Decentralized Multi-Agent Strategy Synthesis under LTL$_f$ Specifications via Exchange of Least-Limiting Advisers

Georg Friedrich Schuppe and Jana Tumova

***Abstract*— We propose a decentralized solution to a high-level task-planning problem for a multi-agent system under a set of possibly dependent LTL$_f$ specifications. We propose an approach where the problem is turned into a number of individual two and a half player stochastic games with reachability objectives. If almost-surely winning strategies cannot be found for them, we deploy so-called least-limiting advisers to restrict agents' behaviours. A key step is treating safety and liveness separately, by synthesizing necessary safety and fairness assumptions and iteratively exchanging them in the form of advisers between the agents. We avoid the state-space explosion problem by computing advisers locally in each game, independently of the model and specification of other agents. The solution is sound, but conservative. We demonstrate its scalability in a series of simulated scenarios involving cleaning of an office-like environment.**

I. Introduction

With the rising deployment of robots, it has become more common that the tasks they are required to accomplish are increasingly sophisticated and that multiple robots need to occasionally collaborate even though they were originally not necessarily meant to operate as a team. To formalize desired tasks and constraints, *Linear Temporal Logic*-based specifications have gained popularity. They combine the advantages of a rich specification language with automated techniques to synthesize provably correct strategies. For instance, *Reactive Synthesis* [1] enables the agents to efficiently compute strategies responding to the environment. These methods are powerful, but come at the cost of doubly exponential complexity.

Immediately applying these techniques to the multi-agent scenario poses several limitations. Traditional, centralized approaches are computationally extremely expensive, and impractical even for smaller systems. They often assume full control over all agents in the system and/or might require the agents to interact in very specific ways. The resulting strategies are therefore not robust against even small changes in a single part of the system. In contrast, decentralized approaches treating every uncontrollable aspect as adversarial behaviour are very conservative and often lead to unrealizability of the problem [2].

Strictness of adversarial behaviour can be alleviated by providing guarantees only when certain assumptions on the environment are met. Refining assumptions over the environment has been used in research related to $GR(1)$ [3], an efficient fragment of LTL. Alur et. al. [4] refine GR(1) specifications in an iterative fashion through counterexamples. *Assume-Guarantee* synthesis generates strategies in two-player games under the assumption that the environment follows a certain specification [5]. Raman et al. provide minimal feedback for unachievable high-level robot behaviours. Here, a human designer adds assumptions by refining the specification [6] [7]. In assume-guarantee compositional design, assumptions and guarantees are used as an interface for system components [8] [9] and are usually designed by hand, but can also be obtained through automata learning [10] or abstraction refinement [11]. In order for assumptions on other parts of the system to be meaningful, they have to be as *least-limiting* or *maximally permissive* as possible. Maximally permissive supervisors are used to synthesize flexible strategies for manufacturing systems [12]. The notion of least-limiting supervisors or maximally-permissive strategies has been a topic of research in other communities, as well. Tumova et. al. [13] formalized least-limiting strategy advisers for safety objectives. However, it is hard to capture the notion of least-limiting or maxmimally permissive for ω-regular objectives, as in general there exists no unique solution [14] [15]. Bernet et. al. [14] defined the notion of *permissive strategies* for parity games as strategies that subsume the behaviour of all memoryless strategies. Chatterjee et. al. [15] generate weakest sufficient assumptions on the environment.

Inspired by these results, in this work, we offer a new reactive synthesis-based solution to multi-agent task planning under individual, possibly dependent LTL$_f$ specifications. The agents are modeled via MDPs and we avoid building a centralized model altogether. Instead, each agent maintains a two and a half player stochastic game with a reachability objective and iteratively prunes or expands it based on exchange of so-called least-limiting advisers with the other agents. A key step is treating safety and liveness separately, by synthesizing necessary safety and fairness assumptions and iteratively exchanging them in the form of advisers between the agents. We show the scalability of the proposed solutions in simulations by computing reactive plans for up to 10 robots, each having 200 states in their MDP within the order of seconds to one minute.

A. Related Work

Different approaches have been used to apply temporal logic-based task planning for multi-agent systems in

Georg Friedrich Schuppe and Jana Tumova are with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, Stockholm, Sweden and are also affiliated with Digital Futures schuppe@kth.se, tumova@kth.se. This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation and the Swedish Research Council (VR) (project no. 2017-05102).

a scalable manner. For instance, Ding et al. compare the advantages of a top-down and a bottom-up approach of LTL task in multi-robot system [16]. Chen et al. [17] distribute a single LTL task between agents so they can be executed independently. This method is only complete for a a particular fragment of LTL. Schillinger et al. coordinate teams of agents with probabilistic models through auctioning and learning [18] [19], whereas we focus on individual, occasionally collaborating agents. Guo et al. treat locally independent LTL tasks and connectivity constraints [20] [21]. A receding horizon approach building a centralized product in a limited horizon is proposed in [22]. Schuppe et al. [23] merges compatible plays in a compositional fashion, but is inefficient in scenarios with high redundancy between executions. Lin et al. synthesize reactive strategies with integrated motion planning for a fragment of LTL [24]. Vasile et al. schedule fleets of robots using Time Window Temporal Logic [25]. Kantaros et al. propose a sampling-based motion planning method for large-scale multi-robot systems under temporal specifications [26]. Lacerda et al. compose Petri nets representing multi-robot team restrictions [27], while we focus on the agents modeled as Markov Decision Processes. DeCastro et al. encode high level motion tasks in LTL in order to synthesize reactive strategies for a team of robots [28].

II. Notation and Preliminaries

For a finite set A, a probability distribution on A is a function $\delta : A \to [0, 1]$ such that $\sum_{a \in A} \delta(a) = 1$. We write $Supp(\delta) = \{a \in A \mid \delta(a) > 0\}$ for the support set of δ. We denote the set of probability distributions on A by $Dist(A)$.

A. Stochastic Games

Definition II.1 (Labelled Stochastic Game)**.** A two and a half player stochastic game (SG) is a tuple $G = ((S, E), Act, \delta, s_0, AP, L)$ with a finite directed graph (S, E), partitions $S = S_1 \uplus S_2 \uplus S_p$ (player 1, player 2, and probabilistic states), $E = E_1 \uplus E_2 \uplus E_p$. Act is the set of actions that can be taken in player 1 or player 2 states. $E_1 \subseteq S_1 \times Act \times S_2, E_2 \subseteq S_2 \times Act \times S_p, E_p \subseteq S_p \times S_1$. For all $s_p \in S_p$ and $s_1 \in S_1$, we have $(s_p, s_1) \in E$ iff $\delta(s_p)(s_1) > 0$, where $\delta : S_p \to Dist(S_1)$ is a probabilistic transition function. $s_0 \in S_1$ is the initial state. AP is a set of atomic propositions and a *state labelling function* $L : S_1 \to 2^{AP}$, assigns to every state the atomic propositions which hold true in there.

