Theoretical Computer Science 412 (2011) 6770–6785

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Specifications for decidable hybrid games

Vladimeros Vladimerou [a,*], Pavithra Prabhakar [b], Mahesh Viswanathan [b], Geir Dullerud [c]

[a] *Department of Automatic Control, University of Lund, LTH, SE-22100, Sweden*

[b] *Computer Science Department, University of Illinois at Urbana-Champaign, UIUC, 61801, IL, USA*

[c] *Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign, UIUC, 61801, IL, USA*

A R T I C L E   I N F O

Article history:
Received 25 February 2010
Received in revised form 30 June 2011
Accepted 24 August 2011
Communicated by J. Esparza

Keywords:
Hybrid systems
Hybrid games
Weighted CTL
Control
Verification

A B S T R A C T

We introduce STORMED hybrid games (SHG), a generalization of STORMED hybrid systems, Vladimerou et al. (2008) [33], which have natural specifications that allow rich continuous dynamics and various decidable properties. We solve the control problem for SHG using a reduction to bisimulation on finite game graphs. This generalizes to a greater family of games, which includes o-minimal hybrid games, Bouyer et al. (2006) [6]. We also solve the optimal-cost reachability problem for Weighted SHG and prove decidability of WCTL for Weighted STORMED hybrid systems.

1. Introduction

Designing reliable open systems requires solving the *control problem* wherein, given a system $\mathcal{M}$ and a requirement ψ, one wants to know if the behaviors of $\mathcal{M}$ can be "controlled" so as to satisfy ψ. Such a control problem is most naturally formalized as a game between a controller and a plant with actions/transitions being partitioned into *controllable* actions, i.e. the controller's choices, and uncontrollable actions, i.e. the moves of the plant, noise, or the environment. The controller synthesis problem is to design a strategy for the controller that ensures that the correctness requirements are met, no matter what the adversarial choices are, while (possibly) meeting certain cost constraints.

In the context of embedded systems, *hybrid games* [17,6,7] have been studied with a view to designing *hybrid* controllers for systems. Such games are defined using hybrid automata, which have finitely many discrete states and continuous variables that evolve with time, and whose discrete transitions have been partitioned into those that are controllable and those that are not. In the version that we consider here, at each step of the game, the controller (and the environment) has two choices: either to let time pass for t time units or to take a controllable (or uncontrollable) transition. If both the controller and the environment pick time, then the system evolves continuously for the shorter of the two durations. If exactly one of them picks a discrete transition, then the discrete transition chosen is taken and finally, in the case when both pick a discrete transitions, the controller's choice is respected. For other versions of hybrid games considered in the literature see related work below.

Our results apply to the *STORMED* specifications [33], and hybrid games which satisfy these are conveniently called STORMED hybrid games (SHG), as well as, in part, to o-minimal hybrid games. These specifications require invariants, guards,

* Corresponding author. Tel.: +46 40700511415.
E-mail addresses: vladimer@control.lth.se (V. Vladimerou), ppbrabha2@illinois.edu (P. Prabhakar), vmahesh@illinois.edu (M. Viswanathan), dullerud@illinois.edu (G. Dullerud).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.08.036

resets, and flows to be described in an *order-minimal* (or o-minimal) theory, and whose resets and flows satisfy certain monotonicity constraints that are found in many real-world applications [33]. When compared to previously studied classes of hybrid games, STORMED hybrid games allow for richer continuous dynamics than rectangular hybrid games [17] and timed games [3,4], and at the same time admit a stronger coupling between the continuous and discrete state components than o-minimal hybrid games [6]. For an example see Section 5.

We consider both *weighted* and *unweighted* versions of these games. In the unweighted case, we show that for regular winning objectives, the controller synthesis problem is decidable, provided the o-minimal theory used to describe the STORMED game is decidable. Our main technical observation shows that under special acyclicity conditions, bisimulation equivalence on the *time-abstract* transition system defined by the STORMED game preserves winning (and losing) states; here, the time-abstract transition system is the labelled transition system semantics of the STORMED game that ignores the distinction between controllable and uncontrollable transitions and abstracts the time when continuous transitions are taken. We show that both STORMED systems and o-minimal systems meet this technical acyclicity condition. Further the observations that the time-abstract transition system for a STORMED automaton has a finite bisimulation quotient [33] (which is effectively constructable when the underlying o-minimal theory is decidable) and the fact that finite games with regular objectives are decidable [25], allow us to conclude the decidability of STORMED hybrid games. We note that o-minimal systems also have finite time-abstract bisimulation quotients, and this gives an alternative proof of decidability of o-minimal games [6].

We also examine weighted versions of SHG, where there is a price on each of the game choices, and the goal is to design optimal (cost) winning strategies for the controller. We show that weighted STORMED games with reachability objectives are decidable (and the controller synthesizable) when the underlying o-minimal theory is decidable. In the games considered here, we avoid zeno plays, that is, the behaviors in which the environment (or controller) can simply pick shorter and shorter time steps, and thereby starve her opponent, by excluding such behaviors in the winning conditions. We observe that when considering non-zeno plays if there is a winning strategy λ for the controller then there is a winning strategy in which the controller does not choose a time step if in the previous step the controller chose a time step shorter than the environment. Based on this technical lemma, we conclude that for non-zeno reachability games, we need to only consider bounded strategies (i.e., those for which every play consistent with the strategy has bounded number of steps); and therefore we can not only compute the cost of the optimal strategy but also synthesize it.

Finally, we consider the problem of model checking *WCTL* properties for weighted STORMED systems. WCTL is a branching-time logic that allows for one to reason about the accumulated costs along computations in addition to regular properties. Once again we show the decidability of the WCTL model checking problem for hybrid systems with the STORMED specifications. Our result here relies on reducing WCTL model checking to CTL model checking on STORMED systems, which was previously shown to be decidable in [33].

Related work. Work on controller synthesis for real-time and hybrid systems has seen a lot of effort since [3] and [21]. Broadly speaking one assumes that the controller can examine the state at various times, and can influence the discrete steps that are taken. Other than [30], most papers typically assume that the controller cannot influence the way the plant evolves continuously. Assuming that the controller can observe the state at certain discrete time instants, it has been shown that the controller synthesis problem is decidable for rectangular hybrid automata [18]. In the dense time setting, there are different formulations of the controller synthesis problem. Assuming that the controller can only enable or disable transition (and not influence when they are taken), it has been shown that the synthesis problem is undecidable for rectangular hybrid automata but decidable for initialized rectangular hybrid automata [17]. When the controller chooses both the transition as well as when it is taken, the problem is known to be decidable for timed automata [21], and o-minimal hybrid automata [6],[1] but undecidable for initialized rectangular automata [19]. We extend these observations to STORMED systems. Symbolic algorithms for the controller synthesis problem first appeared in [13]. The controller synthesis problem has also been considered for dynamical systems (those with one discrete state) [14,27] where dynamical systems is first discretized, and also for switched systems, where the environment has limited power [22]. General categorical conditions on the controller synthesis problem are identified in [16,26].

With dense time, zeno behavior is sometimes a complicating issue for switching dynamics [12]. It is either avoided by imposing syntactic constraints on the game graph [3,4], restricting the kind of game moves allowed [6,7], or by semantic constraints imposed on the winning condition [13,10]. Here we take the approach of avoiding zeno behavior through the winning conditions.

To model resource consumption, weights/prices were added to timed systems, and weighted timed games have been examined since [2,5]. However, synthesis of the optimal cost controller for reachability is undecidable for timed automata [9], but decidable for o-minimal hybrid systems [7] with decidable underlying theories. Model checking timed automata against WCTL properties has been shown to be undecidable [9] but decidable for o-minimal systems [7] with decidable underlying theories. Here we show that optimal reachability and WCTL model checking are decidable for STORMED games.

A partial summary of our results has appeared in [34].

1 Defined on decidable theories.

2. Preliminaries

Equivalence relations and partitions. A binary relation R on a set A is a subset of $A \times A$. We will say aRb to denote $(a, b) \in R$. An *equivalence relation* on a set A is a binary relation R that is reflexive, symmetric and transitive. An equivalence relation partitions the set A into *equivalence classes*: $[a]_R = \{b \in A \,|\, aRb\}$. Let Π_R denote the set of equivalence classes of R. A partition Π of the set A defines a natural equivalence relation $\equiv_\Pi$, where $a \equiv_\Pi b$ iff a and b belong to the same partition in Π. In this paper, we will use the partition Π to mean both the partition, as well as the equivalence relation associated with it. Finally, we will say an equivalence relation R_1 *refines* another equivalence relation R_2 iff $R_1 \subseteq R_2$.

First order logic. In this paper we will consider first order vocabularies consisting of only relation symbols and constant symbols; we will call $\mathcal{A}$ to be a τ-structure if it is a structure over the vocabulary τ. Recall that a k-ary relation $S \subseteq A^k$, where A is the domain of $\mathcal{A}$, is said to be *definable* in the structure $\mathcal{A}$ if there is a formula $\varphi(x_1, x_2, \ldots, x_k)$, with free variables $x_1, \ldots, x_k$, such that $S = \{(a_1, \ldots, a_k) \mid \mathcal{A} \models \varphi[x_i \mapsto a_i]_{i=1}^k\}$. A k-ary function f will be said to be definable if its graph, i.e., the set of all $(x_1, \ldots, x_k, f(x_1, \ldots, x_k))$, is definable. A *theory* $Th(\mathcal{A})$ of a structure $\mathcal{A}$ is the set of all sentences that hold in $\mathcal{A}$. $Th(\mathcal{A})$ is said to be *decidable* if there is an effective procedure to decide membership in the set $Th(\mathcal{A})$. One consequence of this is that it is also decidable to check the emptiness of a definable relation, and whether two definable relations are equal.

O-minimality. A binary relation $\leq$ on a set A is said to be a *total ordering* if it is reflexive, transitive, antisymmetric $((a \leq b \wedge b \leq a) \Rightarrow a = b)$, and total $(a \leq b \vee b \leq a)$. The set A is said to be totally ordered if there exists a total order on it. Given a total order $\leq$, $<$ is the relation such that $a < b$ iff $a \leq b$ and $a \neq b$. An *interval* is a set defined in a totally ordered set using one or two bounds as follows: $\{x : a \sim_1 x \sim_2 b\}$, $\{x : x \sim a\}$, and $\{x : a \sim x\}$, where $\sim, \sim_1, \sim_2 \in \{\leq, <\}$. Trivially, $\{x : a \leq x \leq b\}$ with $a = b$, is an interval consisting of a single point. We write $\mathcal{A} = (A, \leq, \ldots)$ to convey that the τ-structure $\mathcal{A}$ has a total ordering relation $\leq$ and other elements in its structure. A totally ordered first-order structure $\mathcal{A} = (A, \leq, \ldots)$ is *o-minimal* (order-minimal) if every definable set is a finite union of intervals [32]. The theory of this structure is also called o-minimal. Examples of o-minimal structures include $(\mathbb{R}, <, +, -, \cdot, \mathtt{exp})$ and $(\mathbb{R}, <, +, -, \cdot)$, where $+, -, \cdot, \mathtt{exp}$ are the addition, subtraction, multiplication and exponentiation operations on reals, respectively. Additional examples can be found in [31,32]. The theory of $(\mathbb{R}, <, +, -, \cdot)$ is known to be decidable [28].

3. Game graph

Definition 1. A *game graph* (*GG*) $\mathcal{G} = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \rightarrow, L_Q, L_E)$, where

- Q is a set of states,
- Σ_C is a set of controllable actions,
- Σ_U is a set of uncontrollable actions,
- Σ_Q is a set of state labels,
- Σ_E is a set of edge labels,
- $\rightarrow \subseteq Q \times \Sigma_C \times \Sigma_U \times Q$ is a transition function,
- $L_Q : Q \rightarrow \Sigma_Q$ is a state labeling function, and
- $L_E : \Sigma_C \times \Sigma_U \rightarrow \Sigma_E$ is a transition labeling function.

Remark 1. A *transition system* can then be defined as a game graph in which the controllable alphabet Σ_C is a singleton. This captures the situation in which the controller has no choice but to select the only action in Σ_C. Hence we will drop this component from the definition of a transition system.

A game on a game graph is played by two players, namely, a controller and an environment. In each step of the game, the controller selects a controllable action enabled at the state and the environment selects an uncontrollable action. The game proceeds by moving to a new state depending on the actions chosen. Next, we formalize the game.

