}(\mathcal{I}, \xi) = (V_\mathsf{c}^\mathsf{f}, V_\mathsf{o}, V_\mathsf{u}, \iota^\mathsf{f}, \mathcal{T}_\mathsf{c}^\mathsf{f}, \mathcal{T}_\mathsf{e}^\mathsf{f}, \varphi_\mathsf{bad})$ is a symbolic game that differs from $\mathcal{I}$ in the set of controllable variables, the formulas for the transition relations and the formula describing the initial state. We define $V_\mathsf{c}^\mathsf{f} = V_\mathsf{c} \setminus SC$. Thus, the formula ι^f and the transition formula $\mathcal{T}_\mathsf{c}$ for $\mathsf{Player_c}$ do not contain assignments to SC.

The possible options for $\mathsf{Player_c}$ in $\mathcal{F}$ are the same as in $\mathcal{I}$ except for **(C1)**, which is replaced by **(C1$^\mathsf{f}$)** where $\mathsf{Player_c}$ selects an action $\sigma \in \Sigma_\mathsf{c}$ to be executed after a delay determined by $\mathsf{Player_e}$. As now $\mathsf{Player_c}$ updates only the finite-range variables V_c^f, from each $\mathsf{Player_c}$-state he can choose among finitely many possible successors.

The nondeterminism resulting from replacing **(C1)** by **(C1$^\mathsf{f}$)**, i.e, regarding exactly how much time should elapse before the selected controllable action σ is executed, is resolved by $\mathsf{Player_e}$. The action σ can be fired at any time *up to (and including) the first action point* reached after a positive amount of time has elapsed. This is achieved by replacing **(E1)** by **(E1$^\mathsf{f}$)**, where $\mathsf{Player_e}$ can choose to disable further delay transitions at any point. Furthermore, according to **(E1$^\mathsf{f}$)** the duration of the time-elapse transitions is constrained by the action points, regardless of whether $\mathsf{Player_c}$ has chosen to execute a controllable action after a delay or to remain idle. This allows $\mathsf{Player_c}$ to remain idle until reaching an action point and then choose to execute a controllable action.

By the definition, in the game $\mathcal{F}(\mathcal{I},\xi)$ $\mathsf{Player}_\mathsf{e}$ is more powerful than in the game $\mathcal{I}$, while $\mathsf{Player}_\mathsf{c}$ is weaker. Thus, $\mathcal{F}(\mathcal{I},\xi)$ soundly abstracts the await-time game $\mathcal{I}$.

Proposition 2. *For every action-point function $\xi : X_{\mathsf{o}+\mathsf{c}} \to 2^{\mathbb{Q}_{>0}}$, if* $\mathsf{Player}_\mathsf{c}$ *has a (finite-state) winning strategy in the finite-choice await-time game $\mathcal{F}(\mathcal{I},\xi)$, then* $\mathsf{Player}_\mathsf{c}$ *has a (finite-state) winning strategy in the await-time game $\mathcal{I}$.*

Step 2: Predicate abstraction. We now consider a finite set $\mathcal{AP}$ of predicates that contains at least the atomic formulas occurring in some of ι^f and φ_bad. We further require that $\mathcal{AP}$ is precise w.r.t. the (finitely many) choices of $\mathsf{Player}_\mathsf{c}$ in the game $\mathcal{F}$ and w.r.t. every finite-range $v \in V_\mathsf{o}$. That is, $\mathcal{AP}$ contains all boolean variables from $V_{\mathsf{o}+\mathsf{c}}$ plus the predicate $v = d$ for every finite-range $v \in V_{\mathsf{o}+\mathsf{c}}$ and $d \in \mathsf{Dom}(v)$.

We ensure that $\mathcal{AP}$ is precise w.r.t. the action points in $\mathcal{F}$ by including the predicates $x \leq c$ and $x \geq c$ for each $x \in X_{\mathsf{o}+\mathsf{c}}$ and $c \in \xi(x)$. Thus, the observable predicates in $\mathcal{AP}$ (i.e., those referring only to variables in $V_{\mathsf{o}+\mathsf{c}}$) are exactly the observation predicates the controller can track in the current abstraction.

