

Context-Triggered Contingency Games for Strategic Multi-Agent Interaction

Kilian Schweppe¹ and Anne-Kathrin Schmuck¹

Abstract—We address the challenge of reliable and efficient interaction in autonomous multi-agent systems, where agents must balance long-term strategic objectives with short-term dynamic adaptation. We propose context-triggered contingency games, a novel integration of strategic games derived from temporal logic specifications with dynamic contingency games solved in real-time. Our two-layered architecture leverages strategy templates to guarantee satisfaction of high-level objectives, while a new factor-graph-based solver enables scalable, real-time model predictive control of dynamic interactions. The resulting framework ensures both safety and progress in uncertain, interactive environments. We validate our approach through simulations and hardware experiments in autonomous driving and robotic navigation, demonstrating efficient, reliable, and adaptive multi-agent interaction.

I. INTRODUCTION

Autonomous systems are becoming increasingly prevalent and already engage in complex interactions among each other and with humans – such as self-driving cabs entering city traffic or factory robots cooperating with humans to assemble products. While reliability of autonomous system interaction is crucial in such situations and has received overwhelming attention in recent years [1], [2], also *performance* is needed to not make complex procedures stall under increasing autonomous participants.

As an example, consider multiple autonomous cars driving on a highway. When only concerned with safety by avoiding collisions with other cars, an agent might miss its highway exit as it is unable to *actively* enforce merging to the right lane in time. Similarly, autonomous warehouse robots can be caught in a live-lock dynamically giving right to each other, without enforcing progress towards their destination¹.

These examples demonstrate that autonomous system interaction requires both (high-level, long-term) *strategic interaction* – i.e., merging to the right lane when the exit comes closer or always eventually dropping a parcel correctly – and (low-level, short-term) *dynamic interaction* – i.e., anticipating how other agents will give way to actually achieve strategic goals. In addition, this behavior should reliably emerge while agents can only *observe* other agents' behavior in real-time and have no access to their goals.

This paper presents a practical solution to this multi-agent control problem via a novel integration of (high-level) *strategic* and (low-level) *dynamic games* into a so-called *context-triggered contingency game* with a real-time capable

model predictive control (MPC) implementation. The resulting controller ensures the satisfaction of a reactive strategic specification of the ego agent given in linear temporal logic (LTL) while locally optimizing strategic choices based on the estimated actions of other agents through their dynamic interaction. The resulting efficient and reliable interaction of agents is demonstrated in an autonomous driving simulation and on robotic hardware.

A. Related Work

Layered control architectures are known to be essential to address the increasing complexity of autonomous systems [3] while algorithmic game theory [4] is known to capture the mutual influence of interacting agents in a concise fashion. While (high-level, long-term) strategic interactions are usually captured by (logical) games on graphs [5], [6], (low-level, short-term) dynamic interactions are usually captured by (continuous) dynamic games [7]. Existing work usually integrates *one* such game into one layer of a multi-layered control architecture and ignores game-like aspects in other layers. To the best of the authors' knowledge, this is the first work which integrates both types of games across layers while still achieving a real-time capable implementation.

(*High-level*) *Strategic Games*.² There exists a large body of work on the use of *strategic games* derived from LTL specifications in higher layers of multi-level controllers, recently reviewed in [8] and discussed in recent textbooks [9], [10], [11]. Importantly, most solutions consider only a *single agent* which either (i) only interacts with boolean aspects of the environment (i.e., adapts to a door being open/closed, or an alarm being on/off), or (ii) only executes a non-reactive task plan implemented via control barrier functions to allow online adaptation to dynamic uncertainty. A recent exception is [12] which is based on a new representation of strategies in the form of *strategy templates* [13], which are then translated into control Lyapunov functions. While this allows for both boolean interaction with the environment and dynamic adaptations, it is only applicable to a single agent.

In *multi-agent settings* typically a centralized game is solved and the resulting action profile is implemented via (decentralized or multi-agent) control barrier functions, see e.g. [14], [11], [15]. Crucially, these works pre-compute an action profile given specifications for all players which is then realized dynamically. In contrast, we only consider

¹Max Planck Institute for Software Systems, Kaiserslautern, Germany
{kschweppe, akschmuck}@mpi-sws.org

¹See <https://youtu.be/AAhHmqUT9Fs>

²We note that we use the term *strategic game* to refer to a (multi-round) reactive graph game – different from (one-shot) strategic normal-form games.

knowledge of the ego agent’s specification and therefore defer large parts of the strategy selection to the online phase when actions of other agents can be inferred from observations. However, in contrast to multi-level planning & MPC frameworks like [16], [17] we guarantee the eventual satisfaction of the ego agent’s objective without replanning.

(Low-level) Dynamic Games. On the other end of the spectrum, dynamic games arise from multi-agent feedback control problems in highly interactive domains such as autonomous driving [18] and racing scenarios [19], [20]. To practically apply them for feedback control, the resulting nonlinear optimal control problem must be solved in real-time, which induces severe computational challenges. ALGAMES [21] was the first general-purpose solver capable of handling nonlinear equality and inequality constraints. While showing good performance, it can struggle with convergence in scenarios with complex dynamics and constraints [20]. More recently, the dynamic game sequential quadratic programming (DG-SQP) solver was introduced [20]. It shows improved solution quality compared to ALGAMES, but still suffers from long solving times.

While this makes the solution of dynamic games already computationally intense, the situation becomes even more complicated if uncertainty about the behavior of other agents must be taken into account. To address this problem, branch MPC [22] and topology-driven MPC [23] generate multiple independent trajectories and select the most suitable one based on a risk measure. However, these approaches cannot smoothly blend trajectories under high uncertainty. To address this, contingency planning approaches such as [24] were introduced, which have recently been extended to *contingency games* [25], which incorporate multiple interactive future scenarios into a single dynamic game.