Runs and traces. A (finite or infinite) *run* of the game graph $\mathcal{G}$ is a sequence of transitions. We will denote $(q_1, c, u, q_2) \in \rightarrow$ by $q_1 \xrightarrow{c,u} q_2$. A run σ is a sequence $q_0(c_1, u_1)q_1(c_2, u_2)q_2 \cdots$ where $q_i \xrightarrow{c_{i+1},u_{i+1}} q_{i+1}$ for all $i \geq 0$. We denote the first state of the run σ by *first*(σ), thus *first*$(\sigma) = q_0$. We denote a prefix $q_0(c_1, u_1)q_1(c_2, u_2) \cdots (c_i, u_i)q_i$ of a run σ by σ_i. We call σ_{i+1} an extension of σ_i. A run σ is *finite* if $\sigma = \sigma_i$ for some i; in this case we will say that σ is of length $i + 1$. Given a finite run $\sigma = q_0(c_1, u_1)q_1(c_2, u_2) \cdots (c_i, u_i)q_i$, we denote the last state q_i by *last*(σ). We say that $c \in \Sigma_C$ (or $u \in \Sigma_U$) is enabled at q if there exists a $u \in \Sigma_U$ (or $c \in \Sigma_C$) and $q' \in Q$ such that $q \xrightarrow{c,u} q'$. Then we also say that (c, u) is enabled at q if there is a q' such that $q \xrightarrow{c,u} q'$. We say that (c, u) is enabled after a run σ, if it is enabled at *last*(σ). We use *Runs*$(\mathcal{G})$ to denote the set of all infinite runs of $\mathcal{G}$, and *Runs*$(\mathcal{G}, q)$ to denote the set of those starting at q. Similarly we will denote the set of all finite runs of $\mathcal{G}$ by *Runs*$_{fin}(\mathcal{G})$, and those starting at q by *Runs*$_{fin}(\mathcal{G}, q)$.

A *trace* of a run $\sigma = q_0(c_1, u_1)q_1(c_2, u_2)q_2 \cdots$ is the sequence of labels on its states and transitions, i.e., *trace*$(\sigma) = L_Q(q_0)$ $L_E(c_1, u_1)L_Q(q_1)$ $L_E(c_2, u_2)L_Q(q_2) \cdots$. We denote the set of all traces of the runs of $\mathcal{G}$ by *trace*$(\mathcal{G})$ and the set of all traces of runs of $\mathcal{G}$ starting at q by *trace*$(\mathcal{G}, q)$.

Strategies and winning conditions. A *strategy* is a function $\lambda : Runs_{fin}(\mathcal{G}) \to \Sigma_C$ such that if $\lambda(\sigma) = c$ then c is enabled after σ. A run $\sigma = q_0(c_1, u_1)q_1 \cdots$ is *consistent* with a strategy λ if $\lambda(\sigma_i) = c_{i+1}$ for all $i \geq 0$. A *winning condition* $\mathcal{W}$ is a subset of $(\Sigma_Q \Sigma_E)^\omega$. A strategy λ is winning for a state q with respect to the winning condition $\mathcal{W}$ if $trace(\sigma) \in \mathcal{W}$ for all $\sigma \in Runs(\mathcal{G}, q)$ consistent with λ. Then we say that q has a winning strategy λ for $\mathcal{W}$. The *control problem* is a pair $(\mathcal{G}, \mathcal{W})$, where $\mathcal{G}$ is a game graph and $\mathcal{W}$ is a winning condition, and asks to find the set of all states in $\mathcal{G}$ which have a winning strategy for $\mathcal{W}$. The controller synthesis problem asks to construct a winning strategy for all winning states.

The following theorem states that when the game graph is finite, the control problem can be solved for winning conditions which are specified in LTL.

Theorem 1 ([25]). *If the game graph $\mathcal{G}$ is finite, then the LTL control problem is PTIME-complete in the size of $\mathcal{G}$ and 2EXPTIME-complete in the size of the LTL formula.*

Now we define a bisimulation relation on game graphs which will relate states which are either both winning or losing with respect to some objective. The definition we present below is a restricted version of that presented in [1].

Bisimulation. Given two game graphs $\mathcal{G} = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \to, L_Q, L_E)$ and $\mathcal{G}' = (Q', \Sigma'_C, \Sigma'_U, \Sigma_Q, \Sigma_E, \to', L'_Q, L'_E)$ with the same set of state and transition labels, we say that $\mathcal{R} \subseteq Q \times Q'$ is a *bisimulation* on $(\mathcal{G}, \mathcal{G}')$, if for all $(q_1, q'_1) \in \mathcal{R}$, the following conditions hold:

1. $L_Q(q_1) = L'_Q(q'_1)$.
2. For all $c_1 \in \Sigma_C$ enabled at q_1, there exists a $c'_1 \in \Sigma'_C$ enabled at q'_1 such that:
   - for all $u'_1 \in \Sigma'_U$ and $q'_2 \in Q'$ such that $q'_1 \xrightarrow{c'_1, u'_1} q'_2$, there exists $u_1 \in \Sigma_U$ and $q_2 \in Q$ such that $L_E(c_1, u_1) = L'_E(c'_1, u'_1)$, $q_1 \xrightarrow{c_1, u_1}{}' q_2$ and $q_2 \mathcal{R} q'_2$.
3. For all $c'_1 \in \Sigma'_C$ enabled at q'_1, there exists a $c_1 \in \Sigma_C$ enabled at q_1 such that:
   - for all $u_1 \in \Sigma_U$ and $q_2 \in Q$ such that $q_1 \xrightarrow{c_1, u_1} q_2$, there exists $u'_1 \in \Sigma'_U$ and $q'_2 \in Q'$ such that $L'_E(c'_1, u'_1) = L_E(c_1, u_1)$, $q'_1 \xrightarrow{c'_1, u'_1}{}' q'_2$ and $q_2 \mathcal{R} q'_2$.

Remark 2. We observe that the above definition of bisimulation on game graphs reduces to the standard definition of bisimulation for transition systems.

Also we call a bisimulation *finite*, if it is also an equivalence relation with a finite number of equivalence classes.

The following proposition from [1] restated according to our definition of bisimulation relates bisimulations and winning strategies.

Proposition 2. *Let $(\mathcal{G}^1, \mathcal{G}^2)$ be two game graphs over the state labels Σ_Q and transition labels Σ_E. Let $\mathcal{W} \subseteq (\Sigma_Q \Sigma_E)^\omega$ be a winning condition. Let $\mathcal{R}$ be a bisimulation on $(\mathcal{G}^1, \mathcal{G}^2)$ and let $(q_1, q_2) \in \mathcal{R}$. Then there is a winning strategy from q_1 for $\mathcal{W}$ if and only if there is one from q_2.*

Remark 3. We call a bisimulation on $(\mathcal{G}, \mathcal{G})$, a bisimulation on $\mathcal{G}$.

4. Control for hybrid games

Definition 2. A *hybrid game* $\mathcal{H}$ is a tuple $(Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$ where:

- Loc is a finite set of locations,
- Act_C is a finite set of controllable actions,
- Act_U is a finite set of uncontrollable actions,
- $Labels$ is a finite set of state labels,
- $Cont = \mathbb{R}^n$ for some n, is a set of continuous states,
- $Edge \subseteq Loc \times (Act_C \cup Act_U) \times Loc$ is a set of edges,
- $Inv : Loc \to 2^{Cont}$ is a function that associates with every location an invariant,
- $Flow : Loc \times Cont \to (\mathbb{R}^+ \to Cont)$ is a flow function,
- $Guard : Edge \to 2^{Cont}$ is a function that assigns to each edge a guard,
- $Reset : Edge \to 2^{Cont \times Cont}$ is a function mapping an edge to a reset relation,
- $Lfunc : Loc \times Cont \to Labels$ is a state labeling function.

Remark 4. As before, a *hybrid system* is a hybrid game with Act_C a singleton set. Hence we will drop this component from the definition of a hybrid system.

The locations in Loc will be called the discrete (part of) states and the elements of $Cont$ the continuous (part of) states. A state is an element of $Loc \times Cont$, that is, a pair containing a discrete state and a continuous state. The flow function associates with each state a function that describes the evolution of the continuous state with respect to time. A guard is a condition on the continuous part of the state that must hold in order to take a transition. The reset function associates with each edge a *reset*, which is a binary relation that describes how the continuous state changes when a discrete transition is taken. In the above hybrid game, we call n the *dimension* of $\mathcal{H}$.

Remark 5. In contrast to most expositions of hybrid systems, the *Flow* function does not define a vector field, but describes a closed-form solution of the continuous dynamics. We will later impose a semi-group property on the *Flow* function which is guaranteed by closed form solutions of vector fields.

Before giving the semantics of a hybrid game we introduce some notation. We denote by $(l, x) \xrightarrow{t}_{\mathcal{H}} (l, x')$ the fact that starting at some state (l, x) one can let some time t elapse and reach (l, x'), i.e., there exists a $t \geq 0$ such that $Flow(l, x)(t) = x'$ and for all $0 \leq t' \leq t$, $Flow(l, x)(t') \in Inv(l)$. Similarly we denote by $(l, x) \xrightarrow{a}_{\mathcal{H}} (l', x')$ the fact that starting at some state (l, x) one can take a discrete action $a \in Act_C \times Act_U$ and go to (l', x'), i.e., there exists $e = (l, a, l') \in Edge$ such that $x \in Guard(e)$, $x' \in Inv(l')$ and $(x, x') \in Reset(e)$. We will drop the $\mathcal{H}$ whenever it is clear from the context. We will use t, t_1, t_2 and so on to denote an element of $\mathbb{R}_{\geq 0}$, the set of non-negative real numbers.

The semantics of a hybrid game is given in terms of a game graph corresponding to the following game. In each step, the controller selects a time t_1 or a controllable action, and similarly the environment selects a time t_2 or an uncontrollable action. If both of them choose a time, then the game proceeds by a time evolution equal to the minimum of the two times. The one with the minimum time is said to have won this step. If both of them selected the same time, then we non-deterministically declare one of them to have won. If one of them chooses a time and the other an action, then the action is taken, and the one selecting the action wins. Finally if both of them choose an action, then the controllable action is taken, and the controller wins. When both the controller and the environment choose a time step t, we need to be able to non-deterministically choose a winner. Hence we introduce a new set of uncontrollable actions $\{env\} \cdot \mathbb{R}_{\geq 0}$, such that a situation where the environment wins is modeled by a transition on the action $(t, env{\cdot}t)$, whereas the case where the controller wins is modeled by a transition on the action (t, t). (Given two sets S and T, $S \cdot T$ denotes the set $\{s \cdot t \mid s \in S, t \in T\}$.) The transition labels will correspond to the winning player: a transition in which the controller wins is labelled by either an action from Act_C or by $con \cdot \tau$ depending on whether it chose an action or a time. Similarly, a transition in which the environment wins is labelled by either Act_U or $env \cdot \tau$.

Game graph of a hybrid game. Formally, the game graph corresponding to the hybrid game $\mathcal{H} = (Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$ is given by $game(\mathcal{H}) = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \rightarrow, L_Q, L_E)$, where:

- $Q = Loc \times Cont$,
- $\Sigma_C = Act_C \cup \mathbb{R}_{\geq 0}$,
- $\Sigma_U = Act_U \cup \mathbb{R}_{\geq 0} \cup (\{env\} \cdot \mathbb{R}_{\geq 0})$,
- $\Sigma_Q = Labels$,
- $\Sigma_E = Act_C \cup Act_U \cup (\{con, env\} \cdot \{\tau\})$,
- $\rightarrow$ is defined as:
  - for $t_1, t_2 \in \mathbb{R}_{\geq 0}$ such that $(l, x) \xrightarrow{\min(t_1,t_2)}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{(t_1,t_2)} (l', x')$,
  - for $t \in \mathbb{R}_{\geq 0}$ such that $(l, x) \xrightarrow{t}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{(t,env\cdot t)} (l', x')$.
  - for $t \in \mathbb{R}_{\geq 0}$ and $u \in Act_U$ such that $(l, x) \xrightarrow{u}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{(t,u)} (l', x')$.
  - for $c \in Act_C$ and $a \in Act_U \cup \mathbb{R}_{\geq 0}$ such that $(l, x) \xrightarrow{c}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{(c,a)} (l', x')$.
- $L_Q(q, x) = Lfunc(q)$ for all $q \in Q$.
-
  - for $t_1, t_2 \in \mathbb{R}_{\geq 0}$, $L_E(t_1, t_2) = con \cdot \tau$ if $t_1 \leq t_2$ and $env \cdot \tau$ if $t_1 > t_2$.
  - for $t \in \mathbb{R}_{\geq 0}$, $L_E(t, env \cdot t) = env \cdot \tau$.
  - for $t \in \mathbb{R}_{\geq 0}$ and $u \in Act_U$, $L_E(t, u) = u$.
  - for $c \in Act_C$ and $a \in Act_U \cup \mathbb{R}_{\geq 0}$, $L_E(c, a) = c$.