We employ the abstraction procedure from [8] to construct a *finite-state perfect-information* abstract game $G^a(\mathcal{F},\mathcal{AP}) = (Q^a_\mathsf{c}, Q^a_\mathsf{e}, q^a_0, T^a_\mathsf{c}, T^a_\mathsf{e}, B^a)$, which is an explicit safety game. The set of abstract states $Q^a = Q^a_\mathsf{c} \dot\cup Q^a_\mathsf{e}$ is partitioned into the sets of states Q^a_c and Q^a_e that belong to $\mathsf{Player}_\mathsf{c}$ and $\mathsf{Player}_\mathsf{e}$ respectively. The game has a unique initial state q^a_0, and set of error states B^a. The abstract transition relation $T^a = T^a_\mathsf{c} \dot\cup T^a_\mathsf{e}$ is partitioned into T^a_c and T^a_e for the two players and is such that $T^a_\mathsf{c} \subseteq Q^a_\mathsf{c} \times Q^a_\mathsf{e}$ and $T^a_\mathsf{e} \subseteq Q^a_\mathsf{e} \times Q^a$ and each state in Q^a has a successor.

Here a strategy for $\mathsf{Player}_\mathsf{p}$, where $p \in \{\mathsf{c},\mathsf{e}\}$ is a total function $f^a_\mathsf{p} : \mathsf{prefs}_\mathsf{p}(G^a) \to Q^a$ such that for every $\pi \in \mathsf{prefs}_\mathsf{p}(G^a)$, if $f^a(\pi) = q$, then $(\mathsf{last}(\pi), q) \in T^a_\mathsf{p}$.

The soundness of this abstraction guarantees that if $\mathsf{Player}_\mathsf{c}$ has a winning strategy f^a_c in $G^a(\mathcal{F},\mathcal{AP})$, then there exists a finite-state concretization f_c of f^a_c which is a winning strategy for $\mathsf{Player}_\mathsf{c}$ in $\mathcal{F}$ (and hence $\mathsf{Player}_\mathsf{c}$ has a winning strategy in $\mathcal{I}$).

5 Counterexample-Guided Observation Refinement

We now present a procedure for automatically refining the observation predicates in case of a spurious abstract counterexample. This procedure takes into account the two steps of the abstraction phase. Since the predicate abstraction procedure is part of the CEGAR-loop from [8], we refer the reader to the interpolation-based refinement method described there to generate new predicates for $\mathcal{AP}$ in the case when the abstract counterexample does not correspond to a counterexample in the game $\mathcal{F}$. In the following, we focus on the case when the predicate abstraction cannot be further refined due to the fact that the abstract counterexample does correspond to a counterexample in the game $\mathcal{F}$. In this case, we check if it actually corresponds to a concrete counterexample in the game $\mathcal{I}$. We now define a symbolic characterization of the counterexamples that are concretizable in $\mathcal{I}$, and develop a refinement procedure for $\mathcal{F}(\mathcal{I},\xi)$.

Let $\xi : X_{\mathsf{o}+\mathsf{c}} \to 2^{\mathbb{Q}_{>0}}$ be an action-point function, and let $\mathcal{AP}$ be a finite set of predicates. Let $\mathcal{F} = \mathcal{F}(\mathcal{I},\xi)$ be the corresponding finite-choice await-time game, and let $G^a(\mathcal{F},\mathcal{AP}) = (Q^a_\mathsf{c}, Q^a_\mathsf{e}, q^a_0, T^a_\mathsf{c}, T^a_\mathsf{e}, =^a_\mathsf{o}, B^a)$ be its abstraction w.r.t. $\mathcal{AP}$. Suppose that there exists a winning strategy f^a_e for $\mathsf{Player}_\mathsf{e}$ in $G^a(\mathcal{F},\mathcal{AP})$.

Abstract counterexample strategies. A winning strategy f^a_e for $\mathsf{Player}_\mathsf{e}$ in $G^a(\mathcal{F}, \mathcal{AP})$ is an *abstract counterexample*. For a sequence $\rho \in (2^Q)^* \cup (2^Q)^\omega$ of sets of states in a game, we define $\gamma(\rho)$ as the set of paths π such that $|\pi| = |\rho|$ and for every $0 \leq i < |\pi|$ it holds that $\pi[i] \in \rho[i]$. For $\rho_1, \rho_2 \in (2^Q)^* \cup (2^Q)^\omega$ we write $\rho_1 \subseteq \rho_2$ iff $|\rho_1| = |\rho_2|$ and $\gamma(\rho_1) \subseteq \gamma(\rho_2)$. We denote with $\kappa(\mathcal{I})$ and $\kappa(\mathcal{F})$ the perfect-information games for $\mathcal{I}$ and $\mathcal{F}$ defined by knowledge-based subset construction [11].

