Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master Thesis

Symbolic Encodings of Timed Games with Incomplete Information

submitted by

Daniel Dahrendorf

submitted 06.04.2009

Supervisor

Prof. Bernd Finkbeiner, Ph.D.

Advisor

Rayna Dimitrova

Reviewers

Prof. Bernd Finkbeiner, Ph.D.
Prof. Dr.-Ing. Holger Hermanns

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

Statement under Oath I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, den 06.03.2009

Acknowledgement
First and foremost I would like to thank my advisor Rayna Dimitrova, without her help and support this thesis would not have been completed and Prof. Finkbeiner who gave me the opportunity to write it. I would like to thank my girlfriend Jacqy for her understanding and patience during the last months and my family without whose support this would not have been possible. Further I would like to thank everyone who helped me by the compilation of this thesis and during my studies.

Abstract. In this thesis we provide two symbolic encodings for timed games with incomplete information. We take two models [11, 8] and transform them into symbolic game structures which can be used as input for the counterexample-guided abstraction refinement approach for timed games with incomplete information from [14]. We restrict the specifications of the models to be deterministic, fix the resources of the controller and focus on safety games. The challenge is to encode the two models, which are based on timed (game) automata and using different definitions of timed games and specifications. The results are symbolic game structures defined by variables, by their domain and by formulas over predicates. The correctness of the encodings is proved. Since [14] provides sufficient conditions for termination of the refinement loop we give an alternative proof for the decidability of the controller syntheses problem for the considered class of timed games. Further, we provide an implementation for the encoding of the model from [11].

Contents

1 Introduction

1.1 Motivation and Related Work

Timed automata were first introduced by Alur and Dill [2]. Today timed automata are a well established formalism in the field of verification. Given a timed game automaton, where discrete actions are partitioned into uncontrollable and controllable actions, and a specification the problem of controller syntheses is to synthesise a strategy for the controller in a way that the specification is met.
For timed game automata under perfect information this problem was considered in [15, 4, 23] and shown to be decidable under certain conditions. Synthesis algorithms based on backward fix-point computations on the set of winning states were proposed for example in [4, 23] and [12] presented an on-the-fly algorithm for solving timed games. This algorithm is a combination of forward search and back-propagation using zones as underlying data structure. Based on these algorithms efficient tools to analyse timed game automata were developed. Examples are the tool Uppaal Tiga [6], a modification of Uppaal [20] and tools [1] in the tool suite Kronos [9].
Dropping the hypothesis that the controller can observe all uncontrollable actions leads to timed games with incomplete information. This could either be caused by limited sensor capability or by the fact that in software the values of private variables from other components can not be read. In such games there is an additional constraint on the controllers strategy: It cannot depend on information the controller is unable to see.
The controller syntheses problem with incomplete information was shown to be decidable if the resources of the controller (number of controllable clocks and number of constants the clocks can be compared to) are fixed [8]. In the paper [8] Bouyer et al. present an algorithm to synthesis a strategy by constructing a finite untimed game using region transition systems and subset construction.
Invariant stuttering strategies were supposed in [11] to hide information from the controller. Such strategies are not affected by a sequence of uncontrollable or time steps unless changes in the observable information occur. The algorithm from [12] was extended in [11] to deal with timed games under incomplete information and synthesises an invariant stuttering strategy. In [10] a prototype implementation for the on-the-fly algorithm of [11] based on Uppaal Tiga [6] was reported.
An abstraction-refinement based approach for solving infinite-state games with incomplete information was proposed in [14]. It extends the counterexample-guided abstraction refinement from [16]. In [14] timed games are described as symbolic game structures which are defined by a set of variables, by their domains and by

sets of formulas which describe the transition relation and the winning condition. The authors also identify sufficient conditions for the termination of the presented abstraction-refinement loop.
The approach from [14] should be applicable to timed games which are a subclass of infinite state games. A timed game with incomplete information described by a timed game automata cannot be represented by a symbolic game structure straight forward. The rules of the games are different and the ways to hide information from the controller are also varying. An example is the definition of a strategy in [11]. Since the strategy in [14] is not stuttering invariant the stuttering behauvior has to simulated by the representing symbolic game structure.

1.2 Contributions

In this thesis it will be shown that the procedure from [14] can be applied to timed games under incomplete information. We present an encoding given as symbolic game structure of each of the models from [11] and [8]. This is done by defining variables and formulas over predicates which represent the game structures and the rules of the original game in the framework of [14]. As we restrict the external specification in [8] to be deterministic, we stay in the decidability class of timed games with incomplete information and an internal specification. For both encodings we prove the correctness. Further, we show that the symbolic safety games, which result from this encoding satisfy the sufficient conditions for termination of the presented abstraction-refinement loop in [14]. Thus, we give alternative proof of the decidability of the controller syntheses problem for timed games under incomplete information where the player, representing the controller, has a safety objective.

1.3 Organisation of the Thesis

In section 2, we give a short introduction to timed games with perfect information and timed games with incomplete information. In this section we also describe the two models from [8] and [11]. Section 3 defines the symbolic safety game from [14]. The encoding for each of the two models into the framework of symbolic games is presented in section 4, where we also proof the correctness of the two encodings. In section 5 we give the proof for the decidability of the controller syntheses problem for timed games under incomplete information. Selected details of the encoding are discussed in Section 6. The implementation of the encoding of the model, based on observation predicates, is described in section 7.

2 Timed Games with incomplete Information

This section gives an overview of timed games with perfect and incomplete information. The definitions in the first part of this section are general as afterwards two different frameworks of timed games are introduced and described in detail.

2.1 Timed Games with Perfect Information

Timed automata were introduced by Alur and Dill [2] and model the behavior of real-time systems. These automata are finite-state automata that are equipped with auxiliary real valued variables, called clocks. A clock valuation v for a set X^c of clocks is a mapping $v : X^c \to \mathbb{R}_{\geq 0}^{X^c}$. A clock can be bounded by a constant M so its valuation its defined by $v : X^c \to \mathbb{R}_{\geq 0}^{x^c} \cap [0, M]$. We use $\mathbf{0}$ to denote the zero-valuation which reset each x in X^c to 0. With $v[Y]$ we denote for $Y \subseteq X^c$ the valuation assigning 0 (resp. $v(x)$) for any $x \in Y$ (resp. $x \in X^c \setminus Y$). If $t \in \mathbb{R}_{\geq 0}$, the valuation $v + t$ is defined as $(v + t)(x) = v(x) + t$, $\forall x \in X^c$. For a subset Y of X^c the valuation $v[0/Y]$ is defined as: for each clock x, $(v[0/Y])(x) = 0$ if $x \in Y$ and $v(x)$ otherwise.

2.1.1 Definitions

We define the generic form of clock constraints for a set of clocks X^c and define subclasses for the different models if needed [1].

Definition 1. *A clock constraint φ over a set X^c of real valued variables called clocks is defined by the grammar*

$$\varphi ::= \bot \mid \top \mid x \sim k \mid x - y \sim k \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

where $x, y \in X^c$, $\sim \in \{<, \leq, =, \geq, >\}$ and $k \in \mathbb{Q}$.

The subclass $\mathcal{C}(X^c)$ of constraints consists of constraints which compare clocks and differences of clocks against integers and conjunctions of such constraints. The subclass $\mathcal{B}(X^c)$ is a subset of $\mathcal{C}(X^c)$ without differences. They are formally defined as follows.

Definition 2 ($\mathcal{C}(X^c)$ and $\mathcal{B}(X^c)$ [12])**.** *Let X^c be a finite set of real valued clocks. We denote with $\mathcal{C}(X^c)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z}$, $x, y \in X^c$ and $\sim \in \{<, \leq, =, \geq, >\}$. $\mathcal{B}(X^c)$ is*

[1] A set of clocks is usually denoted by X. As in [14] also the set of variables is denoted by X we add a small c in superscript to the set of clocks to distinguish between clocks and variables.

the subset of $\mathcal{C}(X^c)$ *generated by the grammar:* $\varphi ::= \top \mid k_1 \leq x < k_2 \mid \varphi \wedge \varphi$ *where* $k_1, k_2 \in \mathbb{Z}$*,* $k_1 < k_2$*,* $x \in X^c$ *and* $\top$ *is the boolean constant true.*

Definition 3 ([2])**.** *A timed automaton is a tuple* $A = (L, l_0, \Sigma, X^c, \delta, Inv)$ *where* L *is a finite set of locations,* $l_0 \in L$ *is the initial location,* Σ *is the set of actions,* X^c *is a finite set of real-valued clocks and* $\delta \subseteq L \times \mathcal{B}(X^c) \times \Sigma \times 2^{X^c} \times L$ *a finite set of transitions and* $Inv : L \to \mathcal{B}(X)$ *associates to each location its invariant.*

A state of a timed automaton is a pair in the state space $(l, v) \in S \times R_{\geq 0}^{X^c}$. a new state can be reached through a time transition (letting time pass) or by a discrete transition where a new location is reached and possibly clocks are reseted.

Definition 4 ([23])**.** *A timed game automaton (TGA)* G *is a timed automaton with its set of actions* Σ *partitioned into controllable* (Σ_c) *and uncontrollable* (Σ_u) *actions.*

We call a TGA G bounded if all clocks in G are bounded.
A timed game is a two player game on the underlying transition system of a TGA between a controller and the environment. The controller has a given objective. In this thesis we focus on safety games, i.e. the goal of the controller is to keep the game inside a certain subset of S. The objective of the environment is to spoil the game and enforce it to enter a state outside of this set. The controller chooses which controllable actions are allowed to play, but he has no influence on the uncontrollable actions. A strategy tells the controller which actions he has to choose. The controller synthesis problem consists of finding a strategy such that the controller fulfils the objective.

2.1.2 Decidability

The controller synthesis problem is well studied. It is a generalisation of system verification, where the question is if a system can satisfy a property. In [4] is shown that for timed game automata the controller synthesis problem is decidable for an internal winning condition, given as a subset of the state-space of the timed game automata. Safety and reachability objectives are such internal winning conditions.
An deterministic external specification given by a deterministic automaton can be converted to an internal winning condition. This is illustrated in [25] where the controller synthesis problem is shown to be decidable for deterministic external specifications.
In difference to [4, 25] in [15] an a priori limitation of the controllers resources is considered, while the controller synthesis problem is studied for nondeterministic

external specifications, given by timed automata. To limit the resources (of the controller) means that the controller has only a finite number of own clocks and it can only compare a finite number of constants to the clocks. External specifications has to been distinguish between specifications which describe desired and specifications which describe undesired behavior. It was shown in [15] that the controller syntheses problem is decidable if the nondeterministic external specification describes undesired behavior. Table 1 summarizes the results of [4, 25, 15].

Resources	Det. Spec. (Internal/External)	Ext. Non-Deterministic Spec.	
Fixed	Decidable	Undecidable	Decidable
Non-Fixed	Decidable	Undecidable	Undecidable

Table 1: Decidability for Games with complete information

2.2 Timed Games with Incomplete Information

In the last chapter it was assumed that the controller has perfect information about the state of the plant. The hypothesis is unrealistic if we want to model systems where the information of the controller is in limited by the power of sensors or by the impossibility to observe the private variables of other processes.

As the controller has incomplete information about the state of the environment the strategy of the controller for a timed game with incomplete information must depend only on the information the controller has. The different approaches of strategies which only depend on the controllers information will be described later.

In the discrete case, it is well known how to handle games with incomplete information. The classic solution is the knowledge base subset construction proposed by Reif in [22]. Further was shown that incomplete information does not effect the decidability of the controller syntheses problem for discrete games [19].

The results for the decidability of the controller syntheses problem from table 1 changes for timed games with incomplete information. As the undecidable cases stay the same [8] showed that decidability is only possible if the resources for the controller are fixed. This results from the fact that timed games with complete information are a special case of timed games with incomplete information. Table 2 shows the results from [8] for the decidability of the controller syntheses problem for games with incomplete information.

There are different approaches which deal with timed games with incomplete information. Two of them are [8] and [11]. In [8] a timed game with incomplete information is defined by a symbolic alphabet as output of the controller, a timed

Resources	Det. Spec. (Internal/External)	Ext. Non-Deterministic Spec.	
Fixed	Decidable	Undecidable	Decidable
Non-Fixed	Undecidable	Undecidable	Undecidable

Table 2: Decidability for Games with incomplete information

transition system as definition of the plant, and a non deterministic timed automaton which specifies the external undesired behavior. The resources of the controller are limited by an additional fixed granularity. The timed game is solved by constructing a deterministic untimed game, by a subset construction of the region graph of the timed transitions system and the granularity of the controller.
An model which follows the approach from [26] and extend it to timed games is presented in [11]. A bounded finite state time game automaton is used to describe the plant and the winning condition is given by a sequence of observations. The resources of the controller are limited by allowing only a finite set of clocks for the controller, the fact that all clocks are bounded and they could only compared to integers. To solve the game a finite non deterministic game is created using the knowledge based subset construction based on zones. For safety and reachability objectives standard control algorithms can be applied to the finite game.
The papers [11, 8] offer a proof for the decidability of the controller syntheses problem for games with incomplete information and limited resources.

2.3 Two Models for Timed Games with Incomplete Information

2.3.1 Model Based on Synchronized Words

Here we describe the setting from [8]. For a finite alphabet Σ, let Σ^* (resp. Σ^ω) be the set of finite (infinite) sequences of elements in Σ. Let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. Further let X^c be a finite set of clocks.

Definition 5 (Constraints)**.** *The set of constraints (or guards) over a set of clocks X^c denoted by $\mathcal{G}(X^c)$ is given by the syntax*

$$g ::= (x \sim c) \mid \neg g \mid (g \vee g) \mid (g \wedge g)$$

where $x \in X^c$, $c \in \mathbb{Q}_{\geq 0}$ and $\sim$ is one of $<$, $\leq$, $=$, $\geq$ and $>$. The set of valuations of X^c which satisfy a guard $g \in \mathcal{G}(X)$ is denoted by $\llbracket\, g \,\rrbracket$.

Definition 6 (Symbolic alphabet)**.** *Let Σ be an alphabet of actions, and X^c be a finite set of clocks. A symbolic alphabet Γ based on (Σ, X^c) is a finite subset of $\Sigma \times \mathcal{G}(X) \times 2^{X^c}$.*

Definition 7 (Timed automaton). *A timed automaton over* (Σ, X^c) *is a tuple* $\mathcal{A} = (\mathcal{T}, F, \mathcal{F})$ *where* $\mathcal{T} = (Q, q_0, \rightarrow)$, *with* $\rightarrow \subseteq Q \times \Gamma \times Q$, *is a finite state transition system over some symbolic alphabet* Γ *based on* (Σ, X^c), $F \subseteq Q$ *is the set of final states and* $\mathcal{F}$ *is an acceptance condition for infinite behavior (Büchi or parity condition).*

Definition 8 (Path and language). *A path in* $\mathcal{T}$ *is a finite or infinite sequence of transitions:*

$$P = q_0 \xrightarrow{a_1, g_1, Y_1} q_1 \xrightarrow{a_2, g_2, Y_2} q_2 ...$$

where $(q_{i-1}, a_i, g_i, Y_i, q_i) \in \rightarrow$, $\forall i > 0$.

Definition 9 (Timed words). *Let* Σ *be a set of actions and* $\mathbb{R}_{\geq}0$ *the set of non-negative Reals. A timed word* $w = (a_i, t_i)_{1 \leq i}$ *is a sequence in* $(\Sigma \times \mathbb{R}_{\geq}0)^{\infty}$ *which satisfies*

- *Monotonicity:* $\forall i < j : t_i < t_j$
- *If* w *is infinite, non-zenoness:* $\forall t \in \mathbb{R}_{\geq}0 : \exists i : t_i > t$

A symbolic word $\gamma = (b_i, g_i, Y_i)_{i \geq 1} \in \Gamma^{\infty}$ *describes a set of timed words* $tw(\gamma)$. *A timed word* $\sigma = (a_i, t_i)_{i \geq 1} \in \Sigma^{\infty}$ *is in* $tw(\gamma)$ *if there exists a sequence* $v = (v_i)_{i \geq 1}$ *of valuations such that* $|\sigma| = |\gamma| = |v|$ *and*

- $\forall i \geq 1 : a_i = b_i$
- $\forall i \geq 1 : v_{i-1} + t_i - t_{i-1} \models g_i$
- $\forall i \geq 1 : v_i = (v_{i-1} + t_i - t_{i-1})[0/Y_i]$

Let $L_{symb}(\mathcal{A})$ *the language of symbolic words which is accepted from* $\mathcal{A}$ *then* $L(\mathcal{A}) = tw(L_{symb}(\mathcal{A}))$.

Thus a timed automaton accepts a language of symbolic words which represents a set of timed words.

Definition 10 (Granularity). *A granularity is a tuple* $\mu = (X^c, m, max)$ *where* X^c *is a set of clocks,* m *is a positive integer and* $max : X^c \rightarrow \mathbb{Q}^+$ *a function which associates with each clock of* X^c *a positive rational number. The granularity of a finite set of constraints is the tuple* (X^c, m, max) *where* X^c *is the exact set of clocks mentioned in the constraints,* m *is the least common multiple of the denominators of the constants mentioned in the constraints, and* max *records for each* $x \in X^c$ *the* *largest constant it is compared to. A constraint g is called μ-granular if it belongs to some set of constraints of granularity μ. We denote the set of all μ-granular constraints by $\mathcal{G}(\mu)$. A constraint $g \in \mathcal{G}(\mu)$ is called μ-atomic if for all $g' \in \mathcal{G}(\mu)$, either $[\![g]\!] \subseteq [\![g']\!]$ or $[\![g]\!] \cap [\![g']\!] = \emptyset$. Let $atoms_\mu$ denote this set of μ-atomic constraints. For two granularities $\mu = (X^c_\mu, m_\mu, max_\mu)$ and $\nu = (X^c_\nu, m_\nu, max_\nu)$ we use $\mu + \nu$ to denote the combined granularity of μ and ν which is defined by $\mu + \nu = (X^c_\mu \cup X^c_\nu, lcm(m_\mu, m_\nu), max)$ where for a clock $x \in X^c_\mu \cup X^c_\nu$ $max(x)$ is the larger of $max_\mu(x)$ and $max_\nu(x)$, assuming $max_\mu(x) = 0$ if $x \in X^c_\nu \setminus X^c_\mu$ and $max_\nu(x) = 0$ if $x \in X^c_\mu \setminus X^c_\nu$.*

Definition 11 (Timed game)**.** *A timed game is a game between player C (the controller) and player E (the environment). Let Σ be an alphabet of actions, which is partitioned into a set Σ_C of controllable actions, a set Σ^o_E of observable and uncontrollable actions and a set Σ^u_E of unobservable and uncontrollable actions. The set X^c of clocks is partitioned in a similar way into a set X^c_r of readable clocks and a set X^c_u of unreadable clocks. Further let X_{cont} be a set of clocks which can be reseted only by the controller. Let $\hat{\Sigma} = (\Sigma_C, \Sigma^o_E, \Sigma^u_E)$ and $\hat{X}^c = (X^c_r, X^c_u)$ and $\mu = (X^c_r \cup X^c_{cont}, n, max)$ be a granularity. A timed game based on $(\hat{\Sigma}, \hat{X}^c, \mu)$ is a triple $\Omega = (\Gamma, \mathcal{H}, \mathcal{A})$ where Γ is the symbolic alphabet $(\Sigma_C \cup \Sigma^o_E) \times atoms_\mu \times 2^{X^c_{cont}}$, $\mathcal{H}$ is a transition system over (Σ, X^c) and $\mathcal{A}$ is a timed automaton over (Σ, X^c) which specifies the set of undesired behaviors.*

The game rules of the timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{A})$ based on $(\hat{\Sigma}, \hat{X}^c, \mu)$ are as follows. Player C offers a subset of symbolic actions from Γ, and player E responds by choosing an action u from that subset and so on. Player C can offer the empty set of actions, but in that case the game ends and player C looses. Let us fix for the rest of the section the timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{A})$ based on $(\hat{\Sigma}, \hat{X}^c, \mu)$. Thus a play in Ω is a word in Γ^∞.
Figure 1 [7] illustrates a play in Ω where the gray arrows denote the offers of player C and the black arrows denote the choice of player E.
For a finite play a strategy returns a set of symbolic actions in Γ.

Definition 12 (Strategy)**.** *A strategy for player C is a function $f : \Gamma^* \rightarrow 2^\Gamma$ such that $f(\gamma)$ is deterministic for every $\gamma \in \Gamma^*$. A play is according to f if for every prefix $r.u$ of γ, $u \in f(r)$. Let $plays_f(\Omega)$ denote the set of plays according to f where $plays^*_f(\Omega)$ denote the set of finite plays and $plays^\omega_f(\Omega)$ the set of infinite plays.*

To determine if a play γ is winning we consider the synchronized words that can produced by γ in conjunction with $\mathcal{H}$.

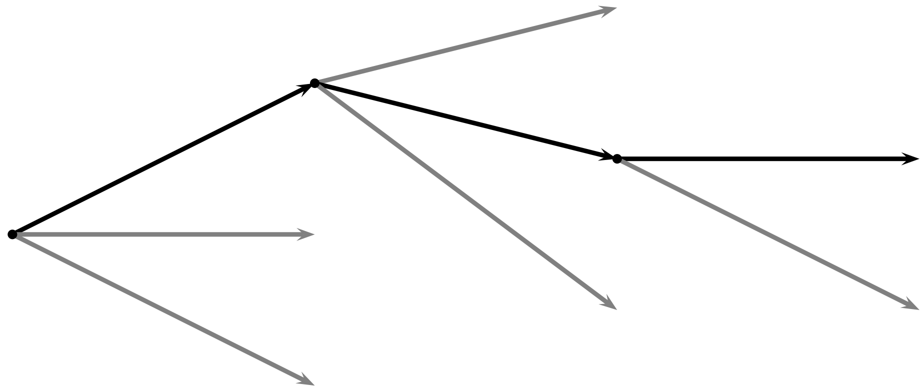

Figure 1: Illustration of a play in Ω

Definition 13 (Synchronized product)**.** *Let $\Sigma = \Sigma_1 \cup \Sigma_2$ be an alphabet, $X^c = X_1^c \cup X_2^c$ be a finite set of clocks and R_1, R_2 be a partition of X^c. Let $\mathcal{T}_1 = (Q_1, q_0^1, \rightarrow_1)$ and $\mathcal{T}_2 = (Q_2, q_0^2, \rightarrow_2)$ be two transition systems over (Σ_1, X_1^c) and (Σ_2, X_2^c) which are allowed to reset clocks only in R_1 and R_2 respectively. The synchronized product $\mathcal{T}_1 \parallel \mathcal{T}_2$ is defined as the transition system $\mathcal{T} = (Q, q_0, \rightarrow)$, where $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$ and the transitions are constructed as follows. Let (q_1, q_2) be a state of Q, let $a \in \Sigma = \Sigma_1 \cup \Sigma_2$ and, for every $i \in \{1, 2\}$ such that $a \in \Sigma_i$, let $q_i \xrightarrow{a, g_i, Y_i} q_i'$ be a transition in $\mathcal{T}_i$. Then there exists a transition $(q_1, q_2) \xrightarrow{a, g, Y} (\overline{q_1}, \overline{q_2})$ with*

- $g = \bigwedge_{i | a \in \Sigma_i} g_i$
- $Y = \bigcup_{i | a \in \Sigma_i} Y_i$
- *for any $i \in \{1, 2\}$, $\overline{q_i} = q_i'$ if $a \in \Sigma_i$ and $\overline{q_i} = q_i$ otherwise.*

We use the notations $(a, g, Y) \restriction j = \epsilon$ if $a \notin \Sigma_j$ and $(a, g, Y) \restriction j = (a, g_j, Y_j)$ otherwise. We extend this notations to words in the obvious way, interpreting ϵ as the empty word.

Definition 14 (Synchronized words)**.** *The set of synchronized words of a symbolic word $\gamma \in \Gamma^\infty$ with respect to $\mathcal{H}$ is denoted by $synw_{\mathcal{H}}(\gamma)$. Let $\mathcal{U}_\Gamma$ be the universal, single state transition system over Γ which accepts all words of Γ^∞. Consider the product $\mathcal{H} \parallel \mathcal{U}_\Gamma$, with respect to the distributed clock-alphabet $((\Sigma, X^c, X^c), (\Sigma_C \cup \Sigma_E^o, X_r^c \cup X_{cont}^c, X_{cont}^c))$. Then $synw_{\mathcal{H}}(\gamma)$ is defined to be the set of all $\gamma' \in L_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ such that $(\gamma' \restriction 2) = \gamma$. Again let us denote with $synw_{\mathcal{H}}^*(\gamma)$ and $synw_{\mathcal{H}}^\omega(\gamma)$ the set of finite and infinite words in $synw_{\mathcal{H}}(\gamma)$.*

A play γ can be interpreted as the "observations" the controllers monitors while a run on the plant which is represent by a time word in $synw_{\mathcal{H}}(\gamma)$.

Definition 15 (Valid strategies)**.** *For a strategy f and a finite play $\gamma \in plays^*_f(\Omega)$, let $\Xi_\gamma = tw(synw^*_{\mathcal{H}}(\gamma))$ denote the set of finite timed words generated by the strategy in conjunction with $\mathcal{H}$ on the finite play γ. The strategy f is non-restricting if whenever $\gamma \in plays^*_f(\Omega)$, $\sigma \in \Xi_\gamma$ and $\sigma.(e,t) \in L^*(\mathcal{H})$ with $e \in \Sigma_E$, it is the case that $\sigma.(e,t) \in \Xi_{\gamma'}$ for some $\gamma' \in plays^*_f(\Omega)$ such that γ is a prefix of γ'. The strategy f is non-blocking if whenever we have $\gamma \in plays^*_f(\Omega)$, $\sigma \in \Xi_\gamma$ and $\sigma.(b,t) \in L^*(\mathcal{H})$ for some $b \in \Sigma$, then there is a word $\sigma.(c,t') \in \Xi_{\gamma'}$, where $c \in \Sigma$, $t' \in \mathbb{R}_{>0}$ and $\gamma' \in plays^*_f(\Omega)$ such that γ is a prefix of γ'.*

Definition 16 (Winning strategies)**.** *A play $\gamma \in \Gamma^\infty$ in Ω is winning for player C whenever $tw(synw^\omega_{\mathcal{H}}(\gamma)) \cap L(\mathcal{A}) = \emptyset$. A strategy f is winning for player C if $tw(synw^\omega_{\mathcal{H}}(plays_f(\Omega))) \cap L(\mathcal{A}) = \emptyset$.*

Thus player C is winning if there exists for a play no timed word in the synchronized words which represents undesired behaviour and is accepted by $\mathcal{A}$.
Example. The timed transition system and the timed automata from figure 2 specify together with the symbolic $\Gamma = (\Sigma_C \cup \Sigma^o_E) \times atoms_\nu \times 2^{X^c_{cont}}$ a timed game based on $(\hat{\Sigma}, \hat{X}^c, \nu)$. The uncontrollable actions Σ_E are partitioned into unobservable actions $\Sigma^u_E = \{u_1\}$ and observable actions $\Sigma^o_E = \{o_1, o_2\}$. The controllable actions are $\Sigma_C = \{r, eject\}$. Player E has only an unobservable clock ($X^c_u = \{x\}$) and player C is equipped with his own clock ($X^c_{cont} = \{y\}$). The granularity ν is given by $\nu = \{\{y\}, 2, 0.5\}$. Player C has to avoid, similar to the example from section 2.3.2, that the plant reaches state l_3 which is specified by the timed automata. This game is controllable as there exists a valid strategy which leads to the play $(o_1, true, \{y\})(r, y = 0.5, \{y\})(r, y = 0.5, \{y\})(r, y = 0.5, \{y\})(r, y = 0.5, \{y\})(eject, y = 0.5, \{y\})(o_1, true, \{y\})$.... Also like in the example from section 2.3.2 the game will be uncontrollable if we take the coarser granularity $\mu = \{\{y\}, 1, 1\}$.

