Trigger-Based Discretization of Hybrid Games for Autonomous Cyber-Physical Systems

Qais Hamarneh
qais.hamarneh@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Maike Schwammberger
schwammberger@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract

Hybrid games are a powerful framework for modelling interactions between cyber-physical systems (CPS). While their high level of nondeterminism allows the modelling of complex systems, it also makes many properties undecidable. This paper presents Hybrid Games with Triggers (HGT), an extension that reduces nondeterminism by embedding agents' rational decision-making as triggers - conditions that, when met, prompt the agents to act. This approach leads to a time-abstract discrete game with a countable state space, where an agent has a winning strategy in the discrete game if and only if they do in the HGT. Our discretisation preserves both the continuous and discrete dynamics of the original hybrid game, providing a more structured, analysable model without sacrificing dynamic expressivity.

CCS Concepts

• **Theory of computation** → **Timed and hybrid models**.

Keywords

Cyber-Physical Systems, Hybrid Systems, Hybrid Games, Discretization, Formal Verification

ACM Reference Format:
Qais Hamarneh and Maike Schwammberger. 2025. Trigger-Based Discretization of Hybrid Games for Autonomous Cyber-Physical Systems. In *28th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3716863.3718032

1 Introduction

With an increase in autonomous systems, modelling interactions between cyber-physical systems (CPS) becomes evermore important. Hybrid games [12], a multi-agent extension of hybrid automata [2], offer a powerful framework for this. Hybrid games are especially relevant in fields like robotics and autonomous driving, where agents need to balance physical limitations with strategic goals. However, while hybrid games are highly expressive, this flexibility introduces challenges in analysing critical properties like safety and reachability [13]. The inherent nondeterminism of hybrid games complicates reasoning about the system's behaviour.

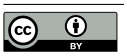
Hybrid games, as conventionally defined [15, 12, 21, 8, 5, 22, 19], primarily account for physical dynamics, often overlooking the goals and rationale of agents. This leaves a high degree of nondeterminism in the dynamics of hybrid games. Yet, in real-world settings, autonomous agents typically do not act randomly: they follow specific, goal-directed behaviours. For example, a vehicle turns when it reaches its intended intersection or changes lanes to avoid an obstacle. Accounting for this rationale can significantly reduce the nondeterminism, thereby simplifying the game analysis.

In this paper, we introduce *Hybrid Games with Triggers* (HGT), an extension of hybrid games that incorporates agents' strategic rationale through the use of *triggers*. A trigger captures the specific conditions under which an agent needs to make a decision, such as a vehicle reaching an intersection or coming within a certain distance of another vehicle. By including these triggers, we reduce the nondeterminism in hybrid games, enabling a structured discretization that preserves both continuous and discrete dynamics.

To leverage triggers, we define a discrete game model where the agent has a winning strategy in the discrete game if and only if the agent has a winning strategy in the HGT. We call this discrete game *time-abstract game with triggers* (TAGT). Unlike the discretization of hybrid games from [12] and [5], TAGT exploits the added knowledge brought by the triggers without imposing any restrictions on the discrete or continuous dynamics of the game. The discrete game consists of a countable state space, keeping only the *active* game states (i.e., where some agent takes an action) while eliminating all *passive* states (i.e., where *nothing* happens).

To prevent a winning strategy that simply impedes the progress of time, we adopt the winning conditions from [7] [6]. We formalize these conditions over TAGT using Alternating-Time Temporal Logic-* (ATL*) [3]. Thus, determining the existence of a winning strategy in an HGT is reduced to an ATL* model-checking problem.
Contribution. We summarise our two key contributions.
(1) Hybrid game with triggers (HGT): An extended model of hybrid games that integrates agents' rationales, allowing for more intuitive reasoning about hybrid properties and strategies (cf. Sect. 3).
(2) Time-abstract game with triggers (TAGT): A time-abstract discretization of HGT that eliminates time nondeterminism, reducing the game to a countable state space. An agent has a winning strategy in the HGT if and only if that agent has a winning strategy in the TAGT. Using ATL*, we formalize winning conditions over TAGTs that prevent Zeno behaviour (cf. Sect. 4).
A research preview on this paper's topic is presented in [10].

Related Work. We adopt the hybrid game definition from [12], which is a multi-agent extension of hybrid automata [2]. In recent years, significant progress has been made in analyzing hybrid games [18, 19, 20], which is a game extension of hybrid programs [17, 16]. In this hybrid game model, one agent controls the system at any given time, with control shifting between agents



HSCC '25, Irvine, CA, USA

ACM ISBN 979-8-4007-1504-4/2025/05
https://doi.org/10.1145/3716863.3718032

throughout the game. In contrast, in the hybrid automata game extension, any agent with an enabled action can act at any time, making it more suitable for our HGT model. In HGT, the agent whose trigger is satisfied first can act and interrupt the game's flow.

Many models exist for discretizing hybrid systems [4]. In most existing approaches to discretization, the model imposes restrictions on either the continuous dynamics, as in rectangular hybrid games [12], or the discrete dynamics, as in O-minimal hybrid games [5, 22]. By incorporating agents' rationale, our approach achieves discretization without imposing constraints on the game's continuous or discrete dynamics.

