Hybrid Games with Triggers*

Qais Hamarneh

Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany

Abstract

Hybrid games are a highly expressive way to model the interaction between cyber-physical systems. This high expressivity comes at the price of decidability. This work proposes an extension to hybrid games that adds the conditions prompting agents to move. We call this extension hybrid games with triggers (HGT). We show how this extension makes it possible to translate a hybrid game into a discrete game with countable state space.

Keywords

Hybrid Games, Hybrid systems, Discrete Games, Verification

1. Introduction

Modelling the interaction between multiple cyber-physical systems (CPS) is a critical problem in computer science, particularly in formal methods. Many approaches were presented to model and reason about this interaction, such as dynamic differential logic [1], and its extension to differential game logic [2], algebraic [3] and coalgebraic [4] approaches among many others.

This work is based on modelling an interaction between multiple CPSs as a hybrid game [5], a multi-agent extension of hybrid automata [6]. Hybrid games allow for discrete and continuous system evolution. Typically, the discrete evolution represents the agents' control, whereas the continuous evolution represents the motion dynamics. While hybrid games are very expressive, this expressivity comes at the price of decidability. In [7], Heinziger et al. show that the reachability in even very restrictive fragments of hybrid automata, like a stopwatch automaton, is undecidable.

There have been multiple attempts to discretize hybrid systems [8] by restricting either the discrete dynamics like in rectangular hybrid games [5] or the continuous dynamics like in o-minimal hybrid games [9]. This work presents a novel extension of hybrid games that allows discretization without restricting system dynamics. It does this by augmenting the hybrid game definition with information about the agents' rationale. This rationale is expressed as quantifier-free formulas of real arithmetic called *triggers*. A trigger is a condition that the agent must act once satisfied. We call this extension *hybrid games with triggers* (**HGT**).

HGTs are inspired by de Alfaro et al.'s timed games [10]. In this version of timed games, each agent declares a time delay, after which they would take an action unless some other agent played first. In essence, our extension replaces the time delay with arithmetic formulas, similar to the guards and invariants of hybrid automata [6], and the timed game with a hybrid game.

Using a logical formula instead of a time delay brings multiple advantages. (1) Formulas reduce the need for perfect information by covering cases without calculating the time required to reach them. For instance, a car safety trigger could look like this:

$$\mathit{free\text{-}ahead}(\mathit{position}) \leq \mathit{braking\text{-}distance}(\mathit{speed}). \tag{1}$$

(2) Unlike time, a small set of formulas like 1 could be sufficient to model a complete system. (3) As we show in this paper, given a countable language of real arithmetic [11], triggers reduce the hybrid game into a discrete game with countable state space.

PhD Symposium of the 19th International Conference on Integrated Formal Methods (iFM) at the University of Manchester, UK, 12 November 2024.

qais.hamarneh@kit.edu (Q. Hamarneh)
https://mase.kastel.kit.edu/team_qais_hamarneh.php (Q. Hamarneh)
0009-0009-9718-1664 (Q. Hamarneh)

In the next Section 1.1, we briefly overview the related work. Section 2 defines the hybrid games this work is based on. The main contribution of this work is presented in sections 3 and 4. In Section 3, we introduce the syntax and semantics of hybrid games with triggers, and we informally define the algorithm to create a discrete game based on an HGT in Section 4. We conclude in Section 5 with a summary and a look at future work.

1.1. Related Work

This work can be seen as a hybrid extension of de Alfaro et al.'s timed games [10]. Multiple points were taken directly from the timed games, like the winning conditions and what happens when multiple triggers are satisfied simultaneously. However, the discretization algorithm is entirely different. The discretization in the timed games relies on the existence of a finite bisimulation of timed automata as a region automata. Hybrid automata do not always offer a finite bisimulation [12].

As already discussed, other ways to discretize hybrid games exist [5][9], but where these methods restrict the game dynamics to allow discretization, our approach rely on adding more information to the game instead of restricting it.