If each state in S_2 has a single outgoing edge, G is called a *Markov Decision Process* (MDP); in that case we omit S_2 from the definition, and use $E_1 \subseteq S_1 \times Act \times S_p$ in the expected way.

Without loss of generality, we assume that for each $a \in AP$, $\neg a$ also belongs to AP and each $L(s)$ contains either an atomic proposition or its negation. For some $s \in S$, we define the set of *enabled* actions as $Act(s) = \{a \in Act \mid (s, a, s') \in E, s' \in S\}$.

Definition II.2 (Plays and Traces)**.** A *play* of game G is an infinite sequence $\pi = s_0 s_1 s_2 \dots$ of states such that $(s_k, a, s_{k+1}) \in E_1 \cup E_2$ or $(s_k, s_{k+1}) \in E_p$ for all $k \geq 0$. We use Π for the set of all plays. Given a play $\pi = s_0 s_1 s_2 \dots \in S^\omega$, we define its trace as $\tau(\pi) = L(s_0)L(s_1)L(s_2) \dots \in (2^{AP})^\omega$.

Definition II.3 (Strategy)**.** A deterministic finite-memory strategy for player i is a function $\lambda : S_i^* \to Act$, where $\lambda(s_1 \dots s_i) \in Act(s_i)$. A memoryless strategy is such that $\lambda(s_1 \dots s_i) = \lambda(s_i)$ for all $s_1 \dots s_i \in S_i^*$ and we use $\lambda : S_i \to Act$.

Strategies λ_1, λ_2 define plays that follow them $\pi_{\lambda_1,\lambda_2} = s_0 s_1 s_2 \dots$, where $(s_k, \lambda_i(s_k), s_{k+1}) \in E_i$ if $s_k \in S_i, i = 1, 2$. For a set of plays $\mathcal{E} \subseteq \Pi$, state s, and strategies λ_1 and λ_2, we denote the probability that a play beginning in s and following λ_1, λ_2 belongs to $\mathcal{E}$ by $Pr_s^{\lambda_1,\lambda_2}(\mathcal{E})$.

Definition II.4 (Objective)**.** An *objective* for a player is a set $\psi \subseteq \Pi$ of plays that are winning for that player. Given a set $F \subseteq S$ of states, the *safety* objective $Safe(F) = \{s_0 s_1 s_2 \dots \in \Pi \mid \forall k \geq 0, s_k \in F\}$ requires only states in F to be visited. The *reachability* objective $Reach(F) = \{s_0 s_1 s_2 \dots \in \Pi \mid \exists k \geq 0, s_k \in F\}$ requires some state in F eventually to be reached.

An objective may also be given in terms of set of traces produced by plays in the expected way.

Definition II.5 (Almost-Sure Winning)**.** Given an objective ψ, strategy λ_1 of player 1 is *almost-sure winning* from state s, if for every strategy λ_2 of player 2, $Pr_s^{\lambda_1,\lambda_2}(\psi) = 1$.

Given an objective ψ, we denote the set of states from which player 1 has an almost-sure winning strategy as $\langle\langle 1 \rangle\rangle(\psi)$. Given an objective ψ, the *cooperative winning set* $\langle\langle 1, 2 \rangle\rangle(\psi)$ is the set of states s where there exists a strategies λ_1, λ_2 for players 1 and 2, such that $Pr_s^{\lambda_1,\lambda_2}(\psi) = 1$.

B. Linear Temporal Logic and Automata

We use LTL interpreted on *finite traces*, called LTL$_f$ [29].

Definition II.6 (Syntax of LTL$_f$)**.** An LTL formula over a set of atomic propositions AP is defined as follows:

$$\phi := true \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U \phi_2 \mid X\phi, \quad a \in AP$$

A trace is finite word over the alphabet 2^{AP}. $X\phi$ requires a proposition to hold in the *next* step. $\phi_1 U \phi_2$ requires ϕ_1 to hold *until* ϕ_2 holds. Operators $\vee, \rightarrow, \leftrightarrow$ are defined as usual. Furthermore, $F\phi = true\, U \phi$ represents ϕ holding *eventually* in the future and $G\phi = \neg F \neg \phi$ *always* holding from now on. For a full introduction and definition of the semantics, we refer to [29].

An LTL$_f$ formula ϕ can be represented by a *deterministic finite automaton* (DFA) [29]. We use the tool MONA [30] to translate an LTL$_f$ formula ϕ into a DFA A_ϕ that precisely accepts $L(\phi)$.

Definition II.7 (Deterministic Finite Automaton)**.** A Deterministic Finite Automaton is a tuple $A = (Q, \Sigma, \Delta, q_i, F)$

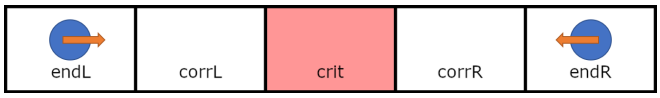

Fig. 1. Running Example. Two robots (the circles) facing each other are tasked to reach the opposite ends of a corridor, but are disallowed to enter the middle cell (crit) simultaneously.

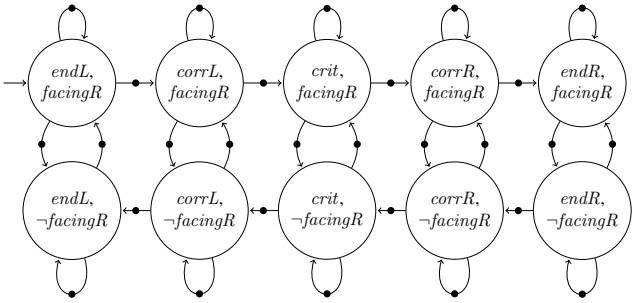

Fig. 2. MPD of the left robot in the running example. States in S_p are depicted as small dots and the outgoing edges from them are associated with probability 1. Propositions that hold in a state are denoted inside of the state.

where Q is the set of states, Σ the alphabet, $\Delta : Q \times \Sigma \to Q$ the transition function, q_i the initial state and F the set of accepting states. A finite *run* of a DFA on a trace $\rho = \rho_0\rho_1 \dots \rho_n$, where $\rho_i \in \Sigma$, is the sequence of states $q_0q_1 \dots q_{n+1}$ such that $q_{i+1} = \Delta(q_i, \rho_i)$ for all $0 \leq i \leq n$. This run is accepting if $q_{i+1} \in F$.