We view hybrid games with triggers as a hybrid generalization of timed games as introduced by de Alfaro et al. [7]. We also adopt the winning conditions from [7], generalized in [6], which prevent players from winning through Zeno behaviour. Solving these timed games depends on the existence of a finite bisimulation of timed automata, the region automata [1]. Hybrid automata, on the other hand, do not always have a finite bisimulation [11]. Consequently, we base our discretization approach on the characteristics of HGT rather than on the underlying hybrid automata.

2 Preliminaries

Our goal is to discretize a hybrid game through the use of triggers. To achieve this, we define two game models: one discrete and one hybrid. for our discrete game model, we adopt the *concurrent game structure* (CGS) from [9], which serves as the semantic model for ATL* logic [3]. We use ATL* to formalize non-Zeno winning conditions (cf. Sect 4.2).

Definition 2.1 (Concurrent Game Structure). A **Concurrent game structure** $\mathcal{S}$ is a tuple:

$$\mathcal{S} = (S, s_0, Agt, Act, AP, label, \delta).$$

- S and Act are non-empty sets of states and actions, respectively.
- $s_0 \in S$ is the initial state.
- Agt and AP are finite non-empty sets of agents and atomic propositions, respectively.
- $label : S \to \mathcal{P}(AP)$ is a labelling function.
- $\delta \subseteq S \times (Agt \to Act) \to S$ is a partial transition function. This is the reason why the game is concurrent: The transition function depends on the actions taken simultaneously by all agents. A decision function $d : Agt \to Act$ assigns each agent $i \in Agt$ an action $d(i) \in Act$. We say, an action $a \in Act$ is *enabled* for agent $i \in Agt$ at state $s \in S$ if there exists a decision function $d : Agt \to Act$ with $d(i) = a$ and $\delta(s, d)$ is defined.

The following concepts and notations are fundamental to defining hybrid games. For the rest of this paper, we assume a set of real-valued variables $X = \{x_1, x_2, \ldots, x_n\}$. The following terms and notation are defined over the set X.

Definition 2.2 (Terms and Constraints). Terms θ and θ' are arithmetic expressions over the variables X:

$$\theta, \theta' \coloneqq x \mid c \mid \theta + \theta' \mid \theta \cdot \theta'$$

where $x \in X$ is a variable and $c \in \mathbb{Q}$ is a constant.
Constraints φ, ψ are predicates over the variables X, of the form:

$$\varphi, \varphi' \coloneqq \theta \geq \theta' \mid \neg\varphi \mid \varphi \wedge \varphi'$$

The set of all Constraints over X is $\mathbb{C}_X$. The constraints *True* and *False* can be defined as usual.

Definition 2.3 (Valuation). A valuation $v : X \to \mathbb{R}$ assigns a real value to each variable. The set of all valuations is called $Val(X)$. The definition of a valuation extends to terms.

We write $v \vDash \varphi$ to indicate that the valuation v satisfies the constraint φ. Otherwise, we write $v \nvDash \varphi$. The set of all valuations that satisfy a constraint φ is the solution set of φ, denoted $[\![\varphi]\!]$:

$$[\![\varphi]\!] = \{v \in \mathbb{R}^n \mid v \vDash \varphi\}$$

In a hybrid game, variables represent dynamically changing values, such as a car's position, velocity, and acceleration. Valuations evolve in one of two ways: continuously or discretely. We use the following notation to represent each type of evolution:

Continuous Evolution: Given a set of differential equations of the form $flow = \{x_i' = \theta \mid x_i \in X\}$, the valuation after a time $t \in \mathbb{R}_{\geq 0}$ where each variable $x \in X$ is updated according to the differential equations in *flow* is denoted $\phi_{flow}(v, t)$. Throughout this work, we assume all differential equations to be solvable.

Discrete Evolution: Given an assignment $x \coloneqq \theta$, the valuation v evolves as follows:

$$v[x \coloneqq \theta](y) = \begin{cases} v(\theta) & \text{if } y = x \\ v(y) & \text{otherwise.} \end{cases}$$

This definition can be extended to a finite, ordered set of assignments $A = \{x_{i_1} \coloneqq \theta_1, x_{i_2} \coloneqq \theta_2, \ldots, x_{i_m} \coloneqq \theta_m\}$:

$$v[A] = v[x_{i_1} \coloneqq \theta_1][x_{i_2} \coloneqq \theta_2] \ldots [x_{i_m} \coloneqq \theta_m]$$

For the hybrid game model, we adopt the hybrid game definition from Henzinger et al. [12]. However, to fit this definition better to our concept of triggers, we redefine the hybrid game to be concurrent. Our redefinition goes counter to the timed games with an element of surprise by De Alfaro et al. [7]. In their approach, each agent chooses a time delay — the rough equivalent of our condition for action — and the action they will take after that time delay. If multiple agents choose the same time delay, the agent who gets to take an action is selected randomly. By contrast, in our concurrent hybrid game, every agent whose condition is met gets to take an action. We argue that a concurrent model like ours here aligns more closely with the actions of agents in real-world autonomous CPS.

Definition 2.4 (Concurrent Hybrid Games). A concurrent hybrid game $\mathcal{G}$ is a tuple:

$$\mathcal{G} = (Loc, l_0, X, v_0, flow, inv, Agt, Act, E, guard, jump) \quad (1)$$

- Loc is a finite nonempty set of locations.
- $l_0 \in Loc$ is the initial location.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of real-valued variables.
- $v_0 \in Val(X)$ is the initial valuation.
- *flow* is a continuous transition relation that associates each location $l \in Loc$ with a set of differential equations $flow_l \coloneqq \{x_i' = \theta_i \mid x_i \in X\}$.
- $inv : Agt \to \mathbb{C}_X$ is a function assigning a constraint, an *invariant*, $inv(l) \in \mathbb{C}_X$ to each location $l \in Loc$.
- $Agt = \{1, \ldots, k\}$ a finite nonempty set of agents.