Remark 6. Observe that the way we have defined hybrid games, it is possible for the environment (or controller) to stall, by repeatedly picking a time transition of shorter and shorter durations, resulting in zeno behavior. We will assume that such zeno behavior is eliminated using an appropriate winning condition. More precisely, we will assume that plays with an infinite sequence of consecutive time transitions labelled $con.\tau$ are won by the environment, and those with an infinite sequence of (not necessarily consecutive) time transitions labelled $env.\tau$ since the last discrete transition, are won by the controller. Please note that these simple fairness objectives can be expressed in a logic like LTL.

Time abstract transition system. We also associate a transition system called *time abstract transition system TATS* with the hybrid game which abstracts away the exact time elapsed during a continuous transition. We will use a new action *time* to represent the abstracted time. Formally, the *TATS* corresponding to the hybrid game $\mathcal{H} = (Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$ is given by $time\text{-}abstract(\mathcal{H}) = (Q, \Sigma_U, \Sigma_Q, \Sigma_E, \rightarrow, L_Q, L_E)$, where:

- $Q = Loc \times Cont$,
- $\Sigma_U = Act_C \cup Act_U \cup (\{con, env\} \cdot time)$,
- $\Sigma_Q = Labels$,
- $\Sigma_E = Act_C \cup Act_U \cup (\{con, env\} \cdot \{\tau\})$,
- $\rightarrow$ is defined as:

   - for $a \in \{con, env\}$ and $t \in \mathbb{R}_{\geq 0}$ such that $(l, x) \xrightarrow{t}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{a.time} (l', x')$.
   - For $a \in Act_C \cup Act_U$ such that $(l, x) \xrightarrow{a}_{\mathcal{H}} (l', x')$, $(l, x) \xrightarrow{a} (l', x')$.
- $L_Q(q, x) = Lfunc(q)$ for all $q \in Q$.
- $L_E(a) = a' \cdot \tau$ if $a = a' \cdot time$, $L_E(a) = a$ otherwise.

Control problem for hybrid games. The control problem for hybrid games is a pair $(\mathcal{H}, \mathcal{W})$ where $\mathcal{H}$ is a hybrid game and $\mathcal{W}$ is a winning condition on the state and transition labels of $game(\mathcal{H})$, and asks to find the set of all states in $game(\mathcal{H})$ from which there is a winning strategy with respect to $\mathcal{W}$. The controller synthesis problem asks to construct a winning strategy for each winning state.

Consistent hybrid game. We say that a hybrid game $\mathcal{H}$ is *consistent* if for all $t_1 < t_2$ and for all $(l, x) \in Loc \times Cont$, $(l, x) \xrightarrow{t_1} (l, x_1)$ and $(l, x) \xrightarrow{t_2} (l, x_2)$ implies $(l, x_1) \xrightarrow{t_2 - t_1} (l, x_2)$. This condition says that if starting from x one can reach x_1 at some time and x_2 at a later time, then one should also be able to start at x_1 and reach x_2 at a later time.

Total order on the bisimulation of a TATS. Let $\simeq$ be a bisimulation on $time\text{-}abstract(\mathcal{H})$ which is also an equivalence relation. Let $P \in \Pi_{\simeq}$. We define $succ_{\simeq}(P)$ to be the set of all classes $P' \in \Pi_{\simeq}$ such that $p \xrightarrow{a \cdot time} p'$ for some $p \in P, p' \in P'$ and $a \in \{con, env\}$. We then define a binary relation $\preceq_{\simeq}$ on $\Pi_{\simeq}$ as follows. $P_1 \preceq_{\simeq} P_2$ if either $P_1 = P_2$ or $P_2 \in succ_{\simeq}(P_1)$. We say a bisimulation $\simeq$ on $time\text{-}abstract(\mathcal{H})$ is *totally ordered* if $\simeq$ is an equivalence relation and for every $P \in \Pi_{\simeq}$, $(succ_{\simeq}(P), \preceq_{\simeq})$ is totally ordered. We will also say $P \prec_{\simeq} Q$ if $P \neq Q$ and $P \preceq_{\simeq} Q$. We will drop the subscript $\simeq$ from $succ_{\simeq}$, $\prec_{\simeq}$ and $\preceq_{\simeq}$ when it is clear from the context.

Next we relate a bisimulation on $time\text{-}abstract(\mathcal{H})$ to one on $game(\mathcal{H})$.

Lemma 3. *Let $\mathcal{H}$ be a consistent hybrid game. Let $\simeq$ be a bisimulation on its TATS $time\text{-}abstract(\mathcal{H})$ which is totally ordered. Then $\simeq$ is also a bisimulation on $game(\mathcal{H})$.*

Proof. Let $game(\mathcal{H}) = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \rightarrow, L_Q, L_E)$, and $time\text{-}abstract(\mathcal{H}) = (Q, \Sigma'_U, \Sigma_Q, \Sigma_E, \rightarrow', L_Q, L'_E)$. Suppose $q_1 \simeq q_2$. We need to show that for any controllable action c_1 from q_1 (or q_2) there is a controllable action c_2 from q_2 (or q_1) such that no matter which uncontrollable action the environment takes from q_2 (from q_1), there is a corresponding uncontrollable action from q_1 (from q_2) such that future behaviors are the "same". We will only consider the case of transitions out of q_1 being mimicked by q_2; the symmetric case of transitions out of q_2 being mimicked by q_1 is similar and skipped.

Let us first consider the case when the controller chooses a non-time action $c \in Act_C$ from q_1. Since $q_1 \simeq q_2$ and c is enabled at q_1, c is also enabled at q_2. Suppose the environment chooses an uncontrollable action u or a time t_2 from q_2. The resulting state q'_2 is such that $q_2 \xrightarrow{c}_{\mathcal{H}} q'_2$. Since $q_1 \simeq q_2$, there exists $q_1 \xrightarrow{c}_{\mathcal{H}} q'_1$ such $q'_1 \simeq q'_2$. Hence if the environment chooses u or a time t_1 from q_1, then the resulting state q'_1 is bisimilar to q'_2.

The main challenge in proving this lemma is in handling the time actions. Suppose the controller chooses a time $t_1 \in \mathbb{R}_{\geq 0}$ enabled at q_1 in $game(\mathcal{H})$. Let q'_1 be the unique state such that $q_1 \xrightarrow{t_1}_{\mathcal{H}} q'_1$. Therefore $q_1 \xrightarrow{con \cdot time}{}' \; q'_1$. Then since $q_1 \simeq q_2$, there exists q'_2 such that $q_2 \xrightarrow{con \cdot time}{}' \; q'_2$ and $q'_1 \simeq q'_2$. This implies that there exists t_2 such that $q_2 \xrightarrow{t_2}_{\mathcal{H}} q'_2$. Therefore t_2 is enabled at q_2 in $game(\mathcal{H})$. The controllable action from q_2 corresponding to t_1 from q_1 is t_2.

Now we need to show that for every uncontrollable transition the environment selects at q_2, we can find one for q_1 with the same label such that they result in equivalent states. Suppose the environment chooses an uncontrollable action from q_2, then it is easy to see that the same uncontrollable action can be taken from q_1 and the resulting behaviors are the same. Suppose the environment chooses t'_2 (or $env \cdot t_2$) from q_2 to q''_2; there are two cases to consider, namely, either $q'_2 \simeq q''_2$ or $q_2 \not\simeq q''_2$.

Case $q'_2 \simeq q''_2$: now if $t_2 \leq t'_2$ then $q_2 \xrightarrow{(t_2, t'_2)} q'_2$ with $L_E(t_2, t'_2) = con \cdot \tau$. In this case we let $t'_1 = t_1$, and so $q_1 \xrightarrow{(t_1, t'_1)} q'_1$ with $L_E(t_1, t'_1) = con \cdot \tau$ as well. On the other hand, if $t'_2 < t_2$ or the environment choose $env \cdot t_2$, then the label of the resulting transition is $env \cdot \tau$. Therefore from q_1 we consider the action $env \cdot t_1$, which also results in a transition with label $env \cdot \tau$.

Case $q'_2 \not\simeq q''_2$: now since $\simeq$ is a bisimulation on $time\text{-}abstract(\mathcal{H})$ there is a t'_1 such that $q_1 \xrightarrow{t'_1}_{\mathcal{H}} q''_1$ and $q''_1 \simeq q''_2$. Further $q''_1 \not\simeq q'_1$, otherwise $q''_1 \simeq q'_1$, $q'_1 \simeq q'_2$ and $q''_1 \simeq q''_2$ would imply $q'_2 \simeq q''_2$, a contradiction. This also implies that $t_1 \neq t'_1$. Observe that if we prove that $t'_1 < t_1$ iff $t'_2 < t_2$, then the transition (t_1, t'_1) from q_1 exactly mimics the transition (t_2, t'_2) from q_2. Suppose $t'_2 < t_2$. We will show that $t'_1 < t_1$. The other direction is similar. Since $q_2 \xrightarrow{t_2}_{\mathcal{H}} q'_2$ and $q_2 \xrightarrow{t'_2}_{\mathcal{H}} q''_2$, consistency of $\mathcal{H}$ implies that $q''_2 \xrightarrow{t_2 - t'_2}_{\mathcal{H}} q'_2$. Therefore, $[q''_2]_{\simeq} \prec [q'_2]_{\simeq}$. Hence $[q''_1]_{\simeq} \prec [q'_1]_{\simeq}$. Suppose for the sake of contradiction that $t'_1 \geq t_1$. We have seen that $t'_1 \neq t_1$, hence $t'_1 > t_1$. We can deduce by an argument similar to the above that $[q'_1]_{\simeq} \prec [q''_1]_{\simeq}$. This contradicts the fact that $\preceq_{\simeq}$ is a total order. $\square$

Our next goal is to solve the controller synthesis problem. Towards this we define a quotient game graph of $game(\mathcal{H})$, which has the property that a winning strategy for this graph can be lifted to a winning strategy for $game(\mathcal{H})$. Hence if the quotient game graph is finite, we may be able to solve the controller synthesis problem for $game(\mathcal{H})$.

Quotient game graph corresponding to game($\mathcal{H}$). Let $\mathcal{H}$ be a hybrid game with controllable actions Act_C and uncontrollable actions Act_U. Let the game graph of $\mathcal{H}$ be $game(\mathcal{H}) = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \rightarrow, L_Q, L_E)$. Let $\simeq$ be a bisimulation on $game(\mathcal{H})$. We define $quo\text{-}game(\mathcal{H}) = (Q', \Sigma_C', \Sigma_U', \Sigma_Q{}', \Sigma_E', \rightarrow', L_Q', L_E')$, where:

- $Q' = \Pi_{\simeq}$, is the set of equivalence classes of $\simeq$.
- $\Sigma_C' = Act_C \cup Q'$.
- $\Sigma_U' = Act_U \cup Q' \cup (\{env\} \cdot Q')$.
- $\Sigma_Q{}' = \Sigma_Q$.
- $\Sigma_E' = \Sigma_E$.
- $\rightarrow'$ is defined as:
  - for $P_1, P_2 \in Q'$, $P \xrightarrow{(P_1,P_2)}{}' P'$, if $P_1, P_2 \in succ(P)$ and either $P' = P_1$ and $P_1 \preceq P_2$ or $P' = P_2$ and $P_2 \preceq P_1$.
  - for $P_1, P_2 \in Q'$, $P \xrightarrow{(P_1,env\cdot P_2)}{}' P'$, if $P_1, P_2 \in succ(P)$, and $P_1 = P_2 = P'$.
  - for $P_1 \in Q'$ and $u_1 \in Act_U$, $P \xrightarrow{(P_1,u_1)}{}' P'$, if $P_1 \in succ(P)$ and there exists $p \in P$ and $p' \in P'$ such that $p \xrightarrow{u_1}_{\mathcal{H}} p'$.
  - for $c_1 \in Act_C$ and $u_1 \in \Sigma_U'$, $P \xrightarrow{(c_1,u_1)}{}' P'$, if there exists $p \in P$ and $p' \in P'$ such that $p \xrightarrow{c_1}_{\mathcal{H}} p'$, and either $u_1 \in Act_U$ and is enabled at some $p \in P$, or $u_1 = P''$ or $env \cdot P''$ for some $P'' \in succ(P)$.
- $L_Q'(P) = L_Q(p)$ for some (all) $p \in P$.
- L_E' is defined as follows:
  - for $P_1, P_2 \in Q'$, $L_E'(P_1, P_2) = a.\tau$, where $a = env$ if $P_2 \prec P_1$, and $a = con$ otherwise.
  - for $P_1, P_2 \in Q'$, $L_E'(P_1, env \cdot P_2) = env \cdot \tau$.
  - for $c_1 \in Act_C$ and $u_1 \in \Sigma_U'$, $L_E'(c_1, u_1) = c_1$.
  - for $u_1 \in Act_U$, $L_E'(P_1, u_1) = u_1$.