While [18] investigated the multi-level solution of a (classical) dynamic game, we are not aware of any existing integration of contingency games into a multi-level control framework, as presented in this paper. In addition, none of the existing solvers for dynamic games is able to handle the complexity of the resulting optimal control problems in real-time. To address this shortcoming, an important contribution of this paper is the development of a novel factor-graph-based approximate solver for contingency games.

B. Contribution & Outline

The key contribution of this paper is the implementation of a real-time capable two-layered reactive planning and control architecture which instantiates and solves a contingency game per strategic game node, leading to a dynamic sequence of contingency games driven by the long-term strategic (LTL) objective of the ego agent (see Fig. 1 for an illustration).

Concretely, this paper presents: 1) the integration of strategy templates [13] and contingency games [25] into a multi-layered *context-triggered contingency game*, which is formalized in Sec. III and illustrated for a robot application scenario in Sec. II; 2) the development of a novel factor-graph-based solver which is demonstrated to outperform existing solutions to contingency games in both speed and scalability

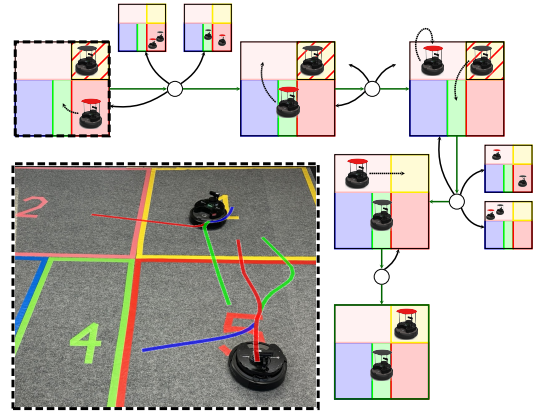


Fig. 1: Mobile Robot Navigation Scenario. Top right: Part of the strategic game graph. Box and circle nodes are owned by environment and ego agent, respectively. Bottom left: Contingency game solution for the dynamic game induced by the top-left context (dashed boundary). Matching colored trajectories illustrate matching intents.

and which enables a real-time capable MPC formulation of the resulting feedback controller, as discussed in Sec. IV; and 3) an experimental validation of the resulting controller on hardware and in simulation presented in Sec. V.

II. PROBLEM SETUP

We focus on scenarios where multiple agents interact both strategically and physically and each agent has its own objective – unknown to the others. Throughout this paper, we take the view of an ego agent with a fixed (long-term) strategic objective. We consider two application scenarios.

(1) A *mobile robot* navigating an unstructured environment containing unknown, potentially moving obstacles (e.g., humans, other agents, or furniture) to fulfill temporal tasks in interaction with other agents and humans. This scenario is illustrated in Fig. 1 and will be used as a running example throughout this paper. The resulting controller was implemented and tested on hardware (see Sec. V).

(2) An *autonomous driving* scenario, where an ego vehicle has to maintain a constant cruising speed while always moving back to the right lane again. This scenario is further described in Sec. V and the resulting controller was implemented and tested in simulation.

A. Interaction Models

(High-level) Reactive Planning Domain. We model all relevant strategic interactions between the (controlled) ego agent and (uncontrolled) agents in its environment as a reactive planning domain $\mathcal{D} = \langle S, s_0, A, AP_D, L \rangle$, where: (i) $S = S_c \cup S_u$ is a set of *contexts*, partitioned into controlled contexts S_c owned by the ego agent and uncontrolled contexts S_u owned by environment agents; (ii) $s_0 \in S$ is the initial context; (iii) $A = A_c \cup A_u$ is the set of actions (modeled as directed edges), partitioned into controlled actions $A_c \subseteq S_c \times S_u$ and uncontrolled actions $A_u \subseteq S_u \times S_c$; (iv) AP_D is a set of boolean variables; (v) $L: S \rightarrow 2^{AP_D}$

is a labeling function that labels each context with a set of propositions which are true in this context.

The planning domain can for example be specified using the planning domain description language (PDDL) [26], a standard language in the field of autonomous systems.

(Low-level) System Dynamics. We assume that all agents p can be modeled by continuous-time affine dynamics

$$\dot{x}^p = f^p(x^p) + g^p(x^p)u^p, \quad (1)$$

where $X \subseteq \mathbb{R}^n$ is the set of admissible physical states, $U \subseteq \mathbb{R}^m$ is the set of admissible inputs, and the functions $f^p : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g^p : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are locally Lipschitz continuous. We assume the existence of a map $\ell^p : X \rightarrow 2^{AP_S^p}$ which maps every continuous state of agent p to a set of its system propositions AP_S^p , i.e., the truth value of these propositions is dependent on the current (continuous) state of an agent and can be actively changed by controlling the dynamics of agents.

Example 1 (Robot Domain and Dynamics): We consider a simplified reactive planning domain consisting of five regions Q_1, \dots, Q_5 as depicted in Fig. 1, a controlled agent R_c , and an uncontrollable environment agent R_u . The states of the planning domain model all possible locations of both agents, and their possible actions to leave or enter particular regions induced by the adjacency of workspace regions. The *state propositions* are naturally defined for $1 \leq i \leq 5$ s.t. c_i, u_i, h_u are set to true if and only if agent R_c , agent R_u or a human is in region Q_i , respectively. Then we have a simple map ℓ^p s.t. $\ell^p(x^p) = p_i$ if $x^p \in Q_i$ for $p \in \{u, c\}$.

Depending on the application, the reactive planning domain \mathcal{D} might need to be refined to truthfully reflect the interaction of AP_S and AP_D . To simplify the presentation, we assume $AP_S \subseteq AP_D$ throughout this paper.