We say that the strategy f^a_e is *concretizable in* $\mathcal{F}$ iff there exists a winning strategy f^κ_e for $\mathsf{Player}_\mathsf{e}$ in $\kappa(\mathcal{F})$ such that for every $\pi^\kappa \in \mathsf{prefs}(f^\kappa_\mathsf{e})$ there exists a $\pi^a \in \mathsf{prefs}(f^a_\mathsf{e})$ such that $\pi^\kappa \subseteq \pi^a$. *Concretizability of* f^a_e *in* $\mathcal{I}$ is defined analogously.

Since $G^a(\mathcal{F}, \mathcal{AP})$ is a finite-state *safety* game, the strategy f^a_e can be represented as a finite tree $\mathsf{Tree}(f^a_\mathsf{e})$ called *strategy tree* for f^a_e, which can be used for constructing a logical formula characterizing the concretizability of f^a_e.

Each node in $\mathsf{Tree}(f^a_\mathsf{e})$ is identified by a unique $n \in \mathbb{N}$ and is labeled with a state in Q^a denoted $\mathsf{state}(n)$. For a node n, we denote with $\mathsf{children}(n)$ the set of all children of n, with $\mathsf{path}(n)$ the sequence of nodes on the path from the root to n and with $\mathsf{pref}(n)$ the prefix in $G^a(\mathcal{F}, \mathcal{AP})$ formed by the sequence of states corresponding to $\mathsf{path}(n)$.

The tree contains a root node 0 labeled with the initial abstract state q^a_0. For each edge from n to m in $\mathsf{Tree}(f^a_\mathsf{e})$ it holds that $(\mathsf{state}(n), \mathsf{state}(m)) \in T^a$. If $n \in \mathsf{Tree}(f^a_\mathsf{e})$, $\mathsf{state}(n) \notin B^a$ and $\mathsf{state}(n) \in Q^a_\mathsf{e}$, then there exists a single child m of n in $\mathsf{Tree}(f^a_\mathsf{e})$ and $\mathsf{state}(m) = f^a_\mathsf{e}(\mathsf{pref}(n))$. If $n \in \mathsf{Tree}(f^a_\mathsf{e})$, $\mathsf{state}(n) \notin B^a$ and $\mathsf{state}(n) \in Q^a_\mathsf{c}$, then for every $s \in Q^a$ with $(\mathsf{state}(n), s) \in T^a_\mathsf{c}$ there exists exactly one child m of n in $\mathsf{Tree}(f^a_\mathsf{e})$ and $\mathsf{state}(m) = s$. If $\mathsf{state}(n) \in B^a$ then $\mathsf{children}(n) = \emptyset$.

Counterexample strategies that are spurious in $\mathcal{F}$. If the counterexample-analysis from [8] reports that f^a_e is not concretizable in $\mathcal{F}$, we refine the set $\mathcal{AP}$ with the predicates generated by the interpolation-based refinement procedure described there and continue. Otherwise, $\mathsf{Player}_\mathsf{e}$ has a winning strategy in $\kappa(\mathcal{F})$, which implies that $\mathsf{Player}_\mathsf{c}$ does not have a winning strategy in $\mathcal{F}$. This fact does not imply that f^a_e is concretizable in $\mathcal{I}$, as the action-point function ξ might just be too imprecise.

Counterexample strategies that are concretizable in $\mathcal{I}$. We now provide a logical characterization of winning strategies for $\mathsf{Player}_\mathsf{e}$ in $G^a(\mathcal{F}, \mathcal{AP})$ that are concretizable in $\mathcal{I}$. The result is a linear arithmetic formula with alternating universal and existential quantifiers corresponding to the alternating choices of the two players. The variables updated by $\mathsf{Player}_\mathsf{e}$ are existentially quantified and the variables updated by $\mathsf{Player}_\mathsf{c}$, including the symbolic constants, are universally quantified.