III. Problem Definition

A. Multi-Agent System Model

Consider a group N of n (heterogeneous) agents individually modelled as labelled MDPs $\mathcal{M}_i = ((S_i = S_{i,1} \uplus S_{i,p}, E_i), Act_i, \delta_i, s_{i,0}, AP_i, L_i)$. Without loss of generality, we assume that the sets of atomic propositions $AP_1, \dots, AP_n$ are pairwise disjunct and $AP = \bigcup_{i=1}^{n} AP_i$.

Example III.1 (Running Example)**.** Figure 1 illustrates two robots placed in the opposite ends of a narrow corridor. The corridor is partitioned into cells, each of which is labeled with a set of atomic propositions that hold there. A state of a robot is determined by its orientation (left, right) and the cell it occupies (endL, corrL, crit, corrR, endR). In each state, a robot can choose to stay, move forward, or turn 180°. As a result, they change their state according to a probabilistic distribution reflecting the uncertain outcome of the chosen action, e.g., due to imprecision in actuation. We thus model each robot as an MDP, similarly as in [31] [32]. Each robot has its own set of atomic propositions that can hold in its state $AP_i = \{endL_i, corrL_i, crit_i, corrR_i, endR_i, facingR_i\}$, $i \in \{1, 2\}$. An MDP for the robot in the leftmost cell is illustrated in Figure 2. Here, we assumed perfect execution of the actions, i.e. $\delta(s_p, s_1) \in 0, 1$ for each $s_p \in S_p$ and $s_1 \in S_1$.

In our setup, each agent chooses and executes a move synchronously at each discrete time step, producing a *joint trace*:

Definition III.1 (Joint Trace)**.** Assuming multiple traces $\tau_i = o_{i,1}o_{i,2} \dots \in (2^{AP_i})^\omega$ produced by plays of $\mathcal{M}_i$, we define the *joint trace* as a sequence:

$$\tau_{joint} = (\bigcup_{i\in[1,n]} o_{i,1})(\bigcup_{i\in[1,n]} o_{i,2}) \dots \in (2^{AP})^\omega$$

B. Task Specification

Each agent is given its own task to satisfy; however, the task may require occasional collaboration with others. For instance, in our running example, the two robots are given a task to reach the opposite end of the corridor. While most of the time they can move independently, they have to collaborate to avoid being in the critical section in the middle at the same time.

Formally, each agent is assigned their own LTL$_f$ task specification ϕ_i over the alphabet AP_ϕ where $AP_i \subseteq AP_\phi \subseteq AP$. We assume that each agent has full knowledge over all atomic propositions AP holding in each time step for each agent.

Example III.2 (Running Example cont.)**.** Formulas expressing the task of reaching the opposite side of the corridor and not entering the critical section at the same time are:

$$\phi_1 = F(endR_1) \wedge G!(crit_1 \wedge crit_2)$$

$$\phi_2 = F(endL_2) \wedge G!(crit_1 \wedge crit_2)$$

C. Problem Definition

Problem III.1. Given a set N of n labelled MDPs $\mathcal{M}_i$ and n LTL$_f$ formulae $\phi_i, i \in N$ over AP, compute a strategy $\lambda_{i,1}$ for player 1 for each $i \in N$ such that there exists a finite prefix on the produced joint trace that satisfies all $\phi_i, i \in N$ with probability 1.

Remark III.1 (Centralized Cooperative Approach)**.** A straightforward approach to solving problems with multiple models and multiple specifications is to compose the models $\mathcal{M}_i, i \in [1, n]$ into a single, global model $\mathcal{M} = \bigotimes_{i\in[1,n]} \mathcal{M}_i$ with composition defined by the synchronous product operator as defined in [33]. Fulfillment of all specification formulae can be assured by conjunction of partial specifications $\phi = \bigwedge_{i\in[1,n]} \phi_i$. Problem III.1 can be solved by constructing $\mathcal{M} \otimes A_\phi$ and solving with standard methods. While those methods are correct and complete in theory, their exponential complexity is an issue that makes the approach practically infeasible.

D. Our Approach

A schematic overview of our approach for two agents is given in Figure 3 and can be scaled up to an arbitrary number of agents. Each agent i starts with the assumption that other agents are adversarial by augmenting the MDP with states $S_{i,2}$ and creating a product stochastic game with a DFA representing ϕ_i. An attempt to find an almost-sure winning strategy to this game with the objective represented by ϕ_i is made; the strategy would constitute a solution to Problem 1 for agent $i \in N$ (see Sec. IV-A). Suppose that this attempt is not successful for one of the agents. Then,

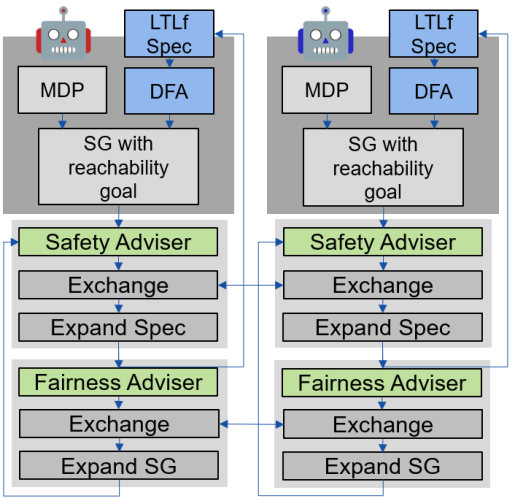

Fig. 3. Schematic overview of the approach with two agents.

minimal, necessary assumptions on the behaviour of some of the agents are imposed and exchanged in an iterative fashion in the form of least-limiting advisers. Inspired by [15], we treat safety and liveness separately, leading to the design of safety and fairness advisers (see Sec. IV-B and IV-C). These advisers prune edges in $E_{i,2}$ or enforce fair usage of them for some of the agents, leading to a higher chance for i to find a winning strategy. The algorithm terminates when a winning strategy for all $i \in N$ is found, meaning that we have a set of strategies that provably solve Problem III.1, or if agents are unable to follow exchanged advisers, leading to concluded unsatisfiability.

Example III.3 (Running Example cont.)**.** Initially, an almost-winning strategy cannot be found for neither of the robots as the requirement to avoid entering the critical section cannot be enforced without some degree of collaboration. A safety adviser is thus synthesized by our approach that, informally, forces the robots to avoid entering the critical section when being in the adjacent state. Obeying this adviser guarantees avoiding violating $G!(crit_1 \wedge crit_2)$. However, a deadlock may occur this way when both robots occur in the state adjacent to the critical section. Our approach synthesises a fairness adviser that prescribes to leave the state ("back off") with a certain probability. This way, one of the robot randomly gives way to the other by turning around, signalling safe traversal of the critical section.