Consider the relation R between the states of $game(\mathcal{H})$ and $quo\text{-}game(\mathcal{H})$ given by $R(p, P)$ iff $p \in P$. The following proposition relates the game graph with its quotient.

Proposition 4. *R is a bisimulation on* $(game(\mathcal{H}), quo\text{-}game(\mathcal{H}))$.

Hence, from Proposition 2, there is a winning strategy from a state p of $game(\mathcal{H})$ iff there is a winning strategy from a state $[p]_{\simeq}$ of $quo\text{-}game(\mathcal{H})$. Next we will explicitly define these strategies.

For every (finite or infinite) run of $game(\mathcal{H})$ there is a corresponding run of $quo\text{-}game(\mathcal{H})$ such that their traces are the same, and vice versa. Let $\sigma \in Runs(game(\mathcal{H}))$ be $p_0(c_1, u_1)p_1 \cdots$. Then the corresponding run $quo\text{-}run(\sigma) \in Runs(quo\text{-}game(\mathcal{H}))$ is given by $P_0(C_1, U_1)P_1 \cdots$, where $P_i = [p_i]_{\simeq}$, $C_i = c_i$ if $c_i \in Act_C$, otherwise if $c_i = t$ then $C_i = [q]_{\simeq}$ where q is such that $p_i \xrightarrow{t}_{\mathcal{H}} q$, and similarly $U_i = u_i$ if $u_i \in Act_U$, otherwise if $u_i = t$ or $env \cdot t$ then $U_i = [q]_{\simeq}$ or $(u, [q]_{\simeq})$, respectively such that $p_i \xrightarrow{t}_{\mathcal{H}} q$. Similarly let $\sigma' = P_0(C_1, U_1)P_1 \cdots$ be a run in $Runs(quo\text{-}game(()\mathcal{H}))$. Since P_0 is not empty, it is easy to see from the definition above that there is a run starting from some $p \in P_0$ whose trace is equivalent to σ'.

Given a strategy λ for $quo\text{-}game(\mathcal{H})$ we can construct a strategy $unquo(\lambda)$ for $game(\mathcal{H})$ as follows. We define $unquo(\lambda)(\sigma)$ as follows. Let $\lambda(quo\text{-}run(\sigma)) = C$. If $C \in Act_C$, then $unquo(\lambda)(\sigma) = C$, otherwise if $C \in Q'$, then $unquo(\lambda)(\sigma) = t$, for some $t \in \mathbb{R}_{\geq 0}$ such that there exists $p' \in C$ with $last(\sigma) \xrightarrow{t}_{\mathcal{H}} p'$. Also given a strategy λ for $\mathcal{H}$, we can define a strategy $quo(\lambda)$ analogously.

The following lemma summarizes the relationship between λ, $unquo(\lambda)$ and $quo\text{-}run(\lambda)$.

Lemma 5. *Let p be a state of $game(\mathcal{H})$ and P a state of $quo\text{-}game(\mathcal{H})$. Let $p \in P$. Then p is winning for $game(\mathcal{H})$ with respect to a winning condition $\mathcal{W}$ if and only if P is winning for $quo\text{-}game(\mathcal{H})$ with respect to the winning condition $\mathcal{W}$. Further given a strategy λ which is winning for p, $quo(\lambda)$ is winning for P. Similarly, if λ is a winning strategy for P, then $unquo(\lambda)$ is a winning strategy for p.*

Proof. Routine and skipped. $\square$

5. Decidability of control

In this section we solve the control problem for some classes of hybrid games. We consider two classes, namely, STORMED hybrid games and o-minimal hybrid games. Let us fix a hybrid game $\mathcal{H} = (Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$ for the rest of this section.

5.1. Hybrid game specifications

Definition 3. A hybrid game $\mathcal{H}$ is said to be *o-minimally* definable if the invariants, flow function, guards, resets and the state labelling functions are definable in some o-minimal theory.

Definition 4. An *o-minimal hybrid game* is an o-minimally defined hybrid game with strong resets, i.e., for every edge $e \in Edge$ of the hybrid game, $Reset(e) = Cont_1 \times Cont_2$ for some $Cont_1, Cont_2 \subseteq Cont$.

Definition 5. A *STORMED hybrid game* is a hybrid game such that there exists a vector ϕ which satisfies the following conditions:

S *Guards are* separable. For all $e_1, e_2 \in Edge$ such that $e_1 \neq e_2$, $dist(Guard(e_1), Guard(e_2)) = \inf\{||x-y|| \mid x \in Guard(e_1), y \in Guard(e_2)\} \geq d_{min}$ for some $d_{\min} > 0$.
T *The flow is time-independent and satisfies the semi-group property (*TISG*).* For every state $(l, x) \in Loc \times Cont$, $Flow(l, x)$ is continuous and $Flow(l, x)(0) = x$, and for all $t, t' \in \mathbb{R}_{\geq 0}$, $Flow(l, x)(t + t') = Flow(l, Flow(l, x)(t))(t')$.
O *o-minimally definable.*
R R*esets are monotonic along vector* ϕ. There exist $\epsilon, \zeta > 0$ such that for all edges $e = (l_1, a, l_2) \in Edge$ and $x_1, x_2 \in Cont$ such that $(x_1, x_2) \in Reset(e)$:
- if $l_1 = l_2$, then either $x_1 = x_2$ or $\phi \cdot (x_2 - x_1) \geq \zeta$,
- otherwise $\phi \cdot (x_2 - x_1) \geq \epsilon||x_2 - x_1||$.

M *Flows are* monotonic *along* ϕ. There exists $\epsilon > 0$ such that for all $l \in Loc$, $x \in Cont$ and $t, \tau \in \mathbb{R}_{\geq 0}$, $\phi \cdot (Flow(l, x)(t + \tau) - Flow(l, x)(t)) \geq \epsilon||Flow(l, x)(t + \tau) - Flow(l, x)(t)||$.
ED *Guards are* ends-delimited *along* ϕ. The set $\{\phi \cdot x \mid x \in Guard(e), e \in Edge\} \subseteq [b^-, b^+]$ for some $b^-, b^+ \in \mathbb{R}$.

STORMED hybrid games are based on STORMED hybrid systems. The constraints imposed by STORMED hybrid systems are realized in some physical systems as follows.

- Monotonicity can be associated with energy or time depletion, or in vehicle control problems, with non-decreasing trajectories.
- The Ends-Delimited property can be present as a deadline on the monotonic direction or a spatial confinement.
- Separability of guards represents infrequency in making control decisions, also based on location or time.
- TISC flows arise naturally, whereas o-minimality is not necessarily a common property, but can be used as an approximation most of the time. Linearization and other model reductions may also result in o-minimal realizations.

We have that STORMED systems have a bounded number of discrete transitions in any execution. This follows from the monotonicity conditions, separability of guards and the condition on ends-delimited. As a matter of fact, the bounded number of discrete transitions, together with a property of o-minimally defined systems is all we need to prove our results.

We believe that the STORMED specifications are natural specifications that enforce an upper bound on the number of discrete transitions of any execution of the system and for that we provide the following example.

5.2. An example

The system examined in this section was first analyzed in a slightly different original form in [29], and revisited in various forms in [24,23] and elsewhere. It defines an aircraft collision avoidance scheme in which an aircraft is to join the trajectory of another aircraft while maintaining a safe distance. The aircraft performs this joining procedure in order to either land or avoid collision in an air traffic congestion policy. In this example, as opposed to [23], only a small part of the procedure is checked for safety, but an exact system is used instead of an abstraction.

5.2.1. Description

The instantaneous locations of two aircraft are (x_1, y_1, θ_1) and (x_2, y_2, θ_2), with x_1, y_1 and x_2, y_2 are the Cartesian locations of the two aircraft on the plane and with θ_1, θ_2 being the counterclockwise angle of their heading with the x axis. The trajectories of the two aircraft are shown with dotted lines in Fig. 1. The motion of the first aircraft does not change. It follows a straight path from position $(x_1, y_1, \theta_1) = (-d_2, 0, 0)$ with velocity v_1 towards the runway. The second aircraft, on the other hand, approaches from $(x_2, y_2, \theta_2) = (-r, -(r + d_2), \pi/2)$, with initial velocity v_2. When $y_2 = -d_2 + r$ the first airplane's position is $x_1 = d_1$. After that point and before it reaches the state where $y_2 = -r$, the second aircraft can choose to start decelerating at a constant rate a_d, accelerating at a constant rate a_a or not change its velocity at all. The deceleration/acceleration or lack thereof will take place until $y_2 = -r$, at which point the second aircraft will continue with its acquired velocity onto the quarter circle path turning into the runway fix on the x axis. The requirement is that the aircraft arrive at a safe distance denoted d_s on the x axis on their final approach. From there it is assumed that they can safely regulate the rest of their landing approach. Clearly, the system can be modeled as a hybrid automaton with three discrete states.

In [23] the authors verify the safe-distance requirement for all times, by abstracting the system to one with linear flows that has aircraft 2 make two instantaneous 45°-clockwise turns in order to merge with aircraft 1 on its runway fix. This is in order to avoid trigonometric functions and be able to use quantifier elimination as they try to verify the safe distance requirement at all times. The abstraction is turned to an over-approximation of the original system by using differential inclusions. The approximation is shown in the right diagram of Fig. 1. In this section a quantifier-free formula will be derived on the parameters for the specification of the system.

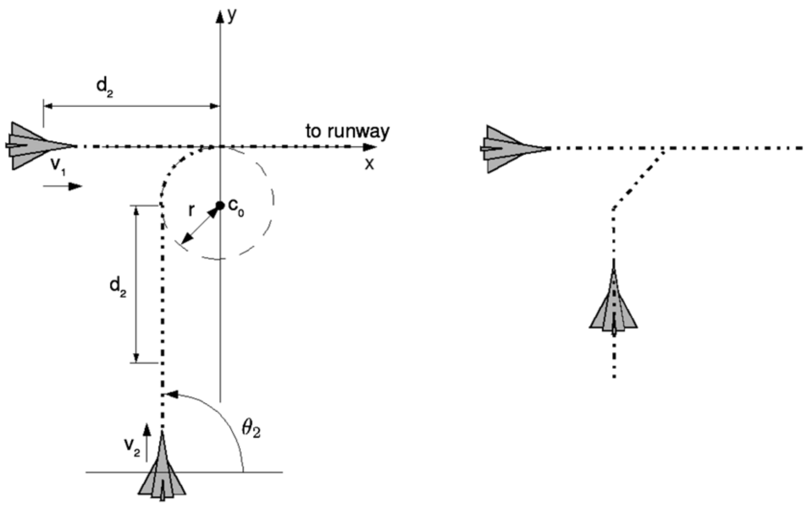

Fig. 1. Right: aircraft 1 and 2 are shown with their trajectories and velocities as indicated. Left: the abstraction in [23].

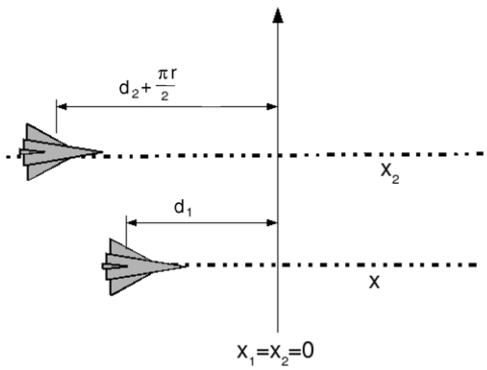

Fig. 2. An equivalent system representation that is STORMED.

One can observe that, since the safety distance requirement is only on the final leg of the route of aircraft 2 and its angular velocity on the circular segment is constant, we can eliminate the y_2, θ_2 components and use a different discrete location to contain the curved path on a straight line of equal length. The remaining components for which we need to verify safety are x_1, x_2 and $\dot{x}_2$ only! Fig. 2 shows how the trigonometric functions can be eliminated.