The label $\mathsf{state}(n)$ of each node n in $\mathsf{Tree}(f^a_\mathsf{e})$ is a state in $G^a(\mathcal{F}, \mathcal{AP})$ and is thus a set of valuations of the abstraction predicates $\mathcal{AP}$. We associate with $\mathsf{state}(n)$ a boolean combination of elements of $\mathcal{AP}$ and thus a formula $\psi^n_\mathsf{st}[V^n]$ (V^n consists of indexed versions of the variables in V and $\mathsf{Dom}(x^n) = \mathsf{Dom}(x)$ for $x \in V$). The formula $\psi^n_\mathsf{st}[V^n]$, which is the disjunction of all conjunctions representing the valuations in $\mathsf{state}(n)$, characterizes the set of states in $\mathcal{I}$ that are in the concretization of $\mathsf{state}(n)$.

As the abstraction is precise w.r.t. the choices of $\mathsf{Player}_\mathsf{c}$ in $\mathcal{F}$, for each node n in $\mathsf{Tree}(f^a_\mathsf{e})$, $\mathsf{state}(n)$ defines a valuation $\mathsf{contr}(n) = \mathsf{state}(n)|_{V^\mathsf{f}_\mathsf{c}}$ of the variables in V^f_c.

For two nodes n and m such that $m \in \mathsf{children}(n)$ and $\mathsf{state}(n) \in Q^a_p$, where $p \in \{\mathsf{c}, \mathsf{e}\}$, we define the formula $\psi^{n,m}_\mathsf{tr}$ that denotes the transitions in $\mathcal{I}$ from states that satisfy ψ^n_st: $\psi^{n,m}_\mathsf{tr}[V^n, V^m] \equiv \psi^n_\mathsf{st} \wedge \mathcal{T}_p[V^n/V, V^m/V']$.

We now turn to the definition of the *quantified strategy-tree formula* $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ that characterizes the concretizability of f^a_{e} in $\mathcal{I}$. We annotate in a bottom-up manner each node $n \in \mathsf{Tree}(f^a_{\mathsf{e}})$ with a quantified linear arithmetic formula φ^n and define $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}})) = \varphi^{n_0}[0/SC^{n_0}]$ where n_0 is the root of $\mathsf{Tree}(f^a_{\mathsf{e}})$. If $\mathsf{path}(n) = n_0 n_1 \dots n_r$, the formula φ^n will have free variables in $SC^n \cup \bigcup_{i=0}^{r-1}(V^{n_i}_{\mathsf{o+c}} \cup SC^{n_i})$ and thus $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ will be a closed formula.

The annotation formula φ^n for a node n describes the set of prefixes in $\kappa(\mathcal{I})$ that are subsumed by $\mathsf{pref}(n)$ and lead to a state from which $\mathsf{Player_e}$ has a winning strategy in $\kappa(\mathcal{I})$ contained in the corresponding subtree of n. Thus, the formula $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ is satisfiable iff $\mathsf{Player_e}$ has a winning strategy in $\kappa(\mathcal{I})$ subsumed by f^a_{e}.

- For a leaf node n (with $\mathsf{state}(n) \in B^a$) with $\mathsf{path}(n) = n_0 n_1 \dots n_r$ we define:

$$\varphi^n \equiv \exists V^n_{\mathsf{o}} \exists V^{n_0}_{\mathsf{u}} \dots \exists V^{n_r}_{\mathsf{u}} \Big(\big(\bigwedge_{i=0}^{r-1} \psi^{n_i,n_{i+1}}_{\mathsf{tr}} \wedge \psi^{n_r}_{\mathsf{st}} \wedge loc^{n_r} = l_{\mathsf{bad}} \big) [\mathsf{contr}(n)/\,V^{\mathsf{f}^n}_{\mathsf{c}}] \Big).$$

- For a non-leaf node n with $\mathsf{state}(n) \in Q^a_{\mathsf{e}}$ and (single) child m, we define:

$$\varphi^n \equiv \exists V^n_{\mathsf{o}} \big(\varphi^m [SC^n/SC^m, \mathsf{contr}(n)/\,V^{\mathsf{f}^n}_{\mathsf{c}}] \big).$$