IV. Solution

A. Individual Strategy Synthesis

In the first step, each agent augments its MDP model $\mathcal{M}_i$ and constructs a turn-based synthesis game, modelling their interaction with other agents through a completely unrestricted player 2. At the same time, the agent's specification ϕ_i is included via DFA A_{ϕ_i} and thus translated to a reachability objective. In the following, we omit index i where clear from the context to simplify the notation.

Definition IV.1 (Synthesis Game)**.** Given a labelled agent MDP $\mathcal{M} = ((S = S_1 \uplus S_p, E), Act, \delta, s_0, AP_i, L)$ and a DFA $A_\phi = (Q, 2^{(AP_\phi)}, q_0, \Delta, F)$, where $AP_i \subseteq AP_\phi \subseteq AP$, we define the environment alphabet $\widehat{\Sigma} = 2^{\widehat{AP}}$, where $\widehat{AP} = AP_\phi \backslash AP_i$. We create a new, labelled stochastic game $\widetilde{G} = ((\widetilde{S}, \widetilde{E}), Act \cup \{\epsilon\}, \tilde{\delta}, \tilde{s}_0, AP_i, \widetilde{L},)$, as follows:

1) *State space.* $\widetilde{S} = \widetilde{S}_1 \uplus \widetilde{S}_2 \uplus \widetilde{S}_p$.
   a) $\widetilde{S}_1 = S_1 \times Q$ are the states of player 1, tracking both the current state of $\mathcal{M}$ and A_ϕ; $\tilde{s}_1 = (s_1, q)$ for $\tilde{s}_1 \in \widetilde{S}_1$.
   b) $\widetilde{S}_2 = S_p \times Q$ are the states of player 2, choosing propositions from $\widehat{\Sigma}$ to represent the actions of other agents; $\tilde{s}_2 = (s_p, q)$ for $\tilde{s}_2 \in \widetilde{S}_2$.
   c) $\widetilde{S}_p = S_p \times Q \times \widehat{\Sigma}$; $\tilde{s}_p = (s_p, q, \sigma)$ for $\tilde{s}_p \in \widetilde{S}_p$.

   Without loss of generality, in practice $\widetilde{S}$ will include only the states reachable from $\tilde{s}_0$.
2) *Edges.* $\widetilde{E} = \widetilde{E}_1 \uplus \widetilde{E}_2 \uplus \widetilde{E}_p$.
   a) $\widetilde{E}_1 = \{(\tilde{s}_1, a, \tilde{s}_2) \mid \tilde{s}_1 = (s_1, q) \in \widetilde{S}_1, a \in Act, \tilde{s}_2 = (s_p, q) \in \widetilde{S}_2,$ and $(s_1, a, s_p) \in E\}$
   b) $\widetilde{E}_2 = \{(\tilde{s}_2, \epsilon, \tilde{s}_p) \mid \tilde{s}_2 \in \widetilde{S}_2, \tilde{s}_p = (\tilde{s}_2, \sigma) \in \widetilde{S}_p, \sigma \in \widehat{\Sigma}\}$
   c) $\widetilde{E}_p = \{(\tilde{s}_p, \tilde{s}_1) \in \widetilde{E} \mid \tilde{s}_p = (s_p, q, \sigma) \in \widetilde{S}_p, \tilde{s}_1 = (s_1, q') \in \widetilde{S}_1, \Delta(q, \sigma \cup L(s_1)) = q')$ and $s_1 \in Supp(\delta(s_p))\}$.
3) *Probabilistic Transition Function.* $\tilde{\delta} : \widetilde{S}_p \to Dist(\widetilde{S}_1)$. Given $\tilde{s}_p = (s_p, q, \sigma) \in \widetilde{S}_p$ and $\tilde{s}_1 = (s_1, q') \in \widetilde{S}_1$, we define:

$$\tilde{\delta}(\tilde{s}_p)(\tilde{s}_1) = \begin{cases} \delta(s_p)(s_1), & \text{if } \Delta(q, \sigma \cup L_s(s_1)) = q' \\ 0, & \text{otherwise.} \end{cases}$$

4) *Initial State.* $\tilde{s}_0 = (s_0, q_0) \in \widetilde{S}_1$.
5) *State Labelling Function.* $\widetilde{L} : \widetilde{S}_1 \to AP_i$. Given $\tilde{s}_1 = (s_1, q) \in \widetilde{S}_1$, $\widetilde{L}(\tilde{s}_1) = L(s_1)$.

Every play of $\widetilde{G}$ corresponds to a run on A_ϕ and a play on $\mathcal{M}$. Given an infinite sequence $\tilde{\pi} = \tilde{s}_0 \tilde{s}_1 \tilde{s}_2 \cdots \in \Pi$, the corresponding play $\pi_\mathcal{M} \in (S_1 \cup S_p)^\omega$ of $\mathcal{M}$ is defined through the first component $s_k \in S$ from the states $\tilde{s}_k = (s_k, q_k) \in \widetilde{S}_1 \cup \widetilde{S}_p$ of $\tilde{\pi}$. In a similar fashion, we define the corresponding run $\pi_A \in Q^\omega$ on A_ϕ through the second component $q_k \in Q$ from the states $\tilde{s}_k = (s_k, q_k) \in \widetilde{S}_1$ of the play $\tilde{\pi}$.

We call a play of $\widetilde{G}$ *accepting*, if there exists an accepting prefix of the corresponding run on A_ϕ. Therefore, to satisfy specification ϕ, a run has to reach a state $(s, q) \in S$ such that $q \in F$. Once such a state is reached, the task is fulfilled. The corresponding game objective is

$$\psi = Reach(\{(s, q) \in S \mid q \in F\}) \tag{1}$$

and we need to decide if $s_0 \in \langle\langle 1 \rangle\rangle(\psi)$. If there exists such a strategy, it is memoryless and deterministic [34].

Lemma IV.1. A memoryless deterministic almost-sure winning strategy for player 1 for the objective $Reach(\{(s, q) \in S \mid q \in F\})$ directly maps to a finite-memory strategy

for player 1 on $\mathcal{M}$ that is almost-surely satisfying the specification ϕ.

Proof: (Sketch) Since every play of $\widetilde{G}$ corresponds to a run on A_ϕ, we can map the memoryless strategy on $\widetilde{G}$ to a finite-memory strategy on $\mathcal{M}$, treating the current state of A_ϕ as the finite memory. □

If an almost-sure winning strategy exist for the game $\widetilde{G}$, we call the game *realizable*. Instances of such games can be solved by state-of-the-art tools, such as PRISM-games [35].