B. (High-level) Strategic Games.

Strategic Tasks as LTL Specifications. To express a strategic task for the ego agent, we use *linear temporal logic* (LTL) [27], which has been extensively used for strategic reactive planning [28]. Given a set AP_φ of atomic task propositions, LTL formulas are recursively defined as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \text{ U } \varphi_2$$

where $p \in AP_\varphi$ is an atomic proposition; \neg and \wedge are the boolean operators' negation and conjunction, respectively; and \bigcirc and U are the temporal operators 'next' and 'until', respectively. Other standard operators such as disjunction (\vee), implication (\Rightarrow), finally (\diamond), and globally (\square) can be derived from the above operators. The semantics of LTL formulas are defined over infinite sequences of sets of atomic propositions and can be found in standard books [27].

Example 2 (Robot Specification): In contrast to state propositions, *task propositions* can additionally model state-independent aspects, e.g. that both agents can be requested to *leave* or *enter* a particular region Q_i by the other agent or by a human operator. Here, the task proposition enter_i^c is true if agent R_c is requested to enter region Q_i , with leave_i^c , leave_i^e and enter_i^c being defined analogously.

As a simple strategic specification for R_c consider the task of always fulfilling an issued enter request for any region, while (i) always avoiding obstacles, (ii) not sharing Q_1 with R_u , and (iii) not sharing Q_4 with a human. This can be formalized by the LTL specification

$$\varphi_G = \left(\bigwedge_{i=1, \dots, 5} \diamond \square \text{enter}_i^c \Rightarrow \diamond c_i \right) \wedge (u_1 \Rightarrow \neg c_1) \wedge (h_4 \Rightarrow \neg c_4). \quad (2a)$$

In addition, we assume that R_u respects a leave request for region Q_1 only when R_c is in region Q_2 , formalized via

$$\varphi_A = (\text{leave}_1^e \Rightarrow \bigcirc \neg u_1) \wedge (\bigcirc \text{leave}_1^e \Rightarrow c_2). \quad (2b)$$

Combining (2a) and (2b) yields the strategic objective $\varphi = \square \varphi_A \Rightarrow \square \varphi_G$ requiring that the ego agent fulfills φ_G when interacting with another agent fulfilling φ_A .

As illustrated above, the satisfaction of a specification is usually not under full control of the ego agent, as some propositions cannot be influenced, e.g. whether R_u enters Q_2 . The strategic cooperation needed to fulfill its specification is captured by a strategic game, as discussed next.

Strategic Two-Player Games. A (high-level) strategic game is a pair $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$, where $\mathcal{D} = \langle S, s_0, A, AP, L \rangle$ is a reactive planning domain, and $\Omega \subseteq S^\omega$ is a set of infinite sequences of contexts that defines the winning condition of the game. A strategic game is typically constructed as the product of an original planning domain \mathcal{D}' (e.g. derived from PDDL) and an LTL specification φ translated into an automaton \mathcal{A} with a winning condition Ω' (see [27] for details), potentially refined by abstractions of the underlying dynamics (1).

The resulting strategic game $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$ is played between the ego agent and the environment agents who move a token over the graph, which models their context-dependent strategic interaction. That is, the agent who owns a context chooses an action and therefore determines the next context.

Example 3 (Strategic Game): An illustrative part of the game graph for the robot scenario is depicted in Fig. 1 (top). The illustration hides the annotation of each context by the current active enter request, which determines which state of the graph needs to be reached by R_c to fulfill φ .

Winning Strategies. The solution of a strategic game $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$ (formalized in Sec. III) equips the ego agent with a strategy to choose actions over time. The problem with the implementation of this solution, however, is that all agents interact highly dynamically. In reality, cars or robots do not patiently wait for others to complete their move such that an action can be taken under full observation. Instead, every agent is moving at the same time and other agents' actions must be anticipated in real-time. This paper provides a novel solution to this problem by equipping every ego-agent context with a contingency game.

C. (Low-level) Contingency Games

A contingency game [25] is a dynamic game that takes multiple possible intents (i.e., discrete actions) Θ^p of players into account. The solution is obtained by solving a carefully constructed optimal control problem, determining the next

input u_{t+1} for the ego agent. In addition to predicting actions of other players, the formulation allows to include additional low-level objectives, such as collision avoidance, trajectory smoothness or energy consumption limitations.

Example 4 (Robot Contingency Game): Consider the top left context in Fig. 1 where $u_1, c_5, enter_c^1$ are true, i.e., the ego agent is in region 5 and requested to enter region 1, which is occupied by the environment agent. As both agents are not allowed to stay in Q_1 together, the ego agent has different optimal strategies, depending on the movement of the environment agent. If the environment agent stays in Q_1 (blue), R_c should proceed towards Q_4 and then Q_2 to request R_u to leave Q_1 (blue). If the environment agent is however already leaving Q_2 (red or green) the ego agent should proceed towards Q_1 (red or green), but not enter until the other agent has left. The resulting contingency game computes beliefs over the intents of the environment agent and computes a weighted blend of the resulting ego-agent trajectories to determine its next control input u_{t+1} .

III. COMBINING STRATEGIC & DYNAMIC GAMES

Given the problem setup discussed in Sec. II, we now formalize the integration of strategic and contingency games. This is done by deferring the choice of strategic actions of the ego agent to the online phase, when observations can be used to infer the current actions of other agents. This is enabled by utilizing a novel, permissive representation of winning strategies for strategic games, called strategy templates [13].

A. (High-level) Strategy Templates

In two-player games, a *strategy template* generalizes the notion of a strategy by succinctly representing an infinite family of strategies through local constraints.