Tight durational concurrent game structures (TDCGS) [13] are intuitively similar to hybrid game with triggers. Transitions in TDCGS carry an integer time delay. This time, however, is treated as a cost and does not reflect the evolution of the continuous dynamics.

2. Preliminaries

Hybrid Game

This definition of a hybrid game is adopted from [5]. The hybrid game is defined over a finite set of real-valued variables X. The set of all valuations $v : X \to \mathbb{R}$ is called $Val(X)$. We extend the notation v to arithmetic terms over X. $Form_X$ is the set of all real arithmetic quantifier-free formulas over X.

Given a valuation $v \in Val(X)$ and an arithmetic term θ, we write $v[\theta/x]$ for the valuation where all variables have the same value as in v, except $v[\theta/x](x) = v(\theta)$. This definition is extended to ordered sets of assignments, where the assignments are executed in order. Given a set of differential equations F, we write $v[F,t]$ for the valuation updated according to the equations in $flow$ after some time $t \in \mathbb{R}_{\geq 0}$ has passed. A hybrid game $\mathscr{G}$ is a tuple:

$$\mathscr{G} = (Loc,\ l_0,\ X,\ v_0,\ flow,\ inv,\ Agt,\ Act,\ E)$$

Loc	a finite nonempty set of locations.
$l_0 \in Loc$	the initial location.
X	a finite nonempty set of real-valued variables with typical elements x_0, x_1.
$v_0 : X \to \mathbb{R}$	the initial valuation.
$flow$	a continuous transition relation that assigns each location $l \in Loc$ a set of differential equations $flow(l) = \{\ \dot{x}_i = \theta_i \mid x_i \in X\}$.
$inv : Loc \to Form_X$	associate each location with an invariant.
$Agt =$	$\{\ 1, 2, \ldots, k\ \}$ a finite nonempty set of agents.
$Act =$	$Act_1 \uplus Act_2 \uplus \ldots \uplus Act_k$ a disjoint union of finite nonempty sets of agents' actions. The typical actions of agent $i \in Agt$ are a_i, b_i.
E	a finite nonempty set of edges representing discrete transition relation with typical elements (l, φ, a_i, A, l') such that: • $l, l' \in Loc$, • $\varphi \in Form_X$ called a *guard*, • $a_i \in Act_i$ an action for some $i \in Agt$, and • A is a finite (possibly empty) ordered set of assignments called a *jump*.

We use the functions $begin(e)$, $guard(e)$, $act(e)$, $jump(e)$ and $end(e)$ to reference the components of an edge $e \in E$.

A valuation v *enables* the edge $e = (l, \varphi, a, A, l') \in E$ (we say the action $act(e)$ is enabled) if:

$$\bullet\ v \vDash inv(l),\quad \bullet\ v \vDash \varphi,\ \text{and}\ \bullet\ v[A] \vDash inv(l')$$

In a hybrid game, a *configuration* is a pair $\langle l, v \rangle$ representing the game's location $l \in Loc$ and valuation $v \in Val(X)$. $\langle l_0, v_0 \rangle$ is the initial configuration. Two types of transitions are possible: A time transition $\langle l, v \rangle \xrightarrow{t} \langle l, v[flow(l), t] \rangle$ for $t \in R_{\geq 0}$ is *legal* if for all $t' \in [0, t]$, $v[flow(l), t'] \vDash inv(l)$. An edge transition $\langle l, v \rangle \xrightarrow{e} \langle end(e), v[jump(e)] \rangle$ for an edge $e \in E$ with $begin(e) = l$ is a legal transition if v enables e. The agent i that takes the action $act(e) \in Act_i$ is called *to blame* for the edge transition. The other agents are called *blameless*. A **play** is an infinite sequence of configurations $(\langle l_0, v_0 \rangle, \langle l_1, v_1 \rangle, \langle l_2, v_2 \rangle, \ldots)$ which starts at the initial configuration and for each $i \in \mathbb{N}_0$, there exists a legal transition $\langle l_i, v_i \rangle \to \langle l_{i+1}, v_{i+1} \rangle$.