2.3.2 Model Based on Observation Predicates

Here we describe the setting for the model used in [11]. The subsets $\mathcal{C}(X^c, M)$ and $\mathcal{B}(X^c, M)$ of clock constraints for a set of clocks X^c are the bounded by an natural number M.

Definition 17 ($\mathcal{C}(X^c, M)$ and $\mathcal{B}(X^c, M)$)**.** *Let X^c be a finite set of real valued clocks. Let M be a natural number. We denote with $\mathcal{C}(X^c, M)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z} \cap [0, M]$, $x, y \in X^c$ and $\sim \in \{<, \leq, =, \geq, >\}$. $\mathcal{B}(X^c, M)$ is the subset of $\mathcal{C}(X^c, M)$ generated*

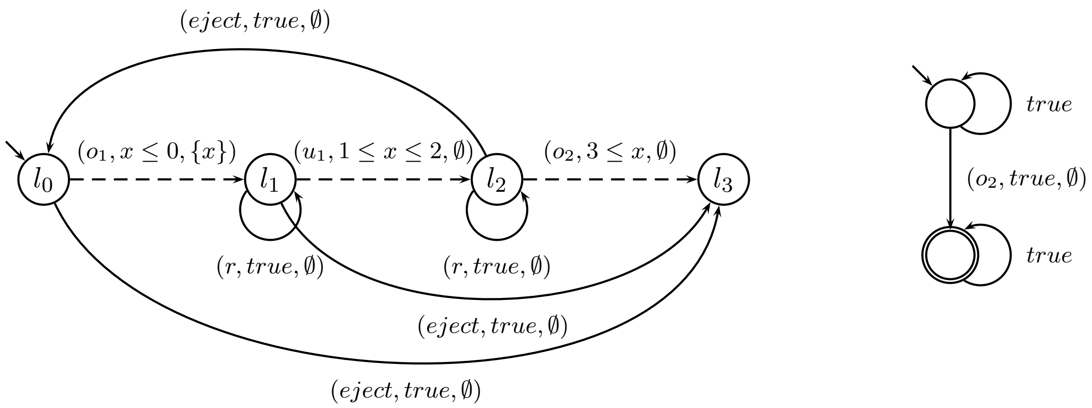

Figure 2: A Timed transition system and a timed automata which specify together with a symbolic alphabet a timed game.

by the grammar: $\varphi ::= \top \mid k_1 \leq x < k_2 \mid \varphi \wedge \varphi$ *where* $k_1, k_2 \in \mathbb{Z} \cap [0, M]$, $k_1 < k_2$, $x \in X^c$ *and* $\top$ *is the boolean constant true. We will use the notation* $V = \mathbb{R} \cap [0, M]$.

Two players, player 1 which corresponds to the controller and player 2 which correspond to the plant, play a timed game on the labeled transition system of an timed game structure.

Definition 18 (Timed Game Structure)**.** *Let* M *be a natural number. An M-timed game structure (M-TGS) is a tuple* $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$ *where:*

- L *is a finite set of locations,*
- $\iota \in L$ *is the initial location,*
- X_1^c, X_2^c *are two disjoint finite sets of real valued clocks.* X_1^c *is the set of clocks of player 1,* X_2^c *is the set of clocks of player 2 and* $X^c = X_1^c \cup X_2^c$,[2]
- Σ_1, Σ_2 *are two disjoint alphabets of actions,* Σ_1 *is the set of actions of player 1 and* Σ_2 *is the set of actions of player 2,*
- $\delta \subseteq (L \times \mathcal{B}(X_2^c, M) \times \Sigma_1 \times 2^{X_2^c} \times L) \cup (L \times \mathcal{C}(X_2^c, M) \times \Sigma_2 \times 2^{X_2^c} \times L)$ *is the transition relation which is partitioned into transitions of player 1 and player 2,*
- $inv : L \rightarrow \mathcal{B}(X_2^c, M)$ *associates to each location its invariant,*
- $\mathcal{P}$ *is a finite set of pairs* (K, φ) *where* $K \subseteq L$ *and* $\varphi \in \mathcal{B}(X^c, M)$, *called observable predicates,*

Definition 19 (Observations)**.** *Two states $(l_1, v_1), (l_2, v_2)$ have the same observation if they satisfy the same observable predicates. An observation is a function $o : \mathcal{P} \rightarrow \{0, 1\}$. Let $\mathcal{O}$ be the set of functions $\{\mathcal{P} \rightarrow \{0, 1\}\}$ called observations of the system. The function $\gamma : \mathcal{O} \rightarrow 2^{L \times [X^c \rightarrow V]}$ maps observations to classes of equivalent states:*

$$\gamma(o) = \left\{ (l, v) \mid \bigwedge_{(K,\varphi) \ \mid \ o(K,\varphi)=1} (l, v) \models (K, \varphi) \wedge \bigwedge_{(K,\varphi) \ \mid \ o(K,\varphi)=0} (l, v) \not\models (K, \varphi) \right\}$$

For a state (l, v) we write $\gamma^{-1}(l, v)$ for the observation o of the state (l, v).

Definition 20 (Semantics of a M-TGS)**.** *The semantics of an M-TGS given by $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$ is a 2-player labeled timed transition system (2-LTTS) $S_H = (S, s_0, \Sigma_1, \Sigma_2, \rightarrow)$ where:*

- $S = \{(l, v) \mid l \in L \wedge v \in V \wedge v \models inv(l)\}$
- $s_0 = (\iota, 0)$
- *The transition relation $\rightarrow$ is composed of*
  - *Discrete transitions. For all $(l_1, v_1), (l_2, v_2) \in S$, for all $\sigma \in \Sigma_1 \cup \Sigma_2$, $((l_1, v_1), \sigma, (l_2, v_2)) \in \rightarrow$ iff there exists a transition $(l, g, a, Y, l') \in \delta$ such that $l = l_1$, $l' = l_2$, $v_1 \models g$ and $v_2 \models v_1[Y]$.*
  - *Time transitions. For all $(l_1, v_1), (l_2, v_2) \in S$, for all $t \in \mathbb{R}_{>0}$, there is a transition $((l_1, v_1), t, (l_2, v_2)) \in \rightarrow$ iff $l_1 = l_2$, $v_2 = v_1 + t$ and for all t', $0 \leq t' < t$, $(l_1, v_1 + t') \in S$ and $\gamma^{-1}(l_1, v_1 + t') = \gamma^{-1}(l_1, v_1)$*

The rules of the game corresponding to a M-TGS are defined as follows: Player 1 proposes an action $\sigma \in \Sigma_1^{ext} = \Sigma_1 \cup \{\Delta\} \cup \{Reset_Y \mid y \in 2^{X_1^c}\}$. The special action Δ is played by player 1 if she does not want to play a discrete action and $Reset_Y$[3] is played if player 1 wants to reset a subset $Y \subseteq X_1^c$ of her own clocks. Once Player 1 has committed to a choice player 2 can play transitions under the following rules:

1. If $\sigma \in \Sigma_1 \cup \{Reset_Y \mid y \in 2^{X_1^c}\}$ player 2 can choose to play as long as the observation does not change, either an action $c \in \Sigma_2 \cup \{\sigma\}$ or let time elapse as long σ is not enabled,
2. If $\sigma = \Delta$ player 2 can choose to play as long as the observation does not change, either an action $c \in \Sigma_2$ or let time elapse,

[2] [11] define only a single set of clocks, but as each player has his own clocks with different rights we define a set of clocks for each player.

[3] [11] do not define the $Reset_Y$ action explicit but use it in an example. Thus we define it here.

3. Player 2 has to give back the turn to player 1 as soon as the observation changes.

Definition 21 (Plays)**.** *A play in H is a infinite sequence ρ^H of the form $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1...(l_n, v_n)c_n\sigma_n...$, where $l_i \in L$, $v_i \in V$, $c_i \in \Sigma_1^{ext}$ and $\sigma_i \in \Sigma_2 \cup \{c_i\}$ such that for all $i \geq 0$ it holds that $((l_i, v_i), \sigma_i, (l_{i+1}, v_{i+1})) \in \rightarrow$ and*

- *either $\sigma_i \in \{c_i\} \cup \Sigma_2$, or*
- *$\sigma_i \in \mathbb{R}_{>0}$ and $\forall 0 \leq t < \sigma_i$, c_i is not enabled in the state $(l_i, v_i + t)$;*
- *if σ_i and σ_{i+1} are in $\mathbb{R}_{>0}$ then $\gamma^{-1}(l_i, v_i) \neq \gamma^{-1}(l_{i+1}, v_{i+1})$.*

A play is consistent iff for all $i \geq 0$ it holds that if $\gamma^{-1}(l_i, v_i) = \gamma^{-1}(l_{i+1}, v_{i+1})$ then $c_i = c_{i+1}$. A play is called initial if it starts in the initial state $(\iota, 0)$. We denote with $Play(H)$ the set of the initial plays in H and with $Play^{co}(H)$ the set of initial consistent plays in H.

Definition 22 (Prefixes)**.** *A prefix in H is a prefix of a play in H that ends in a state of H or ends with an action $c_n \in \Sigma_1^{ext}$. We denote with $Pref(H)$ the set of prefixes of initial plays in H and with $Pref^{co}(H)$ the set of prefixes of the initial consistent plays in H. For a prefix or a play $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1...(l_n, v_n)c_n\sigma_n...$ we denote with $\rho^H(n)$ the prefix up to the state (l_n, v_n). Further we denote for the prefix $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1...\ (l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)$ or $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1...\ (l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)c_n$ with $|\rho^H| = n$ the length and with $last\sigma(\rho^H) = \sigma_{n-1}$ the last played action of the prefix ρ^H.*

In [11] a prefix ends always in a state. We changed the notation as we map in the encoding prefixes of this model to prefixes of the symbolic model. A prefix which ends with an action will be mapped to system states (the player in the symbolic which corresponds to the controller).

Definition 23 (Choice Point)**.** *A prefix $\rho^H = (l_0, v_0)c_0\sigma_0...c_{n-1}\sigma_{n-1}(l_n, v_n)$ is a choice point if either $n = 0$, or $n > 0$ and $\gamma^{-1}(l_{n-1}, v_{n-1}) \neq \gamma^{-1}(l_n, v_n)$*

Definition 24 (Stuttering free observations)**.** *Let $\rho^H = (l_0, v_0)c_0\sigma_0...(l_n, v_n)\ c_n\sigma_n...$ be a consistent play in H. Let $I_{\rho^H} = \{m \mid \rho^H(m)$ is a choice point$\}$. The stuttering free observation $Obs(\rho^H)$ of ρ^H is the sequence of elements of $(\mathcal{O}, \Sigma_1^{ext})$defined by:*

- *If $I_{\rho^H} = \{n_0, n_1, ..., n_k\}$ is finite,*

$$Obs(\rho^H) = \gamma^{-1}(l_{n_0}, v_{n_0})c_{n_0}...\gamma^{-1}(l_{n_k}, v_{n_k})c_{n_k}(\gamma^{-1}(l_{n_k}, v_{n_k})c_{n_k})^\omega$$

- *If* $I_{\rho^H} = \{n_0, n_1, ..., n_k, ...\}$ *is infinite,*

$$Obs(\rho^H) = \gamma^{-1}(l_{n_0}, v_{n_0})c_{n_0}\gamma^{-1}(l_{n_1}, v_{n_1})c_{n_1}...\gamma^{-1}(l_{n_k}, v_{n_k})c_{n_k}...$$

Let $\rho^H = (l_0, v_0)c_0\sigma_0...(l_n, v_n)c_n\sigma_n$ *be a consistent prefix in* H. *The finite stuttering free observation* $Obs^*(\rho^H)$ *of* ρ^H *is the sequence of elements of* $(\mathcal{O}, \Sigma_1^{ext})$ *defined by:*

$$Obs^*(\rho^H) = \gamma^{-1}(l_{n_0}, v_{n_0})c_{n_0}\gamma^{-1}(l_{n_1}, v_{n_1})c_{n_1}...\gamma^{-1}(l_{n_l}, v_{n_l})c_{n_k}$$

Definition 25 (Strategy)**.** *A strategy for player 1 in* H *is defined as function* $\lambda^H : Pref(H) \rightarrow \Sigma_1^{ext}$. *The outcome of a strategy* λ^H *is the set of plays* $\rho^H = (l_0, v_0)c_0\sigma_0...$ $(l_n, v_n)c_n\sigma_n...$ *such that* $l_0 = \iota$, $v_0 = \mathbf{0}$ *and for all* $i \geq 0$ *it holds that* $c_i = \lambda^H(\rho(i))$. *We denote with* $Outcome^H(\lambda^H)$ *this set of plays.*

Definition 26 (OBSI strategy)**.** *A strategy* λ^H *is an observation based stuttering invariant (OBSI) strategy if the following property holds: for all* $\rho_1^H, \rho_2^H \in Pref^{co}(H)$, *let* n_1 *be the maximal value such that* $\rho_1^H(n_1)$ *is a choice point, let* n_2 *be the maximal value such that* $\rho_2^H(n_2)$ *is a choice point, if* $Obs^*(\rho_1^H(n_1)) = Obs^*(\rho_2^H(n_2))$ *then* $\lambda^H(\rho_1^H) = \lambda^H(\rho_2^H)$.

Definition 27 (Winning condition)**.** *Let* $\rho^H \in Play(H)$ *such that* $Obs(\rho^H) = o_0c_0o_1c_1...o_nc_n....$ *The projection* $Obs(\rho^H) \downarrow \mathcal{O}$ *over* $\mathcal{O}$ *of* $Obs(\rho^H)$ *is the sequence* $o_0o_1...o_n....$ *A winning condition* $\mathcal{W}$ *is a stuttering closed*[4] *subset of* $\mathcal{O}^\omega$.

Thus a strategy λ^H is winning in H for $\mathcal{W}$, if and only if, for every play ρ^H in $Outcome^H(\lambda^H) : Obs(\rho^H) \downarrow \mathcal{O} \in \mathcal{W}$

Example. To illustrate the idea of this model we give an example from [11] shown in figure 3. The dotted lines represent the discrete transitions with actions of player 2 (uncontrollable transitions) and the solid lines represent the discrete transitions with actions of player 1 (controllable transitions). This timed game models a box painting production system where player 1 has to eject the box in the right moment. The controllable transitions with the action $Eject?$ from the states l_0, l_1, l_2, l_4 and l_5 to the state l_7 are omitted. Player 1 has incomplete information about the state and the value of an additional clock y. Player 1 knows only if the game is in one of the location sets $E = \{l_0, l_2, l_3, l_5, l_6\}$, $H = \{l_1\}$, $L = \{l_4\}$ or $B = \{l_7\}$ and can compare the value of x to the constant 0.5. The set of observable predicates is defined by

[4] A language is stuttering closed, if dor any word w in the language, the word w' obtained from w by either adding a stuttering step (repeating a letter), or erasing a stuttering step, is also in the language [11].

$\mathcal{P} = \{(E, 0 \leq y < 0.5), (E, 0.5 \leq y), (H, 0 \leq y < 0.5), (H, 0.5 \leq y), (L, 0 \leq y < 0.5), (L, 0.5 \leq y), (B, 0 \leq y < 0.5), (B, 0.5 \leq y)\}$.and the set of actions for player 1 is $\Sigma_1^{ext} = \{Reset_y, Eject!, \Delta\}$. As l_7 is the bad state the player 1 loses if one of the observations $(B, 0 \leq y < 0.5)$ or $(B, 0.5 \leq y)$ are true. Player 1 starts and chooses an action from Σ_1^{ext} then player 2 can play under the rules stated above. If a choice point is reached player 1 proposes a new choice. The choice points are the states where an observation changes. In this case the states where the controllable clock reaches 0.5 or player 2 played an discrete transition such that the new location is not in the same location set as the old location. It is possible to synthesize a OBSI strategy such that none of the observations $(B, 0 \leq y < 0.5)$ or $(B, 0.5 \leq y)$ will be true in any state. If we are in state l_1 we know that we are at least after 10 seconds in state l_3 and can play $Eject!$. Before we just wait or if the clock y has the value 0.5 we reset it. The same works for state l_4 where we know that we are at least after 5 seconds in state l_6. This strategy is just based on the observations.

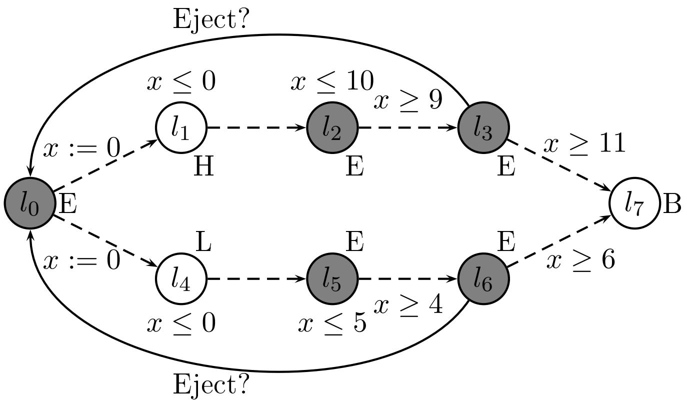

Figure 3: Timed game example from [11]. The controllable transitions with the action $Eject?$ from the states l_0, l_1, l_2, l_4 and l_5 to the state l_7 are omitted.

3 A Symbolic Game Structure with Incomplete Information

In this section we describe the symbolic games as defined in [14]. This is a generic model for representing incomplete information games with possibly infinite state space. Theses are two player games between a system and an environment. They communicate over a finite set of variables, which is partitioned into four sets: X_h, X_i, X_o and $\{t\}$. The set of variables X_h are the private variables of the environment which cannot be observed from the system. The set X_i are the input variables for the system. The system can observe them, but only the environment is allowed to update (write) these variables. The output variables of the system are the variables in X_o which can be observed and updated from the system. The environment can only observe the variables in X_o.
For simplicity of the presentation we introduce ordered tuple which describes the sets of variables. We denote with $\hat{x_h}$ the tuple which contains all variables from X_h, with $\hat{x_i}$ the tuple which contains all variables from X_i and with $\hat{x_o}$ the tuple which contains all variables from X_o.
Sets of states and transitions are represented as formulas over some possibly infinite set $\mathcal{AP}$ of predicates (atomic formulas) over the variables in $X \cup X'$ where X' is the set of the primed versions of the variables in X. Similarly denote $\hat{x_h}'$, $\hat{x_i}'$ and $\hat{x_o}'$ the tuple with the primed variables. With $\mathit{Preds}(\varphi)$ the set of predicates of the formula φ is denoted and for a set of predicates $\mathcal{P}$, the set $\mathit{Obs}(\mathcal{P})$ denotes the set of predicates which contain only observable variable (variables in $X_i \cup X_o$).

Definition 28 (Game Structure with Incomplete Information)**.** *A game structure with incomplete information $C = (S_s, S_e, s_0, \equiv, R_s, R_e)$ consists of a set of states which is partitioned into a set of system states S_s and a set of environment states S_e, an observation equivalence $\equiv$, an initial state s_0 and a transition relation R. The transition relation R is partitioned into the transition relation for the system $R_s \subseteq S_s \times S_e$ and the transition relation for the environment $R_e \subseteq S_e \times S$.*

If in a game structure C for a state $v \in S$ there exists no state $w \in S$ with $(v, w) \in R$ then it is called a *dead-end*.

Definition 29 (Observation Equivalence)**.** *An observation equivalence satisfy the following conditions:*

- *If $v_1 \in S_s$ and $v_2 \in S_e$ then $v_1 \not\equiv v_2$;*
- *For every $v \in S$ and $w_1, w_2 \in S_e$, if $(v, w_1) \in R_s$, $(v, w_2) \in R_s$ and $w_1 \neq w_2$, then $w_1 \not\equiv w_2$;*

- *For every states* $v_1, v_2 \in S$ *and* $w_1 \in S_e$ *such that* $v_1 \equiv v_2$ *and* $(v_1, w_1) \in R_s$, *there exists a state* $w_2 \in S_e$ *such that* $w_1 \equiv w_2$ *and* $(v_2, w_2) \in R_s$.

The observation equivalence can be extended in a natural way to sequences of states.

Definition 30 (Symbolic Game Structure with Incomplete Information)**.** *A symbolic game structure with incomplete information* $\mathcal{C} = (X, init, \mathcal{T}_s, \mathcal{T}_e)$ *consists of a set of variables* X *(partitioned into* X_h, X_i, X_o *and* $\{t\}$*), a formula init over* X *and formulas* $\mathcal{T}_s$ *and* $\mathcal{T}_e$ *over* $X \cup X'$. *A symbolic game structure defines a game structure if the following conditions are satisfied*

- *The following formula is valid*

$$init\{x_h^1/\hat{x_h}, x_i^1/\hat{x_i}, x_o^1/\hat{x_o}, t^1/t\} \wedge init\{x_h^2/\hat{x_h}, x_i^2/\hat{x_i}, x_o^2/\hat{x_o}, t^2/t\}$$
$$\rightarrow x_h^1 \approx x_h^2 \wedge x_i^1 \approx x_i^2 \wedge x_o^1 \approx x_o^2 \wedge t^1 \approx t^2$$

- *The formula* $\mathcal{T}_s$ *implies* $t \approx 1$, $t' \approx 0$, $\hat{x_h}' \approx \hat{x_h}$ *and* $\hat{x_i}' \approx \hat{x_i}$*;*
- *The formula* $\mathcal{T}_e$ *implies* $t \approx 0$ *and* $\hat{x_o}' \approx \hat{x_o}$*;*
- *The following formula is valid*

$$\mathcal{T}_s\{x_h^1/\hat{x_h}\} \leftrightarrow \mathcal{T}_s\{x_h^2/\hat{x_h}\}$$

We require that the domain of each variable in X_o is finite. With $\hat{c}_o$ we denote the tuple of constants corresponding to elements of the domain of $\hat{x}_o$.
The set $Val(X)$ consists of all total functions that map each variable in X to its domain. For a formula φ over $X \cup X'$ and the valuations $v \in Val(X)$ and $w \in Val(X')$ we denote with $\varphi[v, w]$ the truth value of the formula φ for the valuations v and w of the variables.
Let $\mathcal{C}$ be a symbolic game, that satisfies the conditions to define a game structure. Let H,I and O be the corresponding variable domains and the domain of t be $\{0, 1\}$. Then $\mathcal{C}$ together with the variable domains define the game structure $C = (S_s, S_e, s_0, \equiv, R_s, R_e)$ in the following way:

- The set S_s contains the valuations in $Val(X)$ where t is mapped to 1;
- The set S_e contains the valuations in $Val(X)$ where t is mapped to 0;
- s_0 is the state which satisfies the formula *init*;
- Two states are observably equivalent if the valuation of the variables in X_o, X_i and $\{t\}$ are equal;

- The set R_s is defined by $(v, w) \in R_w$ iff $\mathcal{T}_s[v, w]$ is true;
- The set R_e is defined by $(v, w) \in R_w$ iff $\mathcal{T}_e[v, w]$ is true.

Definition 31 (Safety Game)**.** *A safety game with incomplete information* $G = (C, E)$ *consists of a game structure* C *and a set of error states* E*. A symbolic safety game with incomplete information* $\mathcal{G} = (\mathcal{C}, err)$ *consists of a symbolic game structure* $\mathcal{G}$ *and an error formula err which denotes the set of error states.*

Definition 32 (Prefix)**.** *A prefix in* G *is a finite sequence* $\pi = v_0 v_1 ... v_n$ *of states such that for all* $0 \leq j < n$ *it holds that* $(v_j, v_{j+1}) \in R$ *and* $v_0 = s_0$*. If* $last(\pi) \in S_s$ *then it is called a system prefix and if* $last(\pi) \in S_e$ *it is called an environment prefix. The set* $Prefs(G)$ *is the set of prefixes in* G *and* $Prefs_s(G)$ *and* $Prefs_e(G)$ *denote the system and environment prefixes respectively.*

Definition 33 (Play)**.** *A play in* G *is either an infinite sequence of states* $w = v_0 v_1 v_2 ...$ *such that* $v_0 = s_0$ *and for all* $j \geq 0$ $(v_j, v_{j+1} \in R$ *or a prefix* π *such that* $last(\pi)$ *is a dead-end or an error state.*

Definition 34 (Strategy)**.** *A partial function* $f_s : Prefs_s(G) \rightarrow S_e$ *is non-blocking if* $f_s(\pi)$ *defined for every prefix* $\pi \in Prefs_s(G)$ *for which* $last(\pi)$ *not a dead-end and which is compatible with* f_s*, i.e., for all* $0 \leq j < |\pi| - 1$ *with* $\pi[j] \in S_s$ *it holds that* $\pi[j+1] = f_s(\pi[0, j])$*. A strategy for the system is a non-blocking partial function* $f_s : Prefs_s(G) \rightarrow S_e$ *such that if* $f_s(\pi) = v$ *then* $(last(\pi), v) \in R_s$*. A strategy* f_s *is called consistent if iff for all system prefixes* $\pi_1, \pi_2 \in Prefs(f_s)$*, it holds that* $f_s(\pi_1) \equiv f_s(\pi_2)$*. A strategy* f_e *for the environment is defined analogous.*

Definition 35 (Outcome)**.** *The outcome of two strategies* f_s *and* f_e *for the system and the environment is a play* $w = Outcome(f_s, f_e)$ *such that for all* $0 \leq j < |w| - 1$ *if* $w[j] \in S_s$ *then* $w[j+1] = f_s(w[0, j])$ *and if* $w[j] \in S_e$ *then* $w[j+1] = f_e(w[0, j])$*.*

Definition 36 (Winning Strategy)**.** *A system strategy* f_s *is winning iff for every strategy* f_e *of the environment, if* $w = Outcome(f_s, f_e)$ *then* w *is an infinite play and for every* $j \geq 0$*,* $w[j]$ *is not an error state.*

4 Symbolic Encodings of Timed Game Structures with Incomplete Information

In this section we offer for both models described in section 2.2 a symbolic encoding. That means we define a symbolic game structure $\mathcal{G}$ such that this game structure encode the original game structure. We define first the encoding and prove afterwards that this encoding is correct.
The challenge of this encoding is to encode the rules of both games. These rules have to be encoded by the formulas which represents the transition relations.