- Act is a finite nonempty set of agents' actions with a typical element $a \in A$. We assume the existence of a *stutter* action $\perp \in Act$.
- $E \subseteq Loc \times (Agt \to Act) \times Loc$ is a finite nonempty set of edges representing a discrete transition relation with typical elements $e = (l, d_e, l')$. The function $d_e : Agt \to Act$ is a decision function as in the CGS, (cf. Definition 2.1).
- $guard : E \to \mathbb{C}_X$ called an edge *guard*.
- *jump* assigns each edge $e \in E$ a finite (possibly empty) ordered set of discrete assignments $J_e \coloneqq \{x_{i_1} \coloneqq \theta_1, \ldots, x_{i_m} \coloneqq \theta_m\}$, where $x_{i_j} \in X$ for all $1 \le j \le m$.

Semantics. The semantics of the hybrid games are interpreted over a labelled transition system. A *configuration* is a pair $\langle l, v\rangle$ containing the game's location $l \in Loc$ and valuation $v \in Val(X)$. The set of all configurations *Config* represents the states of the transition system, and $\langle l_0, v_0\rangle$ is the *initial configuration*.

The transitions of hybrid games resemble two types of valuation evolutions explained above:

Continuous transition: often called a **time transition**, $\langle l, v\rangle \xrightarrow{t} \langle l, v'\rangle$ for $t \in \mathbb{R}_{\ge 0}$, such that $v' = \phi_{flow_l}(v, t)$.

A time transition $\langle l, v\rangle \xrightarrow{t} \langle l, \phi_{flow_l}(v, t)\rangle$ is *legal* if for all $t' \in [0, t]$ $\phi_{flow_l}(v, t') \vDash inv(l)$.

Discrete transition: often called an **edge** or an **action transition**, $\langle l, v\rangle \xrightarrow{d} \langle l', v'\rangle$ for some decision function $d : Agt \to Act$.

An action transition $\langle l, v\rangle \xrightarrow{d} \langle l', v'\rangle$ is legal if there exists an edge $(l, d, l') \in E$ that is *enabled*. An edge $e = (l, d_e, l') \in E$ is enabled at the configuration $\langle l, v\rangle$ if $v \vDash inv(l), v \vDash guard(e)$ and $v[J_e] \vDash inv(l')$. We also say the action $d_e(i)$ is enabled for the agent $i \in Agt$. We assume the stutter action $\perp$ to be enabled for every agent at every configuration. That means, for every location $l \in Loc$, there exists a *stutter* edge $e_\perp = (l, d_\perp, l) \in E$, such that (1) $d_\perp(i) = \perp$ for every agent $i \in Agt$, (2) $guard(e_\perp) = True$, and (3) $J_{e_\perp} = \{\}$.

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 with the initial configuration. For all consecutive states $\langle l_i, v_i\rangle$ and $\langle l_{i+1}, v_{i+1}\rangle$, there is a legal transition $\langle l_i, v_i\rangle \to \langle l_{i+1}, v_{i+1}\rangle$.

3 Introducing Triggers into Hybrid Games

We introduce a novel extension of hybrid games that we call *triggers*. With triggers, we can incorporate the agents' rationale in the form of conditions that prompt an agent to act. With our extension, the nondeterminism of hybrid games can be significantly reduced.

Example 3.1 (Running Example). Fig. 1 shows three robots: **red** R, and **green** G and **blue** B circles. The radius of all robots is r. The robots are moving in the aisles of a warehouse. The warehouse contains four aisles: two horizontal $H = \{h_1, h_2\}$ and two vertical $V = \{v_1, v_2\}$. The robots continuously move items from one end to the other. The coloured ends indicate each robot's current goal. The robots must also avoid colliding with each other. The arrows attached to the robots show their current direction. The movement and interaction of the robots are modelled as a hybrid game. The agents are the three robots $Agt \coloneqq \{R, G, B\}$. The variables in X

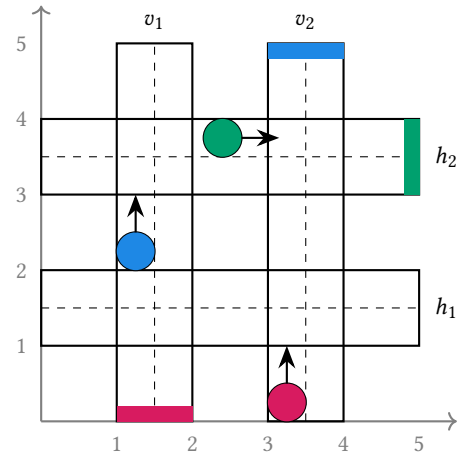


Figure 1: A hybrid game representing 3 robots in a warehouse.

represent their position, speed and direction

$$X \coloneqq \{x_i, y_i, spd_i, dir_i \mid i \in Agt\},$$

where (x_i, y_i) is the position of the centre of robot i. The direction dir_i can have values $\{-1, 1\}$. A positive direction indicates that the coordinates are increasing and vice versa. For simplicity, we assume the robot has only two possible speeds, $\{0, 1\}$, which can change discretely. Additionally, the robot's direction can change in discrete 90° increments, allowing it to turn either left or right relative to its current direction. The initial valuation v_0 is the one shown in Fig. 1. The initial speed of all robots is 1.