5.2.2. Formal game definition

The game definition is as follows:

- $Loc = \{\text{initstate, faststate, slowstate, steadystate, turnstate, finalstate}\}$ and $Cont = \mathbb{R}^3$, i.e. $(x_1, x_2, \dot{x}_2)$.
- The controlled actions are $Act_C = \{\text{actfast, actslow, actsteady}\}$, all from the initstate state to the faststate, slowstate, and steadystate states. The uncontrollable actions are $Act_U = \{\text{actturn, actapproach}\}$ and correspond to the rest of the edges.
- State labels are $Labels = \{\text{initial, safe, collision}\}$, where initial is the label for the states in initstate where $x_1 = -d_1 \wedge x_2 = -(d_2 + \frac{\pi r}{2}) \wedge \dot{x}_2 = v_2$, safe is the label for all the states in finalstate where $|x_1 - x_2| > d_s \wedge x_2 = 0$, and collision is the label for all the rest of the states.
- The edge set is $Edge = \{$(initstate, actfast, faststate), (initstate, actslow, slowstate), (initstate, actsteady, steadystate), (slowstate, actturn, turnstate), (faststate, actturn, turnstate), (steadystate, actturn, turnstate), (turnstate, actapproach, finalstate) $\}$.
- The invariants are given by
  $Inv(\text{initstate}) = Inv(\text{faststate}) = Inv(\text{slowstate}) = Inv(\text{steadystate}) = x_2 \le -r$, $Inv(\text{turnstate}) = x_2 < 0$ and $Inv(\text{finalstate}) = x_2$.
- All the guards leading to states slowstate, faststate, steadystate are $-(d_2 + \frac{\pi r}{2}) < x_2 < -\frac{\pi r}{2}$. The guards to all the transitions to the turnstate state are $x_2 = \frac{\pi r}{2}$ and to the finalstate state is $x_2 = 0$.

- The flows are:

$$Flow_{\mathsf{initstate},(x_1,x_2,v_2)}(t) = (x_1 + v_1 t, x_2 + v_2 t, v_2)$$
$$Flow_{\mathsf{slowstate},(x_1,x_2,v_2)}(t) = \left(x_1 + v_1 t, x_2 + v_2 t + \frac{1}{2}a_d t^2, v_2 + a_d t\right)$$
$$Flow_{\mathsf{faststate},(x_1,x_2,v_2)}(t) = \left(x_1 + v_1 t, x_2 + v_2 t + \frac{1}{2}a_a t^2, v_2 + a_d t\right)$$
$$Flow_{\mathsf{steadystate},(x_1,x_2,v_2)}(t) = (x_1 + v_1 t, x_2 + v_2 t, v_2)$$
$$Flow_{\mathsf{turnstate},(x_1,x_2,v_2)}(t) = (x_1 + v_1 t, x_2 + v_2 t, v_2)$$
$$Flow_{\mathsf{finalstate},(x_1,x_2,v_2)}(t) = (x_1 + v_1 t, x_2 + v_2 t, v_2)$$

Assuming that the system parameters v_2, a_d are such that the possible deceleration while in slowstate will not bring aircraft 2 below a stall velocity, which can be imposed by an extra invariant, the system flows are monotonic. This can be imposed by an extra trivial invariant. The guards are delimited by $x_2 = 0$ and separable by $\min\{d_2, \frac{\pi r}{2}\}$. Everything is defined in the decidable theory $(\mathbb{R}, 1, 0, +, \cdot, <)$; therefore, the system is a STORMED hybrid game and, as we will see in the sequel, a control problem for an LTL winning condition on such a game is decidable.

5.3. Decidability

Theorem 6 *([20,11,33]).* *STORMED hybrid games and o-minimal hybrid games have finite bisimulations of their time-abstract transition systems which are definable in their underlying o-minimal theory. The finite bisimulation can be effectively constructed when the underlying theory is decidable.*

A finite bisimulation is definable in a theory if its equivalence classes are definable in the theory.

Lemma 7. *Hybrid games with TISG flows are consistent.*

Proof. Follows from the definition of *TISG*. □

Lemma 8. *Let $\mathcal{H}$ be an o-minimally defined hybrid game satisfying the TISG property, and let $\simeq$ be a finite bisimulation of its TATS definable in the underlying o-minimal theory. Then $\simeq$ is a totally ordered bisimulation on time-abstract$(\mathcal{H})$.*

Proof. We need to show that for each $P \in \Pi_{\simeq}$, $(succ(P), \preceq)$ is totally ordered. Note that $\preceq$ is reflexive by definition. Let $P_1 \preceq P_2$ and $P_2 \preceq P_3$. To show that $\preceq$ is transitive, we need to show that $P_1 \preceq P_3$. Suppose $P_1 \neq P_2$ and $P_2 \neq P_3$ (otherwise we are done). Let $p_1 \in P_1$. There exist $p_2 \in P_2$ and $p_3 \in P_3$ such that $p_1 \xrightarrow{a\cdot time} p_2$ and $p_2 \xrightarrow{a\cdot time} p_2$. We have from the *TISG* property that the hybrid game is consistent. Hence $p_1 \xrightarrow{a\cdot time} p_3$, which implies $P_1 \preceq P_3$.

Next we need to show that $\preceq$ is anti-symmetric. Let $P_1 \preceq P_2$ and $P_2 \preceq P_1$. Suppose $P_1 \neq P_2$. This violates the o-minimality of $\mathcal{H}$. We will describe the intuition behind the proof here; details can be found in [11]. From every state in P_1, there exists an infinite run that alternates between P_1 and P_2. We can define in the o-minimal theory the set of all times at which such an infinite run is in the equivalence class P_1. This set is not a finite union of intervals, which contradicts the o-minimality. Hence, $\preceq$ is a partial order.

Further, $\preceq$ is totally ordered. To see this, let P_1 and P_2 belong to *succ*(P). Since $P \preceq P_1$ and $P \preceq P_2$, for every $p \in P$, there exist t_1 and t_2 in $\mathbb{R}_{\geq 0}$ such that $p \xrightarrow{t_1}_{\mathcal{H}} p_1$ and $p \xrightarrow{t_2}_{\mathcal{H}} p_2$ for some $p_1 \in P_1$ and $p_2 \in P_2$. Without loss of generality, assume $t_1 \leq t_2$. It follows from the consistency of $\mathcal{H}$, that $p_1 \xrightarrow{t_2 - t_1}_{\mathcal{H}} p_2$, and hence $P_1 \preceq P_2$. □

Theorem 9. *Given a STORMED hybrid game $\mathcal{H}$ and a winning condition $\mathcal{W}$ which is ω-regular, the control problem is decidable if the underlying o-minimal theory is decidable. The controller synthesis problem is also decidable.*

Proof. From Lemma 7, a STORMED hybrid game is consistent, from Theorem 6 it has a finite bisimulation $\simeq$ of its *TATS* which is definable, and from Lemma 8 the bisimulation $\simeq$ is totally ordered. Hence, if the underlying o-minimal theory is decidable, we can construct *quo-game*$(\mathcal{H})$ and solve the control problem on it. Then it follows from Lemma 5 and the control problem is decidable for $\mathcal{H}$. Also, since we can synthesize a winning strategy for *quo-game*$(\mathcal{H})$ from a winning state, it follows from the decidability of the theory and Lemma 5 that we can lift it to synthesize a winning strategy for $\mathcal{H}$ from the corresponding states in $\mathcal{H}$. □

Along the same lines, we have the following.

Theorem 10. *Given an o-minimal hybrid game $\mathcal{H}$ with TISG flows and a winning condition $\mathcal{W}$ which is ω-regular, the control problem is decidable if the underlying o-minimal theory is decidable. The controller synthesis problem is also decidable.* [2]

Our results for o-minimal hybrid games are stronger than the ones in [6] in that we solve the control problem with respect to any ω-regular winning conditions as opposed to just reachability as in [6].

2 The flows considered in [6] are not *TISG*, but have unique suffixes with respect to the partition, we can extend Lemma 3 to obtain the same results.

6. Weighted hybrid games

Now we examine weighted games, which have costs on transitions. The goal is to minimize the accumulated costs while meeting certain qualitative objectives. We will first consider optimal controllers that satisfy given reachability objectives. Then we examine the problem of verifying hybrid systems of the same specifications.

6.1. Weighted hybrid games and optimal-cost reachability problem

A *weighted hybrid game* is a pair $(\mathcal{H}, Cost)$, where $\mathcal{H}$ is a hybrid game and *Cost* is a non-negative and time-non-decreasing function $Cost : Loc \times Cont \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, i.e., $Cost((l, x), t) \geq 0$ for all t, and $Cost((l, x), t_1) \geq Cost((l, x), t_2)$ if $t_1 \geq t_2$. The cost function also satisfies the following additive property: $Cost((l, x), t_1 + t_2) = Cost((l, x), t_1) + Cost((l, Flow(l, x)(t_1)), t_2)$.

Given a weighted hybrid game (*WHG*) $(\mathcal{H}, Cost)$, where $\mathcal{H} = (Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$, the semantics is given by a weighted game graph. A *weighted game graph* (*WGG*) $\mathcal{J}$ is a pair $(\mathcal{G}, Cost)$, where $\mathcal{G} = (Q, \Sigma_C, \Sigma_U, \Sigma_Q, \Sigma_E, \to, L_Q, L_E)$ is a game graph and $Cost : (Q \times \Sigma_C \times \Sigma_U \times Q) \to \mathbb{R}_{\geq 0}$ is a cost function on its transitions. The *WGG* associated with the *WHG* $(\mathcal{H}, Cost)$ is $(\mathcal{G}, Cost')$ where $\mathcal{G} = game(\mathcal{H})$ and $Cost'$ is the function that assigns a weight to the transitions depending on how long the system stays in a particular location. The cost of taking a discrete transition is taken to be 0. More precisely, $Cost'$ on $game(\mathcal{H})$ is defined as follows. Recall that in $game(\mathcal{H})$ $\Sigma_C = Act_C \cup \mathbb{R}$ and $\Sigma_U = Act_U \cup \mathbb{R} \cup (\{env\} \cdot \mathbb{R})$. For $c \in \Sigma_C$ and $u \in \Sigma_U$,

$$Cost'(q, (c, u), q') = \begin{cases} 0 & \text{if } c \in Act_C \text{ or } u \in Act_U \\ Cost(q, \min(c, u)) & \text{if } c \in \mathbb{R}_{\geq 0} \text{ and } u \in \mathbb{R}_{\geq 0} \\ Cost(q, c) & \text{if } c \in \mathbb{R}_{\geq 0} \text{ and } u = env \cdot c. \end{cases}$$

In this section, we will consider the problem of synthesizing optimal cost controllers for reachability objectives. We are given a set of states $Goal \subseteq Loc \times Cont$ which the controller wants to reach. We want to find a strategy which will eventually reach the goal and the worst cost of reaching the goal is minimized. The environment can often avoid reaching the goal by selecting smaller and smaller time steps. We assume that the zeno behavior is eliminated through choosing appropriate winning conditions. We say that the environment *stalls* a play if there are an infinite number of time transitions labelled $env \cdot \tau$ since the last discrete transition. Thus, in the case of reachability objectives we mean that the controller wins if either the play reaches the goal or the environment stalls the play. Otherwise the environment wins.

We now define optimal-cost reachability problem formally using the weighted game graph. Let $(\mathcal{H}, Cost)$ be a *WHG* and $(\mathcal{G}, Cost')$ its *WGG*. Let $Goal \subseteq Q$ be a set of states of $\mathcal{G}$ which we want to reach. Towards this, we define the cost of a run to be the sum of the costs of its transitions till the goal is reached. Given a run $\rho = q_0(c_1, u_1)q_1(c_2, u_2) \ldots \in Runs(\mathcal{G})$ with q_n the first state contained in *Goal*, $Cost(\rho) = \sum_{i=1}^{n} Cost(q_{i-1}, c_i, u_i, q_i)$. If ρ does not contain a state from *Goal*, then its cost is 0. The cost of a strategy λ is the supremum of the cost of all the runs consistent with it. Formally, the *cost of a strategy* λ from a state q is $Cost(\lambda, q) = \sup_\rho\{Cost(\rho) | first(\rho) = q, \rho \text{ is consistent with } \lambda\}$. A run is winning if either it reaches the *Goal* at some time or there are infinitely many consecutive transitions labelled $env \cdot \tau$. A strategy λ is winning for q, if every run starting from q consistent with it is winning. Finally, the *optimal-cost* from a state q is defined as: $Cost_{opt}(q) = \inf_\lambda\{Cost(\lambda, q) | \lambda \text{ is a winning strategy}\}$.