4.1 Notations

Definition 37. *Let X^c be a finite set of real valued clocks, C be a set of variables which contains a clock variable c_x for each clock x in X^c and d be a variable which also represents a clock. For each clock $x \in X^c$, we denote with $c_x \in C$ the corresponding clock variable. The set C' consists of the primed versions of the variables in C. For a constraint φ,*

- *the constraint $\varphi_{[X^c \to C]}$ is obtained from φ by substituting each clock $x \in X^c$ with the corresponding variable $c_x \in C$;*
- *the constraint $\varphi_{[X^c \to C']}$ is obtained from φ by substituting each clock $x \in X^c$ with the corresponding variable $c'_x \in C'$;*
- *the constraint $\varphi_{[Y^c \to 0, X^c \setminus Y^c \to C]}$ is obtained from φ by substituting each clock $x \in Y^c \subseteq X^c$ with the constant 0 and each clock $x \in X^c \setminus Y^c$ with the corresponding variable $c_x \in C$;*
- *the constraint $\varphi_{[X^c \to C + d' - d]}$ is obtained from φ by substituting each clock $x \in X^c$ with $c_x + d' - d$ where c_x is the corresponding clock variable of the clock x.*

4.2 Encoding of the Model Based on Observation Predicates

In this section we define an encoding for the model from [11] which is described in section 2.3.2. We encode a M-TGS $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$ with the winning condition $\mathcal{O}$ (we consider only winning conditions which describe safety games) as a symbolic safety game $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$. We define first the set of variables X for the symbolic safety game structure $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ and explain the idea of the encoding. In the next step the formulas $init$, $\mathcal{T}_s$, $\mathcal{T}_e$ and err will be defined and the correctness of the encoding is proved.

4.2.1 Definition of the Encoding

Let us fix the M-TGS $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$.

Variables of the symbolic game structure. The set of variables for the encoding $\mathcal{G}$ is defined by $X = X_h \cup X_i \cup X_o \cup \{t\}$ where

$$
\begin{aligned}
X_h =& \{location, resetDone\} \cup C \cup \{c_{synch}\} \\
X_i =& Obs \\
X_o =& \{act\} \cup ResetVars
\end{aligned}
$$

and

- The special variable t determines whose turn it is. The domain of t is $\{0, 1\}$.
- The variable *location* models the current location in H. Without loss of generality the domain of *location* is L, which we assume to be a finite subset of $\mathbb{N}$.
- The set C contains a clock variable c_x for each clock x in X_1^c and a clock variable c_x for each clock x in X_2^c to encode the clocks. The set $C = C^s \cup C^e$ is partitioned into the set of system clocks $C^s = \{c_x \mid x \in X_1^c\}$ and the set of environment clocks $C^e = \{c_x \mid x \in X_2^c\}$. The domain of each variable in C is V[5].
- The clock variable c_{synch} will be used to synchronize the clock variables. The domain of c_{synch} is V.
- The boolean variable *resetDone* indicates whether the reset of the system clocks is already done.
- The set *Obs* contains one boolean variable obs_p for each observable predicate $p \in \mathcal{P}$ to encode the observable predicates.
- The variable *act* denotes the action chosen by player 1. The domain of *act* is $\Sigma_1 \cup \{\Delta, Reset\}$ where Σ_1 is the set of actions of player 1, Δ is the special delay action and *Reset* is the action the system plays if it wants to reset system clocks.
- The set $ResetVars$ contains one boolean variable $reset_y \in \{0, 1\}$ for each clock $y \in X_1^c$ which indicates whether x should be reseted or not.

[5] We defined V by $V = \mathbb{R}_{\geq 0}^{X^c} \cap [0, M]$ in section 2.3.2

- The boolean variable *ttAllowed* determines whether the environment is allowed to make a time transition.

Idea of the encoding. While the variables encode the states of a M-TGS H, the rules of a M-TGS H will be encoded in the transition relations in $\mathcal{G}$.

- The choice of an action by player 1 is encoded by a system transition which allows the system to assign a new value $a \in \Sigma_1 \cup \{\Delta, Reset\}$ to the variable *act*. If it assigns *Reset* to *act* it can also set the corresponding variables in $ResetVars$ to true for each own clock it wants to reset. With such a transition the system gives the turn back to the environment.

- Discrete transitions in H are encoded by environment transitions. These transitions assign a new value to the *location* variable and set to 0 those clock variables that correspond to clocks which are reseted in the discrete transition in H. Each variable $obs \in Obs$ is updated according to the new values of the clock variables and the new location. The turn is back to the system if the value of some variable $obs \in Obs$ changes.

- Time transitions in H are encoded by environment transitions which increase all clock variables by the same amount. These transitions are only enabled if there is no transition enabled which encodes a transition $(l, g, a, Y, l') \in \delta$ where $a \in \Sigma_1$ and a is the value of the variable *act* from the current state. To ensure that the time can only elapse until an observation changes or a transition with the action assigned to *act* becomes enabled, the valuation of the clock variables has to remain within the bounds of intervals which are extracted from the observation predicates and the guards of the actions in Σ_1. If such a transition is made the value of the variables in $C \cup \{c_{synch}\}$ are updated, each variable $obs \in Obs$ is updated according to the new values of the clock variables and if some variable $obs \in Obs$ has a different value than before, the environment gives the turn back to the system.

- If the choice of player 1 was a reset action player 2 has to reset the clocks of player 1. We create environment transitions which reset all system clocks c_x where the variable $reset_x$ has the value true. Also we set the variable $resetDone$ to true which indicates that the reset was done and is not executed multiple times.

- To prevent the system in G from getting an advantage by counting the played transitions of environment we give the environment the possibility to play transitions which do not change the values of the variables.

Auxiliary formulas. For a better overview we create abbreviations for sub formulas which are used in the next sections.

- A variable $obs_p \in Obs$ is true if the current state satisfies that the valuation of *location* is in K and satisfies $\varphi_{[X^c \to C]}$ where $p = (K, \varphi)$

  $$\textbf{initialObservation} := \bigwedge_{p=(K,\varphi)\in\mathcal{P}} obs_p \leftrightarrow \left(\bigvee_{l \in K} location \approx l \wedge \varphi_{[X^c \to C]} \right)$$

- A variable obs'_p for $obs_p \in Obs$ is true if after a transition the current state satisfies that the valuation of *location* is in K and satisfies $\varphi_{[X^c \to C]}$ where $p = (K, \varphi)$

  $$\textbf{observationSet} := \bigwedge_{p=(K,\varphi)\in\mathcal{P}} obs'_p \leftrightarrow \left(\bigvee_{l \in K} location' \approx l \wedge \varphi_{[X^c \to C']} \right)$$

- All variables in Obs will remain unchanged

  $$\textbf{observationsStayEqual} := \bigwedge_{p\in\mathcal{P}} obs'_p \approx obs_p$$

- If the observation variables after a transition are unchanged then the environment player keeps the turn, otherwise the system gets the turn

  $$\textbf{nextPlayer} := \textbf{observationsStayEqual} \leftrightarrow t' \approx 0$$

- Increase the values of all clock variables by the same amount

  $$\textbf{clocksUpdated} := \bigwedge_{x \in X^c} \left(c'_x \approx c_x + c'_{synch} - c_{synch} \right)$$

- The value of the synchronizing clock increases by a strictly positive amount

  $$\textbf{timeElapses} := c'_{synch} > c_{synch}$$

- There is no transition with the chosen action denoted by the variable *act* which is enabled in the current state

$$\textbf{chosenActionIsNotEnabled}(\mathbf{l}) := \neg \bigvee_{a \in \Sigma_1} \left(act \approx a \wedge \bigvee_{(l,a,g,Y,l') \in \delta} g_{[X^c \rightarrow C]} \right)$$

- The invariant of the locations l_1 and l_2 holds before and after the transition:

$$\textbf{invariantsHold}(\mathbf{l_1}, \mathbf{l_2}) := inv(l_1)_{[X^c -> C]} \wedge inv(l_2)_{[X^c \rightarrow C']}$$

- If the chosen action is *Reset* than the reset is already done

$$\textbf{noReset} := act \approx Reset \rightarrow resetDone$$

- Each clock variable $c_x \in C^s$, for which the value of $reset_x$ is true is set to 0

$$\textbf{resetSystemClocks} := \bigwedge_{x \in X_1^c} ((reset_x \wedge c'_x \approx 0) \vee (\neg reset_x \wedge c'_x \approx c_x))$$

- All clock variables c_x of the environment with $x \in Y \subseteq X_2^c$ are set to 0

$$\textbf{resetEnvClocks}(Y) := \bigwedge_{x \in Y} c'_x \approx 0 \wedge \bigwedge_{x \in X_2^c \setminus Y} c'_x \approx c_x$$

Formula init. The formula *init* is defined as follows:

$$\begin{aligned} init = & t \approx 1 \wedge location \approx \iota \wedge \textbf{initialObservation} \\ & \wedge \bigwedge_{x \in X^C} c_x \approx 0 \wedge c_{synch} \approx 0 \wedge \bigwedge_{x \in X_1^C} \neg reset_x \\ & \wedge \neg resetDone \wedge act \approx \Delta \wedge ttAllowed \end{aligned}$$

System transitions If it is the turn of player 1 she can choose an action from the set Σ_1^{ext}. The following formula models the choice of player 1 as system transitions:

$$\begin{aligned} \mathcal{T}_s = & t \approx 1 \wedge t' \approx 0 \wedge location' \approx location \wedge \textbf{observationsStayEqual} \\ & \wedge \bigwedge_{x \in X^c} c'_x \approx c_x \wedge c'_{synch} \approx c_{synch} \\ & \wedge resetDone' \approx resetDone \wedge ttAllowed' \approx ttAllowed \end{aligned}$$

Environment transitions. As mentioned above there will be four kinds of environment transitions: time transitions which model that player 2 can let time pass, discrete transitions which model that player 2 plays a discrete action, reset transitions which model that player 2 resets a subset of clocks in X_1^c chosen by player 1 and do nothing transition.
Time transitions. For each location $l \in L$, the formula **timeTransition(l)** represents a time transitions in location l. The environment is allowed to let time elapse by playing a time transition only while

(i) the variables in Obs are unchanged and

(ii) if the value of act is a discrete action $a \in \Sigma_1$ there exists no transition $(l, g, a, Y, l') \in \delta$ where g is satisfied by the current state. Note that the action *Reset* is always enabled and the action Δ is never enabled.

For a location $l \in L$ the set

$$Obs_l = \{\varphi \mid (K, \varphi) \in \mathcal{P},\ l \in K\}$$

consists of all constraints in $\mathcal{B}(X^c, M)$ of observation predicates which could be true in location l. For a location l and an action $a \in \Sigma_1$ the set

$$Ena_{l,a} = \{\varphi \mid (l, \varphi, a, Y, l') \in \delta\}$$

consists of all guards in $\mathcal{B}(X^c, M)$ of transitions $(l, g, a, Y, l') \in \delta$ which could be enabled in location l.

Definition 38. *Let $\varphi \in \mathcal{B}(X^c, M)$ and $x \in X^c$. The constraint $\varphi \downarrow_C x$ is obtained from φ by removing all atomic constraints which contain clocks different from x.*

For each clock $x \in X^c$ a constraint $\varphi \downarrow_C x$ obtained from the constraint φ has the form $k_1 \leq x < k_2$ and describes an interval in $\mathbb{R}_{\geq 0} \cup [0, M]$. We define for a constraint φ in $\mathcal{B}(X^c, M)$ and a clock x the lower and upper bound for that clock.

$$k_1(\varphi, x) = min\{v(x) \mid v(x) \models \varphi \downarrow_C x\}$$
$$k_2(\varphi, x) = sup\{v(x) \mid v(x) \models \varphi \downarrow_C x\}$$

The following formula is satisfied by a valuation if for a constraint φ and a set of clocks Y there exists a valuation, which can be reached by adding time to the clock variables, all clocks are in their interval given by φ.

$$\textbf{couldBeSatisfied}(\varphi, \mathbf{Y}) :=$$
$$\bigwedge_{x \in Y} \left[c_x < k_2(\varphi, x) \wedge \left(c_x < k_1(\varphi, x) \rightarrow \bigwedge_{\substack{y \in Y \\ x \neq y}} c_y + k_1(\varphi, x) - c_x < k_2(\varphi, y) \right) \right]$$

To model the constraints for time transitions, we define the following auxiliary formulas:

- In a state w where $w \models \mathit{location} \approx l$, the environment is allowed to let time pass by increasing the values of the clock variables $c_x \in C$ only until an observation in Obs_l changes

$$\textbf{stopAfterObsChange}(\mathbf{l}) :=$$
$$\bigwedge_{\varphi \in \mathit{Obs}_l} \Bigg[\left(\left(\left(\bigvee_{x \in X^c} c_x < k_1(\varphi, x) \right) \rightarrow \left(\bigvee_{x \in X^c} c'_x \leq k_1(\varphi, x) \right) \right) \right.$$
$$\vee \left(\varphi_{[X^c \rightarrow C]} \rightarrow \left(\bigwedge_{x \in X^c} c'_x \leq k_2(\varphi, x) \right) \right)$$
$$\vee \neg \textbf{couldBeSatisfied}(\varphi, X^c) \Bigg]$$

- In a state w where $w \models \mathit{location} \approx l$, the environment is allowed to let time pass by increasing the values of the clock variables $c_x \in C$ until the action a become enabled where a is the value of the variable act.

$$\textbf{stopAfterChosenActionBecameEnabled}(\mathbf{l}) :=$$
$$\bigvee_{a \in \Sigma_1} \bigwedge_{g \in \mathit{Ena}_{l,a}} \Bigg[\mathit{act} \approx a \wedge \left(\left(\left(\left(\bigvee_{x \in X_2^c} c_x < k_1(g, x) \right) \right.\right.\right.$$
$$\left.\left.\left. \rightarrow \left(\bigvee_{x \in X_2^c} c'_x \leq k_1(g, x) \right) \right) \right) \vee \neg \textbf{couldBeSatisfied}(g, X_2^c) \right) \Bigg]$$
$$\vee\ \mathit{act} \approx \mathit{Reset} \vee \mathit{act} \approx \Delta$$

- There exists a discrete transition which is enabled after a transition

  actionEnabled(**l**) :=
  $$\left(\bigvee_{a\in\Sigma_2}\bigvee_{g\in Ena_{l,a}} g_{[X\to C']}\right) \vee \left(\bigvee_{a\in\Sigma_1}\left(act \approx a \wedge \bigvee_{g\in Ena_{l,a}} g_{[X\to C']}\right)\right)$$

- After a time transition either a discrete transition is enabled or the observation changed which is equivalent to the next state being a system state

  actionEnabledOrObservationChanged(**l**) :=
  $$(\mathbf{actionEnabled}(\mathbf{l}) \wedge \neg ttAllowed' \wedge t' \approx 0) \vee (ttAllowed' \wedge t' \approx 1)$$

Then a formula **timeTransition(l)** for a location $l \in L$ is defined as follows:

$$\begin{aligned}\mathbf{timeTransition}(\mathbf{l}) = & t \approx 0 \wedge \mathbf{nextPlayer} \wedge ttAllowed \\ & \wedge location \approx l \wedge location' \approx location \\ & \wedge \mathbf{invariantsHolds}(\mathbf{l},\mathbf{l}) \wedge \mathbf{chosenActionIsNotEnabled}(\mathbf{l}) \\ & \wedge \mathbf{noReset} \wedge \mathbf{observationSet} \\ & \wedge \mathbf{clocksUpdated} \wedge \mathbf{timeElapses} \\ & \wedge \mathbf{stopAfterObsChange}(\mathbf{l}) \\ & \wedge \mathbf{stopAfterChosenActionBecameEnabled}(\mathbf{l}) \\ & \wedge \mathbf{actionEnabledOrObservationChanged}(\mathbf{l}) \\ & \wedge resetDone' \approx resetDone \wedge act' \approx act \\ & \wedge \bigwedge_{x\in X_1^c} reset'_x \approx reset_x\end{aligned}$$

Discrete transitions. We create a transition in G for each transition $(l, g, a, Y, l') \in \delta$. First we define the following auxiliary formulas

- The guard of the transition is satisfied:

  $$\mathbf{theGuardIsSatisfied}(g) := g_{[X^c\to C]}$$

- If a is in Σ_1 then $act \approx a$:

  $$\mathbf{player1ActionMustBeChosen}(a) := act \approx a$$

An environment transition for a transition $(l, g, a, Y, l') \in \delta$ with $a \in \Sigma_2$ is defined as follows:

$$\begin{aligned}\textbf{discreteTransitionP2}(l, g, a, Y, l') = & t \approx 0 \wedge \textbf{nextPlayer} \\ & \wedge location \approx l \wedge location' \approx l' \\ & \wedge \textbf{theGuardIsSatisfied}(g) \\ & \wedge \textbf{invariantsHold}(l, l') \\ & \wedge \textbf{observationSet} \wedge \textbf{resetEnvClocks}\,(Y) \\ & \wedge act' \approx act \wedge ttAllowed' \wedge c'_{synch} \approx c_{synch} \\ & \wedge \bigwedge_{x \in X_1^c} c'_x \approx c_x \wedge \bigwedge_{x \in X_1^c} reset'_x \approx reset_x\end{aligned}$$

For a transition $(l, g, a, Y, l') \in \delta$ with $a \in \Sigma_1$ an environment transition is defined as follows:

$$\begin{aligned}\textbf{discreteTransitionP1}(l, g, a, Y, l') = & t \approx 0 \wedge \textbf{nextPlayer} \\ & \wedge location \approx l \wedge location' \approx l' \\ & \wedge \textbf{theGuardIsSatisfied}(g) \\ & \wedge \textbf{invariantsHold}(l, l') \\ & \wedge \textbf{player1ActionMustBeChosen}(a) \\ & \wedge \textbf{observationSet} \wedge \textbf{resetEnvClocks}\,(Y) \\ & \wedge act' \approx act \wedge ttAllowed' \wedge c'_{synch} \approx c_{synch} \\ & \wedge \bigwedge_{x \in X_1^c} c'_x \approx c_x \wedge \bigwedge_{x \in X_1^c} reset'_x \approx reset_x\end{aligned}$$

Reset transitions. When the system assigns *Reset* to the variable *act*, the environment has set to 0 the clock variables c_x for which $reset_x$ is true.

For each location $l \in L$ we define the formula

$$\begin{aligned}\mathbf{resetTransition(l)} =& t \approx 0 \wedge \mathbf{nextPlayer}\\ &\wedge location \approx l \wedge location' \approx location\\ &\wedge \mathbf{observationSet} \wedge \mathbf{resetSystemClocks}\\ &\wedge \mathbf{invariantsHolds(l, l)}\\ &\wedge \bigwedge_{x \in X_2^c} c'_x \approx c_x \wedge c'_{synch} \approx c_{synch} \wedge \bigwedge_{x \in X_1^c} reset'_x \approx reset_x\\ &\wedge \neg resetDone \wedge resetDone'\\ &\wedge act \approx Reset \wedge act' \approx act \wedge ttAllowed'\end{aligned}$$

Do nothing transition. These transitions will not change the values of any variables. The environment is allowed to play these transition at any state such that the system will not get an advantage by counting the number of played transitions. As we consider only safety games the environment will not win by playing do nothing transitions. The following formula defines the set of the do nothing transitions

$$\begin{aligned}\mathbf{doNothingTransition} =& t \approx 0 \wedge t' \approx 0 \wedge location' \approx location\\ &\wedge \bigwedge_{x \in X^c} c'_x \approx c_x \wedge c'_{synch} \approx c_{synch} \wedge act' \approx act\\ &\wedge resetDone' \approx resetDone \wedge \bigwedge_{p \in \mathcal{P}} obs'_p \approx obs_p\\ &\wedge \bigwedge_{x \in X_1^c} reset'_x \approx reset_x \wedge ttAllowed' \approx ttAllowed\end{aligned}$$

Transition relation for the environment. The formula $\mathcal{T}_e$ which denotes the transition relation for the environment is defined as:

$$\begin{aligned}\mathcal{T}_e =& \bigvee_{l \in L} \mathbf{timeTransition(l)}\\ &\vee \bigvee_{\substack{(l,g,a,Y,l') \in \delta\\ a \in \Sigma_2}} \mathbf{discreteTransitionP2}(l, g, a, Y, l')\\ &\vee \bigvee_{\substack{(l,g,a,Y,l') \in \delta\\ a \in \Sigma_1}} \mathbf{discreteTransitionP1}(l, g, a, Y, l')\\ &\vee \bigvee_{l \in L} \mathbf{resetTransition(l)} \wedge \mathbf{doNothingTransition}\end{aligned}$$

The Error Formula. As we only consider safety games we can assure that the winning condition $\mathcal{W}$ for H is given as $\mathcal{W} = \mathcal{O}^\omega \setminus \mathcal{E}$, where $\mathcal{E} = \{o_0 o_1 ... \mid \exists j : o_j \in \mathcal{O}_{err}\}$ is the set of erroneous sequences of observations defined by a subset $\mathcal{O}_{err}$ of $\mathcal{O}$. We define the formula err for $\mathcal{G}$ as:

$$err = \bigvee_{o \in \mathcal{O}_{err}} \bigwedge_{p \in \mathcal{P}} obs_p \leftrightarrow o(p)$$

This symbolic game structure satisfies the four conditions to define a valid game structure (see section 3).

4.2.2 Correctness of the Encoding

We will show that the given encoding is correct. First we introduce a mapping from the set of prefixes in a M-timed game structure $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$ with the winning condition $\mathcal{O}$ to sequences of states of the corresponding encoding $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$. The game $G = ((S_s, S_e, s_o, R_s, R_e), E)$ corresponds to the symbolic game $\mathcal{G}$.
A consistent prefix ρ^H in H is mapped via the function map onto a sequence of states π in G. We give a recursive definition of the function $map : Pref^{co}(H) \rightarrow Seq$, where the set Seq denotes the set of finite and infinite sequences of states in G. This is done in the following way:

1. If $\rho^H = (l_0, v_0)$ then $map(\rho^H) = w$ where w is the single state that satisfies $init$.
2. If $\rho^H = (l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)$ ends with a state and $n \neq 0$, then
   a) If $\rho^H(n)$ is a choice point
   $map(\rho^H) = map((l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}) \cdot w_s$
   b) If $\rho^H(n)$ is not a choice point
   $map(\rho^H) = map((l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}) \cdot w_e$
3. If $\rho^H = (l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)c_n$ ends with $c_n \in \Sigma_1^{ext}$ and $n \neq 0$, then
   a) If $\rho^H(n)$ is a choice point
   $map(\rho^H) = map((l_0, v_0)c_0\sigma_0...(l_n, v_n)) \cdot w_e$
   b) If $\rho^H(n)$ is not a choice point
   $map(\rho^H) = map((l_0, v_0)c_0\sigma_0...(l_n, v_n))$

where we define the unique determined states $w_e \in S_e$ and $w_s \in S_s$ as

$$\begin{aligned} w_e \models & t \approx 0 \wedge act \approx act?(c_n) \wedge location \approx l_n \\ & \wedge \bigwedge_{x \in X^c} reset_x \approx reset?(c_n, x) \wedge resetDone \approx resetDone?(\rho^H, \sigma_{n-1}) \\ & \wedge \bigwedge_{x \in X^c} c_x \approx v_n(x) \wedge \bigwedge_{p \in \mathcal{P}} obs_p \approx (\gamma^{-1}(l_n, v_n))(p) \\ & \wedge c_{synch} \approx csynch?(\rho^H, \sigma_{i-1}) \wedge ttAllowed' \approx ttAllowed?(\rho^H, \sigma) \end{aligned}$$

$$\begin{aligned} w_s \models & t \approx 1 \wedge act \approx act?(c_{n-1}) \wedge location \approx l_n \\ & \wedge \bigwedge_{x \in X^c} reset_x \approx reset?(c_{n-1}, x) \wedge \neg resetDone \\ & \wedge \bigwedge_{x \in X^c} c_x \approx v_n(x) \wedge \bigwedge_{p \in \mathcal{P}} obs_p \approx (\gamma^{-1}(l_n, v_n))(p) \\ & \wedge c_{synch} \approx csynch?(\rho^H, \sigma_{n-1}) \wedge ttAllowed' \end{aligned}$$

and the auxiliary functions act?, reset?, resetDone?, csynch? and ttAllowed? as

$$act?(c) = \begin{cases} c, & \text{if } c \in \Sigma_1 \cup \{\Delta\} \\ Reset, & \text{else} \end{cases}$$

$$reset?(c, x) = \begin{cases} 1, & \text{if } c = Reset_Y \text{ and } x \in Y \\ 0, & \text{else} \end{cases}$$

$$resetDone?(\rho, \sigma) = \begin{cases} 1, & \text{if } \sigma = Reset_Y \\ 0, & \text{else if } \rho \text{ is a choice point} \\ resetDone?(\rho_{pre}, last\sigma(\rho)), & \text{else} \end{cases}$$

$$csynch?(\rho, \sigma) = \begin{cases} 0, & \text{if } \rho = (\iota, \mathbf{0}) \\ csynch?(\rho(|\rho| - 1), last\sigma(\rho)) + \sigma, & \text{if } \sigma \in \mathbb{R}_{>0} \\ csynch?(\rho(|\rho| - 1), last\sigma(\rho)), & \text{else} \end{cases}$$

$$ttAllowed?(\rho, \sigma) = \begin{cases} 0, & \text{if } \rho \text{ is not a choice point and } \sigma \in \mathbb{R}_{>0} \\ 1, & \text{else} \end{cases}$$

Sequences of states in G which were generated while the environment played do nothing transitions are not in the range of *map*. As such transitions were just introduced to hide the number of transitions, a sequence with do nothing transitions has

the same corresponding prefix in H as the sequence which is obtained by removing all states from the prefix which where generated by playing do nothing transitions. We solve this problem by introducing equivalence classes of sequences in G.
But first we prove with the three following lemmas that for each prefix π in G generated without do nothing transitions there is a consistent prefix ρ^H in H such that $map(\rho^H) = \pi$ and vice versa.

Lemma 1. *Let* $\rho^H = (l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)c_n \in \mathit{Pref}^{co}(H)$ *with* $c_n \in \Sigma_1^{ext}$ *and let* $\pi = w_0w_1...w_{n-1}w_n \in \mathit{Prefs}(G)$. *If* $map(\rho^H) = \pi$ *then the following hold*

1. *If for a* $\sigma_n \in \mathbb{R}_{>0}$ *and a state* $(l_{n+1}, v_{n+1}) = (l_n, v_n + \sigma_n)$ *the sequence* $\rho^{H'} = \rho^H\sigma_n(l_{n+1}, v_{n+1})$ *is in* $\mathit{Pref}^{co}(H)$, *then there exists a state* w_{n+1} *such that the sequence* $\pi' = \pi \cdot w_{n+1}$ *is in* $\mathit{Prefs}(G)$ *and* $map(\rho^{H'}) = \pi'$.
2. *If for a state* w_{n+1} *with* $w_n \models c_{synch} \approx t_1$, $w_{n+1} \models c_{synch} \approx t_2$ *and* $t_2 > t_1$ *the sequence* $\pi' = \pi \cdot w_{n+1}$ *is in* $\mathit{Prefs}(G)$ *then there exists a* $\sigma_n \in \mathbb{R}_{>0}$ *and a state* $(l_{n+1}, v_{n+1}) = (l_n, v_n + \sigma_n)$ *in* S *such that the sequence* $\rho^{H'} = \rho^H\sigma_n(l_{n+1}, v_{n+1})$ *is in* $\mathit{Pref}^{co}(H)$ *and* $map(\rho^{H'}) = \pi'$.