The locations *Loc* represent the aisles each robot is on:

$$Loc \coloneqq (H \cup V)^3$$

The initial location is $l_0 = (v_1, v_2, h_2)$. The locations' invariants ensure the robots stay within the aisles' borders. The invariant $inv(l_0)$ is defined as follows:

$$\begin{aligned} inv(l_0) \coloneqq & 3 + r \le x_R \le 4 - r \quad \wedge \quad r \le y_R \le 5 - r \wedge \\ & r \le x_G \le 5 - r \quad \wedge \quad 3 + r \le y_G \le 4 - r \wedge \\ & 1 + r \le x_B \le 2 - r \quad \wedge \quad r \le y_B \le 5 - r \end{aligned}$$

The locations' flows determine the robots' continuous movement. The flow $flow_{l_0}$ is defined as follows:

$$\begin{aligned} flow_{l_0} \coloneqq \{ & x_R' = 0, && y_R' = spd_R \cdot dir_R, \\ & x_G' = spd_G \cdot dir_G, && y_G' = 0, \\ & x_B' = 0, && y_B' = spd_B \cdot dir_B\} \end{aligned}$$

The agent's actions allow the robots to *move* ($spd \coloneqq 1$), *stop* ($spd \coloneqq 0$) and turn 90° *left* or *right* with respect to the robot's current direction:

$$Act = \{move, stop, left, right, \perp\}.$$

The edges determine whether the agents can execute their actions. Changing speed is always allowed and is expressed as a self-loop edge in each location. For example, in any location l, the edge e_1

allows robots B and G to stop:

$$e_1 \coloneqq (l,\ d_{e_1} = \{R \mapsto \bot,\ G \mapsto stop,\ B \mapsto stop\},\ l)$$
$$guard(e_1) \coloneqq True,\quad J_{e_1} \coloneqq \{spd_B \coloneqq 0,\ spd_B \coloneqq 0\}.$$

Edges also allow the robots to change tracks by changing direction only at the intersection points. For example, from the initial location l_0, the robot R can turn when it reaches the intersection (h_1, v_2) using the edge e_2:

$$e_2 \coloneqq (l_0,\ d_{e_2} = \{R \mapsto right,\ G \mapsto \bot,\ B \mapsto \bot\},\ (v_1, h_1, h_2))$$
$$guard(e_2) \coloneqq 3 + r \ \le\ x_R \ \le\ 4 - r \ \wedge\ 1 + r \ \le\ y_R \ \le\ 2 - r,$$
$$J_{e_2} \coloneqq \{dir_R \coloneqq -1\}.$$

In this game, we observe a key characteristic of hybrid games: An enabled edge does not indicate whether the agents need to take it. This contributes to a high level of non-determinism. Although each agent has a specific goal, it is not explicitly modelled within the game, meaning it cannot be leveraged in verification processes.

The following section introduces an extension of the hybrid game model for representing an agents' changing goals, thus enabling the verification process to focus solely on edges aligned with the agents' strategic objectives, which we express through *triggers*.

3.1 Triggers

A trigger is a condition that, once met, prompts an agent to act. For example, a vehicle reaching an intersection is prompted to turn, while getting too close to the vehicle ahead prompts it to slow down. With that, triggers form part of agents' strategies for staying safe and reaching their goals. Formally, a trigger is a constraint selected by an agent such that the agent takes action precisely once this constraint is satisfied. In other words, with each discrete transition, each agent chooses a constraint or trigger that they would impose upon the continuous flow. Triggers define the moment of acting; thus, for every trigger, there must be an exact time point when that constraint is satisfied. For example, the constraint $\varphi \equiv x > 2$ for some variable $x \in X$ is not a valid trigger because there might not be an exact moment when φ is satisfied. On the other hand, $\varphi' \equiv x \ge 2$ is a valid trigger.

Definition 3.2 (Trigger). We call a constraint φ a trigger if for every valuation v and every set of differential equations *flow*, either:

- there exists a minimal time $t \in \mathbb{R}_{\ge 0}$ to satisfy φ, i.e.: $\phi_{flow}(v,t) \vDash \varphi$ and for all $t' \in [0,t)$, $\phi_{flow}(v,t') \nvDash \varphi$, or
- φ is never satisfied, i.e. $\phi_{flow}(v,t) \nvDash \varphi$ for all $t \in \mathbb{R}_{\ge 0}$.

In other words, a constraint φ is a trigger if and only if $[\![\varphi]\!]$ is a closed set under the usual topology in $\mathbb{R}^n$.

The set of all trigger constraints is called $\mathbb{C}$:

$$\mathbb{C}_T \coloneqq \{\varphi \in \mathbb{C}_X \mid [\![\varphi]\!] \text{ is a closed set}\}$$

Given a hybrid game configuration $\langle l, v\rangle$, the minimal time to satisfy a trigger for some agent $i \in Agt$ is essential to track when (or whether) the agent i would act.