Strategies. Let $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$ be a strategic game. Then a run $\rho = s_0 s_1 s_2 \dots \in S^\omega$ over \mathcal{D} is an *infinite* sequence of contexts such that $s_0 = s_0$ and for all $i \geq 0$, $(s_i, s_{i+1}) \in A$, i.e., there is an action that takes the system from context s_i to context s_{i+1} . A *control strategy* $\pi_c: S^* S_c \rightarrow A_c$ is a function that maps a sequence of contexts (representing the history of the interaction) to the action that the ego agent will take if it is their turn. A run $\rho = s_0 s_1 s_2 \dots$ is said to be a π_c -run if for all $i \geq 0$, whenever $s_i \in S_c$, then $s_{i+1} = \pi_c(s_0 s_1 \dots s_i)$. A control strategy π_c is called *winning* if all π_c -runs are contained in Ω . The construction of a game \mathcal{G} from an LTL specification φ ensures that this implies that all π_c -runs satisfy φ .

Parity Games. In this paper we consider the class of *parity games* [27] which result from the translation of an LTL formula into a game $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$. In a parity game the winning condition $\Omega = Parity[c]$ is defined by a coloring function $c: S \rightarrow \mathbb{N}$ that assigns a natural number (color) to each state, and a run ρ belongs to $Parity[c]$ if the maximum color that appears infinitely often in ρ is even.

Strategy Templates. Given a strategic game $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$, a *strategy template* [13] is a tuple $\Psi = (U, C, \mathcal{H})$ consisting of three disjoint sets of actions controlled by the ego agent:

(i) *Unsafe actions* $U \subseteq A_i$: actions that the agent is prohibited from taking; (ii) *Co-live actions* $C \subseteq A_i$: actions that may only be taken finitely many times along any run; (iii) *Live-groups* $\mathcal{H} \subseteq 2^{A_i}$: sets of actions such that, if the source state of some $H \in \mathcal{H}$ is visited infinitely often, the agent must take at least one action from the set infinitely often. A run ρ is said to comply with a strategy template Π , if it satisfies all the specified constraints. A strategy π is said to follow a strategy template Π , denoted $\pi \models \Pi$, if all π -runs comply with Π . A strategy template Π is called *winning* if all strategies that follow it are winning. For a detailed formal definition, further intuition and an algorithm to compute a winning strategy template see [13].

B. (Low-level) Contingency Games

We assume a set of agents P where each agent $p \in P$ has a fixed set of intents Θ^p . We assume that for each agent p and for each intent $\theta \in \Theta^p$ its states evolve according to the dynamic constraint $x_{\theta,t+1}^p = f^p(x_{\theta,t}^p, u_{\theta,t}^p)$. For simplicity, we assume the state dimensions n_θ^p and input dimensions m_θ^p are equal to n and m for all players and intents. For ease of notation, we use the shorthand notations $\mathbf{x}_\theta^p = (x_{\theta,t}^p)_{t=1}^T$ and $\mathbf{u}_\theta^p = (u_{\theta,t}^p)_{t=1}^T$ to denote the states and inputs of player p for intent θ aggregated over time, as well as $\mathbf{x}_\theta = (\mathbf{x}_\theta^p)_{p=1}^N$ and $\mathbf{u}_\theta = (\mathbf{u}_\theta^p)_{p=1}^N$ to denote the stacked state and input vectors of all players. In addition, the shorthands $\mathbf{x}_\Theta = (\mathbf{x}_\theta)_{\theta \in \Theta}$ and $\mathbf{u}_\Theta = (\mathbf{u}_\theta)_{\theta \in \Theta}$ are used to denote the states and inputs aggregated over all hypotheses.

Dynamic Objectives. We assume the dynamics and intents of players to be coupled such that the objective function $J^p(\mathbf{x}, \mathbf{u}^p)$ of an agent p might depend on the states of other players, but only on the inputs of p . We define $g^p(\mathbf{x}^p, \mathbf{u}^p)$ and $h^p(\mathbf{x}^p, \mathbf{u}^p)$ to be the equality and inequality constraints of player p , respectively, which model *private constraints* that only depend on the states and inputs of player p , and include the dynamics f^p in the equality constraints. Additionally, we define $\bar{g}(\mathbf{x}, \mathbf{u})$ and $\bar{h}(\mathbf{x}, \mathbf{u})$ to be the *shared constraints* that depend on the states and inputs of all players.

Contingency Games. A contingency game [25] is a dynamic game that takes multiple possible intents of all players into account, which must be inferred by the ego agent via observations. This uncertainty is represented by a discrete probability distribution $b(\theta)$ for $\theta \in \Theta$. The joint hypothesis space for all players is defined as $\Theta = \Theta^1 \times \Theta^2 \times \dots \times \Theta^N$. The intents of all players are assumed to be independent, hence the belief $b(\theta)$ can be written as the product of the individual beliefs $b(\theta) = \prod_{p=1}^N b(\theta^p)$.

For environment agents, the formulation of the control problem becomes a classical dynamic game per intent θ formulated as the coupled optimal control problem

$$\begin{aligned} & \underset{\mathbf{x}_\theta^p, \mathbf{u}_\theta^p}{\text{minimize}} && J_\theta^p(\mathbf{x}_\theta, \mathbf{u}_\theta^p) \\ & \text{subject to} && g_\theta^p(\mathbf{x}_\theta^p, \mathbf{u}_\theta^p) = 0, \\ & && h_\theta^p(\mathbf{x}_\theta^p, \mathbf{u}_\theta^p) \leq 0, \\ & && \bar{g}_\theta(\mathbf{x}_\theta, \mathbf{u}_\theta) = 0, \\ & && \bar{h}_\theta(\mathbf{x}_\theta, \mathbf{u}_\theta) \leq 0. \end{aligned} \tag{3}$$