Proof. The transition $((l_n, v_n)\sigma_n(l_{n+1}, v_{n+1}))$ with $\sigma_n \in \mathbb{R}_{>0}$ is in $\rightarrow$ iff the following condidtions hold

(i) $l_n = l_{n+1}$

(ii) $v_{n+1} = v_n + \sigma_n$

(iii) $\forall 0 \le t < \sigma_n$, $(l_n, v_n + t) \in S$ and $(l_{n+1}, v_{n+1}) \in S$

(iv) $\forall 0 \le t < \sigma_n$, $\gamma^{-1}(l_n, v_n + t) = \gamma^{-1}(l_n, v_n)$

The sequence $\rho^H\sigma_n(l_{n+1}, v_{n+1})$ is in $\mathit{Pref}(H)$ iff $((l_n, v_n)\sigma_n(l_{n+1}, v_{n+1}))$ is in $\rightarrow$ and the following condidtions hold

(v) $\forall 0 \le t < \sigma_n$, $c_n \notin enable(l_n, v_n + t)$

(vi) if σ_{n-1} and σ_n are in $\mathbb{R}_{>0}$ then $\gamma^{-1}(l_n, v_n) \neq \gamma^{-1}(l_{n+1}, v_{n+1})$

1. As the sequence $\rho^{H'}$ is in $\mathit{Pref}^{co}(H)$ we have $((l_n, v_n)\sigma_n(l_{n+1}, v_{n+1}))$ is in $\rightarrow$ and therefore (i)-(vi) hold. We apply map on $\rho^{H'}$

$$\begin{aligned} map(\rho^{H'}) &= map(\rho^H)w_{n+1} \\ &= \pi \cdot w_{n+1} \end{aligned}$$

We created a formula which represents time transitions for each location. The constraints $location' \approx location$ and **clocksUpdated** encode the conditions (i) and (ii). As the invariants are in $\mathcal{B}(X^c, M)$ the constraint invariantsHolds(l,l) is sufficient to encode the condition (iii). The constraints **stopAfterObsChange(l)** and **stopAfterChosenActionBecameEnabled(l)** encode the conditions (i),(ii),(iii), (iv) and (v). Because of (vi) it follows that either ρ^H was a choice point or $\sigma_{n-1} \notin \mathbb{R}_{>0}$. Thus from the definition of *map* follows that $w_n \models ttAllowed$ and we have that (w_n, w_{n+1}) is in R_e. Therefore $\pi' \in Prefs(G)$ and $map(\rho^{H'}) = \pi'$.

2. As $w_n \models c_{synch} \approx t_1$, $w_{n+1} \models c_{synch} \approx t_2$ and $t_2 > t_1$ it follows that $w_n \models ttAllowed$ and the transition (w_n, w_{n+1}) models a time transition. As the transition (w_n, w_{n+1}) is in R_e, the conditions $location' \approx location$, **clocksUpdated**, **invariantsHolds(l,l)**, **stopAfterObsChange(l)** and **stopAfterChosenAction-BecameEnabled(l)** hold and encode (i),(ii),(iii),(iv) and (v). As $w_n \models ttAllowed$ it follows that the condition (vi) is satisfied and thus there exists a sequence in H such that $\pi \cdot w_{n+1} = map(\rho^H)\ \sigma_{n_{n+1}}(l_{n+1}, v_{n+1})$ where $\sigma_{n_{n+1}} = t_2 - t_1$. Therefore $map(\rho^{H'}) = \pi'$ and $\rho^{H'} \in Pref(H)$. If π' is in $Prefs_e(G)$ it holds that for all $p \in P : w_n \models obs_p \leftrightarrow w_{n+1} \models obs_p$. As we do not add a new choice c to the prefix ρ^H and ρ^H is in $Pref^{co}(H)$ it follows that $\rho^{H'}$ is in $Pref^{co}(H)$. □

Lemma 2. *Let $\rho^H = (l_0, v_0)c_0\sigma_0...(l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n)c_n \in Pref^{co}(H)$ with $c_n \in \Sigma_1^{ext}$ and let $\pi = w_0 w_1 ... w_{n-1} w_n \in Prefs_e(G)$. If $map(\rho^H) = \pi$ then the following hold*

1. *If for an action $\sigma_n \in \Sigma_2 \cup \{c_n\}$ and a state (l_{n+1}, v_{n+1}) the sequence $\rho^{H'} = \rho^H \sigma_n (l_{n+1}, v_{n+1})$ is in $Pref^{co}(H)$, then there exists a state w_{n+1} in G such that $\pi' = \pi \cdot w_{n+1} \in Prefs_e(G)$ and $map(\rho^{H'}) = \pi'$.*
2. *If for a state w_{n+1} in G the sequence $\pi' = \pi \cdot w_{n+1}$ is in $Prefs_e(G)$, $t_1 = t_2$ where $w_n \models c_{synch} \approx t_1$ and $w_{n+1} \models c_{synch} \approx t_2$ and $w_n \neq w_{n+1}$, then there exists an action $\sigma_n \in \Sigma_2 \cup \{c_n\}$ and a state (l_{n+1}, v_{n+1}) such that the sequence $\rho^{H'} = \rho^H \sigma_n (l_{n+1}, v_{n+1})$ is in $Pref^{co}(H)$ and $map(\rho^{H'}) = \pi'$.*

Proof. The transition $((l_n, v_n)\sigma_n(l_{n+1}, v_{n+1}))$ with $\sigma_n \in \Sigma_2 \cup \{c_n\}$ is in $\rightarrow$ iff there exists a transition $(l_n, g, \sigma_n, Y, l_{n+1}) \in \delta$ such that the following conditions hold

(i) $v_n \models g$

(ii) $v_{n+1} = v_n[Y]$

(iii) $(l_n, v_n), (l_{n+1}, v_{n+1}) \in S$

1. As $\rho^{H'}$ is in *Pref*(H) the transition $((l_n, v_n), \sigma_n, (l_{n+1}, v_{n+1}))$ is in $\rightarrow$. We apply *map* on $\rho^{H'}$.

$$\begin{aligned} map(\rho^{H'}) &= map(\rho^{H})w_{n+1} \\ &= \pi \cdot w_{n+1} \\ &= \pi' \end{aligned}$$

It holds that (w_n, w_{n+1}) is in R_e because we created a transition in G for each discrete transition $(l_n, g, \sigma_n, Y, l_{n+1})$ in H. So **theGuardisSatisfied**(g), **resetEnvClocks**(Y) and **invariantsHold**(l_n, l_{n+1}) hold because they encode the conditions (i),(ii) and (iii). Thus π' is in *Prefs*(G) and $map(\rho^{H'}) = \pi'$

2. As $w_n \neq w_{n+1}$ and $t_1 = t_2$ the transition $(w_n, w_{n+1}) \in R_e$ models a discrete transition in H and not a do nothing transition. Let a l_{n+1} be the location in L, such that $w_{n+1} \models location \approx l_{n+1}$. Further let v_{n+1} be the valuation in $\mathbb{R}^{X^c}_{>0}$ such that for x in X^c it holds that $v_{n+1}(x) = t$ where $w_{n+1} \models c_x \approx t$. As each discrete transition in G represents a discrete transition in H there exists a transition $(l_n, g, \sigma_n, Y, l_{n+1}) \in \delta$. As (w_n, w_{n+1}) is in R_e the constraints **theGuardIsSatisfied**(g), **resetEnvClocks**(Y) and **invariantsHold**(l_n, l_{n+1}) hold and encode the conditions (i),(ii) and (iii). This means there exists a transition $((l_n, v_n), \sigma_n, (l_{n+1}, v_{n+1}))$ in $\rightarrow$. Thus $\rho^{H'}$ is in *Pref*(H) and from the definition of *map* follows that $map(\rho^{H'}) = \pi'$. If π' is in *Prefs*$_e(G)$ it holds that for all $p \in P : w_n \models obs_p \leftrightarrow w_{n+1} \models obs_p$ as the state w_{n+1} is in S_e. As we do not add a new choice c to the prefix ρ^H and ρ^H is in *Pref*$^{co}(H)$ it follows that $\rho^{H'}$ is in *Pref*$^{co}(H)$. ☐

Lemma 3. *Let* $\rho^H = (l_0, v_0)c_0\sigma_0 ... (l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n) \in Pref^{co}(H)$ *be a choice point and let* $\pi = w_0 w_1 ... w_{n-1} w_n \in Prefs_s(G)$. *If* $map(\rho^H) = \pi$ *then the following hold*

1. *If for an action* c_n *in* Σ_1^{ext} *the sequence* $\rho^{H'} = \rho^H c_n$ *is in* $Pref^{co}(H)$ *then there exists a state* w *and a transition* $(w_n, w_{n+1}) \in R_s$ *such that* $\pi' = \pi \cdot w$ *is in* *Prefs*(G) *and* $map(\rho^{H'}) = \pi'$.
2. *If for a state* w_{n+1} *in* G *and a transition* $(w_n, w_{n+1}) \in R_s$ *the sequence* $\pi' = \pi \cdot w_{n+1}$ *is in* *Prefs*(G) *then there exists an action* c_n *in* Σ_1^{ext} *such that* $\rho^H c_n$ *is in* $Pref^{co}(H)$ *and* $map(\rho^H c_n) = \pi'$.

Proof. **1.** We apply *map* on $\rho^{H'}$.

$$\begin{aligned} map(\rho^{H'}) &= map(\rho^{H})w_{n+1} \\ &= \pi \cdot w_{n+1} \\ &= \pi' \end{aligned}$$

Case 1: $c_n \in \Sigma_1 \cup \{\Delta\}$. As there exists for each $c \in \Sigma_1 \cup \{\Delta\}$ a system transition in G there exists a transition $(w_n, w_{n+1}) \in R_s$ with $w_{n+1} \models act \approx c_n$. Thus π' is in *Prefs*(G) and from the definition of *map* follows $map(\rho^{H'}) = \pi'$.
Case 2: $c_n \in \{Reset_Y \mid y \in 2^{X_1^c}\}$. As there exists for each $c \in \{Reset_Y \mid y \in 2^{X_1^c}\}$ a system transition in G there exists a transition $(w_n, w_{n+1}) \in R_s$ with $w_{n+1} \models act \approx Reset$ and $\forall x \in X_1^c : w \models reset_x \leftrightarrow x \in Y$. Thus π' is in *Prefs*(G) and from the definition of *map* follows $map(\rho^{H'}) = \pi'$.
2. As ρ^H is a choice point it follows that $\gamma^{-1}(l_n, v_n) \neq \gamma^{-1}(l_{n+1}, v_{n+1})$.
Case 1: $w_{n+1} \models act \approx c_n$ with $c_n \in \Sigma_1 \cup \{\Delta\}$. As $\gamma^{-1}(l_n, v_n) \neq \gamma^{-1}(l_{n+1}, v_{n+1})$ the prefix $\rho^H c_n$ is in $Pref^{co}(H)$ and from the definition of *map* follows $map(\rho^H c_n) = \pi'$.
Case 2: $w_{n+1} \models act \approx Reset$. We define $c_n = Reset_Y$ where $\forall x \in X_1^c : w_{n+1} \models reset_x \leftrightarrow x \in Y$. As $\gamma^{-1}(l_n, v_n) \neq \gamma^{-1}(l_{n+1}, v_{n+1})$ the prefix $\rho^H c_n$ is in $Pref^{co}(H)$ and from the definition of *map* follows $map(\rho^H c_n) = \pi'$. ☐

Lemma 2 shows that there is a discrete transition in H iff there is a corresponding environment transition in G and Lemma 1 proves this for time transitions. That each new choice of player 1 correspond to a system transition is proved in Lemma 3.
As consequence of the above three lemmas it follows that for each prefix ρ^H in H, the sequence $map(\rho^H)$ of states in G is a prefix in *Prefs*(G). Also for each prefix π in G which was generated with no do nothing transitions there exists a prefix $\rho^{H'}$ in H such that $\pi = map(\rho^{H'})$. This mapping can be extended to plays.
We define the function *shrink* which removes all transitions which are self loops, especially do nothing transitions, from a sequence π in G. The function defines an equivalence relation over the sequences of states in G.

Definition 39. *Let $\pi = w_0 w_1 ... w_i w_{i+1} w_{i+2} ... w_{n-1} w_n$ be a prefix in G. The prefix unique $\pi' = shrink(\pi)$ is obtained by applying recursively one of the following rules until none of them can be applied anymore:*

- *If $n \geq 2$, $w_i, w_{i+1} \in S_e$ for an $0 \leq i \leq n-2$ and $w_i = w_{i+1}$ then $\pi' = w_0 w_1 ... w_i w_{i+2} ... w_{n-1} w_n$ and π' is in Prefs(G)*

(there exists a transition (w_i, w_{i+2}) in R_e as (w_{i+1}, w_{i+2}) is in R_e and $w_i = w_{i+1}$)

- *If $n \geq 1$, $w_{n-1}, w_n \in S_e$ and $w_{n-1} = w_n$ then $\pi' = w_0 w_1 ... w_i w_{i+1} w_{i+2} ... w_{n-1}$ and π' is in $Prefs(G)$*

For each prefix π in G possibly generated with do nothing transitions there exists prefix π' with $shrink(\pi) = \pi'$ such that π' is in the range of *map*. Figure 4 shows the connection between *map* and *shrink*. Both map to a subset U of $Prefs(G)$ where U is the image of $Pref^{co}(H)$ under *map*. Note that *shrink* is surjective but not injective.

$$\begin{tikzcd}
Pref^{co}(H) \arrow[d, "map"'] \\
U \text{ with } map(Pref^{co}(H)) = U \\
Prefs(G) \arrow[u, "shrink"]
\end{tikzcd}$$

Figure 4: Mapping from prefixes in H onto prefixes in G

Definition 40. *For two prefixes π_1 and π_2 in G it holds that $\pi_1 \equiv^{obs} \pi_2$ iff for some ρ_1^H, ρ_2^H in $Pref(H)$ we have that $map(\rho_1^H) = shrink(\pi_1)$, $map(\rho_2^H) = shrink(\pi_2)$ and $Obs^*(\rho_1^H) = Obs^*(\rho_2^H)$.*

Figure 5 illustrate the meaning of $\equiv^{obs}$ equivalent. Two prefixes in G are $\equiv^{obs}$ equivalent if and only if they correspond to two prefixes in H which have the same finite observation i.e. if one consider only the sequences of system states (by removing the environment states) these sequences are observation equivalent. The both topmost prefixes are prefixes in G. Although it holds that $v_1 \equiv w_1$ and $v_4 \equiv w_3$ the prefixes are not observation equivalent as they differ in the length. Using shrink we obtain the two prefixes in the next row where no transition to the same state are present. Also these prefixes are not observation equivalent. From lemma 2, lemma 1 and lemma 3 follows that there exist two prefixes in H which map to the prefixes in the row above. As $v_1 \equiv w_1$ and $v_4 \equiv w_3$ it follows that the prefixes in H have the same finite observation and thus the both prefixes in G in the first row are $\equiv^{obs}$ equivalent.

If two prefixes are observation equivalent it follows that they are also $\equiv^{obs}$ equivalent. This is proved in the next lemma.

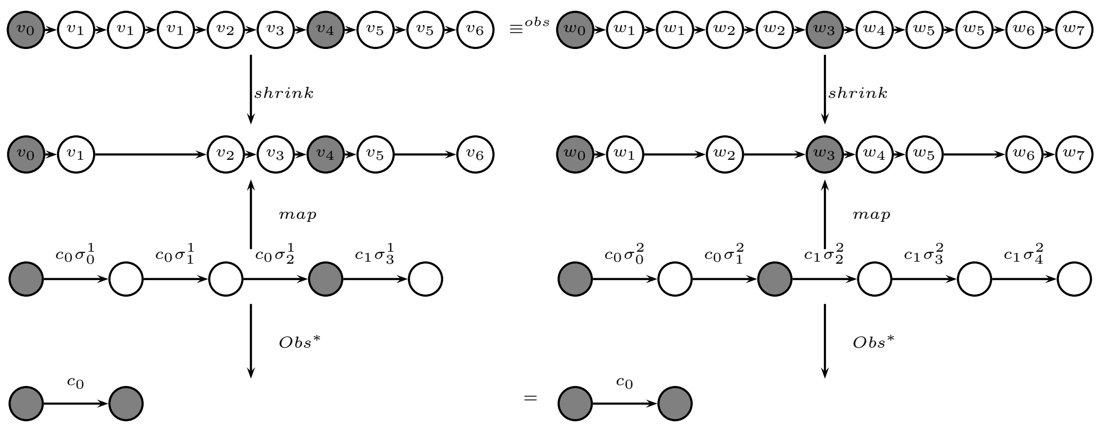

Figure 5: Example of two observation equivalence prefixes. The gray states denote the system prefixes in G and the choice points in H respective.

Lemma 4. *Let π_1 and π_2 be prefixes in Prefs(G). If $\pi_1 \equiv \pi_2$ then $\pi_1 \equiv^{obs} \pi_2$.*

Proof. Let $I_1 = \{n_0, n_1, ..., n_k\}$ be the index set where for all $j \geq 0$ it holds that $\pi_1[n_j]$ is a system state and $I_2 = \{m_0, m_1, ..., n_l\}$ be the index set where for all $j \geq 0$ it holds that $\pi_2[m_j]$ is a system state. As $\pi_1 \equiv \pi_2$ and the variable t is observable it holds that $I_1 = I_2$ and for all $0 \leq j \leq k$ hold that $\pi_1[n_j] \equiv \pi_2[n_j]$. Because of the definition of map it holds for all $0 \leq j \leq k$ that the prefixes $\rho_{1\ j}^H$ and $\rho_{2\ j}^H$ where $map(\rho_{1\ j}^H) = shrink(\pi_1[0,j])$ and $map(\rho_{2\ j}^H) = shrink(\pi_2[0,j])$ are both choice points. As it hold that $\pi_1[n_j] \equiv \pi_2[n_j]$ and $Obs = X_i$ it follows that $\gamma^{-1}(last(\rho_{1\ j}^H)) = \gamma^{-1}(last(\rho_{2\ j}^H))$. As each choice point is exactly mapped to a system state it follows that $Obs^*(\rho_1^H) = Obs^*(\rho_2^H)$ where $map(\rho_1^H) = shrink(\pi_1)$, $map(\rho_2^H) = shrink(\pi_2)$ and thus $\pi_1 \equiv^{obs} \pi_2$. $\square$

We are now able to define a strategy which returns, for prefixes in G which are $\equiv^{obs}$ equivalent, observation equivalent states.

Lemma 5. *If f_s is a consistent winning strategy for the system in the encoding $\mathcal{G}$ of H then there exists a consistent winning strategy f_s' such that if $\pi_1 \equiv^{obs} \pi_2$ then $f_s'(\pi_1) \equiv f_s'(\pi_2)$.*

Proof. For a prefix π in $Prefs_s(G)$ let $\pi_0 \in \{\pi' \in Prefs_s(G) \mid \pi' \equiv^{obs} \pi\}$ and $f_s(\pi_0) = w_0$. We define the strategy f_s' by $f_s'(\pi) = w$ where if $w_0 \models act \approx a$ then $w \models act \approx a$. Further $w \models t \approx 0$ and all other variables have the value of the state $last(\pi)$. For all prefixes π_1, π_2 in $Prefs_s(G)$ it holds that if $\pi_1 \equiv^{obs} \pi_2$ then $last(\pi_1) \equiv last(\pi_2)$ and by the definition of f_s' follows that $f_s'(\pi_1) \equiv f_s'(\pi_2)$. From lemma 4 follows that the strategy f_s' is consistent and as f_s is winning f_s' a consistent winning strategy. $\square$

Using the stated lemmas above we present the proof that given the encoding given as symbolic game for a M-TGS is correct.

Theorem 1. *There exists a consistent winning strategy for the system in the encoding* $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ *of a M-timed game structure* $H = ((L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P}))$ *with the safety winning condition* $\mathcal{O}$ *if and only if player 1 has a OBSI winning strategy in* H.

Proof. Let G be the game $G = ((S_s, S_e, s_o, R_s, R_e), E)$ which corresponds to the symbolic game $\mathcal{G}$. We first show that if the system has a consistent winning strategy in G then player 1 has a OBSI winning strategy in H. For a consistent prefix ρ^H let $lastCP(\rho^H)$ be the longest prefix of ρ^H which is a choice point. This is needed because a system strategy f_s is defined as $f_s : Prefs_s(G) \to S_e$.
Let f_s be a consistent winning strategy in G. From lemma 5 and the definition of $\equiv^{obs}$ it follows that there exists a consistent winning strategy f'_s such that if for two prefixes ρ_1^H and ρ_2^H it holds that $Obs^*(\rho_1^H) = Obs^*(\rho_2^H)$ then $f'_s(map(lastCP(\rho_1^H))) = f'_s(map(lastCP(\rho_2^H)))$. Let f'_s be such a strategy. We define the strategy λ^H by $\lambda^H(\rho^H) = c$ where c is the value of the variable act of the state defined by $w = f'_s(map(lastCP(\rho^H)))$. From lemma 1, lemma 2 and lemma 3 follow that for every $\rho^H \in Outcome(\lambda^H)$ there exists a $\pi \in Outcome(f'_s, f_e)$ for some f_e with $\pi = map(\rho^H)$. As it holds for two prefixes ρ_1^H and ρ_2^H with $Obs^*(\rho_1^H) = Obs^*(\rho_2^H)$ that $f'_s(map(lastCP(\rho_1^H))) \equiv f'_s(map(lastCP(\rho_2^H)))$ and $act \in X_o$ it follows that $\lambda^H(\rho_1^H) = \lambda^H(\rho_2^H)$ and therefore is λ^H an OBSI strategy. As no $\pi \in Outcome(f'_s, f_e)$ for some f_e visits an error state in G, no $\rho^H \in Outcome(\lambda^H)$ visits an error state in H. Thus λ^H is an OBSI winning strategy.
We show now that if there is a OBSI winning strategy for player 1 in H there is a winning strategy for the system in G. Let λ^H be an OBSI winning strategy for player 1 in H, π a system prefix in G and ρ^H a prefix in H such that $shrink(\pi) = map(\rho^H)$. We define the strategy f_s by $f_s(\pi) = w$ where if $last(map(\rho^H \lambda(\rho^H))) \models act \approx a$ then $w \models act \approx a$ and for all $x \in X_1^c$ if $last(map(\rho^H \lambda(\rho^H))) \models reset_x$ then $w \models reset_x$. Further $w \models t \approx 0$ and the other variables have the values of the state $last(\pi)$. From lemma 4 follows that for all prefixes π_1, π_2 in G it holds that if $\pi_1 \equiv \pi_2$ then $\pi_1 \equiv^{obs} \pi_2$. Thus $Obs^*(\rho_1^H) = Obs^*(\rho_2^H)$ where $shrink(\pi_1) = map(\rho_1^H)$ and $shrink(\pi_2) = map(\rho_2^H)$. As λ^H is an OBSI strategy it holds that $\lambda(\rho_1^H) = \lambda(\rho_2^H)$ and from the definition of f_s it follows that $f_s(\pi_1) \equiv f_s(\pi_2)$. Therefore f_s is consistent. From lemma 1, lemma 2 and lemma 3 it follows for every $\pi \in Outcome(f_s, f_e)$ that for some f_e, there exists a $\rho^H \in Outcome(\lambda^H)$ such that $shrink(\pi) = map(\rho^H)$. Therefore f_s defined for each prefix and is a non-blocking strategy. As no $\rho^H \in$

$Outcome(\lambda^H)$ visits an error state in H no $\pi \in Outcome(f'_s, f_e)$ for some f_e visits never an error state in G and hence f_s is a consistent winning strategy. Thus, the theorem is proved. □

4.3 Encoding of the Model Based on Synchronized Words

Again we give first define the set X of variables for the symbolic game structure $\mathcal{G}$ and explain the idea of the encoding. Afterwards the formulas *init*, $\mathcal{T}_s$, $\mathcal{T}_e$ and *err* are defined and the correctness of the encoding is proved. For a timed game $\Omega' = (\Gamma, \mathcal{H}', \mathcal{A})$ we restrict the automaton $\mathcal{A}$ to be deterministic. We use the conversation from [25] to convert Ω' into a timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ with an internal winning condition $\mathcal{I}$ which represents a set of error states as we consider safety games.

4.3.1 Definition of the Encoding

Variables of the symbolic game structure For a timed game $\Omega' = (\Gamma, \mathcal{H}', \mathcal{A})$ based on $(\hat{\Sigma}, \hat{X}^c, \mu)$ with $\hat{\Sigma} = (\Sigma_C, \Sigma_E^o, \Sigma_E^u)$, $\hat{X}^c = (X_r^c, X_u^c)$ and $\mu = (X_r^c \cup X_{cont}^c, n, max)$ let $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ be the conversation to a timed with an internal specification where $\mathcal{I}$ defines a set of error states in $\mathcal{H}$.
For an encoding G of a timed game Ω we define the set of variables $X = X_h \cup X_i \cup X_o$ where:

$$\begin{aligned}
X_h =& \{location, playedAction, obsActionPlayed\} \cup C \cup \{c_{synch}\}\\
X_i =& \{pickedAction\} \cup PickedAtoms \cup PickedResets\\
X_o =& OfferedSymbolicActions
\end{aligned}$$

- The variable t determines whose turn it is. The domain of t is $\{0, 1\}$,
- The variable *location* models the current location from Q. W.l.o.g. the domain of *location* is $Q = \{q_0, q_1, ..., q_p\}$,
- The set C contains a clock variable c_x for each clock specified in $X_r^c \cup X_u^c \cup X_{cont}^c$. The domain of each variable in C is $\mathbb{R}_{\geq 0}$.
- The clock variable c_{synch} will be used to synchronize the clock variables. The domain of c_{synch} is $\mathbb{R}_{\geq 0}$.
- The set *OfferedSymbolicActions* contains for each action $a \in \Sigma_C \cup \Sigma_E^o$, each atom $g \in atoms_\mu$ and each subset $Z \subseteq X_{cont}^c$ of controller clocks a boolean variable $offeredSymAction_{a,g,Z}$. A variable $offeredSymAction_{a,g,Z}$ indicates that the system offered the symbolic action (a, g, Z).

- The variable *pickedAction* identifies which offered action is picked by the environment. The domain of *pickedAction* is $\Sigma_C \cup \Sigma_E^o \cup \{\epsilon\}$,
- The set *PickedAtoms* contains a boolean variable $pickedAtom_g$ for each atom $g \in atoms_\mu$ to encode which atom the environment picked,
- The set *PickedResets* contains a boolean variable for each clock $x \in X^c_{cont}$. to encode which subset of clocks to reset the environment picked,
- The variable *playedAction* keeps track of the played actions. The domain of *playedAction* is $\Sigma \cup \{\epsilon\}$,
- The boolean variable *obsActionPlayed* indicates if an observable action was played after the last transition of the system.

Idea of the encoding A timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ is encoded in the following way.

- The offer of a subset of Γ from player C is encoded in a system transition. This transition allows the system to set the variables $offeredSymAction_{a,g,Z}$ to true if it wants to offer the symbolic action (a, g, Z) and gives the turn to the environment.
- When the environment gets the turn from the system it can choose to make a transition which represents an unobservable action or a transition which represents an observable or a controllable action which is in the offered set. After the environment plays an observable or a controllable action it sets the variable *obsActionPlayed* to true and after that it is only allowed to make transitions which represent an unobservable action or give the turn back to the system. The environment sets the *pickedAction* variable, the *pickedAtom* variables and the *pickedReset* to the corresponding values if it made an observable or a controllable action to track the pick. The clock variables and the variable location track the clock valuation and the current state of the transition system $\mathcal{H}$.
- To prevent the system in $\mathcal{G}$ to get an advantage by counting the played transitions of environment, we give the environment the possibility to play transitions which does not change the value of any of the variables.
- The environment can play a change player transition to give the turn back to the system after it played a observable or controllable action.