Definition 3.3 (Time to Trigger). Given a trigger φ and a configuration $\langle l, v\rangle$, the function *time to trigger* $ttt(\langle l, v\rangle, \varphi)$ returns the minimal time to satisfy φ starting at $\langle l, v\rangle$:

$$(\langle l, v\rangle, \varphi) \coloneqq \begin{cases} t & \text{if } \phi_{flow_l}(v,t) \vDash \varphi \text{ and for all } t' \in [0,t): \\ & \phi_{flow_l}(v,t') \nvDash \varphi. \\ \infty & \text{if } \phi_l(v,t) \nvDash \varphi \text{ for all } t \in \mathbb{R}_{\ge 0}. \end{cases}$$

Example 3.4. Given the following valuation and flow of some location l:

$$valuation: \quad v(x) = 0, \qquad v(y) = 1$$
$$flow_l: \quad x' = 2, \qquad y' = 0$$

The following are examples of triggers along with their *time-to-trigger* function *ttt*:

$$\varphi_1 \coloneqq (x \ge 4 \vee y \ge 10) \qquad ttt(\langle l, v\rangle, \varphi_1) = 2$$
$$\varphi_2 \coloneqq (y \ge 10) \qquad ttt(\langle l, v\rangle, \varphi_2) = \infty$$

Triggers can be used to express that an agent needs to interrupt a continuous flow and take an action. Next, we show how triggers can be incorporated into the hybrid game syntax and semantics.

3.2 Hybrid Games with Triggers

A hybrid game with triggers (HGT) extends the concurrent hybrid game Def. 2.4 with triggers.

Definition 3.5 (Hybrid Games with Triggers). A *hybrid game with triggers* $\mathcal{G}$ is a tuple

$$\mathcal{G} \coloneqq (Loc, l_0, X, v_0, flow, inv, Agt, Act, E, guard, jump, Trig, trig_0), \tag{2}$$

where $Trig \subset \mathbb{C}_T$ is a finite set of triggers and the function $trig_0 : Agt \to Trig$ assigns each agent an initial trigger. The remainder of the definition is a hybrid game as defined in Def. 2.4.

Semantics. The triggers of all players are updated with each action transition. The current triggers are expressed in the triggers function *trig*. The initial triggers are defined by the function $trig_0$. To keep track of the triggers throughout the game, we add the function *trig* to the game configuration. A configuration in an HGT is a triple $\langle l,\ v,\ trig\rangle$. We extend the definition of the *time to trigger* function to configurations:

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

The HGT semantics are interpreted over a labelled transition system similar to the concurrent hybrid games. The set of all configurations is called $Config_T$ and represents the states of the transition system. The transitions of an HGT are defined as follows:

Time transition: $\langle l, v, trig\rangle \xrightarrow{t} \langle l, v', trig\rangle$ is legal if it is legal in the hybrid game definition, and $t \le ttt(\langle l, v, trig\rangle)$.

Action transition: $\langle l, v, trig\rangle \xrightarrow{d} \langle l', v', trig'\rangle$ is legal if it is legal in the hybrid game, and $d(i) = \bot$ for every agent $i \in Agt$ with $v \nvDash trig(j)$.

At configuration $\langle l, v, trig\rangle$, if $v \vDash trig(i)$ for some agent $i \in Agt$, each agent chooses a pair (a, φ) where a is the action the agent chooses, and φ is the trigger for their next action. Agents $j \in Agt$ with $v \nvDash trig(j)$ can only choose the stutter action $\bot$. In an action transition $\langle l, v\rangle \xrightarrow{d} \langle l', v'\rangle$, we say an agent $i \in Agt$ with $v \vDash trig(i)$

is *to blame* for the transition, while agents $j \in Agt$ with $v \not\models trig(j)$ are *blameless*.

Example 3.6. We extend the hybrid game from Example 3.1. The robots use two constraints as triggers: one to stay safe $\texttt{safe}(i)$ and one to reach their goals $\texttt{turn}(i, p)$ for the robot $i \in Agt$ and some intersection or endpoint p. The $\texttt{safe}(i)$ trigger ensures that robot i reacts in time before it collides with another robot:

$$\texttt{safe}(i) \equiv \texttt{free-ahead}(i) \leq \texttt{safety-distance}$$

We assume that `safety-distance` is a constant distance the robots must maintain as an emergency braking distance and $\texttt{free-ahead}(i)$ is a sensor measure of the empty space in front of the robot i. The `turn` triggers allow the robots to turn at an intersection or an endpoint to navigate towards their goals. The trigger $\texttt{turn}(i, (h_1, v_1))$ allows robot i to turn at the intersection (h_1, v_1):

$$\texttt{turn}(i, (h_1, v_1)) \coloneqq 1 + r \leq x_i \leq 2 - r \ \wedge \ 1 + r \leq y_i \leq 2 - r$$

Similarly, the trigger $\texttt{turn}(i, (h_1, 0))$ allow that the robot i to turn at the start of the aisle h_1:

$$\texttt{turn}(i, (h_1, 0)) \equiv x_i = r \ \wedge \ 1 + r \leq y_i \leq 2 - r$$

For the end of aisles, endpoints, the number 0 is replaced with 5.