We now define the following problems on weighted hybrid games.

Definition 6 (*Optimal-cost Reachability Problem*)**.** Given a weighted hybrid game $(\mathcal{H}, Cost)$, a set of states *Goal* of its game graph $(\mathcal{G}, Cost')$, a constant $c \in \mathbb{R}_{\geq 0}$, and a state q of the game graph, the optimal-cost reachability problem is to decide if there exists a winning strategy λ from q such that $Cost'(\lambda, q) \leq c$. The optimal cost of reaching the *Goal* is given by $Cost_{opt}(q)$.

6.2. Weighted STORMED hybrid games and optimal reachability

We now turn to deciding optimal-cost reachability problem for STORMED hybrid games. Our decidability result for optimal controllers relies on the observation that in reachability games, we can focus our attention on games between a controller that is *time consistent and conservative* and an environment that is *conservative*. Plays between such a controller and the environment alternate between a time step (i.e., one labelled by $con \cdot \tau$ or $env \cdot \tau$, depending on who won) and a discrete action. Next, since any STORMED execution has a bounded number of discrete steps, this allows us to focus on bounded strategies when synthesizing optimal controllers, which we show can be effectively constructed. Thus, before presenting the technical details of our decidability result, we define what we mean by time consistent and conservative. For the rest of this section, we fix a STORMED hybrid game $\mathcal{H} = (Loc, Act_C, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$, with cost function *Cost*, that defines a weighted game graph $(\mathcal{G}, Cost')$.

Time consistent and conservative controllers. A controller strategy $\lambda : Runs_{fin}(\mathcal{G}) \to \Sigma_C$ is said to be time consistent and conservative if the following conditions hold.

Conservative On any run σ such that $trace(\sigma) = \rho(con \cdot \tau)$, $\lambda(\sigma) \in Act_C$. In other words, λ will pick discrete controllable action if the last transition was a time step that it won.

Time consistent On any runs σ_1 and σ_2 such that $\lambda(\sigma_1) = t$ and $\sigma_2 = \sigma_1(t, t')q'$ for some q' and $t' < t$, then $\lambda(\sigma_2) = t - t'$. In other words, if σ_2 is an extension of σ_1 consistent with λ in which the last step was a time transition which the environment won, then the controller picks a time step that is consistent with its previous decision.

Conservative environment plays. In a run σ, we will say that the environment played conservatively, if in $trace(\sigma)$ every transition labelled $env \cdot \tau$ is followed by an edge in which the environment choose a discrete action (i.e., the transition contains a symbol from Act_U). Thus, in such plays, the environment does not pick a time transition if it won the previous time transition.

We are now ready to present our main technical observations. We first show that if there is a winning strategy (for the controller) with cost c, there is a time consistent, conservative winning strategy with cost at most c. More precisely,

Lemma 11. *Let λ be a winning strategy from state q. Then there is a time consistent, conservative winning strategy λ' from q such that $Cost(\lambda', q) \le Cost(\lambda, q)$.*

Proof. Let λ be a winning strategy for q with respect to *Goal*. We will construct λ' inductively. More precisely, we will build a sequence of functions λ'_i such that λ'_i will be defined on all runs of length at most i consistent with λ'_{i-1} and not containing *Goal*. Further λ'_i will agree with λ'_{i-1} on all runs of length at most $i-1$. The strategy λ' itself will be the limit of this sequence.

The strategy λ' that we construct will "restrict" the possible plays allowed by λ. Therefore, in order for us to inductively define λ' (and later prove properties about it), we will also need to inductively define functions $f_0, f_1, f_2, \ldots$ such that f_i will map runs of length i consistent with λ'_{i-1} to runs (of unknown length) consistent with λ.

Inductive invariant. We will ensure that following conditions hold during our inductive construction of λ_i and f_{i+1}. We will assume σ is a run of length i consistent with λ'_{i-1} and the only possible state in *Goal* is $last(\sigma)$, and σ' is a run of length $i-1$ consistent with λ'_{i-2} and not containing the goal.

1. $last(\sigma) = last(f_i(\sigma))$ and $f_i(\sigma)$ does not contain a goal state except possibly for $last(f_i(\sigma))$.
2. If σ' is a prefix of σ of length j, such that the label of the last transition in σ' is in $Act_C \cup Act_U \cup \{con \cdot \tau\}$ then $f_j(\sigma')$ is a prefix of $f_i(\sigma)$.
3. $f_i(\sigma)$ is consistent with λ.
4. $Cost(\sigma) = Cost(f_i(\sigma))$.
5. If $\lambda(f_i(\sigma))$ is a time step t and σ does not visit *Goal* then the last label in $trace(\sigma)$ is not $con \cdot \tau$.
6. If the last transition of σ' is labelled $con \cdot \tau$ and σ' does not contain a state from *Goal*, then $\lambda_{i-1}(\sigma')$ is not t.
7. If the last transition of σ' is (t_1, t_2) which the environment won, then $\lambda'_{i-1}(\sigma') = t$ and $t = t_1 - t_2$.

Observe that the last condition ensures that λ' will be time consistent. On the other hand, the second-to-last condition will ensure that λ' is conservative.

Having outlined the intuition behind the construction of λ', we will now present its formal definition. We will begin by first defining λ'_i using λ'_{i-1} and f_i, and then define f_{i+1} using λ'_i and f_i.

Let σ be a run of length $i \ge 0$ consistent with λ'_{i-1}. $\lambda'_i(\sigma)$ is defined based on the form of σ.

- If $\lambda(f_i(\sigma)) \in Act_C$, then $\lambda'_i(\sigma) = \lambda(f_i(\sigma)))$.
- If $\lambda(f_i(\sigma))$ is some time t_0 and the last edge label of $trace(\sigma)$ is not $env \cdot \tau$, then we do the following. Observe that in this case, the last edge label in σ cannot be $con \cdot \tau$, because of the invariant we maintain, and so must be in $Act_C \cup Act_U$. Let $\sigma_0 = f_i(\sigma)$. If $last(\sigma_0)$ is not in *Goal*, then let σ_1 be the run obtained by taking the transition (t_0, t_0) after σ_0. If $\lambda(\sigma_1)$ is a time t_1 and $last(\sigma_1)$ is not in *Goal*, then σ_2 is the run obtained by taking (t_1, t_1) from σ_1, and we repeat this process from σ_2. Thus in general, if $\lambda(\sigma_j)$ is a time t_j and $last(\sigma_j)$ is not in *Goal* then σ_{j+1} is obtained by taking (t_j, t_j) from σ_j. Observe that since λ is a winning strategy, this process cannot go on forever, otherwise it would give a result in a run consistent with λ (since σ_0 is consistent by induction hypothesis) which does not reach the goal and contains an infinite sequence of consecutive time transitions which is winning for the controller (and hence does not contain an infinite sequence of consecutive time transitions which is winning for the environment). Let σ_n be the first run such that $last(\sigma_n) \in Goal$. Then we define $\lambda'_i(\sigma) = \sum_{j=0}^{n-1} t_j$.
- Finally, if $\lambda(f_i(\sigma))$ is some time t_1 and the last edge label of $trace(\sigma)$ is $env \cdot \tau$, then we do the following. Let $\sigma = \sigma'(t, t')q'$ or $\sigma'(t, env \cdot t')$; thus, $\lambda'_{i-1}(\sigma') = t$ and $t' \le t$. Then, $\lambda'_i(\sigma') = t - t'$.

We will now present the formal definition of f. We will define $f_0(q) = q$. Inductively, we need to define f_{i+1} on runs σ of length $i+1$ that are consistent with λ'_i. f_{i+1} is defined as follows.

- Let $\sigma = \sigma'(c, u)q'$, where either $c \in Act_C$ and $u \in \Sigma_U$ or $u \in Act_U$ and $c \in \Sigma_C$. By the invariant that is maintained, $last(\sigma') = last(f_i(\sigma'))$, and so (c, u) is enabled in $last(f_i(\sigma'))$ and will go to the same state. Therefore, define $f_{i+1}(\sigma) = f_i(\sigma')(c, u)q'$.
- Let σ be a run where the last transition is (t_n, u'), where $u' \notin Act_U$; thus, u' is either t' or $env \cdot t_n$. Now, we can write σ as $\sigma'(c, u)q_0(t_1, t'_1)q_1\ (t_2, t'_2)q_2 \cdots\ (t_{n-1}, t'_{n-1})q_{n-1}\ (t_n, u')q_n$, where σ' is a prefix of length j, and either $c \in Act_C$ and $u \in \Sigma_U$ or $u \in Act_U$ and $c \in \Sigma_C$. (The analysis is similar if any of the t'_i is $env \cdot t_i$.) From item 6 of the invariant, we have $t_{n-1} \ge t'_{n-1}$. Further from item 7 of the invariant we have $t_n = t_{n-1} - t'_{n-1}$. Similarly $t_{n-1} = t_{n-2} - t'_{n-2}$. Continuing the argument we obtain $t_n = t_1 - \sum_{j=1}^{n-1} t'_i$.

Let $\sigma'' = \sigma'(c, u)q_0$ be of length j. Then $\lambda(f_j(\sigma''))$ is some time t and the label of the last transition is not $env \cdot \tau$. Hence from the definition of f_j, we have a sequence of runs $\sigma_0, \ldots, \sigma_k$ each consistent with λ such that (a) $\sigma_0 = f_{j+1}(\sigma'(c, u))$, (b) $\lambda(\sigma_i) = t''_{i+1}$, (c) $\sigma_{i+1} = \sigma_i(t''_{i+1}, t''_{i+1})q''_i$, and (d) $t_1 = \sum_{\ell=1}^{k} t''_\ell$. None of the σ_i except possibly for σ_k contains a goal state.

Let $t_{\text{sum}} = \sum_{\ell=1}^{n-1} t'_i + x$ where $x = t_n$ if $u' = env \cdot \tau$ and $x = \min(t_n, t'_n)$ if $u' = t'_n$. Since $t_{\text{sum}} \le t_1$, $t_{\text{sum}} \le \sum_{\ell=1}^{k} t''_\ell$. Hence either $t_{\text{sum}} = \sum_{\ell=1}^{k} t''_\ell$ or there is some $m < k$ such that $\sum_{\ell=1}^{m} t''_\ell \le t_{\text{sum}} < \sum_{\ell=1}^{m+1} t''_\ell$. In the case when $t_{\text{sum}} = \sum_{\ell=1}^{k} t''_\ell$, define $f_{i+1}(\sigma) = \sigma_k$. On the other hand, if $\sum_{\ell=1}^{m} t''_\ell \le t_{\text{sum}} < \sum_{\ell=1}^{m+1} t''_\ell$, define $f_{i+1}(\sigma) = \sigma_m(t_c, t_u)q'$, where $t_c = t''_{m+1}$ and $t_u = t_{\text{sum}} - \sum_{\ell=1}^{m} t''_\ell$.

Observe that our inductive definitions of λ'_i and f_i satisfy all the invariants that we maintain; the costs are preserved because the cost functions are TISC.

Finally, the invariants ensure that λ' satisfies the conditions of the lemma as follows. λ' is winning because any play consistent with λ' can be mapped to play consistent with λ using f_i. The third invariant ensures that the cost of the strategy λ' is bounded by the cost of strategy λ. Finally, conservativeness and time consistency are ensured by invariants 6 and 7 respectively. □

Next, we show that if a time consistent, conservative strategy is winning in all plays where the environment is conservative, then it is winning against all plays. Moreover, the supremum cost is achieved on runs where the environment plays conservatively.