Auxiliary formulas

We define the following auxiliary formulas before defining the formulas that represent the transition relations.

- The picked symbolic action remains unchanged.

  pickStayEqual :=

  $$\mathit{pickedAction}' \approx \mathit{pickedAction} \wedge \bigwedge_{g \in \mathit{atoms}_\mu} \mathit{pickedAtom}'_g \approx \mathit{pickedAtom}_g$$
  $$\wedge \bigwedge_{x \in X^c_{cont}} \mathit{pickedReset}'_x \approx \mathit{pickedReset}_x$$

- The picked symbolic action is reseted.

  pickReset :=

  $$\mathit{pickedAction}' \approx \epsilon \wedge \bigwedge_{g \in \mathit{atoms}_\mu} \neg \mathit{pickedAtom}'_g \quad \wedge \bigwedge_{x \in X^c_{cont}} \neg \mathit{pickedReset}'_x$$

- The offered symbolic actions stayed equal.

  offerStayEqual :=

  $$\bigwedge_{a \in \Sigma_C \cup \Sigma^o_E} \bigwedge_{g \in \mathit{atoms}_\mu} \bigwedge_{Z \in 2^{X^c_{cont}}} \mathit{offeredSymAction}'_{a,g,Z} \approx \mathit{offeredSymAction}_{a,g,Z}$$

- The uncontrollable clocks were updated for a given set of clocks $Y \subseteq (X^c)$ which should be reseted.

  nonControllableClocksUpdated(Y) :=

  $$\bigwedge_{x \in (X^c) \setminus Y} c'_x \approx c_x + c'_{synch} - c_{synch} \wedge \bigwedge_{x \in Y} c'_x \approx 0$$

- The controllable clocks were updated for a given set of clocks $Z \subseteq X^c_{cont}$ which should be reseted.

  controllableClocksUpdated(Z) :=

  $$\bigwedge_{x \in (X^c_{cont}) \setminus Z} c'_x \approx c_x + c'_{synch} - c_{synch} \wedge \bigwedge_{x \in Z} c'_x \approx 0$$

- It is ensured that the time elapses.

  $$\textbf{timeElapses} := c'_{synch} > c_{synch}$$

- The clock values remain unchanged

  $$\textbf{timeStayEqual} := \bigwedge_{x \in (X^c \cup X^c_{cont})} c'_x = c_x \wedge c'_{synch} \approx c_{synch}$$

- There exists for a given action a an offered symbolic action where the offered atom is satisfied and the pick is tracked.

  $$\begin{aligned}&\textbf{updatePickAndCheckAtom(a)} := \\ &\bigvee_{g \in atoms_\mu} \bigvee_{Z \in 2^{X^c_{cont}}} \Bigg(\mathit{offeredSymAction}_{a,g,Z} \wedge g_{[X^c \to C + c'_{synch} - c_{synch}]} \\ &\qquad \wedge \textbf{controllableClocksUpdated}(\mathbf{Z}) \wedge \mathit{pickedAction}' \approx a \\ &\qquad \wedge \bigwedge_{h \in atoms_\mu, h \neq g} \neg \mathit{pickedAtom}'_h \wedge \mathit{pickedAtom}'_g \\ &\qquad \wedge \bigwedge_{x \in Z} \mathit{pickedReset}'_x \wedge \bigwedge_{x \in (X^c_{Cont} \setminus Z)} \neg \mathit{pickedReset}'_x \Bigg)\end{aligned}$$

Formula init The formula *init* that denotes the set of initial states is defined as follows:

$$\begin{aligned} \mathit{init} =& t \approx 1 \wedge \mathit{location} \approx q_0 \\ & \wedge \bigwedge_{x \in X^c \cup X^c_{cont}} c_x \approx 0 \wedge \bigwedge_{a \in \Sigma_C \cup \Sigma^o_E} \bigwedge_{g \in atoms_\mu} \bigwedge_{Z \in 2^{X^c_{cont}}} \neg \mathit{offeredSymAction}_{a,g,Z} \\ & \wedge \mathit{pickedAction} \approx \epsilon \wedge \bigwedge_{g \in atoms_\mu} \neg \mathit{pickedAtom}_g \wedge \bigwedge_{x \in X_{cont}} \neg \mathit{pickedReset}_x \\ & \wedge \neg \mathit{obsActionPlayed} \wedge \mathit{playedAction} \approx \epsilon \end{aligned}$$

System transitions Player C is allowed to choose an arbitrary subset of symbolic actions from Γ.

$$\begin{aligned} \mathcal{T}_s =& t \approx 1 \wedge t' \approx 0 \wedge \mathit{location}' \approx \mathit{location} \\ & \wedge \textbf{timeStayEqual} \wedge \textbf{pickStayEqual} \\ & \wedge \mathit{obsActionPlayed}' \approx \mathit{obsActionPlayed} \wedge \mathit{playedAction} \approx \mathit{playedAction}' \end{aligned}$$

Environment transitions. As a time word $\omega = (a_i, t_i)_{1\leq i}$ satisfies the monotonicity property ($\forall i < j, t_i < t_j$) we do not model time transitions explicitly but instead we combine them with the discrete ones by allowing the environment to let time elapse before taking the discrete action. We call such transitions mixed transitions. We define two other types of transitions: *do nothing transitions* and *change player transitions*. The formula that represents the transition relation for the environment will consist of these three types of transitions.
Mixed transitions. An unobservable transition $(q, g, a, Y, q') \in\rightarrow$ with $a \in \Sigma_E^u$ is encoded by following transition:

$$\begin{aligned}
&\mathbf{unobservableTransition}(q, g, a, Y, q') = \\
&\quad t \approx 0 \wedge t' \approx 0 \wedge \mathit{location} \approx q \wedge \mathit{location}' \approx q' \wedge g_{[X \to C + c'_{synch} - c_{synch}]} \\
&\quad \wedge \mathbf{nonControllableClocksUpdated(Y)} \wedge \mathbf{controllableClocksUpdated}(\emptyset) \\
&\quad \wedge \mathbf{timeElapses} \wedge \mathbf{offerStayEqual} \wedge \mathbf{pickStayEqual} \\
&\quad \wedge \mathit{obsActionPlayed}' \approx \mathit{obsActionPlayed} \wedge \mathit{playedAction}' \approx a
\end{aligned}$$

An observable or controllable transition $(q, g, a, Y, q') \in\rightarrow$ with $a \in (\Sigma_C \cup \Sigma_E^o)$ is encoded in a similar way:

$$\begin{aligned}
&\mathbf{observableTransition}(q, g, a, Y, q') = \\
&\quad t \approx 0 \wedge t' \approx 0 \wedge \mathit{location} \approx q \wedge \mathit{location}' \approx q' \wedge g_{[X \to C + c'_{synch} - c_{synch}]} \\
&\quad \wedge \mathbf{nonControllableClocksUpdated(Y)} \wedge \mathbf{timeElapses} \\
&\quad \wedge \mathbf{offerStayEqual} \wedge \mathbf{updatePickAndCheckAtom(a)} \\
&\quad \wedge \neg \mathit{obsActionPlayed} \wedge \mathit{obsActionPlayed}' \wedge \mathit{playedAction}' \approx a
\end{aligned}$$

Do nothing transitions To make sure that the system cannot get an advantage by counting the played unobservable actions, we add a do nothing transition. This transition can be played from the environment in each location and time point and leaves the values of the variables unchanged except for the variable *playedAction*. The system is not able to distinguish such transition from mixed transitions. As we consider only safety games the environment cannot win by playing a infinite sequence of do nothing transitions.

$$\begin{aligned}
\mathbf{doNothing} = &\, t \approx 0 \wedge t' \approx 0 \wedge \mathit{location}' \approx \mathit{location} \\
&\wedge \mathbf{timeStayEqual} \wedge \mathbf{offerStayEqual} \wedge \mathbf{pickStayEqual} \\
&\wedge \mathit{obsActionPlayed}' \approx \mathit{obsActionPlayed} \wedge \mathit{playedAction}' \approx \epsilon
\end{aligned}$$

Change player transition The change player transition allows the environment to give back the turn to the system after it has played an observable transition.

$$\begin{aligned}\textbf{changePlayer} =& t \approx 0 \wedge t' \approx 1 \wedge location' \approx location \\ & \wedge \textbf{timeStayEqual} \wedge \textbf{offerStayEqual} \wedge \textbf{pickReset} \\ & \wedge obsActionPlayed \wedge \neg obsActionPlayed' \wedge playedAction' \approx \epsilon\end{aligned}$$

So the formula $\mathcal{T}_e$ will be defined as follow

$$\begin{aligned}\mathcal{T}_e = & \bigvee_{(q_s,g,a,Y,q_t)\in\rightarrow a\in(\Sigma_C\cup\Sigma_E^o)} \textbf{observableTransition}(q_s,g,a,Y,q_t) \\ \vee & \bigvee_{(q_s,g,a,Y,q_t)\in\rightarrow a\in\Sigma_E^u} \textbf{unobservableTransition}(q_s,g,a,Y,q_t) \\ \vee & \textbf{ doNothing} \vee \textbf{changePlayer}\end{aligned}$$

The Error Formula Let **errorStates** be the formula which describe the set of error states derived from $\mathcal{I}$. Additionally to the error states, depending on location and time, we introduce error states which will be reached if a strategy is blocking or restricting. Let **errorStates** be the formula which is satisfied for error states derived from $\mathcal{A}$. We define the formula which models the error states as follow:

$$err = \textbf{restricting} \vee \textbf{blocking} \vee \textbf{noChoice} \vee \textbf{errorStates}$$

This formula models the situation when an action $a \in \Sigma_E^o$ is enabled, but it was not chosen from the system. Such a strategy is restricting.

$$\begin{aligned}\textbf{restricting} =& t \approx 0 \wedge \bigvee_{\substack{(q_s,a,g,Y,q_e)\in\rightarrow,\\ a\in\Sigma_E^o}} \Bigg[\big(location \approx q_s \wedge g_{[X^c\rightarrow C]}\big) \wedge \\ & \bigvee_{h\in atoms_\mu} \bigvee_{Z\in 2^X_{cont}} \big(offeredAtomAndReset_{a,h,Z} \rightarrow \neg h_{[X^c\rightarrow C]}\big) \Bigg]\end{aligned}$$

Let $\varphi^{DNF} = \psi_1 \vee \psi_2 \vee ... \vee \psi_n$ be the disjunctive normal form of a constraint $\varphi \in \mathcal{G}(X^c)$. The set of the conjunctive clauses is denoted by $DNF(\varphi) = \{\psi_1, \psi_2, ..., \psi_n\}$. Each contraint in the set $DNF(\varphi)$ is defined by the following grammar

$$\psi ::= (x \sim c) \mid \neg\psi \mid (\psi \wedge \psi)$$

where $\sim$ is one of $<$, $\leq$, $=$, $\geq$ and $>$. A constraint in the set $DNF(\varphi)$ describes an interval of valuations V in $\mathbb{R}_{\geq 0}^{X^c}$ where for all $v \in V$ hold that $v \models \varphi$.

Definition 41. *Let $\varphi \in \mathcal{G}(X^c)$ and $x \in X^c$. The constraint $\varphi \downarrow_B x$ is obtained from φ by removing all atomic constraints which contain clocks different from x.*

Let φ be a constraint in $\mathcal{G}(X^c)$ and ψ a constraint in the set $DNF(\varphi)$. We define for each clock in X^c the values

$$k_1(\varphi, x) = inf\{v(x) \mid v(x) \models \psi \downarrow_B x\}$$
$$k_2(\varphi, x) = sup\{v(x) \mid v(x) \models \psi \downarrow_B x\}$$

and the formulas

$$lessK_1(\psi, x, term) = \begin{cases} term < k_1(\psi, x) & \text{if } [\![\psi \downarrow_B x]\!] \text{ is left closed} \\ term \leq k_1(\psi, x) & \text{if } [\![\psi \downarrow_B x]\!] \text{ is left open} \\ true & \text{if } [\![\psi \downarrow_B x]\!] = \mathbb{R}_{\geq 0} \\ false & \text{if } [\![\psi \downarrow_B x]\!] = \emptyset \end{cases}$$

$$lessK_2(\psi, x, term) = \begin{cases} term < k_2(\psi, x) & \text{if } [\![\psi \downarrow_B x]\!] \text{ is right open} \\ term \leq k_2(\psi, x) & \text{if } [\![\psi \downarrow_B x]\!] \text{ is right closed} \\ true & \text{if } [\![\psi \downarrow_B x]\!] = \mathbb{R}_{\geq 0} \\ false & \text{if } [\![\psi \downarrow_B x]\!] = \emptyset \end{cases}$$

By using these variables und formulas we can determine if for a state w in G there exists a $t \in \mathbb{R}_{\geq}$ which can be added to the values of the variables such that a constraint φ can be true.

$$\begin{aligned} \mathbf{couldBeSatisfied}(\varphi, Y) := \bigvee_{\psi \in DNF(\varphi)} \bigwedge_{x \in Y} \Bigg[& c_x < k_2(\psi, x) \\ & \wedge \Bigg(lessK_1(\psi, x, c_x) \rightarrow \bigwedge_{\substack{y \in Y \\ x \neq y}} lessK_2(\psi, y, c_y + k_1(\psi, x) - c_x) \Bigg) \Bigg] \end{aligned}$$

This formula allows us to descripe blocking states

$$\begin{aligned}
\mathbf{blocking} = & t \approx 0 \wedge \bigwedge_{\substack{(q,a,g,Y,q') \in \rightarrow, \\ a \in \Sigma_E^u}} (location \approx q \rightarrow \neg \mathbf{couldBeSatisfied}(g, X^c)) \\
& \wedge \bigvee_{\substack{(q,a,g,Y,q') \in \rightarrow, \\ a \in \Sigma_E^o \cup \Sigma_C}} (location \approx q \wedge \mathbf{couldBeSatisfied}(g, X^c)) \\
& \wedge \bigwedge_{\substack{(q,a,g,Y,q') \in \rightarrow, \\ a \in \Sigma_E^o \cup \Sigma_C}} \Bigg[(location \approx q \wedge \mathbf{couldBeSatisfied}(g, X^c)) \\
& \rightarrow \bigwedge_{h \in atoms_\mu} \bigwedge_{Z \in 2^{X^c_{cont}}} \big(offeredSymAction_{a,h,Z} \approx 0 \\
& \vee \neg \mathbf{couldBeSatisfied}(g \wedge h, X^c \cup X^c_{cont})) \Bigg]
\end{aligned}$$

So the blocking formula will be true if

(i) no unobservable action will be enabled

(ii) there exists an observable or controllable action which will be enabled

(iii) for all transitions, with observable and controllable actions, which will be enabled it holds that there no offered symbolic action or the conjunction of the guard and the atom could not be satisfied

As mentioned, also the choice of no symbolic action will lead in an error state:

$$\mathbf{noChoice} = t \approx 0 \wedge \bigwedge_{a \in \Sigma_C \cup \Sigma_E^o} \bigwedge_{atoms_\mu} \bigwedge_{Z \in 2^{X^c}_{cont}} offeredSymAction_{a,g,Z} \approx 0$$

This symbolic game structure satisfies also the four conditions to define a valid game structure (see section 3).

4.3.2 Correctness of the Encoding

We show that the given encoding is correct. First we introduce mappings from prefixes of the encoding $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ onto symbolic words, onto timed words and onto sequences of valuations of the corresponding timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ based on $(\hat{\Sigma}, \hat{X}^c, \mu)$.

The mappings $sMap : Pref(G) \to (\Sigma \times \mathcal{G}(X^c \cup X^c_{cont}) \times 2^{X^c \cup X^c_{cont}})^*$, $twMap : Pref(G) \to (\Sigma \times \mathbb{R}_{\geq 0})^*$ and $vMap : Pref(G) \to (\mathbb{R}_{\geq 0}^{X^c})^*$ are defined in the following way.

1. If $\pi = init$ then

$$sMap(\pi) = \epsilon$$
$$twMap(\pi) = \epsilon$$
$$vMap(\pi) = \mathbf{0}$$

2. If $\pi = w_0 w_1 ... w_{n-1} w_n$, $n \geq 1$, $w_n \models playedAction \approx \epsilon$ and either $w_n \in S_s$ (the environment played a change player transition) or $w_n \in S_e$ (the environment played a do nothing transition) or $w_{n-1} \in S_s$ (the system played a system transition) then

$$sMap(\pi) = sMap(w_0 w_1 ... w_{n-1})$$
$$twMap(\pi) = twMap(w_0 w_1 ... w_{n-1})$$
$$vMap(\pi) = vMap(w_0 w_1 ... w_{n-1})$$

3. If $\pi = w_0 w_1 ... w_{n-1} w_n$, $n \geq 1$, $w_{n-1}, w_n \in S_e$ and $w_n \models playedAction \approx a_n$ where $a_n \in \Sigma_C \cup \Sigma^o_E$ (the environment played an controllable or observable transition) then

$$sMap(\pi) = sMap(w_0 w_1 ... w_{n-1})(a, g, Y)$$
$$twMap(\pi) = twMap(w_0 w_1 ... w_{n-1})(a, t)$$
$$vMap(\pi) = vMap(w_0 w_1 ... w_{n-1})v$$

where (a, g, Y), (a, t) and v are determined by w_{n-1} and w_n. Let us fix the following values determined by the states w_{n-1} and w_n:

$$w_{n-1} \models location \approx q_{n-1} \wedge c_{synch} \approx t_{n-1}$$
$$w_n \models location \approx q_n \wedge playedAction \approx a_n \wedge c_{synch} \approx t_n \wedge \bigwedge_{x \in X^c} c_x = t_x$$
$$\wedge\, pickedAtom_h$$

further let Z be a set of clocks such that $x \in Z$ iff $w_n \models pickedReset_x$. Then $(a, t) = (a_n, t_n)$, v is determined by $v(x) = t_x$ and if there exists a $(q_{n-1}, a_n, g_n, Y_n, q_n) \in \to$ such that $v \models g_n$ then $(a, g, Y) = (a_n, g_n \wedge h, Y_n \cup Z)$.

If there exists such a $(q_{n-1}, a_n, g_n, Y_n, q_n) \in\rightarrow$ it is unique as H is time deterministic [8]. Later we show that there exists such a $(q_{n-1}, a_n, g_n, Y_n, q_n) \in\rightarrow$.

4. If $\pi = w_0 w_1 ... w_{n-1} w_n$, $n \geq 1$, $w_{n-1}, w_n \in S_e$ and $w_n \models \mathit{playedAction} \approx a_n$ where $a_n \in \Sigma_E^u$ (the environment played an unobservable transition) then

$$\begin{aligned} sMap(\pi) &= sMap(w_0 w_1 ... w_{n-1})(a, g, Y) \\ twMap(\pi) &= twMap(w_0 w_1 ... w_{n-1})(a, t) \\ vMap(\pi) &= vMap(w_0 w_1 ... w_{n-1})v \end{aligned}$$

where (a, g, Y), (a, t) and v are determined by w_{n-1} and w_n. Let us fix the following values determined by the states w_{n-1} and w_n:

$$\begin{aligned} w_{n-1} \models& \mathit{location} \approx q_{n-1} \wedge c_{synch} \approx t_{n-1} \\ w_n \models& \mathit{location} \approx q_n \wedge \mathit{playedAction} \approx a_n \wedge c_{synch} \approx t_n \wedge \bigwedge_{x \in X^c} c_x = t_x \end{aligned}$$

then $(a, t) = (a_n, t_n)$, v is determined by $v(x) = t_x$ and if there exists a $(q_{n-1}, a_n, g_n, Y_n, q_n) \in\rightarrow$ such that $v \models g_n$ then $(a, g, Y) = (a_n, g_n, Y_n)$.

Let $\mathcal{G} = ((X, \mathit{init}, \mathcal{T}_s, \mathcal{T}_e), \mathit{err})$ be the encoding of Ω and $\mathcal{U}_\Gamma$ the universal, single state transition system over Γ which accepts all words in Γ^∞.

Lemma 6. *Let $\pi = w_0 w_1 ... w_n$ be a system prefix in G, $\eta \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ a finite symbolic word in Ω and $\sigma \in tw(\eta)$ a finite timed word. If it holds that $sMap(\pi) = \eta$ and $twMap(\pi) = \sigma$ and there exists a state $w_{n+1} \in S_e$ such that $\pi' = \pi \cdot w_{n+1}$ is in $\mathit{Prefs}_e(G)$ then it holds that $sMap(\pi') = \eta$ and $twMap(\pi') = \sigma$*

Proof. The proof follows directly from the definition of $sMap$ and $twMap$. □

Lemma 7. *Let $\pi = w_0 w_1 ... w_n$ be a environment prefix in the encoding G, $\eta = (a_1, g_1, Y_1)(a_2, g_2, Y_2)...(a_m, g_m, Y_m) \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ a finite symbolic word in Ω and $\sigma \in tw(\eta)$ a finite timed word. If $sMap(\pi) = \eta$, $twMap(\pi) = \sigma$ and $vMap(\pi) = v_0 ... v_m$ it holds that*

1. *If there exists a state $w_{n+1} \in S_e$ such that $w_{n+1} \models \mathit{playedAction} \approx a$ with $a \in \Sigma_E^u$ (the environment played an unobservable transition) and $\pi' = \pi \cdot w_{n+1}$ is in $\mathit{Prefs}_e(G)$ then there exist a symbolic action $(a_{m+1}, g_{m+1}, Y_{m+1}) \in \Sigma_E^u \times \mathcal{G}(X^c) \times 2^{X^c}$ such that $\eta' = \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})$ is in $L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ and a timed word $\sigma' = \sigma \cdot (a_{m+1}, t_{m+1}) \in tw(\eta')$ and it holds that $sMap(\pi') = \eta'$ and $twMap(\pi') = \sigma'$.*

2. *If there exists a state* $w_{n+1} \in S_e$ *such that* $w_{n+1} \models playedAction \approx a$ *with* $a \in \Sigma_C \cup \Sigma_E^o$ *(the environment played a controllable or observable transition) and* $\pi' = \pi \cdot w_{n+1}$ *is in* $Prefs_e(G)$ *then there exist a symbolic action* $(a_{m+1}, g_{m+1}, Y_{m+1}) \in (\Sigma_c \cup \Sigma_E^o) \times \mathcal{G}(X^c \cup X_{cont}^c) \times 2^{X^c \cup X_{cont}^c}$ *such that* $\eta' = \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})$ *is in* $L_{symb}^*(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ *and a timed word* $\sigma' = \sigma \cdot (a_{m+1}, t_{m+1}) \in tw(\eta')$ *and it holds that* $sMap(\pi') = \eta'$ *and* $twMap(\pi') = \sigma'$.

3. *If there exists a* $w_{n+1} \in S$ *such that* $w_{n+1} \models playedAction \approx \epsilon$ *(the environment played a do nothing transition or a change player transition) and* $\pi' = \pi \cdot w_{n+1} \in Prefs(G)$ *then it holds that* $sMap(\pi') = \eta$ *and* $twMap(\pi') = \sigma$.

4. *If there exist a symbolic action* $(a_{m+1}, g_{m+1}, Y_{m+1}) \in \Sigma_E^u \times \mathcal{G}(X^c) \times 2^{X^c}$ *(player 2 played an unobservable action) such that* $\eta' = \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})$ *is in* $L_{symb}^*(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ *and a timed word* $\sigma' = \sigma \cdot (a_{m+1}, t_{m+1}) \in tw(\eta')$ *then there exists a state* $w_{n+1} \in S_e$ *such that the prefix* $\pi' = \pi \cdot w_{n+1}$ *is in* $Prefs_e(G)$ *and it holds that* $sMap(\pi') = \eta'$ *and* $twMap(\pi') = \sigma'$.

5. *If the following three conditions hold*
   - *There exist a symbolic action* $(a_{m+1}, g_{m+1}, Y_{m+1}) \in (\Sigma_c \cup \Sigma_E^o) \times \mathcal{G}(X^c \cup X_{cont}^c) \times 2^{X^c \cup X_{cont}^c}$ *such that* $\eta' = \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})$ *is in* $L_{symb}^*(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ *(player 2 played a controllable or observable action)*
   - *There exists a timed word* $\sigma' = \sigma \cdot (a_{m+1}, t_{m+1}) \in tw(\eta)$
   - *It holds that* $w_n \models \neg obsActionPlayed \wedge offeredSymAction_{a_{m+1}, h_{m+1}, Z_{m+1}}$ *with* $(a_{m+1}, h_{m+1}, Z_{m+1}) = (a_{m+1}, g_{m+1}, Y_{m+1}) \upharpoonright 2$

   then there exists a state $w_{n+1} \in S_e$ *such that the prefix* $\pi' = \pi \cdot w_{n+1}$ *is in* $Prefs_e(G)$ *and it holds that* $sMap(\pi') = \eta'$ *and* $twMap(\pi') = \sigma'$.

Proof. The following conditions are equivalent

- The symbolic word $\eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})$ is in $L_{symb}^*(\mathcal{H} \parallel \mathcal{U}_\Gamma)$
- The symbolic word $(\eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1})) \upharpoonright 1$ is in $L_{symb}^*(\mathcal{H})$
- There exists a path

  $$P = q_0 \xrightarrow{a_1, g_1', Y_1'} q_1 \xrightarrow{a_2, g_2', Y_2'} q_2 \ldots q_m \xrightarrow{a_{m+1}, g_{m+1}', Y_{m+1}'} q_{m+1}$$

  with $(q_{i-1}, a_i, g_i', Y_i', q_i) \in \rightarrow$ and $(a_i, g_i', Y_i') = (a_i, g_i, Y_i) \upharpoonright 1$, $\forall 0 < i \leq m+1$.