If $\widetilde{G}$ is realizable for all agents, Problem 1 is solved. If not, we limit the agents by imposing advice on their behaviour such that the existence of a strategy can be guaranteed. Any linear-time property can be decomposed into a safety and a liveness component [33]. The safety component of an objective ψ can be expressed as $\psi_s = \mathit{Safe}(\langle\langle 1,2\rangle\rangle(\psi))$. From all states in $\langle\langle 1,2\rangle\rangle(\psi_s)$, called live states, ψ can be satisfied almost-surely. The fairness component is to make sure that player 1 can satisfy ψ almost-surely regardless of the choices of player 2. We use this separation to compute safety and fairness assumptions that enable the existence of a winning strategy for player 1, inspired by [15]. These are communicated in the form of least-liming advisers to the other agents.

B. Minimal Safety Assumptions and Least-Limiting Safety Advisers

A safety assumption is a set $E_s \subseteq \widetilde{E}_2$ of edges that player 1 needs to remove from $\widetilde{G}$ in order to find an almost-sure winning strategy for objective ψ_s. Such safety assumptions should be sufficient, realizable by the other agents, and as little restrictive as possible for the other agents. We transfer the results from [15] for deterministic games to stochastic games and almost-sure winning strategies:

Definition IV.2 (Safe-Sufficiency)**.** Given an objective ψ, a safety assumption $E_s \subseteq \widetilde{E}_2$ is safe-sufficient if player 1 has an almost-sure winning strategy for the objective

$$\begin{aligned}&\mathit{AssumeSafe}(\psi, E_s) = \psi_s \cup \\ &\{\pi = s_0 s_1 s_2 \ldots \mid \exists k \geq 0, (s_k, s_{k+1}) \in E_s\}\end{aligned}$$

Definition IV.3 (Minimality)**.** A safety assumption E_s is minimal if $|E'_s| \leq |E_s|$ for all safety assumptions $E'_s \in E_2$. The unique, minimal safety assumption can be computed as

$$E_s = \{(s,s') \in E_2 \mid s \in \langle\langle 1,2\rangle\rangle\psi \text{ and } s' \notin \langle\langle 1,2\rangle\rangle\psi\}, \tag{2}$$

E_s can be computed in a straightforward way by a modified reachability algorithm on graph $\widetilde{G}$ in quadratic time [15].

Lemma IV.2. $\tilde{s} \in \langle\langle 1,2\rangle\rangle(\psi)$ if and only if $\tilde{s}$ is live for the objective $\mathit{AssumeSafe}(\psi, E_s)$.

The lemma follows directly from the definitions and intuitively, it says that if player 2 respects the safety assumption E_s, it is only the strategy of player 1 that determines whether a play of $\widetilde{G} = ((\widetilde{S}, \widetilde{E} \backslash E_s), \mathit{Act} \cup \{\epsilon\}, \tilde{\delta}, \tilde{s}_0, AP_i, \widetilde{L},)$ belongs to ψ_s or not.

1) From Safety Assumptions to Advisers: The assumption E_s is specific to the stochastic game $\widetilde{G}$ and hence cannot be directly communicated as an adviser to the other agents. Instead, we communicate the advice in the form of atomic propositions, which all agents have access to.

Definition IV.4 (Simplest Precondition)**.** The simplest precondition pre for any edge $\tilde{e}_2 = (\tilde{s}_2, \tilde{s}_p) \in \widetilde{E}_2$ is defined as $\mathit{pre}(\tilde{e}_2) = \widetilde{L}(\tilde{s}_1)$ where $\tilde{s}_1 \in \widetilde{S}_1$ s.t. $(\tilde{s}_1, \tilde{s}_2) \in \widetilde{E}$.

The simplest precondition is a set of atomic propositions that have to hold before an edge in safe assumption is taken.

Definition IV.5 (Safety Adviser)**.** A *safety adviser* is a set of tuples:

$$\mathit{SafeAdv} = \{(\mathit{pre}, \sigma) \mid \mathit{pre} \in AP_i, \sigma \in \widehat{\Sigma}_i\}$$

We call pre the precondition and σ the *advice*; given that agent i satisfies pre, other agents should not satisfy σ in their next state.

2) Incorporating Safety Advisers: After agent j broadcasts its safety adviser $\mathit{SafeAdv}_j$, it can assume the advice to be followed by other agents. Here, we propose a way of incorporating adviser from j into the strategy planning of an agent i. Given a safety adviser $\mathit{SafeAdv}_j$, agent i creates an LTL$_f$ formula for each tuple $(\mathit{pre}, \sigma) \in \mathit{SafeAdv}_j$, as follows

$$\phi_{(\mathit{pre},\sigma),i} = G(\mathit{pre} \rightarrow \neg X \mathit{proj}_i(\sigma)), \tag{3}$$

where $\mathit{proj}_i(\sigma) = \{a \mid a \in \sigma \text{ and } a \in AP_i\}$. If $\mathit{proj}_i = \varnothing$, agent i is not affected by the advice-tuple.

We can incorporate all advisers from all agents into the specification of the agent i through conjunction:

$$\phi_{s,i} = \bigwedge_{\forall(\mathit{pre},\sigma) \in \mathit{SafeAdv}_j, j \in N} \phi_{(\mathit{pre},\sigma),i} \tag{4}$$

A specific case occurs for $j = i$, when instead of imposing the safety adviser through an aditional LTL$_f$ formula, we directly remove edges from $\widetilde{E}_2$.

Lemma IV.3. Let $\mathit{SafeAdv}_j$ be a safety adviser tuple from agent j. Assuming all other agents $i \neq j$ can individually satisfy $\phi_{s,i}$, then the safety assumptions E_s of agent j are necessarily yielded.

Proof: For a given $(\mathit{pre}, \sigma) \in \mathit{Adv}_s$, assume that all agents individually satisfy $\phi_{(\mathit{pre},\sigma),i}$. In any resulting joint trace $\tau = \tau_{joint}$, if $\mathit{pre} \subseteq \tau_k$ for some k, $\mathit{proj}_i(\sigma) \not\subseteq \tau_{k+1}$. Since all propositions in AP belong to some agent, $\sigma \not\subseteq \tau_{k+1}$. □

3) Iterative Exchange of Safety Advisers: The iterative exchange of safety advisers is summarized in Algorithm 1. After expanding their specification with formulae (4), each agent constructs a new synthesis game with updated specification or prunes the game and checks for a winning strategy (lines 2-6). If there is none, the agent continues to compute

additional least limiting safety advisers and broadcasts them. This is repeated until no further safety advisers are necessary or unrealizability is concluded (line 7-12).