Lemma 12. *Let λ be a time consistent, conservative strategy. Let R denote the collection of all runs consistent with λ starting from q and let $R_C \subseteq R$ be those runs in which the environment is conservative. If all the runs in R_C are winning then λ is a winning strategy from q. Moreover, $Cost(\lambda, q) = \sup_{\rho\in R} Cost(\rho) = \sup_{\rho\in R_C} Cost(\rho)$.*

Proof. Recall that we use R to denote the collection of all runs consistent with λ starting from q and $R_C \subseteq R$ to be those runs in which the environment is conservative. Suppose all the runs in R_C are winning. We need to show that all runs in R are winning. Suppose $\sigma \in R$ is not winning. Then σ does not reach the goal and does not contain an infinite sequence of $env \cdot \tau$ labels. Further since a $con \cdot \tau$ is necessarily followed by a discrete transition, we have only finite sequences of transitions labelled by $env \cdot \tau$ or $con \cdot \tau$ and $con \cdot \tau$ appears only at the end as λ is a conservative strategy. Consider a maximal sequence of time transitions in σ: $q_1(t_1, t'_1)q_2 \cdots (t_n, t'_n)q_n$. We can replace this by $q_1(t_1, \sum_{i=1}^{n} t'_i)q_n$ and the resulting sequence will be consistent with λ and have the same cost as the original run (because λ is time-consistent and the cost-function is additive). Hence σ' obtained by replacing every such maximal sequence by a single transition is in R_C and is not winning, a contradiction.

Consider $\sigma \in R$ which is winning. If it does not reach goal, then its cost is 0, then the σ' obtained above will also have cost 0 and is in R_C. If σ reaches goal, then ρ be the prefix of σ such that $last(\rho)$ is the first state in ρ which is in *Goal*. Then the ρ' obtained by merging transitions as above is in R_C and has the same cost as ρ or equivalently σ. Hence for every $\sigma \in R$, there is a $\sigma' \in R_C$ such that $Cost(\sigma) \le Cost(\sigma')$, hence $Cost(\lambda, q) \le \sup_{\rho\in R_C} Cost(\rho)$. But $\sup_{\rho\in R_C} Cost(\rho) \le \sup_{\rho\in R} Cost(\rho) = Cost(\lambda, q)$. □

Based on Lemmas 11 and 12, we can conclude the following:

Corollary 13. *Let λ be a conservative and time-consistent strategy. Any run σ which is consistent with λ and in which the environment is conservative does not have two consecutive time labels, i.e., does not contain the two consecutive $a.\tau$ where a is con or env.*

This along with the fact that the number of discrete transitions in any execution of a STORMED game is bounded allows us to conclude that we can restrict ourselves to bounded strategies.

Theorem 14 ([33])**.** *The number of discrete transitions in any run σ on the game graph induced by a STORMED game is bounded by a constant ν.*

Let us formally define a bounded strategy.

Definition 7. A strategy λ is *n-bounded* from a state q if it is conservative and time-consistent and every run from q consistent with λ in which the environment is conservative has at most n discrete transitions.

Thus we have the following observation about the existence of n-bounded strategies for weighted STORMED games.

Lemma 15. *If a state q of a weighted STORMED game has a winning strategy with cost c, then there is a n-bounded winning strategy from q of cost at most c.*

The above lemma implies that to solve the optimal-cost reachability problem we need to search only for n-bounded strategies. Following is an observation about optimal-bounded strategies. From now on we assume that the hybrid game is a weighted STORMED game.

Lemma 16. *Let q be a winning state and λ be an n-bounded optimal winning strategy. If $\lambda(q) = c$ for some $c \in Act_C$ and $q \xrightarrow{c,u} q'$ for $u \in \Sigma_U$, then λ is a $(n-1)$-bounded optimal winning strategy for q'. If $\lambda(q) = t$ for some $t \in \mathbb{R}_{\geq 0}$ and $q \xrightarrow{t,t'} q'$ or $q \xrightarrow{t,env\cdot t} q'$, then $\lambda(q') = c$ for some $c \in Act_C$ and λ is a $(n-1)$-bounded optimal winning strategy for q'', where $q' \xrightarrow{c,u} q''$ for some $u \in \Sigma_U$. If $\lambda(q) = t$ for some $t \in \mathbb{R}_{\geq 0}$ and $q \xrightarrow{t,u} q'$ for $u \in \Sigma_U$, then λ is a $(n-1)$-bounded optimal winning strategy for q'.*

We can now use a backward algorithm presented in [7] to compute the optimal cost of reaching the goal from a state q. Given a state q and a $n \in \mathbb{N}$, we define $c_n(q)$, the optimal cost of reaching $Goal \subseteq Loc \times Cont$ from q in at most n steps.

- $c_0(q) = 0$ if $q \in Goal$, $c_0(q) = \inf_{q \xrightarrow{t}_{\mathcal{H}} q', q' \in Goal} \{Cost(q,t) \mid \nexists t' \leq t, u \in Act_U, q \xrightarrow{t'}_{\mathcal{H}} q', q' \xrightarrow{u} q'', q'' \notin Goal\}$ if there exists t such that $q \xrightarrow{t}_{\mathcal{H}} q'$, $q' \in Goal$, ∞ otherwise.
- $c_{n+1}(q) = \inf_{q \xrightarrow{t}_{\mathcal{H}} q', q' \xrightarrow{c}_{\mathcal{H}} q''} \max(Cost(q,t) + c_n(q''), \sup_{q \xrightarrow{t'}_{\mathcal{H}} p', p' \xrightarrow{u} p'', t' \leq t} Cost(q,t') + c_n(p''))$.

Lemma 17. *For every $\epsilon > 0$, for every q such that $c_n(q) < \infty$, there exists a definable n-bounded winning strategy λ from q such that $Cost(q, \lambda) \leq c_n(q) + \epsilon$.*

Proof. $c_n(q)$ is taken to be the infimum cost over all enabled pairs (t, c). Hence given any ϵ, one can find a pair (t, c) enabled at q such that the cost of the expression within max is within $[c_n, c_n + \epsilon/n)$. In each step, there is a choice of (t, c) which is within ϵ/n from the optimal cost. Hence the cost of the strategy itself is within ϵ from the optimal cost. □

Lemma 18. *If λ is an n-bounded winning strategy from q, then $Cost(q, \lambda) \geq c_n(q)$.*

Theorem 19. *Given a Weighted STORMED hybrid game $(\mathcal{H}, Cost)$, whose underlying theory $\mathcal{M}$ is decidable, a state q of $\mathcal{H}$, a constant $c \in \mathbb{R}_{\geq 0}$ and a set of state Goal of $\mathcal{H}$, all of which are definable in $\mathcal{M}$, the optimal-cost reachability problem is decidable. In fact, we can define the optimal cost of reaching Goal.*

Proof. Since the number of discrete transitions in a STORMED game is bounded, the optimal cost of reaching *Goal* is equal to $c_n(q)$, for a computable n. $c_n(q)$ is definable in the o-minimal theory. To solve the optimal-cost reachability problem, we need to be able to determine if the number defined by $c_n(q) \leq c$. But since c is definable, we can decide if the inequality holds. $c_n(q)$ is also $Cost_{opt}(q)$. □

6.3. Model checking Weighted STORMED systems

In this section, we consider the problem of model-checking weighted hybrid systems with respect to a Weighted branching time logic called Weighted Computation Tree Logic (*WCTL*) which was introduced in [8,9].

A weighted STORMED hybrid system is the hybrid system version of a weighted STORMED game, that is, a weighted STORMED hybrid system is a weighted STORMED game with a single controllable action. Hence the semantics of a weighted STORMED system is given in terms of the weighted transition graph as for the case of weighted STORMED games.

First, let us define the logic *WCTL*. Given a structure $\mathcal{M}$ and an alphabet Σ_Q, a formula in $WCTL(\mathcal{M}, \Sigma_Q)$ is defined inductively as:

$$\phi ::= a \mid \phi \vee \phi \mid \neg\phi \mid E\phi U_{\sim c}\phi \mid A\phi U_{\sim c}\phi$$

where $a \in \Sigma_Q$ is an atomic proposition, $\sim \in \{<, \leq, =, \geq, >\}$ and c is an $\mathcal{M}$-definable constant.

Given a weighted transition system $(T, Cost)$ with a set of state labels Σ_Q and a state q, and a $WCTL(\mathcal{M}, \Sigma_Q)$ formula ϕ, the satisfaction relation $T, q \models \phi$ is defined inductively as follows:

$$\begin{array}{lcl}
T, q \models a & \Leftrightarrow & a \in L_Q(q). \\
T, q \models \neg\phi & \Leftrightarrow & T, q \not\models \phi. \\
T, q \models \phi_1 \vee \phi_2 & \Leftrightarrow & T, q \models \phi_1 \text{ or } T, q \models \phi_2. \\
T, q \models E\phi_1 U_{\sim c}\phi_2 & \Leftrightarrow & \text{there exists a maximal run } \rho \text{ from } q \text{ in } T \text{ such that} \\
 & & T, \rho \models \phi_1\, U_{\sim c}\, \phi_2. \\
T, q \models A\phi_1 U_{\sim c}\phi_2 & \Leftrightarrow & \text{for every maximal run } \rho \text{ from } q \text{ in } T,\ T, \rho \models \phi_1\, U_{\sim c}\, \phi_2.
\end{array}$$

Recall that ρ_i denotes the prefix of ρ of length i. Let $\rho[i]$ denote the last state of ρ_i. Below $Cost(\rho_i)$ denotes the sum $\sum_{j=1}^{i} Cost(q_{j-1}, c_j, u_j, q_j)$, where $\rho = q_0(c_1, u_1)q_1(c_2, u_2)\ldots$.

$$\begin{array}{lcl}
T, \rho \models \phi_1 U_{\sim c}\phi_2 & \Leftrightarrow & \exists i \geq 0 \text{ such that } T, \rho[i] \models \phi_2, \\
 & & \text{for all } 0 \leq i' < i,\ T, \rho[i'] \models \phi_1 \text{ and } Cost(\rho_i) \sim c.
\end{array}$$

The next theorem states that the problem of model-checking weighted STORMED hybrid systems against *WCTL* formulas is decidable.

Theorem 20. *Given a weighted STORMED hybrid system $\mathcal{H}$ definable in a decidable o-minimal structure $\mathcal{M}$, a definable state q of $\mathcal{H}$ and a WCTL($\mathcal{M}$, Σ_Q) formula ϕ, where Σ_Q is the set of state labels of game($\mathcal{H}$), the problem of whether game($\mathcal{H}$), $q \models \phi$ is decidable.*

Proof. We solve the problem by reducing it to the problem of model-checking a bounded discrete horizon o-minimally definable hybrid system against a *CTL* formula, which is shown to be decidable in [15,33].

Given a weighted STORMED hybrid system $\mathcal{H} = (Loc, Act_U, Labels, Cont, Edge, Inv, Flow, Guard, Reset, Lfunc)$ and a *WCTL*($\mathcal{M}$, Σ_Q) formula ϕ, we construct the hybrid system $\mathcal{H}' = (Loc', Act'_U, Labels', Cont', Edge', Inv', Flow', Guard', Reset', Lfunc')$ such that $\mathcal{H}, q \models \phi$ iff $\mathcal{H}', q' \models t(\phi)$, where q' is a state of $\mathcal{H}'$ corresponding to q, and ϕ and $t(\phi)$ is a *CTL* formula over $\Sigma_Q{}'$.

Informally, to construct $\mathcal{H}'$ for a given ϕ, we add a variable corresponding to every subformula of the form $\phi_1 U_{\sim c} \phi_2$ of ϕ. In $\mathcal{H}'$, the variables corresponding to these subformulas evolve with rate 0 and at some point start evolving according to the cost function. In the formula we ensure that the point at which a subformula starts evolving according to the cost function aligns with the point where the particular subformula is interpreted. The value of the variable at any point captures the cost since it started evolving according to the cost function. We introduce a label for each subformula which is true only if the value of the cost function in a particular state satisfies the constraint imposed by the subformula. We modify $\phi_1 U_{\sim c} \phi_2$ so that at the state chosen for satisfaction of ϕ_2, the proposition corresponding to the variable for $\phi_1 U_{\sim c} \phi_2$ also holds.