1. As the transition (w_n, w_{n+1}) is in R_e and each transition in R_e corresponds to a transition $(q, a, g, Y, q') \in\rightarrow_{\mathcal{H}}$ and $\mathcal{H}$ is deterministic there exists a unique transition $(q_m, a_{m+1}, g'_{m+1}, Y'_{m+1}, q_{m+1}) \in\rightarrow_{\mathcal{H}}$ with $v_m + t_{m+1} - t_m \models g'_{m+1}$ where

$$\begin{aligned} w_n \models & location \approx q_m \wedge c_{synch} \approx t_m \\ w_{n+1} \models & location \approx q_{m+1} \wedge playedAction \approx a_{m+1} \wedge c_{synch} \approx t_{m+1} \end{aligned}$$

The condition $v_m + t_{m+1} - t_m \models g'_{m+1}$ holds as $vMap(\pi) = v_m$ and the guard g_{m+1} is encoded in the transition (w_n, w_{n+1}) with $g'_{m+1[X \rightarrow C + c'_{synch} - c_{synch}]}$. Therefore $sMap(\pi')$ exists and we apply $sMap$ and $twMap$ on π'

$$\begin{aligned} sMap(\pi') &= sMap(\pi) \cdot (a_{m+1}, g_{m+1}, Y_{m+1}) \\ &= \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1}) \end{aligned}$$

$$\begin{aligned} twMap(\pi') &= twMap(\pi) \cdot (a_{m+1}, t_{m+1}) \\ &= \sigma \cdot (a_{m+1}, t_{m+1}) \end{aligned}$$

As a_{m+1} is in Σ_E^u there is no transition $(s, a_{m+1}, h_{m+1}, Z_{m+1}, s)$ in $\rightarrow_{\mathcal{U}_\Gamma}$, where s is the single state of $\mathcal{U}_\Gamma$. Thus $(a_{m+1}, g'_{m+1}, Y'_{m+1}) = (a_{m+1}, g_{m+1}, Y_{m+1})$. As $(q_m, a_{m+1}, g_{m+1}, Y_{m+1}, q_{m+1})$ is in $\rightarrow_{\mathcal{H}}$ there exists a path

$$P' = q_0 \xrightarrow{a_1, g'_1, Y'_1} q_1 \xrightarrow{a_2, g'_2, Y'_2} q_2 ... q_m \xrightarrow{a_{m+1}, g_{m+1}, Y_{m+1}} q_{m+1}$$

such that $(q_{i-1}, a_i, g'_i, Y'_i, q_i) \in\rightarrow_{\mathcal{H} \| \mathcal{U}_\Gamma}$ and $(a_i, g'_i, Y'_i) = (a_i, g_i, Y_i) \upharpoonright 1$, $\forall 0 < i < m$. Thus η' is in $L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ and $sMap(\pi') = \eta'$. The sub formula $g'_{m+1[X \rightarrow C + c'_{synch} - c_{synch}]}$ ensures that $v_m + t_{m+1} - t_m \models g_{m+1}$ and thus it holds that σ' in $tw(\eta')$ and $twMap(\pi') = \sigma'$

2. As the transition (w_n, w_{n+1}) is in R_e and each transition in R_e corresponds to a transition $(q, a, g, Y, q') \in\rightarrow_{\mathcal{H}}$ and $\mathcal{H}$ is deterministic there exists a unique transition $(q_m, a_{m+1}, g'_{m+1}, Y'_{m+1}, q_{m+1}) \in\rightarrow_{\mathcal{H}}$ with $v_m + t_{m+1} - t_m \models g'_{m+1}$ where

$$\begin{aligned} w_n \models & location \approx q_m \wedge c_{synch} \approx t_m \\ w_{n+1} \models & location \approx q_{m+1} \wedge playedAction \approx a_{m+1} \wedge c_{synch} \approx t_{m+1} \end{aligned}$$

The condition $v_m + t_{m+1} - t_m \models g'_{m+1}$ holds as $vMap(\pi) = v_m$ and the guard g_{m+1} is encoded in the transition (w_n, w_{n+1}) with $g'_{m+1[X \rightarrow C + c'_{synch} - c_{synch}]}$.

Therefore $sMap(\pi')$ exists and we apply $sMap$ and $twMap$ on π'

$$\begin{aligned} sMap(\pi') &= sMap(\pi) \cdot (a_{m+1}, g_{m+1}, Y_{m+1}) \\ &= \eta \cdot (a_{m+1}, g_{m+1}, Y_{m+1}) \end{aligned}$$

$$\begin{aligned} twMap(\pi') &= twMap(\pi) \cdot (a_{m+1}, t_{m+1}) \\ &= \sigma \cdot (a_{m+1}, t_{m+1}) \end{aligned}$$

We define the symbolic action $(a_{m+1}, h_{m+1}, Z_{m+1})$ s.t. $w_{n+1} \models pickedAtom_{h_{m+1}}$ and for all $x \in X^c_{cont} : x \in Z_{m+1}$ iff $w_{n+1} \models pickedReset_x$. As a_{m+1} is in $\Sigma^o_E \cup \Sigma_C$ and U_Γ accepts all words in Γ^∞ there exists a transition $(s, a_{m+1}, h_{m+1}, Z_{m+1}, s)$ in $\rightarrow_{U_\Gamma}$, where s is the single state of $\mathcal{U}_\Gamma$. From the definition of $sMap$ it follows that $(a_{m+1}, g_{m+1}, Y_{m+1}) = (a_{m+1}, g'_{m+1} \wedge h_{m+1}, Y'_{m+1} \cup Z_{m+1})$. As $(q_m, a_{m+1}, g'_{m+1}, Y'_{m+1}, q_{m+1})$ is in $\rightarrow_\mathcal{H}$ there exists a path

$$P' = q_0 \xrightarrow{a_1, g'_1, Y'_1} q_1 \xrightarrow{a_2, g'_2, Y'_2} q_2 ... q_m \xrightarrow{a_{m+1}, g'_{m+1}, Y'_{m+1}} q_{m+1}$$

such that $(q_{i-1}, a_i, g'_i, Y'_i, q_i) \in \rightarrow_{\mathcal{H} \| \mathcal{U}_\Gamma}$ and $(a_i, g'_i, Y'_i) = (a_i, g_i, Y_i) \upharpoonright 1$, $\forall 0 < i \leq m$. Thus η' is in $L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ and $sMap(\pi') = \eta'$. The sub formula $\mathbf{updatePickAndCheckAtom(a)}$ ensures that $v_m + t_{m+1} - t_m \models g_{m+1}$ and thus it holds that σ' in $tw(\eta')$ and $twMap(\pi') = \sigma'$

3. The proof follows directly from the definition of $sMap$ and $twMap$.

4. As $\eta' \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ and $a \in \Sigma^u_E$ there exists a path

$$P = q_0 \xrightarrow{a_1, g'_1, Y'_1} q_1 \xrightarrow{a_2, g'_2, Y'_2} q_2 ... \xrightarrow{a_m, g'_m, Y'_m} q_m \xrightarrow{a_{m+1}, g_{m+1}, Y_{m+1}} q_{m+1}$$

such that $(q_{i-1}, a_i, g'_i, Y'_i, q_i) \in \rightarrow_{\mathcal{H} \| \mathcal{U}_\Gamma}$ and $(a_i, g'_i, Y'_i) = (a_i, g_i, Y_i) \upharpoonright 1$, $\forall 0 < i < m+1$ and $(a_i, g_i, Y_i) \upharpoonright 2 = \epsilon$. There is an environment transition for each $(q, a, g, Y, q') \in \rightarrow_\mathcal{H}$ with $a \in \Sigma^u_E$ where q and q' is encoded by the variable *location*, the action a is encoded by the variable *pickedAction*, the guard g is encoded by the sub formula $g_{[X \rightarrow C + c'_{synch} - c_{synch}]}$ and the set Y is encoded by the sub formula $\mathbf{nonControllableClocksUpdated(Y)}$. Therefore and as $\sigma' \in tw(\eta')$ which means that $v_m + t_{m+1} - t_m \models g$ there exists a unique transition $(w_n, w_{n+1}) \in R_e$ such that

$$\begin{aligned} w_n &\models location \approx q_m \wedge c_{synch} \approx t_m \\ w_{n+1} &\models location \approx q_{m+1} \wedge playedAction \approx a_{m+1} \wedge c_{synch} \approx t_{m+1} \end{aligned}$$

Thus π' is in $Prefs(G)$ and from the definition of $sMap$ and $twMap$ follows that $sMap(\pi \cdot w') = \eta'$ and $twMap(\pi') = \sigma'$.

5. As $\eta' \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ and $a \in \Sigma^o_E \cup \Sigma_C$ there exists a path

$$P = q_0 \xrightarrow{a_1, g'_1, Y'_1} q_1 \xrightarrow{a_2, g'_2, Y'_2} q_2 \ldots \xrightarrow{a_m, g'_m, Y'_m} q_m \xrightarrow{a_{m+1}, g'_{m+1}, Y'_{m+1}} q_{m+1}$$

such that $(q_{i-1}, a_i, g'_i, Y'_i, q_i) \in \rightarrow_{\mathcal{H} \parallel \mathcal{U}_\Gamma}$ and $(a_i, g'_i, Y'_i) = (a_i, g_i, Y_i) \restriction 1$, $\forall 0 < i \leq m+1$ and $(a_i, g_i, Y_i) \restriction 2 \neq \epsilon$. Thus there exists a unique transition $(q_m, a_{m+1}, g'_{m+1}, Y'_{m+1}, q_{m+1})$ in $\mathcal{H}$ and a transition $(s, a_{m+1}, h_{m+1}, Z_{m+1}, s)$ in $\mathcal{U}_\Gamma$, where s is the single state in $\mathcal{U}_\Gamma$, s.t. $(a_{m+1}, g_{m+1}, Y_{m+1}) = (a_{m+1}, g'_{m+1} \wedge h_{m+1}, Y'_{m+1} \cup Z_{m+1})$. We defined a formula for each $(q, a, g, Y, q') \in \rightarrow_{\mathcal{H}}$ with $a \in \Sigma^o_E \cup \Sigma_C$ where q and q' is encoded by the variable *location*, the action a is encoded by the variable *pickedAction*, the guard g is encoded by the sub formula $g_{[X \rightarrow C + c'_{synch} - c_{synch}]}$ and the set Y is encoded by the sub formula $\mathbf{nonControllableClocksUpdated(Y)}$. Further the guard h_{m+1} and the set Z_{m+1} are encoded in the sub formula $\mathbf{updatePickAndCheckAtom(a)}$. Therefore and as $\sigma' \in tw(\eta')$ which means that $v_m + t_{m+1} - t_m \models g \wedge h$ there exists a unique transition $(w_n, w_{n+1}) \in R_e$ such that

$$\begin{aligned} w_n &\models location \approx q_m \wedge c_{synch} \approx t_m \\ w_{n+1} &\models location \approx q_{m+1} \wedge playedAction \approx a_{m+1} \wedge c_{synch} \approx t_{m+1} \\ &\quad \wedge pickedAction \approx a_{m+1} \wedge pickedAtom_{h_{m+1}} \wedge pickedReset_{Z_{m+1}} \end{aligned}$$

Thus π' is in $Prefs(G)$ and from the definition of $sMap$ and $twMap$ follows that $sMap(\pi \cdot w') = \eta'$ and $twMap(\pi') = \sigma'$.

$\square$

Definition 42. *A symbolic word γ with unobservable actions is mapped by hide: $(\Sigma \times \mathcal{G}(X^c \cup X^c_{cont}) \times 2^{X^c \cup X^c_{cont}})^\infty \rightarrow \Gamma^\infty$ to a symbolic word which contains only the observable part of the symbolic actions. The function hide is defined by $hide(\gamma) = \gamma \restriction 2$.*

Lemma 8. *Let π_1, π_2 be two prefixes in G. If $\pi_1 \equiv \pi_2$ then $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$.*

Proof. As $sMap$ maps all states which are results of a do nothing transition, a change player transition and a system transition to ϵ we do not have to consider such transitions. Further $hide$ maps all $(a, g, Y) \in \Sigma_E^u \times \mathcal{G}(X^c) \times 2^{X^c}$ to ϵ. So we have only to consider transitions $(w_1, w_2) \in R_e$ which model observable and controllable actions where $w_1 \models \mathit{pickedAction} = \epsilon$ and $w_2 \models \mathit{playedAction} \approx a \wedge \mathit{pickedAction} \approx a$. Let $I = \{n_0, n_1, ..., n_k\}$ be the index set such that $\pi_1[n_j] \models \mathit{pickedAction} = \epsilon$ and $\pi_1[n_j + 1] \models \mathit{playedAction} \approx a \wedge \mathit{pickedAction} \approx a$ where $a \in \Sigma_E^o \cup \Sigma_C$ for $0 \leq j \leq k$. For a $0 \leq j \leq k$ let $(a, g_1, Y_1) = last(sMap(\pi_1(j)))$ and $(a, g_2, Y_2) = last(sMap(\pi_2(j)))$ such that $(a, g_1, Y_1) \upharpoonright 2 = (a, h_1, Z_1)$ and $(a, g_2, Y_2) \upharpoonright 2 = (a, h_2, Z_2)$. As $\mathit{PickedAtoms} \subseteq X_i$, $\mathit{PickedReset} \subseteq X_i$ and $\pi_1 \equiv \pi_2$ it follows that $h_1 = h_2$ and $Z_1 = Z_2$. Thus $(a, g_1, Y_1) \upharpoonright 2 = (a, g_2, Y_2) \upharpoonright 2$. As $last(sMap(\pi_1(j))) \upharpoonright 2 = last(sMap(\pi_2(j))) \upharpoonright 2$ holds for all $0 \leq j \leq k$ it follows that $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$. □

Lemma 9. *If there exists a consistent winning strategy f_s for the system in $\mathcal{G}$ then there exists a consistent winning strategy f'_s such that for all prefixes $\pi_1, \pi_2 \in Prefs(G)$ it holds that $f'_s(\pi_1) \equiv f'_s(\pi_2)$ if $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$.*

Proof. For a prefix π in $Prefs_s(G)$ let $\pi_0 \in \{\pi' \in Prefs_s(G) \mid hide(sMap(\pi)) = hide(sMap(\pi'))\}$ and $f_s(\pi_0) = w_0$. We define a strategy f'_s for the prefix by $f'_s(\pi) = w$ where $w \models \mathit{offeredSymAction}_{a,h,Z}$ iff $w_0 \models \mathit{offeredSymAction}_{a,h,Z}$, $w \models t \approx 0$, and the values of the other variables are equal to $last(\pi)$. If it holds that $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$ then $f'_s(\pi_1) \models \mathit{offeredSymAction}_{a,h,Z}$ iff $f'_s(\pi_2) \models \mathit{offeredSymAction}_{a,h,Z}$. As $last(\pi_1) \equiv last(\pi_2)$ it follows that $f'_s(\pi_1) \equiv f'_s(\pi_2)$. For two prefixes π_1, π_2 in $Prefs_s(G)$ with $\pi_1 \equiv \pi_2$ it follows from lemma 8 that $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$. Therefore the strategy f'_s is consistent and as f_s is winning f'_s a consistent winning strategy. □

Theorem 2. *There exists a consistent winning strategy for the system in the encoding $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ of a timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ if and only if player C has a valid winning strategy in Ω.*

Proof. First we show that the system has a consistent winning strategy if player C has valid winning strategy. Let f be a valid winning strategy for player C in Ω and π a system prefix in the encoding G. Let $\eta \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ be a finite symbolic word in Ω such that $sMap(\pi) = \eta$ and $\gamma = hide(\eta)$ is a finite play in Ω. The set $C = f(\gamma) \subseteq \Gamma$ consists of the offered symbolic actions. Now we define $f_s(\pi) = w_e$

such that $w_e \models \mathit{offeredSymAction}_{a,g,Z}$ iff $(a, g, Z) \in C$ and the values of all other variables have the same value as in the state $last(\pi)$. From Lemma 8 follows that for all prefixes $\pi_1, \pi_2 \in \mathit{Prefs}(G)$ with $\pi_1 \equiv \pi_2$ it holds that $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$. It follows from $f(hide(sMap(\pi_1))) \equiv f(hide(sMap(\pi_2)))$ that $f_s(\pi_1) \equiv f_s(\pi_2)$ and therefore f_s is consistent. From Lemma 6 and 7 follows that for every $\pi \in \mathit{Outcome}(f_s, f_e)$ for some f_e there exists a symbolic word $\eta \in L^*_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ where $hide(\eta) \in plays_f(\Omega)$ such that $sMap(\pi) = \eta$ and a timed word $\sigma \in tw(\eta)$ such that $twMap(\pi) = \sigma$. As f is a valid winning strategy it holds that for each timed word in $tw(synw^\omega_{\mathcal{H}}(plays_f(\Omega)))$ $\mathcal{H}$ will not visit an error state described by $\mathcal{I}$, f is not blocking and f is not restricting. Thus $\omega \in \mathit{Outcome}(f_s, f_e)$ for some f_e is infinite and for every $j \geq 0$ it holds that $\omega[j] \not\models err$. As f_s is defined for each prefix it is a non-blocking strategy. Therefore f_s is a consistent winning strategy.

We show now that player C has a valid winning strategy if the system has a consistent winning strategy. Let f_s be a consistent winning strategy in G. From lemma 9 follows that there exists a strategy f'_s such that for all prefixes $\pi_1, \pi_2 \in \mathit{Prefs}(G)$ if $hide(sMap(\pi_1)) = hide(sMap(\pi_2))$ it holds that $f'_s(\pi_1) \equiv f'_s(\pi_2)$. Let f'_s be such a strategy. Let γ be a finite play in Ω. If $synw^*_{\mathcal{H}}(\gamma) = \emptyset$ or for all $\eta \in synw^*_{\mathcal{H}}(\gamma)$ it holds that $tw(\eta) = \emptyset$ then there is no corresponding path in $\mathcal{H}$ for γ and we define $f(\gamma) = \emptyset$. Else we choose a $\eta \in synw^*_{\mathcal{H}}(\gamma)$ and a $\sigma \in tw(\eta)$. From Lemma 7 follows that for every $\eta \in synw^\omega_{\mathcal{H}}(plays_f(\Omega))$ and every timed word $\sigma \in tw(\eta)$ there exists a $\pi \in \mathit{Outcome}(f'_s, f_e)$ for some f_e with $sMap(\pi) = \eta$ and $twMap(\pi) = \sigma$. Let π the prefix such that $sMap(\pi) = \eta$ and $twMap(\pi) = \sigma$. We define $f(\gamma) = C$ where C is a subset of Γ such that $(a, h, Z) \in C$ iff $f'_s(\pi) \models \mathit{offeredSymAction}_{a,h,Z}$. The strategy f is valid cause as no $\pi \in \mathit{Outcome}(f'_s, f_e)$ for some f_e visits an error state cause of blocking or restricting reasons. Therefore f is a valid winning strategy. Thus the theorem is proved. $\square$

5 An Alternative Proof for the Decidability of the Encoded Models

As already mentioned in section 2.2 a timed game is only decidable if the resources of the controller are fixed (assuming we have a deterministic specification or an external non deterministic specification with undesired behaviors). In this section we provide an alternative proof for the decidability.
In [14] was shown that the abstraction-refinement based approach presented there is a decision procedure for the controller syntheses problem, for a class of infinite-state safety games for which a region algebra that satisfies certain conditions exists. We show that the encodings we have presented result into symbolic games which satisfy this criteria, i.e. in games for which the abstraction-refinement loop from [14] is guaranteed to terminate. Together with the correctness of the two encodings this yields that the procedure from [14] is a decision procedure for the controller syntheses problem for timed games with incomplete information and safety objective. For the symbolic analysis predicate transformers are needed.

Definition 43. *For a formula φ and $\hat{c}_o \in \hat{C}_o$, $Pre_s(\hat{c}_o, \varphi)$ is a formula such that $v \models Pre_s(\hat{c}_o, \varphi)$ iff there exists a state $w \models \varphi \wedge \hat{x}_o \approx \hat{c}_o$ such that $(v, w) \in R_s$ and for a formula φ, $Pre_e(\varphi)$ is a formula such that $v \models Pre_e(\varphi)$ iff there exists a state $w \models \varphi$ such that $(v, w) \in R_e$.*

The region algebra which has to exists for a symbolic safety game to ensure that the refinement loop terminates is defined as follow.

Definition 44 (Region algebra)**.** *A region algebra for a symbolic safety game $\mathcal{G} = (\mathcal{C}, err)$ is a pair (R, Obs), where $R \subseteq 2^S$ is a set of regions such that $\bigcup R = S$ and $Obs \subseteq R$ is a subset of observable regions, with the following properties:*

(1) The set R is closed under boolean operations: for every r_1, r_2 in R, the sets $r_1 \cup r_2$, $r_1 \cap r_2$ and $S \setminus r_1$ are in R;

(2) The sets $\{v \in S \mid v \models t \approx 0\}$ and $\{v \in S \mid v \models t \approx 1\}$ are in Obs;

(3) For every $\hat{c}_o \in \hat{C}_o$ the set $\{v \in S \mid v \models \hat{c}_o \approx \hat{x}_o\}$ is in Obs;

(4) Every region is expressible over the set of predicates $\mathcal{AP}$, for every predicate p with $p \in Preds(init)$ or $p \in Preds(err)$ it holds that the set $\{v \in S \mid v \models p\}$ is in R, and the transformers Pre_s and Pre_e are defined in such a way that for every $r \in R$, for every $\hat{c}_0 \in \hat{C}_o$, and predicate p with $p \in Preds(Pre_e(r))$ or $p \in Preds(Pre_s(\hat{c}_0, r))$ it holds that for every $r' \in R$, the sets $r' \cap \{v \in S \mid v \models p\}$ and $r' \cap \{v \in S \mid v \models \neg p\}$ are in R.

(5) *For every $\pi_1, \pi_2 \in Prefs(G)$, if each of $last(\pi_1)$ and $last(\pi_2)$ is a dead-end or an error state and there exists an index j such that $\pi_1[j] \not\equiv \pi_2[j]$, then there exists an index $0 \leq k \leq j$ and an observable region $r \in Obs$ such that $\pi_1[k] \in r$ and $\pi_2[k] \notin r$.*

In the next two sections we define region algebras for the two encodings and prove that they satisfy the conditions from above.

5.1 Region Algebra for the Encoding of the Model Based on Observation Predicates

Let M be a natural number and $H = (L, \iota, X^c, \delta, \Sigma_1, \Sigma_2, inv, \mathcal{P})$ be a M-TGS with the safety winning condition $\mathcal{O}$. Let $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ be the symbolic safety game that encodes the M-TGS H. First we define the following sets of predicates.

$$\begin{aligned} P_o =& \{t \approx 0, t \approx 1\} \\ & \cup \{act \approx \sigma \mid \sigma \in \Sigma_1^{ext}\} \\ & \cup \{reset_x \approx 0 \mid x \in X_1^c\} \cup \{reset_x \approx 1 \mid x \in X_1^c\} \end{aligned}$$

$$P_i = \{obs_p \approx 0 \mid p \in \mathcal{P}\} \cup \{obs_p \approx 1 \mid p \in \mathcal{P}\}$$

$$\begin{aligned} P_h =& \{location \approx l \mid l \in L\} \\ & \cup \{resetDone \approx 0, resetDone \approx 1\} \\ & \cup \{ttAllowed \approx 0, ttAllowed \approx 1\} \\ & \cup \{c_x \sim k \mid x \in X^c, k \in \mathbb{Z} \cap [0, M]\} \\ & \cup \{c_x - c_y \sim k \mid x, y \in X^c, k \in \mathbb{Z} \cap [0, M]\} \\ & \cup \{c_{synch} \sim k \mid k \in \mathbb{Z} \cap [0, M]\} \\ & \cup \{c_x - c_{synch} \sim k \mid x \in X^c, k \in \mathbb{Z} \cap [0, M]\} \end{aligned}$$

With these sets we define the pair $(R, Obs)_G$ for the symbolic safety game $\mathcal{G}$ as follows

- R is the set of sets $\{v \in S \mid v \models \varphi\}$ where φ is generated by the following grammar:

  $$\varphi = p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

  where $p \in P_o \cup P_i \cup P_h$

- *Obs* is the set of sets $\{v \in S \mid v \models \varphi\}$ where φ is generated by the following grammar:

$$\varphi = p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

where $p \in P_o \cup P_i$

Lemma 10. *The pair $(R, Obs)_G$ is a finite region algebra for the encoding $\mathcal{G}$ of the M-TGS H.*

Proof. First we prove that the sets R and Obs are finite. It is sufficient to show that P_o, P_i and P_h are finite as the set of non-equivalent formulas generated by the grammar $\varphi = p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$ over a finite set of predicates is finite. From the definition of a M-TGS [11] follows that the sets Σ^{ext}, L, X^c and $\mathcal{P}$ are finite and as M is fixed it follows directly that P_o, P_i and P_h are finite[6].
Now we prove that $(R, Obs)_G$ is a region algebra for the symbolic safety game $\mathcal{G}$.

(1) Let $r_1, r_2 \in R$ where $r_1 = \{v \in S \mid v \models \varphi_{r_1}\}$ and $r_2 = \{v \in S \mid v \models \varphi_{r_2}\}$

(i)

$$\begin{aligned} r_1 \cup r_2 &= \{v \in S \mid v \models \varphi_{r_1} \text{ or } v \models \varphi_{r_2}\} \\ &= \{v \in S \mid v \models \varphi_{r_1} \vee \varphi_{r_2}\} \in R \end{aligned}$$

(ii)

$$\begin{aligned} r_1 \cap r_2 &= \{v \in S \mid v \models \varphi_{r_1} \text{ and } v \models \varphi_{r_2}\} \\ &= \{v \in S \mid v \models \varphi_{r_1} \wedge \varphi_{r_2}\} \in R \end{aligned}$$

(iii)

$$\begin{aligned} S \setminus r_1 &= \{v \in S \mid v \not\models \varphi_{r_1}\} \\ &= \{v \in S \mid v \models \neg\varphi_{r_1}\} \end{aligned}$$

The formula $\neg\varphi_{r_1}$ can be transformed to an equivalent formula where the negation only occurs in the level of the predicates. For each predicate $x \approx c$ where x is discrete variable and c a constant in the domain $dom(x)$ of x

[6] When [11] define a non-deterministic game for H they define a finite alphabet which models the choice of player 1.

holds

$$\neg(x \approx c) \equiv \bigvee_{c' \in dom(x), c' \neq c} x \approx c'$$

and for each predicate $c_1 \sim k$ or $c_2 - c_1 \sim k$ where $c_1 \in \{c_x \mid x \in X^c\} \cup \{c_{synch}\}$, $c_2 \in \{c_x \mid x \in X^c\}$, $\sim \in \{<, \leq, \approx, \geq, >\}$ and $k \in \mathbb{Z} \cap [0, M]$ holds

$$\begin{aligned}
\neg(c_1 < / \leq / \geq / > k) &\equiv c_1 \geq / > / < / \leq k \\
\neg(c_1 \approx k) &\equiv c_1 < k \vee c_1 > k \\
\neg(c_2 - c_1 < / \leq / \geq / > k) &\equiv c_2 - c_1 \geq / > / < / \leq k \\
\neg(c_2 - c_1 \approx k) &\equiv c_2 - c_1 < k \vee c_2 - c_1 > k
\end{aligned}$$

Thus it holds that $S \setminus r_1 \in R$.

(2) By definition it holds that the sets $\{v \in S \mid v \models t \approx 0\}$ and $\{v \in S \mid v \models t \approx 1\}$ are in Obs;

(3) As for each $x_o \in X_o$ the set P_o contains a predicate $x_o \approx c$ for all $c \in dom(x_o)$ it holds that for every $\hat{c}_o$ the set $\{v \in S \mid v \models \hat{x}_o \approx \hat{c}_o\}$ is in Obs.