For the ego agent, however, all possible intents are taken into account simultaneously. This is achieved by modifying the objective function to optimize the expected value over all hypotheses, and by duplicating all constraints per intent:

$$\underset{\mathbf{x}_\Theta^c, \mathbf{u}_\Theta^c}{\text{minimize}} \quad \sum_{\theta \in \Theta} b(\theta) J_\theta^c(\mathbf{x}_\theta, \mathbf{u}_\theta^c) \quad (4a)$$

$$\text{subject to} \quad g_\theta^c(\mathbf{x}_\theta^c, \mathbf{u}_\theta^c) = 0 \quad \forall \theta \in \Theta, \quad (4b)$$

$$h_\theta^c(\mathbf{x}_\theta^c, \mathbf{u}_\theta^c) \leq 0 \quad \forall \theta \in \Theta, \quad (4c)$$

$$\bar{g}_\theta(\mathbf{x}_\theta, \mathbf{u}_\theta) = 0 \quad \forall \theta \in \Theta, \quad (4d)$$

$$\bar{h}_\theta(\mathbf{x}_\theta, \mathbf{u}_\theta) \leq 0 \quad \forall \theta \in \Theta, \quad (4e)$$

$$u_{\theta_1, t}^c = u_{\theta_2, t}^c \quad \forall \theta_1, \theta_2 \in \Theta, \\ \forall t = 1, \dots, t_b \quad (4f)$$

The last line (4f) is a *contingency constraint* which ensures that all control inputs up to the *branching time* t_b are equal and influenced by all possible contingencies, weighted by the belief for each scenario. After t_b , it is assumed that certainty is obtained about the intent of the other agents, i.e., there exists an intent θ with $b(\theta) = 1$. We refer to [25] for a discussion of the branching time, its computation heuristics, its influence on contingency game solutions.

As in classical optimal control, the solution of (3) determines the trajectory of each player p consisting of a sequence of states $x_1^p, x_2^p, \dots, x_T^p \in \mathbb{R}^n$ and control inputs $u_1^p, u_2^p, \dots, u_{T-1}^p \in \mathbb{R}^m$, if $b(\theta)$ is given. To implement this solution in a model predictive fashion, $b(\theta)$ needs to be estimated from predicted trajectories and u_1^c is used as the next control input for the ego agent in every time step. We refer the reader to [25] for more details.

C. Extracting Context-Dependent Intents

In order to connect strategic games to contingency games, we now discuss how intents for all agents can be extracted from a particular context s (i.e., a particular node in Fig. 1) to formulate a contingency game for this context s .

Environment Agents. We discussed in Sec. II that *state propositions* are directly connected to the low-level dynamics of agents, as their truth value is determined by their (controlled) continuous behavior. Given a strategic game $\mathcal{G} = \langle \mathcal{D}, \Omega \rangle$ and the current context s , we therefore obtain the possible intents of an environment agent p by extracting

$$\Theta_s^p := \{L(s') \cap AP_S^p \mid (s, s') \in A_u\}$$

from \mathcal{D} , yielding a context-dependent set $\Theta_s := \times_{p \in E} \Theta_s^p$.

Ego Agent. Given a context s and a specific environment agent intent $\theta \in \Theta_s$, we observe that the latter is connected to an action $(s, s') \in A_u$ where $s' \in S_c$ is an ego agent context. Given that we have pre-computed a strategy template Ψ over \mathcal{G} , we can directly extract $\Psi|_{s'} = (U|_{s'}, C|_{s'}, H|_{s'})$ as the unsafe, colive and live actions originating in s' .

We now utilize $\Psi|_{s'}$ to derive a context- and intent-dependent objective function $J_{s, \theta}^c(\mathbf{x}_\theta, \mathbf{u}_\theta^c)$ to be used in the context-dependent version of (4). Towards this end, we observe that live actions are those which allow the ego

agent to make progress towards the satisfaction of its (high-level) specification, and should therefore be preferred. We therefore choose a goal state x_{goal} for each live action $(s', s'') \in H|_{s'}$ if $H|_{s'} \neq \emptyset$, s.t. $x_{goal} \in \ell^{c-1}(\alpha)$ where $\alpha \in (L(s'') \setminus L(s')) \cap AP_S^c$, i.e., x_{goal} is a representative goal state (e.g., the center) of the state space region associated with the state proposition α which must become true in s'' but is not yet true in s' . If $H|_{s'} = \emptyset$, we choose an edge $(s', s'') \in A_c \setminus C|_{s'} \setminus U|_{s'}$ instead, which always exists. Given x_{goal} , the objective function $J_{s, \theta}^c$ follows straightforwardly.

D. Extracting Context-Dependent Constraints

While the previous discussion shows how strategy templates allow us to extract local dynamic goals from the strategic game that enable progress towards satisfying a (long-term) objective, it is equally important to ensure that no (context-triggered) safety violation (as specified in φ) occurs. In our example, such safety violations occurred if a region was shared with other agents, i.e. a safety constraint which was only active in a context where another agent was present.

We again use the local strategy template $\Psi|_{s'}$ to include such context-dependent avoid constraints into the contingency game. Towards this goal we define $\kappa := L(s')$ as the current context and $\mathcal{A} := \bigcup_{(s', s'') \in C|_{s'} \cup U|_{s'}} L(s'') \cap AP_S$ as the current propositional avoid set and utilize control barrier functions to formalize their dynamic effect.

Control barrier functions (CBFs) [29] are a popular tool for guaranteeing safety properties in dynamical systems. Typically, CBFs are formulated for continuous time dynamics, and are utilized in the form of safety filters to keep the physical state of a control system inside a safe set. To formalize this notion, consider a set \mathcal{C} defined as the superlevel set of a continuously differentiable function $h : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$, i.e., $\mathcal{C} = \{x \in D : h(x) \geq 0\}$. Here, safety is defined w.r.t. forward invariance, i.e., when starting inside the safe set \mathcal{C} , the system is guaranteed to stay in \mathcal{C} .