The set *Trig* can now be defined as follows:

$$Trig \coloneqq \{\texttt{safe}(i) \wedge \texttt{turn}(i, p) \mid i \in Agt, p \text{ is an intersection or an endpoint}\}$$

The initial trigger function $trig_0$ is defined as follows:

$$trig_0(B) = \texttt{safe}(B) \wedge \texttt{turn}(B, (h_2, v_1))$$
$$trig_0(R) = \texttt{safe}(R) \wedge \texttt{turn}(R, (h_1, v_2))$$
$$trig_0(G) = \texttt{safe}(G) \wedge \texttt{turn}(G, (h_2, 5))$$

The triggers for the robots B and R are satisfied at the same time (circles with the label 1 in Fig. 2). Therefore, these two robots can take concurrent actions, creating the following decision function:

$$d \coloneqq \{B \mapsto right, R \mapsto left, G \mapsto \bot\}$$

Each robot chooses a new trigger, creating the new trigger function $trig_1$:

$$trig_1 \coloneqq \{B \mapsto \texttt{safe}(B) \wedge \texttt{turn}(B, (h_2, v_2)),$$
$$R \mapsto \texttt{safe}(R) \wedge \texttt{turn}(R, (h_1, v_1)),$$
$$G \mapsto \texttt{safe}(G) \wedge \texttt{turn}(G, (h_2, 5))\}$$

Fig. 2 shows how the game progresses further in the circles labelled with 2. In this running example, note how guards and invariants set fixed conditions for when actions *can* occur, while triggers define when the agents *want* or *need* to take actions and can be updated throughout the game. Consider the following explanation for this: a robot can change its speed at any point and may have the option to turn at multiple intersections. However, triggers determine which intersection the robot will take to reach its destination and under which conditions it will change its speed.

4 Time Abstract Game with Triggers

We introduce a discrete game abstraction of a hybrid game with triggers (HGT). We call this discrete game *time-abstract game with triggers* (TAGT). TAGT consists of a countable state space. An agent has a winning strategy in the TAGT if and only if that agent has a winning strategy in the HGT.

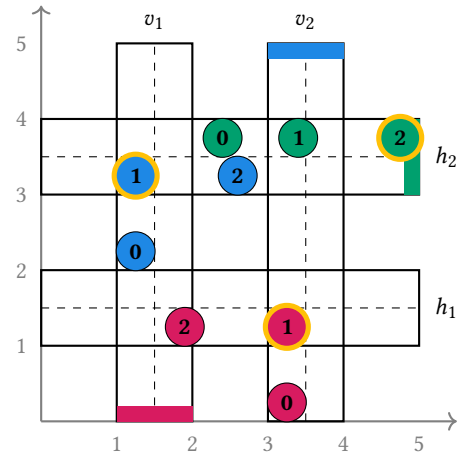


Figure 2: Progression of an HGT representing 3 robots in a warehouse. Yellow circles indicate satisfied triggers.

Intuitively, the discrete game takes advantage of the reduced nondeterminism brought by the triggers: in the TAGT, the times where none of the triggers are satisfied are skipped over, knowing that no actions can be taken in these times. Fig. 3 illustrates this intuition. This ensures that the game progresses solely based on agents' actions and the conditions defined by their triggers, creating a predictable path through the game states. Fig. 2 shows the progression of the HGT from Example 3.6 in discrete steps.

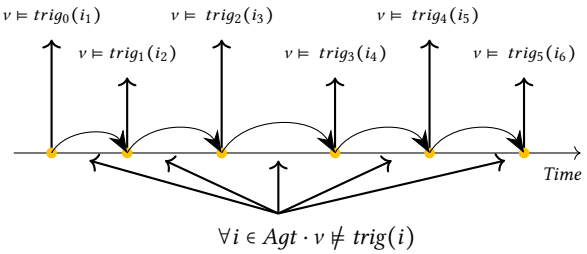


Figure 3: Skipping over times where no trigger is satisfied.

It is important to note that HGT does not restrict the discrete or continuous dynamics of the hybrid games, except for the assumption of the differential equations' solvability. Maintaining the hybrid game's expressive power means that the reachability question in the abstract discrete game remains semi-decidable.

4.1 TAGT as Instance of CGS

We model the TAGT as a concurrent game structure (cf. Def. 2.1). This game model fits intuitively to the TAGT and serves as the semantic model for ATL* [3] and strategy logic [14].

Definition 4.1 (TAGT as an instance of CGS). Given an HGT $\mathcal{G}$:

$$\mathcal{G} \coloneqq (Loc, l_0, X, v_0, flow, inv, Agt, Act, E, guard, jump, Trig, trig_0),$$

the following CGS $\mathcal{T}$ is called a time-abstract game with triggers TAGT:

$$\mathcal{T} = (S, s_0, Agt, Act_w, AP, label, \delta) :$$

- The agent set Agt is the same as in the hybrid game.
- The action set Act_w contains all pairs in $Act \times Trig$ and the action $wait$, which players must take to allow for a time transition:

$$Act_w = (Act \times Trig) \cup \{wait\}$$

- The set of states $S \subset Config_T$ is a countable subset of configurations of the HGT. S can be inductively defined as follows:
  - The initial state is the initial configuration $s_0 = \langle l_0, v_0, trig_0 \rangle \in S$
  - If $s = \langle l, v, trig \rangle \in S$, then:
    $\langle l, v[ttt(s)], trig \rangle \in S$ if $v \not\models trig(i)$ for all $i \in Agt$
    $\langle l', v[J_e], trig' \rangle \in S$ for each $trig' : Agt \to Trig$ if the edge $e = (l, d_e, l') \in E$ is enabled at s in $\mathcal{G}$.
- The transition function δ is defined based on the HGT transitions. Given a state $s = \langle l, v, trig \rangle \in S$, then:
  Case $v \not\models trig(i)$ **for any** $i \in Agt$**:**
  $\delta(s, d_{wait}) = \langle l, v[\mathrm{ttt}(s)], trig \rangle$ where $d_{wait}(i) = wait$ for all $i \in Agt$.
  Case $v \models trig(i)$ **for some** $i \in Agt$**:**
  $\delta(s, d'_e) = \langle l', v[J_e], trig' \rangle$ for all $e = (l, d_e, l')$ enabled at s and all $trig' : Agt \to Trig$, such that for each agent $j \in Agt$, $d'_e(j) = (a_j, \varphi_j)$ where $\varphi_j = trig'(j)$ and $a_j = d_e(j)$.
- $AP = Loc \cup Trig$
- The labelling function $label$ is defined as follows:

$$label : \langle l, v, trig \rangle \mapsto \{l\} \cup \{trig(i) \mid i \in Agt.\ v \models trig(i)\}$$

The state space can be visualized as a *states tree* for which we show the top three layers in Fig. 4. States with no satisfied triggers have only one child, whereas states with at least one satisfied trigger have a child for each enabled edge and each trigger function $trig : Agt \to Trig$. This makes each layer of the tree finite, and thus, the state space overall is countable. FEach

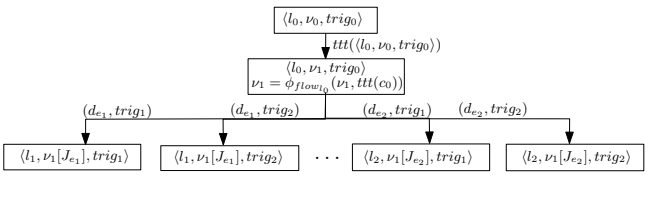


Figure 4: First layers of the discrete game tree.

4.2 Excluding Zeno Behaviour

The winning conditions discussed in [7] and [6] prevent an agent from winning by blocking the progression of time, the so-called Zeno behaviour. We describe how these winning conditions can be adapted to the TAGT in the form of an ATL* [3] formula.

A play is winning for agent i if it satisfies the desired outcome ϕ_i (expressed as an ATL* property over AP) and time diverges td or time converges tc, provided i is not to blame for the time convergence. These conditions ensure that an agent cannot win by halting time, thereby encouraging realistic strategies.

To express these winning conditions, we assume the existence of a global clock $z \in X$, which progresses continuously without interruption, i.e. is never updated in any edge and has the flow $z' = 1$ in all locations. Furthermore, we extend the atomic propositions of TAGT with the following propositions:

- *tick*: The proposition *tick* is *True* if the state is reached through a time transition, and this transition causes the global clock z to advance by an integer value. That is, *tick* is *True* in state $s \in S$ if there exists a state $r \in S$ such that $\delta(r, d_{wait}) = s$ and $\lfloor \nu_s(z) \rfloor - \lfloor \nu_r(z) \rfloor = 1$, where ν_s and ν_r are the valuations at states s and r, respectively.
- $Blame = \{blame_1, \ldots, blame_k\}$: Each $blame_i$ is *True* if the state is reached via an action transition where agent $i \in Agt$ is responsible for the transition. Specifically, $blame_i$ is *True* in state $s \in S$ if there exists a state $r \in S$ with $\delta(r, d) = s$ and $d(i) = (a, \varphi)$ where $a \neq \perp$.

The existence of a winning strategy for a coalition of agents $I \subseteq Agt$ can be expressed using the ATL* notation:

$$\langle\!\langle I \rangle\!\rangle (\Phi_I \wedge \Box \Diamond tick) \vee (\Diamond \Box \neg tick \wedge \bigwedge_{i \in I} \Diamond \Box \neg blame_i)$$

In the above formula, Φ_I is the desired outcome of the agents in I. The $\langle\!\langle I \rangle\!\rangle$ is an ATL* operator that says *there exists a strategy for the coalition I to enforce the following formula.* This formula ensures that $tick$ appears infinitely often, i.e. time diverges, or if time converges, the agents in I are not to blame for the convergence. robot

5 Conclusion

We introduce Hybrid Games with Triggers (HGT), a novel extension of hybrid games that incorporates agents' rationale into the game model. This extension reduces nondeterminism, enabling the discretization of hybrid games. Importantly, HGT achieves this without imposing restrictions on system dynamics. HGT uses triggers to focus only on active states. In this way, our approach simplifies reasoning about safety and reachability in multi-agent CPS.

Our work opens new directions for more efficient verification and analysis of autonomous CPS, potentially influencing future work on game-based models for complex CPS interactions. Specifically, we recognize two important directions for future research on the application of HGT:

Bounded HGT: We show how the time abstract game with triggers (TAGT) reduces an HGT to a discrete game with a countable set of states. Nonetheless, in a TAGT, reachability remains undecidable. Therefore, to reduce a TAGT to a finite number of states, we propose introducing termination conditions, e.g. a time-bound. However, this is only possible if Zeno-runs are excluded. In future research, we will explore realistic termination conditions and explore the decidability of bounded HGTs.

AI Integration: HGT can be integrated into AI systems to improve verification and explainability. Triggers, which function as IF-THEN statements, are straightforward to extract in rule-based AI, but they are often less transparent in learning-based AI. Future work could focus on incorporating triggers into AI to enhance interpretability. For example, an AI could be trained to identify or create triggers, allowing it to operate according to HGT semantics by remaining inactive until a trigger condition is met. Alternatively, triggers could be learned from the AI's behaviour, providing a basis for explaining or verifying the system's properties.