(4) (i) The formula *init* is defined as follow:

$$\begin{aligned}
& t \approx 1 \wedge location \approx \iota \wedge \textbf{initialObservation} \\
& \wedge \bigwedge_{x \in X^c} c_x \approx 0 \wedge c_{synch} \approx 0 \wedge \bigwedge_{x \in X_1^c} \neg reset_x \\
& \wedge \neg resetDone \wedge act \approx \Delta \wedge ttAllowed
\end{aligned}$$

It holds that for all $p \in Preds(init)$ the predicate p is in $P_o \cup P_i \cup P_h$. Thus by definition the set $\{v \in S \mid v \models p\}$ is in R;

(ii) The formula *err* contains only observation predicates. As $P_i = \{obs_p \approx 0 \mid p \in \mathcal{P}\} \cup \{obs_p \approx 1 \mid p \in \mathcal{P}\}$ it holds that for all $p \in Preds(err)$ the set $\{v \in S \mid v \models p\}$ is in R;

(iii) Let $r \in R$, φ_r the formula which represents r and $p \in Preds(Pre_e(\varphi_r))$. If $\{v \in S \mid v \models p\}$ is in R then it holds with (1) that for all $r' \in R$ the sets $r' \cap \{v \in S \mid v \models p\}$ and $r' \cap \{v \in S \mid v \models \neg p\}$ are in R. Let us consider the following formula which is equivalent to $Pre_e(\varphi_r)$

$$\exists \hat{x}'.\mathcal{T}_e \wedge \varphi_r'$$

where φ_r' is the formula obtained from φ where all variables are primed. Constructing the disjunctive normal form of this formula results in the equivalent

$$\exists \hat{x}'. \bigvee_{0 \leq i \leq n} \varphi_i \equiv \bigvee_{0 \leq i \leq n} \exists \hat{x}'.\varphi_i$$

We now transform each existentially quantified conjunction φ_i to a equivalent quantifier free formula. In a first step we eliminate all primed discrete variables from φ_i. Each predicate which contains a primed discrete variable x' is of the form $x' \approx x$ or $x' \approx k$ where x is the unprimed version of the variable and $k \in dom(x)$. For each discrete variable x we remove one predicate of the form $x' \approx x$ or $x' \approx k$ and substitute each occurrence of the variable X' with x or k respectively. Thus we get for each conjunction φ_i a conjunction ψ_i which contains no primed discrete variables. In a second step we eliminate all primed clock variables. Each conjunction ψ_i' can be transformed to a system of linear inequalities. Applying the Fourier-Motzkin-Elimination to eliminate all primed clock variables we get an equivalent system of inequalities without the primed clock variables. Let χ_i denote the conjunction which represents the resulting system for ψ_i. Thus we define

$$Pre_e(\varphi_r) := \bigvee_{0 \leq i \leq n} \chi_i$$

As all predicates with no primed variables in $Preds(\varphi_r \wedge \mathcal{T}_e)$ are in $P_o \cup P_i \cup P_h$ we have only to check the predicates which results of the substitution of the discrete variables and of the Fourier-Motzkin-Elimination. For the substitution of the discrete variables it is easy to see that these predicates are in $P_o \cup P_i \cup P_h$ or will be evaluated to $true$. All predicates in $Preds(\varphi_r \wedge \mathcal{T}_e)$ which contain primed clock variables are of the form $c_x' \approx c_x$, $c_x' \sim k$, $c_x' - c_y' \sim k$, $c_{synch}' \sim k$, $c_{synch}' \approx c_{synch}$ or $c_x' \approx c_x + c_{synch}' - c_{synch}$ where $x, y \in X^c$, $\sim \in \{<, \leq \approx, \geq, >\}$ and $0 \leq k \leq M$. Thus the predicates from the resulting Fourier-Motzkin-Elimination are of the form (or can be transformed to) $c_x \sim k$, $c_x - c_y \sim k$, $c_x - c_{synch} \sim k$ or $c_{synch} \sim k$. As theses predicates are all in P_h it follows that for all $p \in Preds(Pre_e(\varphi_r))$ it holds that $\{v \in S \mid v \models p\} \in R$;

(iv) Analogously to (iii) with the formula

$$\hat{x}'_o \approx \hat{c}_o \wedge \varphi_r \wedge \mathcal{T}_s$$

(5) Let $\pi_1, \pi_2 \in Prefs(G)$ such that $last(\pi_1)$ and $last(\pi_2)$ are dead-ends or error states. Further let $\pi_1 \not\equiv \pi_2$ and j be the smallest index such that $\pi_1[j] \not\equiv \pi_2[j]$. As $\pi_1[j] \not\equiv \pi_2[j]$ there must be at least one variable in $Obs(X)$[7] such that the value of this variable is different in $\pi_1[j]$ and $\pi_2[j]$. Let φ be the formula such that each variable in $obs(X)$ is equal to the valuation in $\pi_1[j]$. By definition is $\{v \in S \mid v \models \varphi\}$ in Obs and it holds that $\pi_1[j] \in \{v \in S \mid v \models \varphi\}$ As there is at least one variable in $obs(X)$ which differs in the value it follows that $\pi_2[j] \notin \{v \in S \mid v \models \varphi\}$.

□

5.2 Region Algebra for the Encoding of the Model Based on Synchronized Words

Let $\Omega = (\Gamma, \mathcal{H}, \mathcal{I})$ be a timed game with an internal winning condition based on $(\hat{\Sigma}, \hat{X}^c, \mu)$ with $\hat{\Sigma} = (\Sigma_C, \Sigma_E^o, \Sigma_E^u)$, $\hat{X}^c = (X_r^c, X_u^c)$. $\mu = (X_r^c \cup X_{cont}^c, n_\mu, max_\mu)$ Let $\mathcal{G} = ((X, init, \mathcal{T}_s, \mathcal{T}_e), err)$ be the encoding of Ω. Let $\nu = (X^c, n_\nu, max_\nu)$ be the granularity of the transition system $\mathcal{H}$ and $\mu + \nu = (X^c \cup X_{cont}^c, n, max)$ the combined granularity[8]. For a clock $x \in X^c \cup X_{cont}^c$ let $Con(x)$ be the set of constants which are fit to the combined granularity $\mu + \nu$ (The set of constants $k \in \mathbb{Q}$ such that n is a common multiplier of the denominator of k and $0 \leq k \leq max(x)$). The

[7] See section 3 for definition.
[8] See section 2.3.1 for definition.

sets of predicates are defined by

$$
\begin{aligned}
P_o =& \{t \approx 0, t \approx 1\} \\
& \cup \{\mathit{offeredSymbolicActions}_{a,g,Z} \approx 0 \mid a \in \Sigma_C \cup \Sigma_E^o, g \in atoms_\mu, Z \subseteq X_{cont}^c\} \\
& \cup \{\mathit{offeredSymbolicActions}_{a,g,Z} \approx 1 \mid a \in \Sigma_C \cup \Sigma_E^o, g \in atoms_\mu, Z \subseteq X_{cont}^c\}
\end{aligned}
$$

$$
\begin{aligned}
P_i =& \{\mathit{pickedAction} \approx a \mid a \in \Sigma_C \cup \Sigma_E^o \cup \{\epsilon\}\} \\
& \cup \{\mathit{pickedAtom}_g \approx 0 \mid g \in atoms_\mu\} \\
& \cup \{\mathit{pickedAtom}_g \approx 1 \mid g \in atoms_\mu\} \\
& \cup \{\mathit{pickedReset}_Z \approx 0 \mid Z \subseteq X_{cont}^c\} \\
& \cup \{\mathit{pickedReset}_Z \approx 1 \mid Z \subseteq X_{cont}^c\}
\end{aligned}
$$

$$
\begin{aligned}
P_h =& \{c_x - c_{synch} \sim k \mid x \in X^c \cup X_{cont}^c, k \in Con(x), \sim \in \{<, \leq, \approx, \geq, >\}\} \\
& \{c_x - c_y \sim k \mid x, y \in X^c \cup X_{cont}^c, k \in Con(x), \sim \in \{<, \leq, \approx, \geq, >\}\} \\
& \cup \{c_x \sim k \mid x \in X^c \cup X_{cont}^c, k \in Con(x), \sim \in \{<, \leq, \approx, \geq, >\}\} \\
& \cup \{c_{synch} \sim k \mid k \in Con(x), \sim \in \{<, \leq, \approx, \geq, >\}\} \\
& \cup \{\mathit{location} \approx l \mid l \in Q\} \\
& \cup \{\mathit{playedAction} \approx a \mid a \in \Sigma \cup \{\epsilon\}\} \\
& \cup \{\mathit{obsActionPlayed} \approx 0, \mathit{obsActionPlayed} \approx 1\}
\end{aligned}
$$

With these sets we define like for the first model the pair (R, Obs) for the encoding $\mathcal{G}$ by

- R is the set of sets $\{v \in S \mid v \models \varphi\}$ where φ is generated by the following grammar:

  $$\varphi = p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

  where $p \in P_o \cup P_i \cup P_h$

- Obs is the set of sets $\{v \in S \mid v \models \varphi\}$ where φ is generated by the following grammar:

  $$\varphi = p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

  where $p \in P_o \cup P_i$

Lemma 11. *The Pair (R, Obs) is a finite region algebra for the encoding $\mathcal{G}$ of the timed game Ω.*

Proof. The proof is in the most parts identical with the proof for lemma 10. So we show only that the sets R and Obs are finite and the 4th condition for a region algebra for symbolic games is fulfilled for the pair $(R, Obs)_{\mathcal{G}}$.
Again the sets R and Obs are finite if P_o, P_i and P_h are finite. The set Q is finite as $\mathcal{H}$ is a finite transition system and the sets Σ and X^c are finite by definition. As the granularity μ is fixed the set X^c_{cont} is finite and both n and for all $x \in X^c \cup X^c_{cont}$ $max(x)$ are fixed. Thus P_h is finite. The set $Con(x)$ for a clock $x \in X^c \cup X^c_{cont}$ is by definition also finite and therefore P_o and P_i are finite.

(4) (i) The formula *init* is defined as follows

$$
\begin{aligned}
& t \approx 1 \wedge location \approx q_0 \wedge \neg obsActionPlayed \wedge playedAction \approx \epsilon \\
& \wedge \bigwedge_{x \in X^c \cup X^c_{cont}} c_x \approx 0 \wedge \bigwedge_{a \in \Sigma_C \cup \Sigma^o_E} \bigwedge_{g \in atoms_\mu} \bigwedge_{Z \in 2^{X^c_{cont}}} \neg offeredSymbolicAction_{a,g,Z} \\
& \wedge pickedAction \approx \epsilon \wedge \bigwedge_{g \in atoms_\mu} \neg pickedAtom_g \wedge \bigwedge_{x \in X_{cont}} \neg pickedReset_x
\end{aligned}
$$

As one can see easily it holds that all $p \in Preds(init)$ are in $P_o \cup P_i \cup P_h$ and thus $\{v \in S \mid v \models p\} \in R$.

(ii) The formula *err* is defined as follows

$$err = \mathbf{restricting} \vee \mathbf{blocking} \vee \mathbf{noChoice} \vee \mathbf{errorStates}$$

The predicates in $Preds(err)$ of the form $offeredSymbolicAction_{a,h,Z} \approx 0$ or $location \approx q$ where $q \in Q$, $a \in \Sigma$, $Z \subseteq X^c_{cont}$ and $h \in atoms_\mu$ are by definition in $P_o \cup P_i \cup P_h$. Also the predicates in $Preds(g_{[X^c \rightarrow C]})$ and $Preds(h_{[X^c \rightarrow C]})$ where $g \in \mathcal{G}(\mathcal{H})$, $h \in atoms_\mu$ are in P_h as the predicates of the form $c_x \sim k/n$ where $x \in X^c \cup X^c_{cont}$, $k \in \mathbb{Z} \cap [0, n \cdot max(x)]$, $\sim \in \{<, \leq, \approx, \geq, >\}$ are in P_h. The values $k_1(g,x)$ and $k_2(g,x)$ with $g \in \mathcal{G}(\mathcal{H})$ and $x \in X^c$ are computed from the guards in $\mathcal{H}$ and thus it holds that both values are in $\mathbb{Z} \cap [0, n \cdot max(x)]$. Thus the resulting predicates of $lessK_1(g,x,term)$ and $lessK_2(g,x,term)$ where $term$ is of the form c_x or $c_y + k_1(g,x) - c_x$ are in P_h and therefore all predicates in $Preds(\mathbf{couldBeSatisfied}(g \wedge h, X^c \cup X^c_{cont}))$ are in P_h. With the same argumentation over the guards and the atoms it holds for all $p \in Preds(\mathbf{couldBeSatisfied}(g \wedge h, X^c \cup X^c_{cont}))$ that $p \in P_h$. Thus it holds for all $p \in \mathit{Preds}(err)$ that $\{v \in S \mid v \models p\} \in R$.

(iii) Identically to lemma 10.

(iv) Identically to lemma 10.

□

5.3 Termination

Here we state the results from [14] and show that the games resulting from the encodings satisfy the sufficient condition for termination of the refinement loop. The refinement loop terminates if only finitely many different abstract states are generated by the refinement algorithm. The algorithm obtain refinement predicates through interpolation of real valued variables. These computed interpolants are restricted at each refinement step to some finite language $\mathcal{L}_b$.

Definition 45. *A rectangular predicate over $Obs(X)$ is a conjunction of rectangular inequalities of the form $ax \lhd c$, where $x \in Obs(X)$, $a \in \{-1, 1\}$, $\lhd \in \{<, \leq\}$ and c is an integer constant. For $m \in \mathbb{N}$, a rectangular predicate φ is called m-bounded if for each conjunct $ax \lhd c$ of φ, $|c| \leq m$. Let $\mathcal{L}_m$ be the set of all m-bounded rectangular predicates over $obs(X)$.*

Theorem 3. *Let $\mathcal{G} = (\mathcal{C}, err)$ be a symbolic safety game with incomplete information. If there exists a finite region algebra (R, Obs) for $\mathcal{G}$ with $Obs = \mathcal{L}_m$ for some $m \in \mathbb{N}$ then the refinement loop terminates.*

The set *Obs* of both region algebras for the two encodings contains predicates over discrete variables. We interpret the discrete variables as reals by substituting the negations and inequalities by disjunctions of equalities as described in the last sections. This is possible as each of them has finite domains. For both region algebras we can transform the set *Obs* into a set of rectangular inequalities. Thus the theorem applies to the resulting games which provide an proof for the decidability of the control synthesis problem for the covered classes of timed games.

6 Selected Details of the Encoding

In this section we discuss the details of the encodings. In both encoding the variable c_{synch} is used to ensures that to all clock variables the same amount of time is added. This variable was just introduced for simplicity. The variable c_{synch} can be substituted by any clock variable as can be seen in the following example.

$$
\begin{aligned}
c'_x &= c_x + c'_{synch} - c_{synch} \\
c'_y &= c_y + c'_{synch} - c_{synch} \\
c'_z &= c_z + c'_{synch} - c_{synch} \\
c'_{synch} &> c_{synch}
\end{aligned}
\quad \Leftrightarrow \quad
\begin{aligned}
c'_x &= c_x + c'_z - c_z \\
c'_y &= c_y + c'_z - c_z \\
c'_z &> c_z
\end{aligned}
$$

Let us consider first the encoding of the model based on observation predicates from section 4.2. Here we encountered the following problems:

- How to hide the number of transitions played by the environment?
- How to ensure that the environment does not reset the system clock variables multiple times?
- How to model the rule that the environment is not allowed to play two time transitions in succession?
- How to model the rule that player 2 is only allowed to let time pass until an observation change?
- How to model the rule that player 2 is only allowed to let time pass until the chosen controllable action becomes enabled?

The most of these problems were already touched in the definition of the encoding. Here we go into some details. In the model from [11], OBSI strategies are used while in the symbolic model a strategy has to be consistent. An OBSI strategy just depends on the finite stuttering observations of a prefix: i.e. on the observations obtained from the choice points. That means in general player 1 does not know what and how many transitions were played between two choice points. A consistent system strategy depends on the observable variables of the states, but here the system knows how many environment states are between two system prefixes. Thus the system could get an advantage by counting the environment states and construct an strategy which does not correspond to an strategy in the original game.
This is illustrated in the following example. Given the game structure for an M-TGS stated in figure 6, player 1 has an own clock y and can retrieve the set of observations

$\mathcal{O} = \{(\{l_0, l_1\}, 0 \leq y < 1), (\{l_0, l_1\}, 1 \leq y), (\{l_2, l_3, l_4\}, 0 \leq y < 1), (\{l_2, l_3, l_4\}, 1 \leq y), (\{l_5\}, 0 \leq y < 1), (\{l_5\}, 1 \leq y)\}$. The safety property is $\Box\neg\{l_5\}$. In state l_2

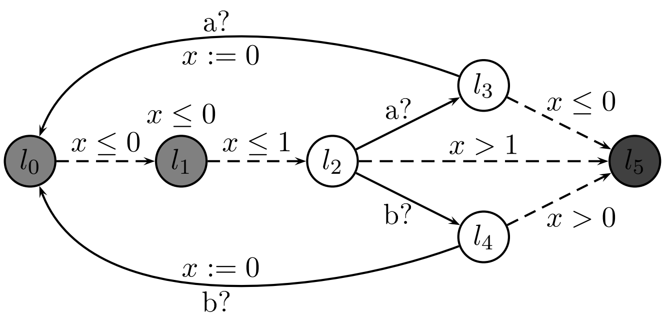

Figure 6: Timed game with incomplete information which is uncontrollable

player 1 has to decide if he either want to choose $a!$ or $b!$ else player 2 could play the uncontrollable transition which leads to the error state l_5. Player 1 should propose $a!$ if player 2 let time pass and else $b!$. This game is uncontrollable as player 1 do not know if player 2 let in state l_1 time t pass (if t is less than 1) or not as such prefixes will have the same finite observation. The problem is that although these prefixes have the same finite observations the mappings to the encoded game have different lengths as can been see in figure 7. The grey states denote the system states in G and the choice points in H. Note that $v_4 \equiv v_4'$ but the two prefixes in G are not equivalent. Thus the system can construct an winning strategy f_s by choosing $b!$ if there only two states between the system states and else $a!$. Such a strategy does not correspond to a winning strategy in the original game. This was solved by introducing the do nothing transitions. The strategy f_s is not longer a winning strategy as the environment could play a do nothing transition to pad the prefixes to

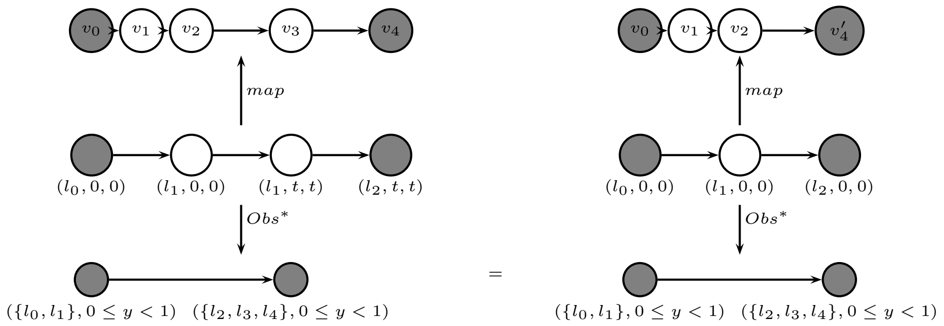

Figure 7: Two observation equivalence prefixes where the mapping is different $(0 < t < 1)$.

an arbitrarily length. Figure 8 shows another example how different prefixes could be padded with do nothing transition.

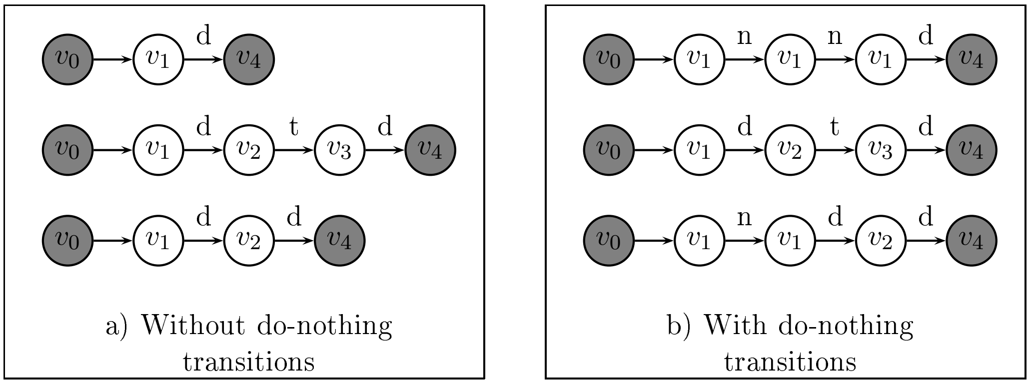

Figure 8: Prefixes in a encoding G where $\{v_0, v_1, v_2, v_3, v_4\} \subseteq S$ is a subset of states, $\{(v_0, v_1), (v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)\} \subseteq R$ represents time and discrete transitions and $(v_1, v_1) \in R$ represents a do nothing transition. The transitions are labeled where d stands for a discrete transition, t for a time transition and n for a do-nothing transition.

The multiple-reset and time transition repetition problem was resolved by introducing two flags *resetDone* and *ttAllowed*. The variable *resetDone* is set to true if a reset was done, as the name says and set to false if the system gets the turn back. Further no transitions which represent time transitions can be played if the variable *act* has the value *Reset* and the variable *resetDone* is false. The variable *ttAllowed* is set to false if a time transition was played and set to true if the system gets the turn back, a discrete or reset transition was played. As a transition which represents a time transition can only be played if *ttAllowed* is true the environment is not able to play two time transitions in succession
The rule that player 2 is only allowed to let time pass until an observation changes is modeled in the following way. Each time constraint φ of an observation predicate describes, because it is in $\mathcal{B}(X^c, M)$, a left-closed and right-open interval in $\mathbb{R}_{\geq 0}^{X^c}$. The constants $k_1(\varphi, x)$ and $k_2(\varphi, x)$ in $\mathbb{R}_{\geq 0}$ describe the lower and upper bounds for a clock x of such an interval. For a valuation are three possibilities. (1) The observation predicate is false and by letting time pass it will become true. Here at least one clock x is below their lower bound $k_1(\varphi, x)$. Either for the new valuation the observation predicate will evaluate again to false as x is below $k_1(\varphi, x)$ or observation predicate is true and at least one clock x is equal to their lower bound $k_1(\varphi, x)$. In the figure 9 this is the case for the valuation b. (2) The observation predicate is true. Here either for the new valuation the observation predicate will evaluate again to true or observation predicate is false and at least one clock x is equal to their

upper bound $k_2(\varphi, x)$. In the figure 9 this is the case for the valuation c. (3) The observation predicate is false and can not be enabled by letting time pass. In the figure 9 this is the case for the valuations a and d.
The first two cases can be checked easy and the new valuation can be restricted that either one clock is equal to the lower bound (case (1)) or all clock less or equal to the upper bound (case (2)). Case (3) is a more difficult as we have to check that the observation predicate can not be enabled by letting time pass. This can not be done by checking the old or new valuation as this not avoid cases, like illustrated in figure 9 for the valuations e, where transitions "goes through" an observation predicate. This was solved by introducing the formula **couldBeSatisfied**(φ, Y) for a clock constraint φ and a set of clocks Y. The formula is satisfied if all values of the clocks in Y are below their upper bound of φ (else φ cannot be satisfied) and for all clocks $x \in Y$ where the value is below the lower bound $k_1(\varphi, x)$ it holds that the distance to the lower bound $k_1(\varphi, x)$ is less than the distance of all others clocks $y \in Y$ to their upper bound $k_2(\varphi, y)$ i.e. if a clock enter its interval no other clock leave its interval.
The formula **stopAfterObsChange** combines the three cases and has to be satisfied if the environment wants to play a transition which models a time transition.
Example Consider the valuations $v_a = (1, 2)$, $v_b = (1, 1)$, $v_c = (3, 2)$, $v_d = (5, 2)$ and the clock constraint $\varphi = 2 \leq x < 4 \wedge 1 \leq y < 3$. Figure 9 shows the corresponding clock valuations and the zone for φ. It follows that

- $v_a \not\models$ **couldBeSatisfied**$(\varphi, \{x, y\})$
- $v_b \models$ **couldBeSatisfied**$(\varphi, \{x, y\})$
- $v_c \models$ **couldBeSatisfied**$(\varphi, \{x, y\})$
- $v_d \not\models$ **couldBeSatisfied**$(\varphi, \{x, y\})$

The rule that player 2 is only allowed to let time pass until the chosen controllable action is enabled is encoded similar by the rule for the observations. But in this case we only have to consider the cases when a guard becomes enabled or it can not be satisfied.
In the encoding of the model with the synchronized words we encountered similar problems as in the model before but because of the construction of the strategy and the winning condition (see section 2.3.1) there were also different challenges:

- How to simulate the game which is based on synchronized words?

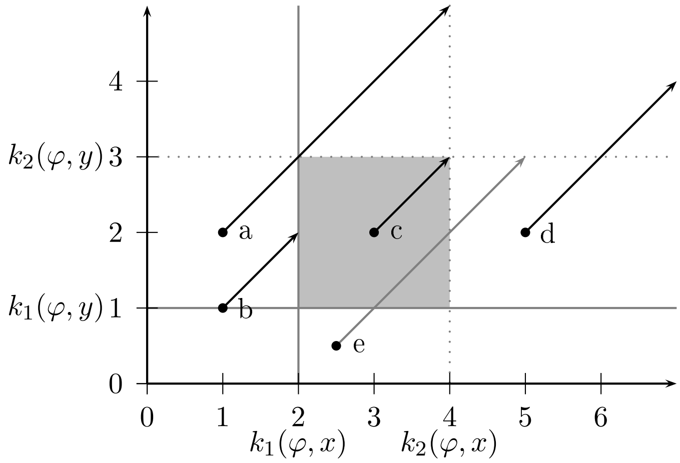

Figure 9: Clock valuations

- How to encode the requirements on a strategy to be non-blocking and non-restricting in the winning condition?
- How to model the winning condition?

In the model of [8] a play is just a symbolic word which were build in such a way that player C offered a subset of actions and player E picks one action. The timed transition system comes in the play only by checking the winning condition and that a strategy is valid. This is done by building the set of synchronized words. In the encoding we have to consider the transition system concurrently with the choices and give the environment the ability to play unobservable (and uncontrollable) transitions between two observable (and uncontrollable) or controllable transitions. This was done by letting the environment, after the system offered a subset of actions encoded by the *offeredAtomAndReset* variables, first play a sequence (possibly empty) of unobservable transitions. Afterwards letting it play an observable or controllable transition and then again a sequence (possibly empty) of unobservable transitions. This encoding simulates the construction of the synchronized words.
In [8] only valid strategies are considered. To simulate valid strategies a strategy for the system which has a non valid behavior will end in an error state. For restricting strategies this was straight forward. The formula **restricting** which define error states checks if in a state a observable transition of player E is enabled but the corresponding *offeredAtomAndReset* variable was not set by the system. The formula **blocking** is more complex. Similarly to the formula which simulates the letting-time-pass-rule for player 2 in the model above we introduced a **couldBeSatisfied** formula. As guards could contain disjunctions they describe a set of intervals in the clock valuation space. So the bounds were defined from the conjunctions of the

distributive normal form. Also it was not possible to use only the bounds but we had to check if such intervals are left or right open or closed. Thus the **couldBe-Satisfied** formula depend also on the type of the intervals. So a strategy is blocking if there exists no unobservable transition which will be enabled and there exists a observable or controllable transition which will be enabled and the corresponding *offeredAtomAndReset* variable was not set by the system.
The winning condition in [8] is given by a nondeterministic automaton $\mathcal{A}$. Thus a strategy f is winning for player C iff

$$tw(synw^{w}_{\mathcal{H}}(plays_f(\Omega))) \cap L(\mathcal{A}) = \emptyset$$

The set of plays winning for player C can be described by an automaton $\mathcal{B}$ which is deterministic [8], but this automaton can describe an nondeterministic timed automaton. It is unsure if it is possible to encode a timed game with incomplete information and external nondeterministic specification to a symbolic game structure with error formula. Our solution was to restrict our self to deterministic specifications which describe safety conditions. Using deterministic specifications we can apply the conversion from [25] to obtain a timed game with internal winning condition. This results in the fact that this thesis does not cover the class of timed games with external nondeterministic specification.