In order to define context-dependent avoid constraints we translate both the context κ and the avoid set \mathcal{A} into sets $\mathcal{C}^\kappa \subseteq \ell^{-1}(\kappa)$ and $\mathcal{C}^{\mathcal{A}} \subseteq \ell^{-1}(\mathcal{A})$ with associated CBFs h^κ and $h^{\mathcal{A}}$. In particular, if each proposition $\alpha \in \mathcal{A}$ corresponds to some CBF h^α , and each proposition $\gamma \in \kappa$ corresponds to a CBF h^γ , then $h^{\mathcal{A}}$ and h^κ can be defined as $h^{\mathcal{A}} = \bigwedge_{\alpha \in \mathcal{A}} h^\alpha$ and $h^\kappa = \bigwedge_{\gamma \in \kappa} h^\gamma$.

Context-Dependent Optimization Constraints. Referring back to the intuition that avoid constraints should only hold in a given context, we formalize context-dependent avoid constraints as a Boolean combination of CBFs [30] s.t.

$$[h^\kappa(x) \Rightarrow \neg h^{\mathcal{A}}(x)] = \max(-h^\kappa(x), -h^{\mathcal{A}}(x)).$$

This formulation allows to model the situation that the context changes externally (i.e., a human enters the room) and new avoid constraints are activated. In addition, it is important to note that some constraints may conflict with the current target. To address this, avoid constraints are only activated when the belief for the corresponding context exceeds a specified threshold. From a practical point of

view, the outlined procedure to extract context-dependent constraints results in a large number of often redundant functions h_θ . We therefore simplify the resulting Boolean expressions before solving a contingency game. Simplifying the avoid objectives reduces the number of constraints in the resulting optimization problem, while the combination of identical future behavior minimizes the number of intents.

E. Context-Triggered Contingency Games.

Combining the above constructions, we get a multi-level game, where at every time step $t \in \mathbb{N}$ the context $s(t)$ and state $x^p(t)$ may change for every agent p . The current context $s(t)$ is influenced in real-time by external inputs (i.e., enter requests issued by a human operator), the movement of environment agents (i.e., entering or leaving a specific region), and the movement of the ego agent (i.e., entering or leaving a specific region). The strategic game graph along with the computed strategy templates ensures that the logical dependencies of these context changes w.r.t. the given high-level objective φ induce an optimal control problem at the current time step t (as discussed in Sec. III-C and Sec. III-D), such that the behavior emerging from the real-time solution of this game and its MPC implementation indeed satisfies the ego agent’s objective φ . Note that this implies that one might solve a different game in every time step.

IV. SOLVING CONTINGENCY GAMES EFFICIENTLY

The solution to a dynamic game constitutes a Nash equilibrium (NE), where no player can improve the cost of their strategy without another player having to increase their cost. Here, we are interested in finding a generalized Nash equilibrium (GNE) which generalizes NEs by allowing each player’s strategy to depend on the strategies of other players. This models the interaction of players in the control problem, and allows anticipating the reaction of other players to own actions. Note that this assumes that players act rationally, but non-cooperatively. Hence, solving contingency games requires finding the solution to a Generalized Nash Equilibrium Problem (GNEP).

A. Existing Solvers

Most existing solvers approach this problem by searching for local equilibria only, by first deriving Karush-Kuhn-Tucker (KKT) conditions and rewriting them into a general Mixed Complementarity Problem (MCP). The PATH solver [31] remains the de-facto standard solver for MCP problems. Despite its effectiveness, it was not originally designed with real-time performance in mind, and is not able to provide solutions at high frequency. ALGAMES [21] is an augmented Lagrangian based solver specifically designed for dynamic game applications, yet it does not scale well with the number of players as required for contingency games. More recently, DG-SQP [20] was proposed and demonstrated improved solution quality and robustness compared to both PATH and ALGAMES, particularly for challenging racing scenarios. However, it is still limited by the method’s computational complexity and relatively high solving times, making it unsuitable for real-time applications.

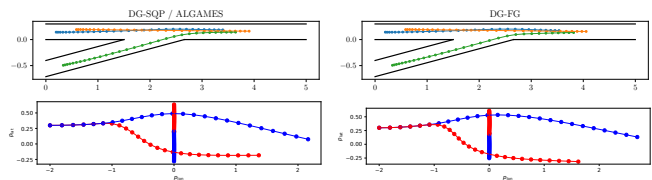


Fig. 2: Solution trajectories for the lane merge (top) and crosswalk (bottom) scenarios. DG-FG solution (right) differs slightly from the reference (left), but remains very similar.

B. A Novel Factor Graph Based Solver

As an alternative to the derivation of an MCP from the KKT conditions, our novel *Dynamic Game Factor Graph* (DG-FG) solver for contingency games utilizes penalty functions to transform the (GNEP) with inequality constraints into an unconstrained, though non-differentiable, optimization problem. This approach has been previously studied in [32], where penalty-based reformulations of GNEPs were shown to provide globally convergent solution methods. The advantage compared to the approach based on the KKT conditions is that no additional variables corresponding to the Lagrange multipliers need to be introduced, which drastically decreases the problem size. Although penalty methods result in nonsmooth terms, they still work well in practice and can be solved efficiently with an appropriate solver.

DG-FG is based on a direct model of the dynamic game by a factor graph, similarly to [33], with the following main advantages: First, the factorization naturally induces sparsity, allowing efficient optimization by considering only the local neighborhoods of each variable. Second, algorithms for solving factor graphs work incrementally, allowing online computation even for large factor graphs.

DG-FG is implemented using SymForce [34], a solver for factor graphs using the Levenberg-Marquardt (LM) algorithm, which computes derivatives symbolically to exploit sparsity and common subterms, and generates optimized C++ code suitable for real-time execution.

C. Benchmarks

We evaluate DG-FG by comparing its performance to DG-SQP, ALGAMES, and PATH using a ramp merge scenario with three cars from [21] and a crosswalk scenario with one pedestrian and one car from [25], all modelled using nonlinear dynamics. Performance is measured by solve time and assessed in a Monte Carlo study in which the initial conditions of each agent are randomly perturbed around nominal values. Each Monte Carlo sample consists of five sequential steps, enabling the solver to warm-start from the previous solution. This procedure provides a more realistic assessment of solver performance in dynamic games.