Algorithm 1: Iterative Exchange of Safety Advisers

Data: $\mathcal{M}_1, \dots, \mathcal{M}_n$ and $\phi_1, \dots, \phi_n$
1 **while** *any* ϕ_i *changed* **do**
2 **for** $i \leftarrow 1$ **to** n **do**
3 compute $\widetilde{G}_i$ from $\mathcal{M}_i$ and ϕ_i;
4 prune $\widetilde{G}_i$ through all previously computed **own** safety advisers;
5 **if** $s_0 \notin \langle\langle 1,2 \rangle\rangle(\psi)$ **then**
6 **return** False;
7 **for** $i \leftarrow 1$ **to** n **do**
8 compute $E_{s,i}$ as in (2);
9 compute $SafAdv_i$ as in Definition IV.5;
10 **for** $j \leftarrow 1$ **to** n **do**
11 **if** $i \neq j$ **then**
12 $\phi_j = \phi_j \wedge \phi_{s,j}$ as in (4);

Lemma IV.4. Algorithm 1 is guaranteed to terminate, either by not requiring additional safety advisers or by concluding unsatisfiability.

Proof: Advisers constructed as in Definition IV.5 only use symbols from AP, so the total amount of possible advisers is finite. If an iteration in Algorithm 1 did not lead to termination, it must have added safety advisers. Eventually, the algorithm terminates either through concluding realizability, unrealizability or by constructing all possible advisers.

C. Locally Minimal Fairness Assumptions and Least-Limiting Fairness Advisers

The synthesis game of each agent has been pruned so that it contains only live states. By imposing *fairness* on choices of some of the agents, we can guarantee existence of a winning strategy for player 1 with the original objective ψ (1) [33].

A fairness assumption is a set $E_l \subseteq E_2$ of edges that need to be chosen fairly (i.e. infinitely many times upon infinitely many visits to their source). Similarly as for the safety assumptions, we desire the fairness assumptions to be sufficient and minimal. Minimality criterion is defined analogously to Definition IV.3.

Definition IV.6 (Sufficiency)**.** Given an objective ψ, a fairness assumption E_l is sufficient if player 1 has an almost-sure winning strategy for the objective

$$AssumeFair(\psi, E_l) = \psi \cup \{\pi = s_0 s_1 s_2 \dots \mid \exists (s, s') \in E_l \text{ s.t. } s_k = s \text{ for infinitely many } k \text{ but } s_{k+1} = s' \text{ only finitely often.}\}$$

In stochastic games, we can enforce fairness by the following construction: We prepend an edge $(s, a, s') \in E$ with a probabilistic state s_p. From s_p, we choose with a non-zero probability to enter a copy of state s from which only action a leading to s' can be taken; and with a non-zero probability to enter the original state s with potentially other actions enabled. An example is depicted in Figure 4.

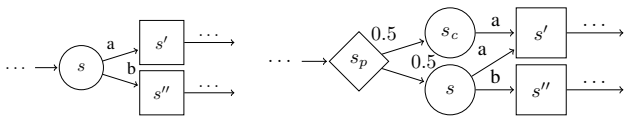

Fig. 4. Enforcing fairness on (s, a, s') by preprending a probabilistic state.

1) Locally Minimal Fairness Assumptions: Since computing a *minimal* set of fairness assumptions is NP-hard [15], we instead compute *locally minimal* fairness assumptions E_l^*. The procedure is outlined in Algorithm 2, resulting in a runtime depending linearly on $|\widetilde{E}_2|$. We start by constructing a new stochastic game by expanding each edge $e \in \widetilde{E}_2$ to be fair, i.e. $E_{l,0}^* = E_2$, with the construction described above. If in $\widetilde{G}'(E_{l,i}^*)$, player 1 has a winning strategy, we remove edges from $E_{l,i}^*$ until it is not possible to do so without player 1 not having a winning strategy anymore. The resulting set $E_{l,i}^*$ is locally minimal, as no set $E' \subseteq E_{l,i}^*$ is sufficient. The order in which we remove edges from $E_{l,i}^*$ is chosen randomly. A suitable heuristic is subject for our future research.

Algorithm 2: Locally Minimal Fairness Assumptions

Data: $\widetilde{G} = ((\widetilde{S}, \widetilde{E}), Act, \tilde{\delta}, \tilde{s}_0, AP_i, \widetilde{L},)$,
$\psi = Reach(\{(s, q) \in S' \mid q \in F\}$
1 $E_{l,i}^* = E_{l,0}^* = \widetilde{E}_2$;
2 Construct $\widetilde{G}'(E_{l,i}^*)$ based on construction in IV-C.1;
3 **if** $\tilde{s}_i' \notin \langle\langle 1 \rangle\rangle(\psi)$ *for* $\widetilde{G}'(E_{l,i}^*)$ **then**
4 **return** False;
5 **for** $i \leftarrow 1$ **to** $|\widetilde{E}_2|$ **do**
6 **if** $\exists e \in E_{l,i}^*$ *s.t.* $\tilde{s}_i' \in \langle\langle 1 \rangle\rangle(\psi)$ *for* $\widetilde{G}'(E_{l,i}^* \setminus \{e\})$ **then**
7 $E_{l,i+1}^* = E_{l,i}^* \setminus \{e\}$;
8 **else**
9 **return** $E_{l,i}^*$;

2) From Fairness Assumptions to Advisers: From a set E_l^*, we construct a fairness adviser $FairAdv$.

Definition IV.7 (Fairness Adviser)**.** A *fairness adviser* is a set of tuples:

$$FairAdv = \{(pre, \sigma) \mid pre \in AP_i, \sigma \in \widehat{\Sigma}_i\}$$

We call pre the precondition and σ the *advice*; given that agent i satisfies pre, other agents should satisfy σ with non-zero probability in their next state.

3) Incorporating Fairness Advisers: Given a fairness adviser $FairAdv_j$, agent i expands their synthesis game $\widetilde{G}$ to a new game $\widetilde{G}'$ by prepending each player-1 state with a

probabilistic state, as in Figure 4, enforcing fairness on the subset of actions that lead to states $\tilde{s}$ satisfying $proj_i(\sigma) \subseteq \widetilde{L}(\tilde{s})$. If no such choice exists, then player 1 cannot fulfill the adviser in that state. In the proposed framework of this paper, we conclude that this state must be avoided and prune all player-1 choices leading to this state. Own fairness advisers $FairAdv_i$ are incorporated by applying the fairness construction in Figure 4 to corresponding player-2 states.

This construction guarantees that all fairness advisers are followed, and hence if a winning strategy exists for each agent's $\widetilde{G}'$ with objective ψ, then the collection of such winning strategies is a solution to Problem 1. On the other hand, the construction possibly enforces stronger constraints than needed, making the overall approach conservative.

Lemma IV.5. Let (pre, σ) be a fairness adviser tuple from agent i. Assuming all agents $j \in N$ individually have an almost-sure winning strategy in $\widetilde{G}'$ with objective ψ, the fairness assumption of agent i is necessarily yielded.