- For a non-leaf node n with $\mathsf{state}(n) \in Q^a_{\mathsf{c}}$, we first define a formula $\varphi^{n,m}$ for each node $m \in \mathsf{children}(n)$. The definition of $\varphi^{n,m}$, i.e., the treatment of the symbolic constants SC^m in φ^m, depends on $\mathsf{contr}(m)$, i.e., on the choice made by $\mathsf{Player_c}$ in the game $\mathcal{F}$. For successors m where $\mathsf{Player_c}$ chose to allow $\mathsf{Player_e}$ to decide when to execute the controllable action, we quantify universally over the variables in SC^m, adding a condition which restricts their values to ones that are valid choices of await points for $\mathsf{Player_c}$ in $\mathcal{I}$. In order to ensure intermediate action points, we further require that each c^m_x is smaller than the smallest action point in $\xi(x)$ that is larger than the current value of x. This gives a condition θ^m on the symbolic constants SC^m at node m and we define $\varphi^{n,m} \equiv \forall SC^m(\theta^m \rightarrow \varphi^m)$, where

$$\theta^m \equiv \big(\bigvee_{x \in X_{\mathsf{o+c}}} c^m_x > 0 \big) \wedge \bigwedge_{x \in X_{\mathsf{o+c}}} (c^m_x > 0 \rightarrow c^m_x > x^n) \wedge \bigwedge_{\substack{x \in X_{\mathsf{o+c}} \\ c \in \xi(x)}} \big(x^n < c \rightarrow c^m_x < c \big).$$

For successors m where $\mathsf{Player_c}$ chose to execute a controllable action immediately, to reset some controllable clocks, or to remain idle, we substitute SC^m by 0, i.e., $\varphi^{n,m} \equiv \varphi^m[0/SC^m]$ (which agrees with the transition relation $\mathcal{T}_{\mathsf{c}}$ in $\mathcal{I}$). Finally:

$$\varphi^n \equiv \exists V^n_{\mathsf{o}} \big(\bigwedge_{m \in \mathsf{children}(n)} \varphi^{n,m} [\mathsf{contr}(n)/\,V^{\mathsf{f}^n}_{\mathsf{c}}] \big).$$

The formula $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ characterizes the concretizability of f^a_{e} in $\mathcal{I}$. Thus, if $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ is satisfiable, then $\mathsf{Player_c}$ has no finite-state winning strategy in $\mathcal{I}$.

Proposition 3. *For every winning strategy f^a_{e} for* $\mathsf{Player_e}$ *in $G^a(\mathcal{F}, \mathcal{AP})$, the formula* $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ *is satisfiable iff the strategy f^a_{e} is concretizable in $\mathcal{I}$.*

Extracting refinement action points from a model. We now consider the case when the formula $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ is unsatisfiable. Since $\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ is a closed formula, its negation $\Phi = \neg\mathsf{QTF}(\mathsf{Tree}(f^a_{\mathsf{e}}))$ is satisfiable. In Φ all symbolic constants (indexed accordingly) are existentially quantified. Our goal is to compute witnesses for the symbolic constants that can be used for refining the action-point function ξ to eliminate in the resulting finite-choice game the winning strategies for $\mathsf{Player}_\mathsf{e}$ subsumed by f^a_e.

Consider a block $\exists SC^n$ of existentially quantified symbolic constants in Φ. The block $\exists SC^n$ is preceded by the blocks of universal quantifiers $\forall V_\mathsf{o}^{n_i}$ for $i = 0, \dots, r$, where $\mathsf{path}(n) = n_0 n_1 \dots n_r$. Thus, a witness $\mathsf{w}(c)$ for a symbolic constant $c \in SC^n$ for the satisfiability of Φ is a function $\mathsf{w}(c) : \mathsf{Vals}(V_\mathsf{o}^{n_0}) \times \dots \times \mathsf{Vals}(V_\mathsf{o}^{n_r}) \to \mathbb{Q}_{\geq 0}$.

Assume for now that we have a tuple of witness functions for the variables in SC^n of the following form. For some $k \in \mathbb{N}_{>0}$, there are k positive rational constants $a_1, \dots, a_k$ and each a_i is associated with some variable $x \in X_{\mathsf{o+c}}$. The functions are such that we have a case split with k cases according to the valuation v of the observable variables along the prefix, such that in case i we have $\mathsf{w}(c_x)(v) = a_i$, where x is the variable associated with a_i and $\mathsf{w}(c_y)(v) = 0$ for all other $c_y \in SC^n$.