Remark 1. *In this paper, we assume all differential equations to be solvable. Even with solvable differential equations, the question of whether there exists a play that reaches a certain configuration is undecidable in a hybrid game [7].*

2.0.1. Example

Figure 1 shows an orange and a green robot moving on tracks in a warehouse. The warehouse contains 6 tracks, 2 horizontal $H = \{ h_1, h_2 \}$ and 4 vertical $V = \{ v_1, v_2, v_3, v_4 \}$. The robots continuously pick items from storage units (small circles) and take them to the processing units (small squares). The robots must also avoid collisions with each other. To model this system as a hybrid game, we define the locations as the current tracks of the two robots $Loc = (H \cup V) \times (H \cup V)$. The current location is (h_1, v_3). The invariants define the end points of each track. The flow in each location describes the motion of each robot based on its current speed. Intersection points are edges that allow the robots to change tracks. Self-loop edges change a robot's speed without changing its track.

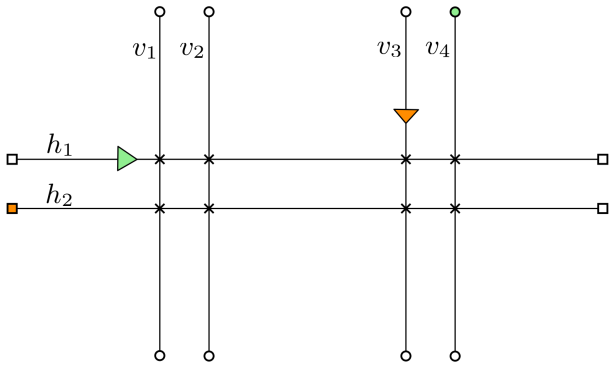

Figure 1: Robots on tracks.

3. Hybrid Game with Triggers (HGT)

In this section, we introduce an extension to the definition of hybrid games. We call this extension *hybrid games with triggers* or (HGT) for short. A trigger is an agent-declared quantifier-free formula of real arithmetic that, once satisfied, prompts the agent to take an action. Intuitively, the trigger is the reason the agent moves. An autonomous vehicle could set a trigger to be *not enough free space ahead or the desired intersection is reached.* According to the case that comes first, the car would have to take an action. An edge transition only happens when some agent has a satisfied trigger. With each edge transition, each agent gets to update their triggers. We call the set of all possible triggers $Form_{Trig} \subseteq Form_X$.

The definition of HGT extends the definition of hybrid games as follows:

$$\mathcal{G} := (Loc,\ l_0,\ X,\ v_0,\ flow,\ inv,\ Agt,\ Act_\perp,\ E_\perp,\ Form_{Trig}, trig_0)$$

The function $trig_0 : Agt \to Form_{Trig}$ is the initial triggers function. The set $Act_\perp$ includes a *stutter* action $\perp \notin Act$, available for every agent. If an agent's trigger is satisfied and this agent has no enabled actions in Act, the agent must take the stutter action $\perp$. This indicates that there is a self-loop edge for every location l, $(l, True, \perp, \emptyset, l) \in E_\perp$. A **configuration** in a HGT is a triplet $\langle l, v, trig \rangle$. The initial configuration is $\langle l_0, v_0, trig_0 \rangle$.

Trigger Formula: A formula φ is called a trigger formula if and only if for every valuation v and every map $flow$ assigning a differential equation to each variable in X, there exists a minimum time $t \in \mathbb{R}_{\geq 0}$ to satisfy φ, i.e. such that $v[flow, t] \models \varphi$ and for all $0 \leq t' < t$, $v[flow, t'] \not\models \varphi$, or if φ is never satisfied, i.e. $v[flow, t] \not\models \varphi$ for all $t \in \mathbb{R}_{\geq 0}$. In other words, a formula φ is a trigger formula ($\varphi \in Form_{Trig}$) if and only if its solution set is a closed set under the usual topology in $\mathbb{R}^{|X|}$.