Fig. 2 shows solutions for two different initial conditions for both scenarios. In both cases, DG-FG trajectories deviate only slightly from the reference and remain qualitatively similar. The similarity can be improved by fine-tuning the cost function weights. Furthermore, Table I lists solve times for both benchmarks. DG-FG provides the fastest solve times by a large margin in all instances.

		$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$
DG-SQP	mean/std (s)	0.017 ± 0.005	0.060 ± 0.012	0.275 ± 0.192	0.485 ± 0.159	0.977 ± 1.143
ALGAMES	mean/std (s)	0.033 ± 0.004	0.133 ± 0.009	0.218 ± 0.030	0.300 ± 0.052	0.408 ± 0.020
DG-FG	mean/std (s)	0.003 ± 0.002	0.015 ± 0.006	0.022 ± 0.014	0.027 ± 0.011	0.039 ± 0.019

		$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$
PATH	mean/std (s)	0.027 ± 0.227	0.066 ± 0.245	0.283 ± 1.321	1.184 ± 3.659	1.744 ± 4.764
DG-SQP	mean/std (s)	0.105 ± 0.092	0.432 ± 0.380	0.870 ± 0.547	1.460 ± 0.994	1.905 ± 1.031
DG-FG	mean/std (s)	0.001 ± 0.001	0.002 ± 0.003	0.004 ± 0.005	0.006 ± 0.005	0.008 ± 0.008

TABLE I: Solve times (mean and standard deviation in seconds) for lane merge (top) and crosswalk (bottom) scenarios.

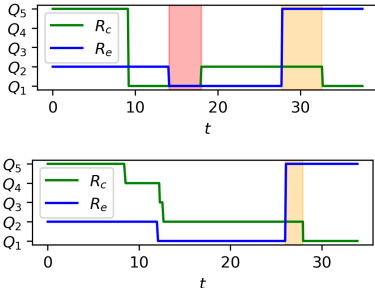


Fig. 3: Comparison of the region heat-map resulting from experimental trajectories when R_c is running a context-triggered controller with (bottom) and without (top) contingency games. Specification violation is indicated in red (top).

V. EXPERIMENTS

A. Robot Navigation Scenario

We first consider the robot navigation example depicted in Fig. 1 and discussed as a running example in Sec. II. The resulting controller is validated³ on a TurtleBot4 platform running ROS2 jazzy. The DG-FG solver is invoked in real-time with a frequency of 20 Hz, resulting in a smooth trajectory in the presence of multiple simultaneous intents.

We compare the performance of the resulting controller to a baseline where no contingency game is invoked in the lower level in Fig. 3 via the following scenario: Initially, R_u is in Q_2 , while R_c is in Q_5 and is requested to enter Q_1 . After 12 seconds R_u enters Q_1 and leaves again after 25 seconds. We recorded the corresponding trajectories on the testbed and show the resulting region heat map for each agent in Fig. 3 without (top) and with (bottom) the integration of contingency games. If no contingency game is used, R_c directly moves into Q_1 , causing a violation of the constraint that R_u and R_c should never be together in Q_1 as soon as R_u enters Q_1 (red region in Fig. 3 (top)). This is prevented by invoking a contingency game, as R_c now anticipates that R_u attempts to move into Q_1 and therefore directly approaches Q_2 . We also see that the contingency game solution results in R_c reaching Q_1 faster after R_u has left (orange region in Fig. 3) as it already moves towards Q_1 when anticipating that R_u will be leaving Q_1 .

³Recordings of the experiments are available as supplementary material.

B. Highway Scenario

This scenario consists of a highway with three lanes. The objective of the controlled autonomous car (ego agent) is to keep its desired cruising speed, while safely overtaking slower vehicles when necessary. In doing so, it must comply with traffic rules, avoid collisions, and maintain a safe distance from other vehicles. For simplicity, the highway is assumed to be completely straight, meaning all vehicles move only along the y -direction. Furthermore, it is assumed that all other vehicles also follow the traffic rules.

The ego agent’s objectives and traffic rules were specified in LTL and translated into a strategic game. The dynamics of each agent are modeled using the nonlinear kinematic bicycle model. The resulting context-triggered contingency game is validated in simulation. The scenario involves three cars: the car in front of the ego vehicle is slow, while a fast car is on lane 3 (Fig. 4a). After observing that the fast car intends to merge into lane 2 (Fig. 4b), the ego vehicle delays its overtaking maneuver and waits until the fast car has passed (Fig. 4c). During this period, it reduces speed to avoid a collision. After switching to lane 2, it is observed that the car in front is also slowing down, so the ego vehicle switches to lane 3 to safely overtake both cars (Fig. 4e-g).

We compare the performance of our control framework to invoking a classical MPC that only depends on the current context in the lower level as a baseline. Fig. 5 illustrates the resulting trajectories of different low-level controllers with two different horizon lengths during the initial part of the scenario. In both cases, the MPC fails to anticipate the intent of the other driver. While the MPC with long horizon is still able to take evasive action, the MPC with short horizon merges in front of the other car, forcing it to brake risking a collision. In contrast, with contingency games, the intent of the other vehicle is correctly anticipated (after it switched lanes), causing the ego vehicle to proactively remain in its lane. Notably, even with a short horizon, the controller behaves similarly to the MPC with a long horizon.