Let $\mathsf{id} : X_{\mathsf{o+c}} \to \mathbb{N} \cap [1, |X_{\mathsf{o+c}}|]$ be an indexing function for the clock variables $X_{\mathsf{o+c}}$. Thus, each a_i is associated with an index $d_i \in [1, |X_{\mathsf{o+c}}|]$ of a variable in $X_{\mathsf{o+c}}$.

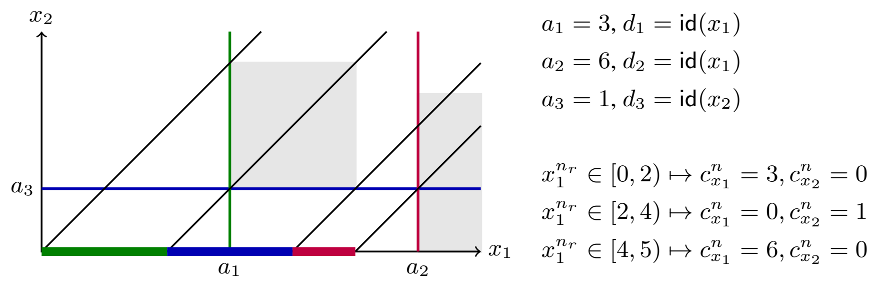
Example. Consider an example with $X_{\mathsf{o+c}} = \{x_1, x_2\}$, where in the abstract state $\mathsf{state}(n_r)$ we know that the value of $x_1^{n_r}$ is in $[0, 5]$ and the value of $x_2^{n_r}$ is 0, and the "good" values for SC^n are depicted as the gray sets on Fig. 5. The figure shows an example where $k = 3$ and each a_i is associated with the shown variable-index d_i. $\square$

More formally, we assume the existence of a function $b : \mathsf{Vals}(V_\mathsf{o}^{n_0}) \times \dots \times \mathsf{Vals}(V_\mathsf{o}^{n_r}) \to \mathbb{N} \cap [1, k]$, such that for each $c_x^n \in SC^n$, the witness function $\mathsf{w}(c_x^n)$ is such that for every valuation $v \in \mathsf{Vals}(V_\mathsf{o}^{n_0}) \times \dots \times \mathsf{Vals}(V_\mathsf{o}^{n_r})$ we have $\mathsf{w}(c_x^n)(v) = a_i$ for some i iff $b(v) = i$ and $d_i = \mathsf{id}(x)$ and $\mathsf{w}(c_x^n)(v) = 0$ otherwise.

We can then refine ξ as follows: For each $x \in X_{\mathsf{o+c}}$ we add to the set $\xi(x)$ all positive constants a such that $\mathsf{w}(c_x^n)(v) = a$ for some node n and valuation v. The form of the function $\mathsf{w}(c_x^n)$ implies that the number of these constants is finite.

For a given $k \in \mathbb{N}_{>0}$, we restrict the possible witnesses for SC^n to the above form by strengthening the formula Φ. To this end, we replace the condition θ^n for SC^n that was used in the construction of Φ a stronger one, $\theta^n \wedge \theta_k^n$ where θ_k^n is defined below.

The formula θ_k^n refers to a fresh bounded integer variable b^n with domain $\mathbb{N} \cap [1, k]$, and the variables from a set $A_k^n = \{a_1^n, \dots, a_k^n\}$ of k fresh rational variables and from a set $D_k^n = \{d_1^n, \dots, d_k^n\}$ of k fresh bounded integer variables. The variable b^n is existentially quantified together with the symbolic constants in SC^n. The variables in A_k^n and D_k^n are free in the resulting formula. We define the formula θ_k^n as:


x_2
a_3
x_1
a_1
a_2
$a_1 = 3, d_1 = \mathsf{id}(x_1)$
$a_2 = 6, d_2 = \mathsf{id}(x_1)$
$a_3 = 1, d_3 = \mathsf{id}(x_2)$
$x_1^{n_r} \in [0,2) \mapsto c_{x_1}^n = 3, c_{x_2}^n = 0$
$x_1^{n_r} \in [2,4) \mapsto c_{x_1}^n = 0, c_{x_2}^n = 1$
$x_1^{n_r} \in [4,5) \mapsto c_{x_1}^n = 6, c_{x_2}^n = 0$
Fig. 5. Example of witnesses for symbolic constants $c_{x_1}^n$ and $c_{x_2}^n$.