This restriction eliminates formulas like $x > 2$ for a variable $x \in X$ where no exact time exists when it is first satisfied if the valuation $v(x) = 0$ and $flow(x) = 1$. On the contrary, $x \geq 2$ is a valid trigger formula.

Given a valuation v, a flow $flow$ and a trigger φ, we define the function *time to trigger* or $ttt(v, flow, \varphi)$ to return the minimum time required to satisfy the trigger φ if it exists and ∞ otherwise. This definition is extended to configurations $\langle l, v, trig \rangle$ to return the minimum time required to satisfy any of the formulas if such a time exists:

$$ttt(\langle l, v, trig \rangle) = \min \{ ttt(v,\ flow,\ trig(i)) \mid i \in Agt \}.$$

The definition of *legal* transitions in HGT is more restrictive than that in hybrid games. A transition is $\langle l, v, trig \rangle \to \langle l', v', trig' \rangle$ legal if and only if it fulfils the following conditions:

- $\langle l, v \rangle \to \langle l', v' \rangle$ is legal in the hybrid game,
- a time transition $\langle l, v, trig \rangle \xrightarrow{t} \langle l, v[flow(l), trig], trig \rangle$ does not change the triggers function $trig$, and $t \leq ttt(\langle l, v, trig \rangle)$, and
- an edge transition $\langle l, v, trig \rangle \xrightarrow{e} \langle end(e), v[jump(e)], trig' \rangle$ if $v \models trig(i)$ for the agent $i \in Agt$ to blame for the transition.

Along with each edge transition, each player i gets to choose a new trigger $\varphi_i \in Form_{Trig}$ creating a new trigger function mapping $trig'(i) = \varphi$. Similar to [10], if more than one trigger is satisfied at the same time, the agent who gets to take an action is chosen at random.

3.0.1. Example

We go back to the example shown in Figure 1. In the current configurations, the green robot has the trigger:

$$green.pos = intersection(h_1, v_4) \vee free\text{-}ahead(green.pos) \leq braking\text{-}distance(green.spd)$$

The orange robot's trigger is similar but has $intersection(h_2, v_3)$.

In this example, we can notice that robots (agents) do not need to calculate the time needed for the trigger to be satisfied when selecting one. Another observation is that a small set of trigger formulas is often sufficient for many systems. This feature makes reasoning about the system significantly easier.

Remark 2. *Contrary to guards and invariants, triggers are not part of the game structure but rather part of the players' strategies. A player could choose different triggers in the same location (see Example 3.0.1). While it is possible to extend the game structure by adding more locations and more restrictive guards and invariants to embed the triggers into the game structure, this is not always possible with a finite set of locations.*

Winning Conditions: We adopt the winning conditions from [10]. The idea of these winning conditions is that an agent cannot win by preventing time from progressing. A play is a winning play for agent i if time diverges td and the play fulfils their goal ϕ_i or if time converges tc, but the agent i is not to blame ($blameless_i$) for the time convergence, i.e. the agent i only takes a finite number of actions during the entire play. The set of winning plays for agent i with the desired outcome ϕ_i is $(WC(\phi_i) \cap td) \cup (blameless_i \setminus td)$, where $WC(\phi)$ is the set of plays that fulfils ϕ. As noted in [10], according to these winning conditions, both agents can lose in a two-agent game where one agent has the goal ϕ and the other $\neg\phi$. This is when the two agents infinitely take turns blocking time from progressing. I.e. time converges, and neither agent is *blameless*.