Acknowledgments

This research was supported by the Innovation Campus for Future Mobility (www.icm-bw.de) and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263 (www.sfb1608.kit.edu/index.php). A special thanks to our in-house textician.

References

[1] Rajeev Alur. 1999. Timed automata. In *Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Proceedings* (Lecture Notes in Computer Science). Nicolas Halbwachs and Doron A. Peled, (Eds.) Vol. 1633. Springer, 8–22. doi:10.1007/3-540-48683-6_3.

[2] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. 1995. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138, 1, 3–34. doi:10.1016/0304-3975(94)00202-T.

[3] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time temporal logic. *J. ACM*, 49, 5, 672–713. doi:10.1145/585265.585270.

[4] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. 2000. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88, 7, 971–984. doi:10.1109/5.871304.

[5] Patricia Bouyer, Thomas Brihaye, and Fabrice Chevalier. 2010. O-minimal hybrid reachability games. *Log. Methods Comput. Sci.*, 6, 1. http://arxiv.org/abs/0911.4833.

[6] Thomas Brihaye, François Laroussinie, Nicolas Markey, and Ghassan Oreiby. 2007. Timed concurrent game structures. In *CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings* (Lecture Notes in Computer Science). Luís Caires and Vasco Thudichum Vasconcelos, (Eds.) Vol. 4703. Springer, 445–459. doi:10.1007/978-3-540-74407-8_30.

[7] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. 2003. The element of surprise in timed games. In *CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003, Proceedings* (Lecture Notes in Computer Science). Roberto M. Amadio and Denis Lugiez, (Eds.) Vol. 2761. Springer, 142–156. doi:10.1007/978-3-540-45187-7_9.

[8] Sheetal Dharmatti and Mythily Ramaswamy. 2006. Zero-sum differential games involving hybrid controls. *Journal of optimization theory and applications*, 128, 1, 75–102.

[9] Valentin Goranko and Govert van Drimmelen. 2006. Complete axiomatization and decidability of alternating-time temporal logic. *Theor. Comput. Sci.*, 353, 1-3, 93–117. doi:10.1016/J.TCS.2005.07.043.

[10] Qais Hamarneh. 2024. Hybrid games with triggers. In *Proceedings of the PhD Symposium at the 19th International Conference on integrated Formal Methods in Manchester 2024 (iFM 2024), Manchester, United Kingdom, November 12, 2024* (CEUR Workshop Proceedings). Vol. 3860. CEUR-WS.org, 7–13. https://ceur-ws.org/Vol-3860/paper%5C_2.pdf.

[11] Thomas A. Henzinger. 1995. Hybrid automata with finite bisimulations. In *Automata, Languages and Programming*. Zoltán Fülöp and Ferenc Gécseg, (Eds.) Springer Berlin Heidelberg, Berlin, Heidelberg, 324–335. ISBN: 978-3-540-49425-6.

[12] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. 1999. Rectangular hybrid games. In *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings* (Lecture Notes in Computer Science). Jos C. M. Baeten and Sjouke Mauw, (Eds.) Vol. 1664. Springer, 320–335. doi:10.1007/3-540-48320-9_23.

[13] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. 1998. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57, 1, 94–124. doi:10.1006/JCSS.1998.1581.

[14] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. 2014. Reasoning about strategies: on the model-checking problem. *ACM Trans. Comput. Log.*, 15, 4, 34:1–34:47. doi:10.1145/2631917.

[15] Anil Nerode, Jeffrey B. Remmel, and Alexander Yakhnis. 1996. Hybrid system games: extraction of control automata with small topologies. In *Hybrid Systems IV, Proceedings of the Fourth International Workshop on Hybrid Systems, Ithaca, NY, USA, October 1996* (Lecture Notes in Computer Science). Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, (Eds.) Vol. 1273. Springer, 248–293. doi:10.1007/BFB0031565.

[16] André Platzer. 2022. Correction to: differential dynamic logic for hybrid systems. *J. Autom. Reason.*, 66, 1, 173. doi:10.1007/S10817-021-09608-W.

[17] André Platzer. 2008. Differential dynamic logic for hybrid systems. *J. Autom. Reason.*, 41, 2, 143–189. doi:10.1007/S10817-008-9103-8.

[18] André Platzer. 2015. Differential game logic. *ACM Trans. Comput. Log.*, 17, 1, 1. doi:10.1145/2817824.

[19] André Platzer. 2017. Differential hybrid games. *ACM Trans. Comput. Log.*, 18, 3, 19:1–19:44. doi:10.1145/3091123.

[20] Jan-David Quesel and André Platzer. 2012. Playing hybrid games with keymaera. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings* (Lecture Notes in Computer Science). Bernhard Gramlich, Dale Miller, and Uli Sattler, (Eds.) Vol. 7364. Springer, 439–453. doi:10.1007/978-3-642-31365-3_34.

[21] Claire J. Tomlin, John Lygeros, and S. Shankar Sastry. 2000. A game theoretic approach to controller design for hybrid systems. *Proc. IEEE*, 88, 7, 949–970. doi:10.1109/5.871303.

[22] Vladimeros Vladimerou, Pavithra Prabhakar, Mahesh Viswanathan, and Geir E. Dullerud. 2011. Specifications for decidable hybrid games. *Theor. Comput. Sci.*, 412, 48, 6770–6785. doi:10.1016/J.TCS.2011.08.036.