Proof: If all agents $j \in N$ individually have an almost-sure winning strategy in the game $\widetilde{G}'$, then each agent chooses actions that lead to $proj_i(\sigma)$ with non-zero probability every time the preconditions pre are met, meaning that σ is fulfilled with non-zero probability in the step after pre occurs. If pre occurs infinitely often, σ will occur infinitely often with probability 1. □

4) Iterative Exchange of Safety and Fairness Advisers: Incorporating fairness advisers can lead to unrealizability of the newly created game $\widetilde{G}'$, leading to the need of additional safety and/or fairness assumptions. We iteratively exchange advisers until no further advisers are necessary or unrealizability is concluded. The overall procedure is summarized in Algorithm 3.

Algorithm 3: Complete Adviser-Based Framework

```
Data: M_1, ..., M_n and φ_1, ..., φ_n
1  while any SafAdv_i or FairAdv_i changed do
2      converge safety advisers as in Algorithm 1;
3      for i ← 1 to n do
4          compute E*_{l,i} as in Algorithm 2;
5          Construct G̃'(E*_{l,i}) based on construction in
             IV-C.1;
6          if s̃'_i ∉ ⟨⟨1⟩⟩(φ)) for G̃'(E*_{l,i}) then
7              return False;
8          compute FairAdv_i as in Definition IV.5;
9      for i ← 1 to n do
10         for j ← 1 to n do
11             incorporate FairAdv_j as in IV-C.3;
```

Lemma IV.6. Algorithm 3 is guaranteed to terminate, either by converging on safety and fairness advisers or by concluding unrealizability.

Proof of Lemma IV.6 is analogous to the proof of Lemma IV.4.

D. Analysis

Theorem IV.1 (Correctness)**.** Either Algorithm 3 returns unrealizability or the joint trace of strategies computed by satisfy all specifications almost-surely.

Proof: (Sketch) From Lemma IV.4 and Lemma IV.6 it follows that the algorithm terminates. Consider the case where it terminated by concluding realizability. In that case, every agent i has a winning strategy for their individual synthesis game (line 6). They must follow all broadcasted safety and fairness advisers, as they are incorporated into the game either through the specification or through expanding the game. Conversely, all of their broadcasted advisers can be assumed to be followed (Lemma IV.3, Lemma IV-C.3). Hence, each agent's play corresponds to an accepting run on its respective DFA $A_{\phi.i}$ and thus the joint trace has a finite prefix that accepts all specification formulae, by which it solves Problem 1. □

Remark IV.1 (Complexity)**.** Solving a stochastic game with safety or reachability objective can be done in quadratic time [15]. Safety advisers require solving a single stochastic game with a safety objective and can thus also be computed in quadratic time. Locally minimal fairness advisers require solving multiple stochastic games, limited by the edges of current game. While fairness advisers grow the games only by a linear factor, safety advisers can grow the DFA by a doubly exponential factor. Since it is hard to put a limit on the amount of generated advisers, we do not provide a tight upper bound but in the worst case, our approach meets the complexity of reactive synthesis, which is proven to be 2EXP-complete. In all our experiments presented in Section V, both safety and fairness adviser computation converged after one or two rounds.

V. Simulation Results

We implemented our approach in Python3 using MONA [30] and *ltlf2dfa* to obtain DFA for LTL$_f$ specifications and PRISM-games [35] to solve our stochastic game instances. Computation has been performed on an Intel i5-7600K processor and 16GB RAM. The implementation alongside installation instructions is available on Github at `https://github.com/gschup/ltlf-assum`.

To illustrate the scalability of the approach, we simulated an office-like environment, partitioned into 10x5 labelled cells. A schematic view of the office is shown in Figure 5. A state of a robot is determined by its orientation (left, right, up, down) and the cell it occupies. In each state, a robot can choose to stay, move forward, or turn 90°.

A. Independent Tasks

In the first scenario, each of n robots is assigned to empty all ℓ bins in the environment. The task formula for robot i is:

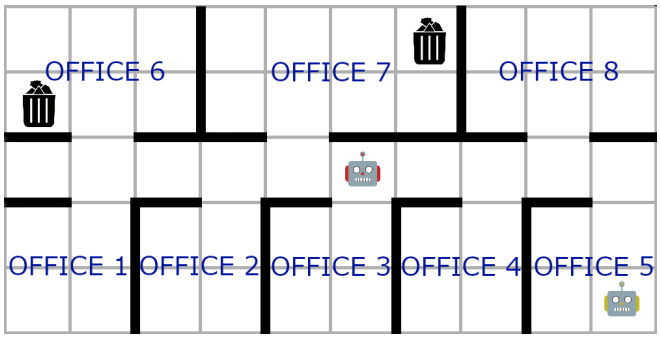

Fig. 5. Instance of the office scenario with two robots and two bins.

		bin robots				
		1	2	3	4	5
bins	1	0.538s	0.945s	1.256s	1.699s	2.102s
	2	1.561s	2.929s	4.580s	5.784s	7.086s
	3	7.148s	12.848s	17.479s	24.686s	30.549s

TABLE I

AVERAGE RUNTIMES FOR SCENARIO 1

$$\phi_i = \bigwedge_{\forall k \in \{1,\dots,\ell\}} F bin_{i,k},$$

where $bin_{i,k}$ is an atomic propositions encoding that the position of robot i coincides with the position of bin k.

Average runtimes of 10 runs for $n = 1,\dots,5$ and $\ell = 1,\dots,3$ with random starting positions of robots and bins are listed in Table I. Since the robots are completely independent, a rising number of agents leads to a linear increase in runtime, as games are solved separately and no advisers need to be exchanged. Adding more bins to each robot's task increases the DFA size and therefore the game size and runtimes, as expected.

B. Spillage cleaning tasks

In addition to n robots emptying bins, we add m cleaning robots to the scenario. Each cleaning robot $j \in \{1,\dots,m\}$ needs to clean detected spillage in office no. o_j. It is also required to guarantee that none of the bin-emptying robots enter the affected office in order to prevent further damage. The task formula for the bin emptying robots is defined as above, the cleaning robots have the following task:

		bin robots				
		1	2	3	4	5
cleaning robots	1	7.487s	10.704s	13.929s	16.896s	21.681s
	2	12.766s	15.694s	19.724s	23.369s	29.032s
	3	20.399s	23.524s	26.839s	30.770s	37.978s
	4	29.046s	32.074s	36.786s	40.195s	47.520s
	5	43.427s	45.509s	49.680s	52.495s	66.732s

TABLE II

AVERAGE RUNTIMES FOR SCENARIO 2

		bin robots				
		1	2	3	4	5
cleaning robots	1	8.884s	11.112s	13.486s	15.869s	18.887s
	2	15.348s	18.269s	19.565s	22.699s	26.838s
	3	23.529s	25.288s	27.326s	30.395s	33.639s
	4	28.826s	32.210s	33.479s	37.456s	41.082s
	5	34.259s	39.343s	41.691s	45.421s	48.764s

TABLE III

AVERAGE RUNTIMES FOR SCENARIO 3

$$\phi_j = F off_{j,o_j} \wedge G(\bigwedge_{\forall i \in \{1,\dots,n\}} \neg off_{i,o_j})$$

Average runtimes of 10 runs for $n = 1,\dots,5$, $m = 1,\dots,5$ and $\ell = 1$ with random starting positions of robots and bins are listed in Table II. To guarantee task satisfaction, each cleaning robot has to broadcast safety advisers to the bin-emptying robots, disallowing them to enter the office they were assigned to clean. Adding bin-emptying robots to the scenario leads to small increases in computation time, as the computed safety advisers are efficient to implement. Adding additional cleaning robots results in a larger increase in runtime, as each needs to compute safety advisers.