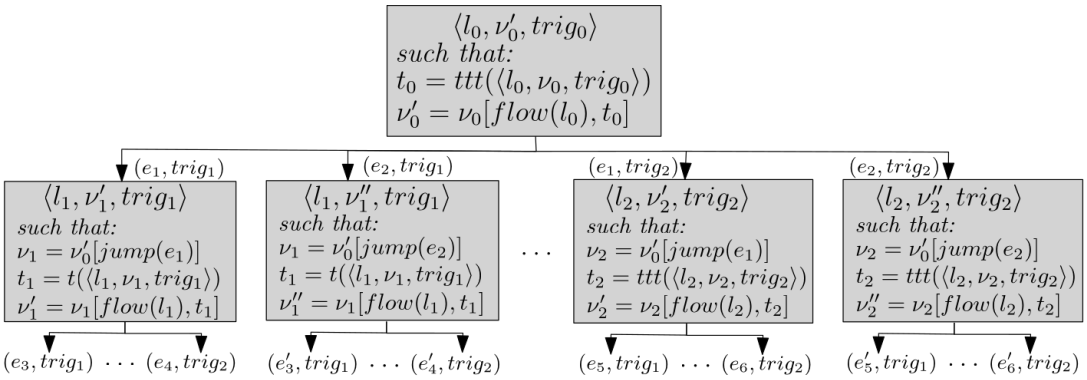

Figure 2: Time-abstract discrete game structure.

4. Discretizing a Hybrid Game with Triggers

In this section, we show an intuitive way to define a discrete game, such that agent $i \in Agent$ has a winning strategy in the HGT if and only if the same i has a winning strategy in the discrete game. The intuition of the discrete game is to skip any time steps where no triggers are satisfied. This allows time to move in discrete steps. At any configuration $\langle l, v, trig \rangle$ with $v \not\models trig(i)$ for all $i \in Agt$, the game can progress by $ttt(\langle l, v, trig \rangle)$. If a trigger is satisfied, any enabled edge can be taken and the trigger function gets updated.

Due to space limitations, we only briefly describe the discrete game in this paper.

The discrete game is structured as a concurrent game structure (CGS) [14]. The players are the agents of the HGT Agt with the addition of the player $Random \notin Agt$ to select the agent who gets to take an action when more than one trigger is satisfied. While only one action is taken in each step, the game is concurrent to allow all agents to select new triggers simultaneously. The actions available for the agents are $Act \times Form_{Trig}$. The actions available for $Random$ are the set Agt.

The set of states S of the discrete game is defined inductively:

- $s_0 = \langle l_0, v_0, trig_0 \rangle \in S$ is the initial state.
- If the state $s = \langle l, v, trig \rangle \in S$, then:
  - if no trigger is satisfied $v \not\models trig(i)$ for all $i \in Agt$ and $ttt(\langle l, v, trig \rangle) \in \mathbb{R}_{\geq 0}$, then $\langle l, v[flow(l), ttt(\langle l, v, trig \rangle)], trig \rangle \in S$,
  - otherwise for every $e \in E_\perp$ enabled at s and for every function $trig' : Agt \to Trigger$, the state $\langle end(e), v[assign(e)], trig' \rangle \in S$.

A time-abstract game is visualized in Figure 2, where the game evolves only based on the players' choices.

Given that the number of edges $E_\perp$ is finite and the number of trigger formulas is countable [11], each layer of the tree is countable. The state space of the entire discrete game is, therefore, countable. Each discrete game state is labelled with formulas that its valuation satisfies and are relevant to the agents' winning conditions. These formulas serve as atomic propositions in the discrete game. Additionally, the states are labelled according to their position in the tree. We use a set of atomic propositions inspired by [10] to prevent agents from winning by blocking the passing of time. The boolean proposition *tick* is true if the global time has passed an integer value compared to the state's parent in the tree. The atomic propositions $blame_i$ for $i \in Agt$ express the agent whose action reached this state.

A play is winning for an agent i if (1) the play satisfies the desired outcome ϕ (expressed as a temporal property) and has an infinite number of states marked with *tick* or (2) the play has a finite number of states marked with *tick* and a finite number of states marked with $blame_i$. The existence of a winning strategy for the agent i can be then expressed in the notation of ATL* [14] as follows:

$$\langle\!\langle \{ i \} \rangle\!\rangle (\phi \wedge \Box \Diamond tick) \vee (\Diamond \Box \neg tick \wedge \Diamond \Box \neg blame_i) \tag{2}$$

This reduces the winning conditions in a hybrid game with triggers to an ATL* model checking problem over countable state space. This is shown to be decidable in [15, 16].