$$\theta^n_k \equiv \exists b^n \bigwedge_{i=1}^{k} \Big(b^n = i \to \bigwedge_{x \in X_{\mathsf{o+c}}} \big((\mathsf{id}(x) = d^n_i \to c^n_x = a^n_i) \wedge (\mathsf{id}(x) \neq d^n_i \to c^n_x = 0)\big)\Big).$$

The refinement procedure iterates over the values of $k \geq 1$, at each step constructing a formula Φ_k by replacing each constraint θ^n in Φ by $\theta^n \wedge \theta^n_k$. For each $k \geq 1$, Φ_k is a strengthening of Φ and Φ_{k+1} is weaker than Φ_k. The procedure terminates if a k for which the formula Φ_k is satisfiable is reached. In this case we use the values of the newly introduced variables from A^n_k to refine the action-point function ξ. Thus, if it terminates, Algorithm 1 returns a new action-point function ξ' such that for every $x \in X_{\mathsf{o+c}}$, we have $\xi'(x) \supseteq \xi(x)$. The new action points for x are extracted from a model for Φ_k as the values of those variables a^n_i for which d^n_i is equal to $\mathsf{id}(x)$, and they suffice to eliminate all strategies f^κ_{e} winning for Player$_{\mathsf{e}}$ in $\mathcal{F}$ that are in the concretization of f^a_{e}.

Algorithm 1: Computation of refinement action points

```
Input: satisfiable $\Phi$ with $\theta^{n_i}$ for $SC^{n_i}$, for $i = 1, \dots, m$; function $\xi : X_{o+c} \to 2^{\mathbb{Q}_{>0}}$
Output: function $\xi' : X_{o+c} \to 2^{\mathbb{Q}_{>0}}$
$\xi'(x) := \xi(x)$ for every $x \in X_{o+c}$; sat := false; k := 0;
while sat == false do { k++; construct $\Phi_k$; sat := check($\Phi_k$); }
$\mathcal{M}$ = model($\Phi_k$);
foreach $(n, i) \in (\{n_1 ... n_m\} \times \{1, ..k\})$ with $\mathcal{M}(a^n_i) > 0$ do
    forall $x \in X_{o+c}$ with $id(x) = \mathcal{M}(d^n_i)$ do $\xi'(x) := \xi'(x) \cup \{\mathcal{M}(a^n_i)\}$;
return $\xi'$
```

Proposition 4. *Let f^a_{e} be a winning strategy for* Player$_{\mathsf{e}}$ *in $G^a(\mathcal{F}, \mathcal{AP})$. If Algorithm 1 terminates, it returns an action-point function ξ' such that for the await-time game with fixed action points $\mathcal{F}' = \mathcal{F}(\mathcal{I}, \xi')$,* Player$_{\mathsf{e}}$ *has no winning strategy f^κ_{e} in $\kappa(\mathcal{F}')$ such that for every $\pi^\kappa \in \mathsf{prefs}(f^\kappa_{\mathsf{e}})$ there exists a $\pi^a \in \mathsf{prefs}(f^a_{\mathsf{e}})$ with $\pi^\kappa \subseteq \pi^a$.*

6 Results and Conclusions

We developed a prototype implementation of the presented extension of the CEGAR procedure from [8] to the case of await-time games. We applied our prototype to the safety controller synthesis problem for the Box Painting Production System and the Timed Game For Sorting Bricks examples due to Cassez et al. [4]. We encoded the synthesis problems as await-time games and, starting with empty sets of action points, applied our method to compute observations for which the plants are controllable.

In Table 1 we report on the results from our experiments preformed on an Intel Core 2 Duo CPU at 2.53 GHz with 3.4 GB RAM. We present the maximal number of explored states in the intermediate abstractions, the size of the abstract strategy, the number of action points in the final game, as well as the number of refinement iterations for the await-time game with fixed action points and the number of refinement iterations of the CEGAR loop. In order to demonstrate that our method performs well in situations where fine granularity is needed to win the game, i.e., when the constraints occurring in the plant involve large constants and the differences between certain guards