Next we present the formal definitions. Define $C_\psi = \{\psi_1 U_{\sim c} \psi_2 \mid \psi_1 U_{\sim c} \psi_2 \text{ is a subformula of } \psi\}$. Let us fix a *WCTL* formula ϕ. Let $k = |C_\phi|$ and $f : [k] \to C_\psi$ be a bijection. Let $Z_\psi = \{zero_\varphi \mid \varphi \in C_\psi\}$ and $B_\psi = \{comp_\varphi \mid \varphi \in C_\psi\}$. Define $\mathcal{H}'_\phi = (Loc', Act'_U, Labels', Cont', Edge', Inv', Flow', Guard', Reset', Lfunc')$, where:

- $Loc' = Loc \times 2^k$.
- $Act'_U = Act_U \cup \{\tau\}$.
- $Labels' = \Sigma_Q \cup Z_\phi \cup B_\phi$.
- $Cont' = Cont \times \mathbb{R}^k$.
- $Edge' = Edge'_1 \cup Edge'_2$ where $Edge'_1 = \{((l, S), a, (l', S')) \mid (l, a, l') \in Edge, S \subseteq S'\}$ and $Edge'_2 = \{((l, S), \tau, (l, S')) \mid S \subset S'\}$.
- $Inv'((l, S)) = Inv(l) \times \mathbb{R}^k$.
- Given $x \in Cont'$, we denote x by (x_r, x_c) where $x_r \subseteq Cont$ is the projection of x to the first n components (where $Cont = \mathbb{R}^n$) and $x_c \in \mathbb{R}^k$ is the projection of x to the last k components.
  $Flow'((l, S), (x_r, x_c))(t) = (Flow(l, x_r)(t), x'_c)$, where if $j \notin S$ then the j-th component of x'_c is the same as the j-th component of x_c, and if $j \in S$, then the j-th component of x'_c is $Cost((l, x_j), t)$ where x_j is the j-th component of x_c.
- $\mathcal{G}'(e) = \mathcal{G}(e) \times \mathbb{R}^k$ if $e \in Edge'_1$, $\mathcal{G}'(e) = Cont \times \mathbb{R}^k$ otherwise.
- $Reset'(e) = \{((x_r, x_c), (x'_r, x_c)) \mid (x_r, x'_r) \in Reset(e), x_c \in \mathbb{R}_{\geq 0}\}$ if $e \in Edge'_1$, $Reset'(e) = \{(x, x) \mid x \in Cont \times \mathbb{R}^k_{\geq 0}\}$, otherwise.
- $Lfunc'((l, S), (x_r, x_c)) = Lfunc(l, x_r) \cup Z \cup B$, where $Z = \{zero_{f^{-1}(i)} \mid i \in S\}$, and $B = \{comp_{f^{-1}(j)} \mid x_j \sim c\}$, where x_j is the j-th component of x_c}.

$\mathcal{H}'_\phi$ satisfies all the conditions of STORMED except for the separability of the guards. Nevertheless, $\mathcal{H}'_\phi$ has a bounded number of discrete transitions along any execution, since $\mathcal{H}$ itself had a bounded number of transitions along any execution and the newly added edges can be taken only finitely many times (due to the condition that an edge from $Edge'_2$ requires that the second component of the location strictly increases in size). Hence, due to results from [15,33], we can conclude that $\mathcal{H}'_\phi$ has a finite computable bisimulation and model-checking $\mathcal{H}'_\phi$ with respect to any *CTL* formula is decidable.

We now define the *CTL* formula $t(\phi)$ corresponding to ϕ inductively. The X operator here is the "next" operator of *CTL*. Given a formula ψ, let $F_\psi = \bigwedge_{\psi' \in C_\psi} \neg zero_{\psi'}$.

$$\begin{aligned}
t(a) &= a. \\
t(\neg\phi) &= \neg t(\phi). \\
t(\phi_1 \vee \phi_2) &= t(\phi_1) \vee t(\phi_2) \\
t(E\phi_1 U_{\sim c} \phi_2) &= EX(zero_{\phi_1 U_{\sim c} \phi_2} \wedge E((t(\phi_1) \wedge F_{\phi_1}) U (t(\phi_2) \wedge F_{\phi_2} \wedge comp_{\phi_1 U_{\sim c} \phi_2})). \\
t(A\phi_1 U_{\sim c} \phi_2) &= AX(zero_{\phi_1 U_{\sim c} \phi_2} \implies A((t(\phi_1) \vee \neg(F_{\phi_1} \wedge F_{\phi_2})) U \\
&\quad\; (\neg(F_{\phi_1} \wedge F_{\phi_2}) \vee (t(\phi_2) \wedge comp_{\phi_1 U_{\sim c} \phi_2})))).
\end{aligned}$$

Given a state $(l, x) \in Loc \times Cont$ of $\mathcal{H}$, and a subformula ψ of ϕ, let $Ext((l, x), \psi)$ defines a set of states of $\mathcal{H}_\phi$ as follows. $Ext((l, x), \psi) = \{((l, S), x, y) \mid S \subseteq [k], y \in \mathbb{R}^k, S \cap f^{-1}(C_\psi) = \emptyset, y = (y_1, \ldots, y_k), y_i = 0, \forall i \in f^{-1}(C_\psi)\}$.

Proposition 21. *Let $q \in Loc \times X$ and ψ be a subformula of ϕ. Then*
$\mathcal{H}, q \models \psi$ iff for all $q' \in Ext(q, \psi)$, $\mathcal{H}_\phi, q' \models t(\psi)$.

Since $\mathcal{H}'_\phi$ has a finite bisimulation quotient which can be constructed when the underlying o-minimal theory is decidable, we can effectively check if $\mathcal{H}_\phi, q' \models t(\psi)$ which is a model-checking problem for the *CTL* formula. Hence we can model-check $\mathcal{H}$ with respect to a *WCTL* formula. □

7. Conclusion

We have provided results for controller design for LTL winning conditions and optimal-cost reachability conditions for a general class of hybrid games and weighted hybrid games respectively. Our results apply to systems with rich continuous dynamics as well as a strong coupling between the discrete and continuous dynamics (they do not require strong resets), called STORMED hybrid games. At the same time, by providing a connection between the time-abstract bisimulation and the bisimulation on game graphs we have extended the reachability-only results from [6] to general LTL game specifications for other classes of hybrid games such as o-minimal hybrid games. In addition, we have shown decidability for the optimal reachability game for weighted STORMED hybrid games and decidability of WCTL for weighted (closed) STORMED hybrid systems.

References

[1] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, Moshe Y. Vardi, Alternating refinement relations, in: Proceedings of the Ninth International Conference on Concurrency Theory, CONCUR98, in: LNCS, vol. 1466, Springer-Verlag, 1998, pp. 163–178.

[2] Rajeev Alur, Salvatore La Torre, George J. Pappas, Optimal paths in weighted timed automata, in: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01, Springer-Verlag, London, UK, 2001, pp. 49–62.

[3] Eugene Asarin, Oded Maler, Amir Pnueli, Symbolic controller synthesis for discrete and timed systems, in: Hybrid Systems II, in: LNCS, vol. 999, Springer, 1995, pp. 1–20.

[4] Eugene Asarin, Oded Maler, Amir Pnueli, Joseph Sifakis, Controller synthesis for timed automata 1, in: Proc. IFAC Symposium on System Structure and Control, 1998, pp. 469–474.

[5] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, Frits W. Vaandrager, Minimum-cost reachability for priced timed automata, in: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01, Springer-Verlag, London, UK, 2001, pp. 147–161.

[6] Patricia Bouyer, Thomas Brihaye, Fabrice Chevalier, Control in o-minimal hybrid systems, in: LICS'06: Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society, Washington, DC, USA, 2006, pp. 367–378.

[7] Patricia Bouyer, Thomas Brihaye, Fabrice Chevalier, Weighted o-minimal hybrid systems are more decidable than weighted timed automata, in: LFCS, Springer, 2007, pp. 69–83.

[8] Thomas Brihaye, Véronique Bruyère, Jean-François Raskin, Model-checking for weighted timed automata, in: Yassine Lakhnech, Sergio Yovine (Eds.), FORMATS/FTRTFT, in: Lecture Notes in Computer Science, vol. 3253, Springer, 2004, pp. 277–292.

[9] Thomas Brihaye, Véronique Bruyère, Jean françois Raskin, Model-checking for weighted timed automata, in: Proceeding of FORMATS-FTRTFT04, in: Lect. Notes Comput. Sci., vol. 3253, Springer, 2004, pp. 277–292.

[10] Thomas Brihaye, Tom Henzinger, Vinayak Prabhu, Jean-Franois Raskin, Minimum-time reachability in timed games, in: ICALP 2007 Automata, Languages and Programming, July 2007, pp. 825–837.

[11] Thomas Brihaye, Christian Michaux, On the expressiveness and decidability of o-minimal hybrid systems, J. Complexity 21 (4) (2005) 447–478.

[12] F. Cassez, T.A. Henzinger, J.-F. Raskin, A comparison of control problems for timed and hybrid systems, in: HSCC 02, in: LNCS, vol. 2289, Springer, 2002, pp. 134–148.

[13] L. de Alfaro, T.A. Henzinger, R. Majumdar, Symbolic algorithms for infinite state games, in: Proceedings of CONCUR, 2001, pp. 536–550.

[14] Georgios E. Fainekos, Antoine Girard, George J. Pappas, Hierarchical synthesis of hybrid controllers from temporal logic specifications, in: Proceedings of the 10th international conference on Hybrid systems: computation and control, HSCC'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 203–216.

[15] Raffaella Gentilini, Klaus Schneider, B. Mishra, Successive abstractions of hybrid automata for monotonic ctl model checking, in: Sergei N. Artëmov, Anil Nerode (Eds.), LFCS, in: Lecture Notes in Computer Science, vol. 4514, Springer, 2007, pp. 224–240.

[16] E. Haghverdi, P. Tabuada, G.J. Pappas, Bisimulation relations for dynamical, control, and hybrid systems, Theoretical Computer Science 342 (2) (2005) 229–261.

[17] A. Henzinger, B. Horowitz, R. Majumdar, Rectangular hybrid games, in: Proc. 10th International Conference on Concurrency Theory, CONCUR'99, in: Lecture Notes in Computer Science, vol. 1664, Springer, 1999, pp. 320–335.

[18] T.A. Henzinger, P.W. Kopke, Discrete time control for rectangular hybrid automata, Theoretical Computer Science 221 (1999) 369–392.

[19] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, Pravin Varaiya, What's decidable about hybrid automata? in: Proc. 27th Annual ACM Symp. on Theory of Computing, STOC, 1995, pp. 373–382.

[20] G. Lafferierre, G.J. Pappas, S. Sastry, O-minimal hybrid systems, in Mathematics of Control, Signals, and Systems, vol. 13, March 2000, pp. 1–21.

[21] Oded Maler, Amir Pnueli, Joseph Sifakis, On the synthesis of discrete controllers for timed systems, in: Ernst Mayr, Claude Puech (Eds.), STACS 95, in: Lecture Notes in Computer Science, vol. 900, Springer, Berlin / Heidelberg, 1995, pp. 229–242.

[22] Thomas Moor, J.M. Davoren, Robust controller synthesis for hybrid systems using modal logic, in: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01, Springer-Verlag, London, UK, 2001, pp. 433–446.

[23] Y. Pang, M.P. Spathopoulos, Hao Xia, Reachability and optimal control for linear hybrid automata: a quantifier elimination approach, IJC 80 (5) (2007) 731–748.

[24] André Platzer, Edmund M. Clarke, Computing differential invariants of hybrid systems as fixedpoints, in: ICAV, 2008, pp. 176–189.

[25] R. Rosner, Modular synthesis of reactive systems, Ph.D. Thesis, 1992. Weizmann Institute of Science.

[26] P. Tabuada, Controller synthesis for bisimulation equivalence, Systems and Control Letters 57 (6) (2008) 443–452.

[27] P. Tabuada, G.J. Pappas, Linear temporal logic control of discrete-time linear systems, IEEE Transactions on Automatic Control 51 (12) (2006) 1862–1877.

[28] Alfred Tarski, A Decision Method for Elementary Algebra and Geometry, 2nd edition, University of California Press, 1951.

[29] C. Tomlin, G.J. Pappas, S. Shankar Sastry, Conflict resolution for air traffic management: a case study in multi-agent hybrid systems. Technical Report UCB/ERL M96/38, EECS Department, University of California, Berkeley, 1996.

[30] C.J. Tomlin, J. Lygeros, S. Shankar Sastry, A game theoretic approach to controller design for hybrid systems, Proceedings of the IEEE 88 (7) (July 2000) 949–970.

[31] L. van den Dries, C. Miller, On the real exponential field with restricted analytic functions, Israel Journal of Mathematics (85) (1994) 19–56.

[32] Lou van den Dries, Tame Topology and O-minimal Structures, Cambridge University Press, 1998.

[33] Vladimeros Vladimerou, Pavithra Prabhakar, Mahesh Viswanathan, Geir Dullerud, STORMED hybrid systems, in: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II, ICALP'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 136–147.

[34] Vladimeros Vladimerou, Pavithra Prabhakar, Mahesh Viswanathan, Geir E. Dullerud, STORMED hybrid games, in: Rupak Majumdar, Paulo Tabuada (Eds.), HSCC, in: Lecture Notes in Computer Science, vol. 5469, Springer, 2009, pp. 480–484.