7 Implementation of the Encoding of the Observation Based Model

7.1 Description of the Implementation

In this section we describe the implementation in java of the encoding for a safety game of the model based on observation predicates [11]. The tool takes two XML files which represent a safety game and returns a XML file which contain a representation of the symbolic encoding. We used JDOM [17] to read and write the XML files. As the M-TGS defined in [11] is similar to the TGS which is used in Uppaal Tiga [6] except for the observation predicates, we used the Uppaal Tiga XML specification [24] and defined for the observation predicates an own XML specification (see appendix). The actions of player 1 are modeled as channels and annotating the controllable transitions with synchronisation labels. In the additional file also the Bound M for the M-TGS, the clocks which are used by player 1 and the safety winning condition are defined. The clocks of player 1 are not defined in the Uppaal Tiga XML to distinguish them from the clocks of player 2.

The tool takes both files and creates a XML file which represents the symbolic encoding. The output format can looked up in the appendix (see appendix).

7.2 Experiments

We took the example from [11] which is shown in figure 3 in section 2.3.2 and encoded it with the tool.

This timed game models a box painting production system where the system has to eject the box in the right state. The controller has incomplete information about the state and the value of an additional clock y. The controller knows only if the game is in one of the location sets $E = \{l_0, l_2, l_3, l_5, l_6\}$, $H = \{l_1\}$, $L = \{l_4\}$ or $B = \{l_7\}$ and can compare the value of y to a fixed constant m. The set of observable predicates is defined by $\mathcal{P} = \{(E, 0 \leq y < m), (E, m < y), (H, 0 \leq y < m), (H, m < y), (L, 0 \leq y < m), (L, m < y), (B, 0 \leq y < m), (B, m < y)\}$ where m is a parameter and the set of actions for the controller is $\Sigma_1 = \{Reset_y, Eject!, \Delta\}$. As l_7 is the bad state the controller looses if one of the observations $(B, 0 \leq y < m)$ or $(B, m < y)$ is true. We fix $m = 0.5$. For this value the game is controllable [11]. The TGS was modeled in Uppaal Tiga (see appendix) and the clock y, the observation predicates and the safety conditions where defined in the additional file (see appendix). With these file as input we computed with the tool for the example the symbolic encoding. We did not include this file as the XML format blowed up the file to around 10000 lines.

7.3 Possible Additional Experiments

At the time when this thesis was written the tool which solves the symbolic model from [14] was unfortunately not finished. A nice result of these thesis would have been to compare the results of [11] for the example from the last section with the result of tool with the encoding as input. The number of explored states and the size of the strategy proposed in [11] for different values of the parameter m would have been a base for a first comparison of the two approaches.
As there is now a prototype for Uppaal Tiga which takes the observation predicate model as input [10] there would be further the possibility to compare different systems.

8 Outlook

We showed that we are able to define encodings for the two considered models of timed games. Both encodings just encode the original game directly into a symbolic game with safety winning condition, but do not use the possible benefits of abstraction refinement.
In both models the strategy of the controller is based on points where the controller can observes changes of the plant. That means either an observable transition was taken or the evaluation of the given predicates over the observable clocks which limit the resources of the controller changes. The behavior of the plant is restricted on the one hand by the choices of the controller and on the other hand by the points where the controller gets the turn back as this limits the possibility to let time pass. In [11] observation predicates are used to limit the resources of the controller and the time transitions of the environment as they determine the states where the controller can choose controllable actions.
The fixed granularity in [8] has a similar effect. It constraints the behavior of the plant by restricting the time when controllable actions are allowed to play. After a controllable or observable action was played the controller makes a new choice.
Thus in the encoding of the both models the predicates which are provided that the controller can distinguish two states are given a priori. Using abstraction refinement it should be possible to generate these predicates in the refinement procedure. We are looking for a model which does not provide such predicates a priori and make use of this more powerful machinery. This leads to models where the controller can compare his own clocks to an arbitrary set of constants.
A model which does not use a fixed granularity or fixed observation predicates needs an other mechanism to limit the time transitions of the environment and determine the states where the controller is allowed to choose the controllable actions. Further the controller synthesis problem is in general undecidable for models where the controller can compare his own clocks to an arbitrary set of constants [8].
The following two approaches could be used to determine the points where the controller can choose the controllable action and knows the exact value of his own clocks.

(1) There exists fixed points in time (independent from any clock in the game) where the controller is asked to choose an action.

(2) There exists fixed locations where the controller is asked to choose an action.

In both approaches the controller has incomplete information as he is not able to

observe the uncontrollable clocks and only knows the set of locations instead the concrete location. He will also get the turn back if the plant reach a new location set. The first approach could model in practice for example a controller which checks periodically the state of the plant or is waked up if something changes.
In a symbolic game structure which represent such approaches, predicates which are needed to distinguish two states are derived from the transition system by the abstraction refinement procedure.
We believe that for the proposed models the controller synthesis problem is decidable for safety winning conditions. Although the controller can compare his own clocks to arbitrary constants, through the restrictions when he gets the turn back we could gain the decidability back. This assumption has to be proved.
Example. In the following we give an example with the idea of (2). The controller has a finite set of clocks $\{y\}$ and can propose an action from a finite set $\{0, 1\}$. He has also the possibility to reset his clock by playing r or play the special action delay Δ. The rules are similar to the model from [11]. The different is that the time transitions are not limited by observation predicates and the controller gets the turn only back if the observation of the location changes. If it is the turn of the controller he knows the exact value of y. The transition system is shown in figure 10. The states are partitioned into sets of observation equivalent locations $E = \{l_0, l_1, l_3\}$, $H = \{l_{2,4}\}$ and $B = \{l_5\}$. The controller must avoid that the plant visit the error location l_5. He has to distinguish by the value of his own clock y if the plant is in location l_2 or in l_4 and to chose the right action, either 0 or 1. The game is controllable as the controller can check his clock in the location set H and can determine by the value of his own clock in which location the plant is.

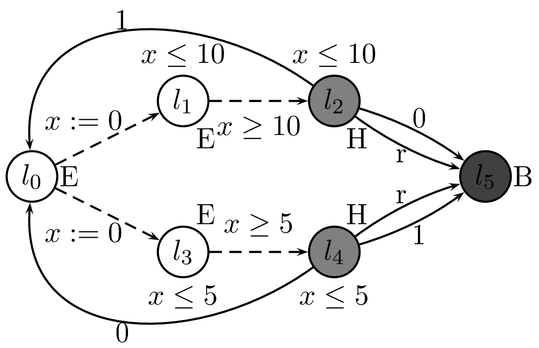

Figure 10: Example of a game without fixed granularity or observation predicates.

As we can construct a finite region algebra for this example the refinement loop will terminate. We model the game with the following set variables

$$X_h = \{x, loc, rd\},\ X_i = \{locObs, y\},\ X_o = \{a\}$$

where

$$\begin{aligned}&t \in \{0,1\}, loc \in \{0,..,5\}, x, y \in \mathbb{R} \cap [0, M], a \in \{0, 1, \Delta, r\},\\ &locObs \in \{E, H, B\}, rd \in \{0,1\}\end{aligned}$$

and the transition relation formulas which can looked up in the appendix. The following sets of predicates generate a finite region algebra (R, Obs) for symbolic games which satisfies the conditions described in [14].

$$\begin{aligned}R^p = \{&t \approx 0, t \approx 1, a \approx 0, a \approx 1, a \approx r, a \approx \Delta, rd \approx 0, rd \approx 1\\ &y \approx 0, y > 0, y \leq 5, y > 5, y \leq 10, y > 10,\\ &locObs \approx E, locObs \approx H, locObs \approx B,\\ &loc \approx 0, loc \approx 1, loc \approx 2, loc \approx 3, loc \approx 4, loc \approx 5,\\ &x \approx 0, x > 0, x \leq 5, x > 5, x \leq 10, x > 10\}\end{aligned}$$

$$\begin{aligned}Obs^p = \{&t \approx 0, t \approx 1, a \approx 0, a \approx 1, a \approx r, a \approx \Delta,\\ &y \approx 0, y > 0, y \leq 5, y > 5, y \leq 10, y > 10,\\ &locObs \approx E, locObs \approx H, locObs \approx B\}\end{aligned}$$

Thus the refinement loop will terminate for this game. This example shows that is it possible to create games with a safety winning condition for which the controller synthesis problem is decidable and which take advantage of the abstraction refinement procedure. For the proposed models we made restrictions that the controller is awaked at fixed points, either depending on time or location. The question is if it is possible to create models with less or better restrictions or even totally different (better) models, which also make use of the abstraction refinement benefits.

9 Conclusion

In this thesis we have proposed two symbolic encodings of timed games with incomplete information. In both encodings the environment can play self loops denoted as do nothing transition to hide the number played transitions. This avoids that the system can take an advantage by counting the played transitions and guarantees that the system has the same amount of information as in the original games. Formulas were introduced which ensures that in the encoding of the model from [11] transitions which represent time transitions are behave correct. That means the environment can only let time pass until the observation of the controller changes. Similar formulas ensure that non valid strategies in the encoding of the model from [8] lead to an error state.

We proved the correctness of the two encodings by introducing a mapping and showing that there exists a winning strategy in the encoding if and only if there exists a winning strategy for the original game.

For both models we consider the class of timed games with incomplete information with deterministic specification and fixed resources for the controller. For this class it is known that the controller syntheses problem is decidable. In the case of non fixed resources of the controller the problem becomes undecidable while for nondeterministic specifications that specify undesired behavior it is still decidable. However it is questionable if it is possible to encode timed games with incomplete information with a nondeterministic specification as a symbolic game structure where the error condition is given by an error formula.

The provided implementation will be useful to encode systems given in the syntax of [11]. The timed game automaton can be modelled in Uppaal Tiga[6] and an additional XML file defines the limited resources of the controller. The resulting XML file can be used as input for a tool which implements the abstraction refinement approach from [14]. In this way it is possible to compare the both approaches.

We showed that for each of the two models there exists a region algebra which satisfies the sufficient conditions that the refinement loop in [14] terminates. Together with the proof of correctness for both models this provides an alternative proof for the decidability of the controller syntheses problem for the considered class of timed games with incomplete information.

List of Figures

List of Tables

References

[1] K. Altisen and S. Tripakis. Tools for controller synthesis of timed systems. In *2nd Workshop on Real-Time Tools (RT-TOOLS'2002)*, july 2002.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211:1–13, 1999.

[4] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata 1.

[5] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, May 2008.

[6] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. UPPAAL-Tiga: Time for Playing Games! In H. H. W. Damm, editor, *Computer Aided Verification 19th International Conference on Computer Aided Verification (CAV07)*, volume 4590 of *Lecture Notes in Computer Science*, pages 121–125, Berlin Germany, 2007. Springer.

[7] P. Bouyer. `http://www.lsv.ens-cachan.fr/~bouyer/files/cav03.ps`.

[8] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *To appear in Proceedings of CAV 2003*, Lecture Notes in Computer Science. Springer Verlag, 2003.

[9] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems (tool-presentation for ftrtft '98). In *FTRTFT*, pages 298–302, 1998.

[10] F. Cassez. Efficient on-the-fly algorithms for partially observable timed games. In *Proc. of the 5th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'07) Proc. of the 5th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'07)*, volume 4763 of *Lecture Notes in Computer Science*, pages 5–24, Salzburg, Austria France, 10 2007. Springer.

[11] F. Cassez, A. David, K. G. Larsen, D. Lime, and J. F. Raskin. Timed control with observation based and stuttering invariant strategies. In *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis*, number to appear in LNCS, page to appear. Springer, 2007.

[12] F. Cassez, R. David, E. Fleury, and K. G. Larsen. Efficient on-the-fly algorithms for the analysis of timed games. In *In CONCUR 05, LNCS 3653*, pages 66–80. Springer, 2005.

[13] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, , and M. Stoelinga. The element of surprise in timed games. In *Proceedings CONCUR*, volume 2761 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[14] R. Dimitrova and B. Finkbeiner. Abstraction refinement for games with incomplete information. In *FSTTCS*, 2008.

[15] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 571–582, London, UK, 2002. Springer-Verlag.

[16] T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *In Proc. 30th Int. Colloquium on Automata, Languages, and Programming (ICALP), volume 2719 of LNCS*, pages 886–902. Springer, 2003.

[17] JDOM project. `http://www.jdom.org`.

[18] B. Khoussainov and A. Nerode. *Automata Theory and Its Applications*. Birkhauser Boston, 2001.

[19] O. Kupferman and M. Y. Vardi. Synthesis with incomplete informatio. In *In Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, 2000.

[20] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct. 1997.

[21] O. L. NGUENA TIMO. Model-cheking game : case of event-recording logic. In Y. S. e. K. Assogba, E. Badouel, editor, *Proceedings of the 8th African Conference on Research in Computer Science (CARI)*, pages 91–98, Nov 2006.

[22] J. H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274–301, Oct. 1984.

[23] J. Sifakis, O. Maler, and A. Pnueli. On the synthesis of discrete controllers for timed systems. In *STACS '95: Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science*, pages 229–242. Springer, 1995.

[24] Uppaal Team. `http://www.it.uu.se/research/group/darts/uppaal/flat-1_1.dtd`.

[25] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of 30th Conf. Decision and Control*, pages 1527–1528, Brighton, UK, 1991.

[26] M. D. Wulf, L. Doyen, J. françois Raskin, and U. L. D. Bruxelles. A lattice theory for solving games of imperfect information. In *In HSCC, LNCS 3927*, pages 153–168. Springer, 2006.

Appendix

XML Specification for the Symbolic Game Structure

```
<?xml version="1.0" encoding="UTF-8"?>

<!element game (variables,transitions,init,err,observations?)>

<!element variables  (variable*)>
<!element variable  (name,type,lower?,upper?)>
<!-- name of the variable -->
<!element name  (#PCDATA)>
<!-- type of the variable: int,real -->
<!element type  (#PCDATA)>
<!-- if the variable is bounded from below: lower bound -->
<!element lower (#PCDATA)>
<!-- if the variable is bounded from above: upper bound -->
<!element upper (#PCDATA)>

<!-- Each transition is described by two formulas: guard and update.
The guard is a formula over un-primed variables. The update is a
formula over primed and un-primed variables. The transition relation
for the system player has to bescribed as follows: for each possible
assignment to the output variables there exists a separate transition
in which the update is a conjunction of equalities. -->
<!element transitions (transition*)>
<!element transition  (guard,update)>
<!element guard   (formula)>
<!element update  (formula)>

<!element init  (formula)>
<!element err   (formula)>

<!-- A conjunction of formulas defining the variables corresponding to
the observation predicates. -->
<!element observations (formula)>
<!element formula   (and|or|not|eq|lt|le|gt|ge|true|false)>
<!element and   (and|or|not|eq|lt|le|gt|ge|true|false)*>
<!element or    (and|or|not|eq|lt|le|gt|ge|true|false)*>
<!element not   (and|or|not|eq|lt|le|gt|ge|true|false)>

<!-- Predicate symbols in atomic formulas: eq,lt,le,gt,ge -->
<!-- eq stands for equality-->
<!element eq    ((var|const|mul|sum|sub),(var|const|mul|sum|sub))>
<!-- eq stands for less then -->
<!element lt    ((var|const|mul|sum|sub),(var|const|mul|sum|sub))>
<!-- eq stands for less or equal -->
<!element le    ((var|const|mul|sum|sub),(var|const|mul|sum|sub))>
<!-- eq stands for greater than -->
<!element gt    ((var|const|mul|sum|sub),(var|const|mul|sum|sub))>
<!-- eq stands for greater or equal -->
<!element ge    ((var|const|mul|sum|sub),(var|const|mul|sum|sub))>
```

```

<!element true  (#PCDATA)>
<!element false (#PCDATA)>

<!-- A term is a variable name, a constant, a variable multiplied by a
constant, <sum> t0...tn </sum> which stands for (t0 + ...+ tn) or
<sub> t0...tn </sub> which stands for (t0 - t1 ...- tn)-->
<!element var   (#PCDATA)>
<!element const (#PCDATA)>
<!element mul (const,var)>
<!element sum   (var|const|mul|sum|sub)*>
<!element sub   (var|const|mul|sum|sub)*>

<!-- Determines whether a variable is observable or not-->
<!attlist variable obs (true|false) "true" >
<!-- Determines whether a variable is an output variable for the
system or not-->
<!attlist variable out (true|false) "false" >
<!attlist transition player (system|environment) "environment" >
<!-- Determines whether the variable is primed or not-->
<!attlist var prime (true|false) "false" >

```

XML Specification for the Additional Information

```
<?xml version="1.0" encoding="UTF-8"?>

<!element additionalInformation ( mBound, controllerClocks, observations, errorObservationSets ) >

<!element mBound EMPTY >
<!attlist mBound value NMTOKEN #REQUIRED >

<!-- Each controllable clock is described by a unique name -->
<!element controllerClocks ( controllerClock+ ) >
<!element controllerClock ( name ) >

<!-- Each observation is described by a name, a time constraint and a set of locations -->
<!element observations ( observation+ ) >
<!element observation ( name?, timeConstraint?, location* ) >
<!element name ( #PCDATA ) >
<!element location ( name ) >
<!element timeConstraint ( #PCDATA ) >

<!-- Each observation set which describes a set of error states is described by observations definend in observation -->
<!element errorObservationSets ( observationSet+ ) >
<!element observationSet ( observation+ ) >
<!attlist observation name NMTOKEN #IMPLIED >
```

Uppaal XML File for an Encoded Example

```
<?xml version='1.0' encoding='utf-8'?>
<!doctype nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN' 'http://www.it.uu.se/research/group/darts/uppaal/flat-1_1.dtd'><nta>
  <declaration>// Place global declarations here.
</declaration>
  <template>
    <name x="5" y="5">Main</name>
    <declaration>// Place local declarations here.
chan eject;
clock x;</declaration>
    <location id="id0" x="-160" y="32">
      <name x="-168" y="48">l4</name>
      <label kind="invariant" x="-168" y="0">x&lt;=0</label>
    </location>
    <location id="id1" x="-64" y="32">
      <name x="-72" y="48">l5</name>
      <label kind="invariant" x="-80" y="0">x&lt;=10</label>
    </location>
    <location id="id2" x="32" y="32">
      <name x="24" y="48">l6</name>
    </location>
    <location id="id3" x="128" y="-64">
      <name x="144" y="-72">l7</name>
    </location>
    <location id="id4" x="32" y="-160">
      <name x="24" y="-192">l3</name>
    </location>
    <location id="id5" x="-64" y="-160">
      <name x="-72" y="-192">l2</name>
      <label kind="invariant" x="-88" y="-144">x&lt;=20</label>
    </location>
    <location id="id6" x="-160" y="-160">
      <name x="-168" y="-192">l1</name>
      <label kind="invariant" x="-176" y="-144">x&lt;=0</label>
    </location>
    <location id="id7" x="-256" y="-64">
      <name x="-288" y="-72">l0</name>
    </location>
    <init ref="id7"/>
    <transition>
      <source ref="id7"/>
      <target ref="id3"/>
      <label kind="synchronisation" x="0" y="160">eject?</label>
      <nail x="-192" y="160"/>
      <nail x="192" y="160"/>
    </transition>
    <transition>
      <source ref="id0"/>
      <target ref="id3"/>
      <label kind="synchronisation" x="0" y="128">eject?</label>
      <nail x="-64" y="128"/>
```

```
    <nail x="128" y="128"/>
  </transition>
  <transition>
    <source ref="id1"/>
    <target ref="id3"/>
    <label kind="synchronisation" x="0" y="96">eject?</label>
    <nail x="0" y="96"/>
    <nail x="64" y="96"/>
  </transition>
  <transition>
    <source ref="id7"/>
    <target ref="id3"/>
    <label kind="synchronisation" x="0" y="-312">eject?</label>
    <nail x="-192" y="-288"/>
    <nail x="192" y="-288"/>
  </transition>
  <transition>
    <source ref="id6"/>
    <target ref="id3"/>
    <label kind="synchronisation" x="0" y="-280">eject?</label>
    <nail x="-64" y="-256"/>
    <nail x="128" y="-256"/>
  </transition>
  <transition>
    <source ref="id5"/>
    <target ref="id3"/>
    <label kind="synchronisation" x="0" y="-248">eject?</label>
    <nail x="0" y="-224"/>
    <nail x="64" y="-224"/>
  </transition>
  <transition>
    <source ref="id4"/>
    <target ref="id7"/>
    <label kind="synchronisation" x="-128" y="-120">eject?</label>
    <nail x="-64" y="-96"/>
    <nail x="-160" y="-96"/>
  </transition>
  <transition>
    <source ref="id2"/>
    <target ref="id7"/>
    <label kind="synchronisation" x="-128" y="-32">eject?</label>
    <nail x="-64" y="-32"/>
    <nail x="-160" y="-32"/>
  </transition>
  <transition controllable="false">
    <source ref="id2"/>
    <target ref="id3"/>
    <label kind="guard" x="40" y="-32">x&gt;=12</label>
  </transition>
  <transition controllable="false">
    <source ref="id1"/>
    <target ref="id2"/>
```

```
      <label kind="guard" x="-32" y="32">x&gt;=8</label>
    </transition>
    <transition controllable="false">
      <source ref="id0"/>
      <target ref="id1"/>
    </transition>
    <transition controllable="false">
      <source ref="id7"/>
      <target ref="id0"/>
      <label kind="assignment" x="-224" y="-8">x=0</label>
    </transition>
    <transition controllable="false">
      <source ref="id6"/>
      <target ref="id5"/>
    </transition>
    <transition controllable="false">
      <source ref="id4"/>
      <target ref="id3"/>
      <label kind="guard" x="40" y="-112">x&gt;=22</label>
    </transition>
    <transition controllable="false">
      <source ref="id5"/>
      <target ref="id4"/>
      <label kind="guard" x="-32" y="-184">x&gt;=18</label>
    </transition>
    <transition controllable="false">
      <source ref="id7"/>
      <target ref="id6"/>
      <label kind="assignment" x="-224" y="-136">x=0</label>
    </transition>
  </template>
  <system>// Place template instantiations here.
Process = Main();

// List one or more processes to be composed into a system.
system Process;</system>
</nta>
```

Additional Information XML File for an Encoded Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<!doctype additionalInformation SYSTEM "additionalInformation.dtd">

<additionalinformation>
  <mbound value="22" />
  <controllerclocks>
    <controllerclock>
      <name>y</name>
    </controllerclock>
  </controllerclocks>
  <observations>
    <observation>
      <name>o1</name>
      <timeconstraint>0 &lt;= y &amp;&amp; y &lt; 1</timeconstraint>
      <location>
        <name>l0</name>
      </location>
      <location>
        <name>l2</name>
      </location>
      <location>
        <name>l3</name>
      </location>
      <location>
        <name>l5</name>
      </location>
      <location>
        <name>l6</name>
      </location>
    </observation>
    <observation>
      <name>o2</name>
      <timeconstraint>1 &lt;= y</timeconstraint>
      <location>
        <name>l0</name>
      </location>
      <location>
        <name>l2</name>
      </location>
      <location>
        <name>l3</name>
      </location>
      <location>
        <name>l5</name>
      </location>
      <location>
        <name>l6</name>
      </location>
    </observation>
    <observation>
      <name>o3</name>
```

```xml
      <timeconstraint>0 &lt;= y &amp;&amp; y &lt; 1</timeconstraint>
      <location>
        <name>l1</name>
      </location>
    </observation>
    <observation>
      <name>o4</name>
      <timeconstraint>1 &lt;= y</timeconstraint>
      <location>
        <name>l1</name>
      </location>
    </observation>
    <observation>
      <name>o5</name>
      <timeconstraint>0 &lt;= y &amp;&amp; y &lt; 1</timeconstraint>
      <location>
        <name>l4</name>
      </location>
    </observation>
    <observation>
      <name>o6</name>
      <timeconstraint>1 &lt;= y</timeconstraint>
      <location>
        <name>l4</name>
      </location>
    </observation>
    <observation>
      <name>o7</name>
      <timeconstraint>0 &lt;= y &amp;&amp; y &lt; 1</timeconstraint>
      <location>
        <name>l7</name>
      </location>
    </observation>
    <observation>
      <name>o8</name>
      <timeconstraint>1 &lt;= y</timeconstraint>
      <location>
        <name>l7</name>
      </location>
    </observation>
  </observations>
  <errorobservationsets>
    <observationset>
      <observation name="o7"/>
    </observationset>
    <observationset>
      <observation name="o8"/>
    </observationset>
    <observationset>
      <observation name="o7"/>
      <observation name="o8"/>
    </observationset>
```

```
  </errorobservationsets>
</additionalinformation>
```

Transition Relation Formulas for the Example from Section 8.

$\mathcal{T}_s$	guard	update
	$t \approx 1$	$t' \approx 0 \wedge x' \approx x \wedge y' \approx y \wedge loc' \approx loc \wedge locObs' \approx locObs \wedge rd' \approx rd$

$\mathcal{T}_e$	guard	update
d	$t \approx 0 \wedge loc \approx 0$	$t \approx 0 \wedge loc' \approx 1 \wedge x' \approx 0 \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx E \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 0$	$t \approx 0 \wedge loc' \approx 3 \wedge x' \approx 0 \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx H \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 1$	$t \approx 1 \wedge loc' \approx 2 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx E \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 3$	$t \approx 1 \wedge loc' \approx 4 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx H \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 2 \wedge a \approx 1$	$t \approx 1 \wedge loc' \approx 0 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx E \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 2 \wedge (a \approx 0 \vee a \approx r)$	$t \approx 0 \wedge loc' \approx 5 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx B \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 4 \wedge (a \approx 1 \vee a \approx r)$	$t \approx 0 \wedge loc' \approx 5 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx B \wedge a' \approx a$
d	$t \approx 0 \wedge loc \approx 4 \wedge a \approx 0$	$t \approx 1 \wedge loc' \approx 0 \wedge x' \approx x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx E \wedge a' \approx a$
t	$t \approx 0 \wedge loc \approx 0$	$t \approx 0 \wedge loc' \approx loc \wedge y' \approx y + x' - x \wedge rd' \approx rd$ $\wedge locObs' \approx locObs \wedge a' \approx a$
t	$t \approx 0 \wedge loc \approx 1 \wedge x < 10$	$t \approx 0 \wedge loc' \approx loc \wedge x' \leq 10 \wedge y' \approx y + x' - x$ $\wedge rd' \approx rd \wedge locObs' \approx locObs \wedge a' \approx a$
t	$t \approx 0 \wedge loc \approx 2 \wedge x < 5$	$t \approx 0 \wedge loc' \approx loc \wedge x' \leq 5 \wedge y' \approx y + x' - x$ $\wedge rd' \approx rd \wedge locObs' \approx locObs \wedge a' \approx a$
r	$t \approx 0 \wedge rd \approx 0$	$t \approx 0 \wedge loc' \approx loc \wedge x' \leq x \wedge y' \approx 0 \wedge rd' \approx 1$ $\wedge locObs' \approx locObs \wedge a' \approx a$
dn	$t \approx 0$	$t \approx 0 \wedge loc' \approx loc \wedge x' \leq x \wedge y' \approx y \wedge rd' \approx rd$ $\wedge locObs' \approx locObs \wedge a' \approx a$