VI. CONCLUSION

We presented context-triggered contingency games, a two-layered architecture combining strategic and dynamic interactions by extracting the intents of each agent from strategy templates. Our DG-FG solver outperforms existing solvers, enabling safe real-time planning under LTL in interactive scenarios. We validate the approach through hardware and simulation experiments, showing clear improvements over

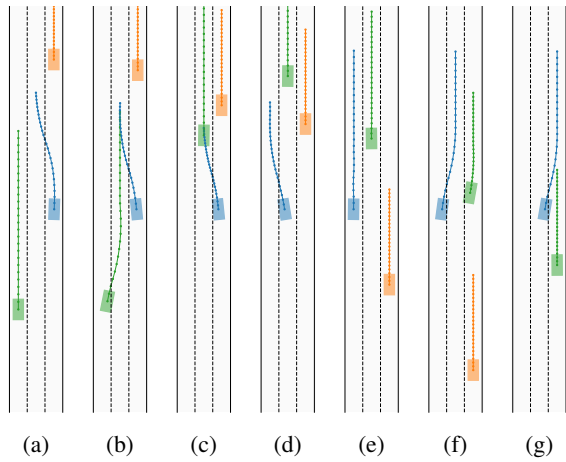


Fig. 4: Overtaking maneuver with three vehicles.

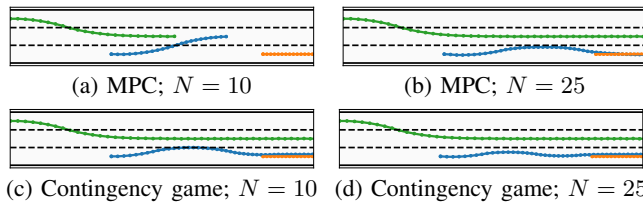


Fig. 5: Comparison of reactive and predictive controllers with varying horizon lengths.

classical MPC baselines. Future work includes accounting for uncertainties in the dynamics and objectives of other agents, investigating more gradual intent representations for human-robot interaction, and scaling the architecture to more complex multi-agent scenarios.

REFERENCES

- [1] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.
- [2] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, no. Volume 7, 2024, pp. 47–72, 2024.
- [3] N. Matni, A. D. Ames, and J. C. Doyle, "Towards a theory of control architecture: A quantitative framework for layered multi-rate control," 2024.
- [4] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.
- [5] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
- [6] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, p. 110692, 2023.
- [7] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and systems*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.
- [8] X. Yin, B. Gao, and X. Yu, "Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges," *Annual Reviews in Control*, vol. 57, p. 100940, 2024.
- [9] P. Tabuada, *Verification and control of hybrid systems*, 2009th ed. New York, NY: Springer, June 2009.
- [10] C. Belta, B. Yordanov, and E. Aydin Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017.
- [11] L. Lindemann and D. V. Dimarogonas, *Formal Methods for Multi-Agent Feedback Control Systems*. MIT Press, 2025.
- [12] S. P. Nayak, L. N. Egidio, M. D. Rossa, A.-K. Schmuck, and R. M. Jungers, "Context-triggered abstraction-based control design," *IEEE Open Journal of Control Systems*, vol. 2, pp. 277–296, 2023.
- [13] A. Anand, S. P. Nayak, and A. Schmuck, "Synthesizing permissive winning strategy templates for parity games," vol. 13964, pp. 436–458, 2023.
- [14] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.
- [15] Y. Huang, N. Xu, M. Guo, J. Li, and L. Shen, "Distributed and reactive controller synthesis for multi-agent systems under finite horizon temporal logic tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 13 485–13 500, 2025.
- [16] S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Multi-robot mission planning in dynamic semantic environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023.
- [17] Y. Zhang, C. Pezzato, E. Trevisan, C. Salmi, C. H. Corbato, and J. Alonso-Mora, "Multi-modal mppi and active inference for reactive task and motion planning," 2023.
- [18] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," 2018.
- [19] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," 2018.
- [20] E. L. Zhu and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria," 2022.
- [21] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Algames: a fast augmented lagrangian solver for constrained dynamic games," *Autonomous Robots*, vol. 46, no. 1, pp. 201–215, November 2021.
- [22] Y. Chen, U. Rosolia, W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Interactive multi-modal motion planning with branch model predictive control," 2021.
- [23] O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 110–126, 2025.
- [24] K. A. Mustafa, D. J. Ornia, J. Kober, and J. Alonso-Mora, "Racp: Risk-aware contingency planning with multi-modal predictions," 2024.
- [25] L. Peters, A. Bajcsy, C.-Y. Chiu, D. Fridovich-Keil, F. Laine, L. Ferranti, and J. Alonso-Mora, "Contingency games for multi-agent interaction," 2023.
- [26] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone, *An introduction to the planning domain definition language*. Springer, 2019, vol. 13.
- [27] C. Baier and J.-P. Katoen, *Principles of model checking*, ser. The MIT Press. London, England: MIT Press, April 2008.
- [28] H. Kress-Gazit, M. Lahijanjan, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 211–236, 2018.
- [29] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, June 2019.
- [30] P. Glotfelter, J. Cortes, and M. Egerstedt, "Boolean composability of constraints and control synthesis for multi-robot systems via nonsmooth control barrier functions," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, pp. 897–902.
- [31] S. P. Dirkse and M. C. Ferris, "The path solver: a nonmonotone stabilization scheme for mixed complementarity problems," *Optimization Methods and Software*, vol. 5, no. 2, pp. 123–156, January 1995.
- [32] F. Facchinei and C. Kanzow, "Penalty methods for the solution of generalized nash equilibrium problems," *SIAM Journal on Optimization*, vol. 20, no. 5, pp. 2228–2253, January 2010.
- [33] M. King-Smith, P. Tsotras, and F. Dellaert, "Simultaneous control and trajectory estimation for collision avoidance of autonomous robotic spacecraft systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2022, pp. 257–264.
- [34] H. Martiros, A. Miller, N. Bucki, B. Solliday, R. Kennedy, J. Zhu, T. Dang, D. Pattison, H. Zheng, T. Tomic, P. Henry, G. Cross, J. VanderMey, A. Sun, S. Wang, and K. Holtz, "Symforce: Symbolic computation and code generation for robotics," 2022.