C. Collision-free cleaning

Finally, we consider the scenario, where the bin-emptying robots are disallowed to enter the dirty office only when the cleaning robots are working there to avoid collisions, replacing ϕ_j for each of the cleaning robot with

$$\phi_j = F(off_{j,o_j} \wedge \bigwedge_{\forall i \in \{1,\dots,n\}} \neg off_{i,o_j}).$$

Average runtimes of 10 runs for $n = 1,\dots,5$, $m = 1,\dots,5$ and $\ell = 1$ with random starting positions of robots and bins are listed in Table III. This time, fairness advisers have to be exchanged. For small games, solving the task is more efficient than in scenario 2 that required exchange of safety advisers, as safety advisers grow the size of the individual games by growing the DFA. Since implementing fairness advisers is efficient, adding additional bin emptying robots does not lead to a big increase in runtime.

VI. CONCLUSION AND FUTURE RESEARCH

We have proposed a novel, decentralized approach to the problem of task planning for heterogeneous multi-agent systems under LTL$_f$ specifications using least-limiting advisers. The results show the approach to be effective in scenarios with sparse dependency.

Future work includes expanding on the fairness handling to address the conservativeness of the method. the agent models to include costs for actions will enable the framework to optimize for certain aspects like completion time or energy consumption. Another interesting aspect is the inclusion of humans in the environment, as advisers only come in two forms and are potentially understandable by humans.

References

[1] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1989, pp. 179–190.

[2] M. Kloetzer and C. Belta, "Dealing with nondeterminism in symbolic control," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2008, pp. 287–300.

[3] R. Bloem, R. Ehlers, S. Jacobs, and R. Könighofer, "How to handle assumptions in synthesis," *arXiv preprint arXiv:1407.5395*, 2014.

[4] R. Alur, S. Moarref, and U. Topcu, "Counter-strategy guided refinement of gr (1) temporal logic specifications," in *2013 Formal Methods in Computer-Aided Design*. IEEE, 2013, pp. 26–33.

[5] K. Chatterjee and T. A. Henzinger, "Assume-guarantee synthesis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2007, pp. 261–275.

[6] V. Raman and H. Kress-Gazit, "Automated feedback for unachievable high-level robot behaviors," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 5156–5162.

[7] ——, "Towards minimal explanations of unsynthesizability for high-level robot behaviors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 757–762.

[8] K. Ghasemi, S. Sadraddini, and C. Belta, "Compositional synthesis via a convex parameterization of assume-guarantee contracts," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–10.

[9] M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Assume-guarantee verification for probabilistic systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2010, pp. 23–37.

[10] S. Chaki, E. Clarke, N. Sinha, and P. Thati, "Automated assume-guarantee reasoning for simulation conformance," in *International Conference on Computer Aided Verification*. Springer, 2005, pp. 534–547.

[11] A. Komuravelli, C. S. Păsăreanu, and E. M. Clarke, "Assume-guarantee abstraction refinement for probabilistic systems," in *International Conference on Computer Aided Verification*. Springer, 2012, pp. 310–326.

[12] Y. Chen, Z. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness-enforcing petri net supervisor for flexible manufacturing systems," *IEEE Transactions on automation science and engineering*, vol. 8, no. 2, pp. 374–393, 2010.

[13] J. Tumova and D. V. Dimarogonas, "Synthesizing least-limiting guidelines for safety of semi-autonomous systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 5714–5719.

[14] J. Bernet, D. Janin, and I. Walukiewicz, "Permissive strategies: from parity games to safety games," *RAIRO-Theoretical Informatics and Applications*, vol. 36, no. 3, pp. 261–275, 2002.

[15] K. Chatterjee, T. A. Henzinger, and B. Jobstmann, "Environment assumptions for synthesis," in *International Conference on Concurrency Theory*. Springer, 2008, pp. 147–161.

[16] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[17] Y. Chen, X. C. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global LTL specifications," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2718–2723.

[18] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 253–267.

[19] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical LTL-task mdps for multi-agent coordination through auctioning and learning," *The international journal of robotics research*, 2019.

[20] M. Guo, J. Tumova, and D. V. Dimarogonas, "Cooperative decentralized multi-agent control under local LTL tasks and connectivity constraints," in *53rd IEEE conference on decision and control*. IEEE, 2014, pp. 75–80.

[21] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[22] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[23] G. F. Schuppe and J. Tumova, "Multi-agent strategy synthesis for ltl specifications through assumption composition," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 533–540.

[24] A. Partovi, R. R. da Silva, and H. Lin, "Reactive integrated mission and motion planning for mobile robotic manipulators," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 3538–3543.

[25] A. Mosca, C.-I. Vasile, C. Belta, and D. M. Raimondo, "Multi-robot routing and scheduling with temporal logic and synchronization constraints," in *Proceedings of the 2019 2nd International Conference on Control and Robot Technology*, 2019, pp. 40–45.

[26] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[27] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, p. 103289, 2019.

[28] J. A. DeCastro, J. Alonso-Mora, V. Raman, D. Rus, and H. Kress-Gazit, "Collision-free reactive mission and motion planning for multi-robot systems," in *Robotics research*. Springer, 2018, pp. 459–476.

[29] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. Association for Computing Machinery, 2013, pp. 854–860.

[30] J. G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm, "Mona: Monadic second-order logic in practice," in *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 1995, pp. 89–110.

[31] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe ltl specifications." in *IJCAI*, 2015, pp. 1587–1593.

[32] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, 2011.

[33] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[34] J.-P. Katoen, "The probabilistic model checking landscape," in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 2016, pp. 31–45.

[35] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos, "PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time," in *Proc. 32nd International Conference on Computer Aided Verification (CAV'20)*, ser. LNCS, vol. 12225. Springer, 2020, pp. 475–487.