5. Discussion and Conclusion

Extending hybrid games with triggers has multiple advantages and applications beyond the decidable fragment of hybrid games brought on by triggers.

In addition to the significant decidability results, triggers could help improve system understandability. An AI system, for instance, could be trained to choose (or form) a trigger formula and not act again until this formula is satisfied. Such a feature would have major benefits to AI verification and explainability.

In summary, hybrid games with triggers (HGT) offer a powerful framework for reasoning about and verifying multi-agent hybrid systems. We show that by incorporating agents' rationale into the game model, HGT can effectively reduce a hybrid game to a decidable discrete game without restricting its continuous or discrete dynamics.

Acknowledgments

This research was supported by the Innovation Campus for Future Mobility (www.icm-bw.de) and by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1608 Convide (www.sfb1608.kit.edu/index.php).

References

[1] A. Platzer, Differential dynamic logic for hybrid systems, J. Autom. Reason. 41 (2008) 143–189. URL: https://doi.org/10.1007/s10817-008-9103-8. doi:10.1007/S10817-008-9103-8.

[2] A. Platzer, Differential game logic, ACM Transactions on Computational Logic (TOCL) 17 (2015) 1–51.

[3] P. Höfner, B. Möller, An algebra of hybrid systems, The Journal of Logic and Algebraic Programming 78 (2009) 74–97.

[4] R. Neves, L. S. Barbosa, Hybrid automata as coalgebras, in: Theoretical Aspects of Computing–ICTAC 2016: 13th International Colloquium, Taipei, Taiwan, ROC, October 24–31, 2016, Proceedings 13, Springer, 2016, pp. 385–402.

[5] T. A. Henzinger, B. Horowitz, R. Majumdar, Rectangular hybrid games, in: J. C. M. Baeten, S. Mauw (Eds.), CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings, volume 1664 of *Lecture Notes in Computer Science*, Springer, 1999, pp. 320–335. URL: https://doi.org/10.1007/3-540-48320-9_23. doi:10.1007/3-540-48320-9_23.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, Theor. Comput. Sci. 138 (1995) 3–34. URL: https://doi.org/10.1016/0304-3975(94)00202-T. doi:10.1016/0304-3975(94)00202-T.

[7] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, J. Comput. Syst. Sci. 57 (1998) 94–124. URL: https://doi.org/10.1006/jcss.1998.1581. doi:10.1006/JCSS.1998.1581.

[8] R. Alur, T. A. Henzinger, G. Lafferriere, G. J. Pappas, Discrete abstractions of hybrid systems, Proceedings of the IEEE 88 (2000) 971–984.

[9] P. Bouyer, T. Brihaye, F. Chevalier, O-minimal hybrid reachability games, Logical Methods in Computer Science 6 (2010).

[10] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, M. Stoelinga, The element of surprise in timed games, in: R. M. Amadio, D. Lugiez (Eds.), CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003, Proceedings, volume 2761 of *Lecture Notes in*

Computer Science, Springer, 2003, pp. 142–156. URL: https://doi.org/10.1007/978-3-540-45187-7_9. doi:10.1007/978-3-540-45187-7_9.

[11] A. Tarski, A decision method for elementary algebra and geometry, in: B. F. Caviness, J. R. Johnson (Eds.), Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer Vienna, Vienna, 1998, pp. 24–84.

[12] T. A. Henzinger, Hybrid automata with finite bisimulations, in: Z. Fülöp, F. Gécseg (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 324–335.

[13] F. Laroussinie, N. Markey, G. Oreiby, Model-checking timed atl for durational concurrent game structures, in: International Conference on Formal Modeling and Analysis of Timed Systems, Springer, 2006, pp. 245–259.

[14] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713. URL: https://doi.org/10.1145/585265.585270. doi:10.1145/585265.585270.

[15] S. Schewe, Atl* satisfiability is 2exptime-complete, in: International colloquium on automata, languages, and programming, Springer, 2008, pp. 373–385.

[16] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: On the model-checking problem, ACM Transactions on Computational Logic (TOCL) 15 (2014) 1–47.