	A. States	A. Strategy	Act. Points	OBS Iter.	CEGAR Iter.	Time	TIGA
Paint	626	55	2	2	8	73.50	0.08
Paint-100	573	49	2	2	5	29.65	3.57
Paint-1000	573	49	2	2	5	29.53	336.34
Paint-10000	560	76	2	2	7	54.85	> 1800
Paint-100000	614	55	2	2	7	52.88	> 1800
Bricks	1175	125	3	3	3	24.85	0.05
Bricks-100	1175	175	3	3	3	25.16	2.63
Bricks-1000	1175	176	3	3	3	25.29	302.08
Bricks-10000	1175	176	3	3	3	25.83	> 1800
Bricks-100000	1175	175	3	3	3	25.40	> 1800

Table 1. Results from experiments with our prototype on Box Painting Production System and Timed Game For Sorting Bricks: number of states in largest intermediate abstraction, size of abstract strategy for the controller, number of action points in final abstraction, number of iterations of the respective refinement loops and running time (in seconds). Results from experiments with UPPAAL-TIGA with fixed observations (controllable cases): running time (in seconds).

and invariants are small, we constructed multiple instances of each example. Instances Bricks$-N$ and Paint$-N$, where $N \in \{100, 1000, 10000, 100000\}$ were obtained by adding the constant N to all positive constants occurring in the plants.

The results show that the size of the abstract games and strategies generated by our approach depend on the number of action points and predicates and not on the size of the constants in the plant. This is in contrast with approaches based on fixed granularity, where strategies involve counting modulo the given granularity.

Since the problem of synthesizing observation predicates for timed games under incomplete information is out of the scope of existing synthesis tools, a relevant comparison is not possible. However, we used the tool UPPAAL-TIGA, which supports timed games with partial observability and fixed observations, on the problem instances constructed as explained above. For the Box Painting Production System we used observation $y \in [0, 1)$ and for the Timed Game For Sorting Bricks we used observation $[0, 0.5)$ (given as $y \in [0, 1)$ by scaling accordingly). One can see in Table 1 that, although on the small instances the running times are better compared to our approach, on instances where fine granularity is needed, our approach *synthesizes* good observations considerably faster than it takes UPPAAL-TIGA to solve the game with given fixed granularity.

Conclusions. We presented a method to automatically compute observation predicates for timed controllers with safety objectives for partially observable plants. Our approach is based on the CEGAR-paradigm and can be naturally integrated into the CEGAR-loop for games under incomplete information. The observation refinement procedure could be beneficial to methods for solving timed games with fixed observations that are not necessarily CEGAR-based. The bottleneck in such approaches is the enumeration of granularities, which leads to a dramatic increase in the number of state-sets, that need to be explored, and the size of the resulting strategies. As we demonstrated, in some cases, when a reasonable number of action points suffices for controllability, our approach can be extremely successful. This opens up a promising opportunity for synergies between the CEGAR-paradigm and specialized techniques for timed systems.

Acknowledgement This work is partially supported by the German Research Foundation (DFG) as part of SFB/TR 14 AVACS and by a Microsoft Research PhD Scholarship.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, Apr. 1994.
2. P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *Proc. CAV'03*, volume 2725 of *LNCS*. Springer, 2003.
3. F. Cassez. Efficient on-the-fly algorithms for partially observable timed games. In *Proc. FORMATS'07*, volume 4763 of *LNCS*. Springer, 2007.
4. F. Cassez, A. David, K. G. Larsen, D. Lime, and J.-F. Raskin. Timed control with observation based and stuttering invariant strategies. In *Proc. ATVA'07*, volume 4762 of *LNCS*, 2007.
5. K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. In *Proc. CSL'06*, volume 4207 of *LNCS*. Springer, 2006.
6. L. de Alfaro and P. Roy. Solving games via three-valued abstraction refinement. In *Proc. CONCUR*, volume 4703, pages 74–89. Springer-Verlag, 2007.
7. M. De Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In *Proc. HSCC'06*, LNCS, pages 153–168. Springer, 2006.
8. R. Dimitrova and B. Finkbeiner. Abstraction refinement for games with incomplete information. In *Proc. FSTTCS'08*, volume 08004 of *Dagstuhl Seminar Proceedings*, 2008.
9. B. Finkbeiner and H.-J. Peter. Template-based controller synthesis for timed systems. In C. Flanagan and B. König, editors, *TACAS*, volume 7214 of *Lecture Notes in Computer Science*, pages 392–406. Springer, 2012.
10. T. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *Proc. ICALP'03*, volume 2719 of *LNCS*, pages 886–902. Springer, 2003.
11. J. H. Reif. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci.*, 29(2):274–301, 1984.