Decomposition of Finite LTL Specifications for Efficient Multi-Agent Planning

Philipp Schillinger, Mathias Bürger and Dimos V. Dimarogonas

Abstract Generating verifiably correct execution strategies from *Linear Temporal Logic* (LTL) mission specifications avoids the need for manually designed robot behaviors. However, when incorporating a team of robot agents, the additional model complexity becomes a critical issue. Given a single finite LTL mission and a team of robots, we propose an automata-based approach to automatically identify possible decompositions of the LTL specification into sets of independently executable task specifications. Our approach leads directly to the construction of a team model with significantly lower complexity than other representations constructed with conventional methods. Thus, it enables efficient search for an optimal decomposition and allocation of tasks to the robot agents.

1 Introduction

High-level planning based on Linear Temporal Logic (LTL) specifications creates the opportunity to deploy robots in increasingly sophisticated scenarios while being able to provide guarantees regarding correctness and optimality, e.g. [17, 16, 14]. In such scenarios, systems often benefit from utilizing multiple agents in order to flexibly distribute workload. Instead of executing tasks sequentially, they can be allocated to different agents and carried out in parallel. Nonetheless, considering

Philipp Schillinger, Mathias Bürger
Corporate Research - Cognitive Systems (CR/AEY2), Robert Bosch GmbH Renningen, 70465 Stuttgart, Germany. e-mail: philipp.schillinger@de.bosch.com, e-mail: mathias.buerger@de.bosch.com

Philipp Schillinger, Dimos V. Dimarogonas
KTH Centre for Autonomous Systems and ACCESS Linnaeus Center, EES, KTH Royal Institute of Technology, Stockholm, Sweden. e-mail: schillin@kth.se, e-mail: dimos@kth.se
Third author was supported by the H2020 ERC Starting Grant BUCOPHSYS and the Swedish Research Council (VR).

multiple agents for a set of tasks introduces significant additional planning complexity.

Consider as a motivating example a hospital station. In addition to caring for patients, nurses are required to refill medical supplies, guide visitors, deliver meals, or clean equipment. To support them, a multi-robot system can automate required transportation. Similarly, workflows in an intelligent factory can be improved using a multi-robot system. Machines for assembly can monitor required components and request supplies if they run low. In such scenarios, missions are typically given as one specification, implicitly based on a set of independent finite tasks, e.g., deliver meals to all rooms or supply a machine with several components. In the following, we formally define problems of this type, based on an LTL mission specification and a system model, with the goal to obtain execution strategies for all robots such that the mission is guaranteed to be fulfilled with minimal costs.

Approaches for multi-agent task planning include Mixed-Integer Linear Program (MILP) formulations [9, 21], numerical vector-based allocation [1], or market-based contracting [23]. However, these approaches involve forming a team product or require separate cost calculation for all combinations of allocation options, given that these options are explicitly known. An automata-based approach is proposed in [10] and assumes synchronous team motions to plan motion sequences from an LTL specification. This assumption is relaxed in [19] by a two-phase model reduction approach, and in [20, 3] by employing trace-closed languages [18] to abstract over asynchronous motions of the agents. However, these approaches only decompose the LTL specification into independent tasks in the special case of disjoint agent alphabets. Their focus is rather to coordinate execution between a known team of agents given explicit allocation options.

If task allocation to the agents is explicitly provided, approaches like [15] and [8] take communication and motion coordination between the agents into account; [13, 11] choose a Petri-Net based approach for explicit coordination whereas [6] uses a game theoretic approach for negotiation, able to consider adversarial agents. In contrast, our approach does not assume known allocation and instead decomposes the mission such that execution does not require coordination between the agents, enabling them to operate independently based on their assigned task.

The contributions of this paper are as follows: (i) A formal definition of the decomposition of finite LTL specifications into tasks is introduced and several relevant properties of such decompositions are discussed. (ii) It is shown how such decompositions can be efficiently identified in an automaton representation of the LTL specification. (iii) Based on these results, a method is presented for constructing a team model of tractable complexity with respect to the team size that can be used for efficient multi-agent planning. The proposed approach and its computational advantages are illustrated on an example system setup motivated by the scenarios mentioned above. Specifically, we evaluate state space complexity of the team model and planning time based on a ROS implementation of our approach.

2 Preliminaries

2.1 LTL Semantics

An LTL specification ϕ over a set of atomic propositions Π identifies a set of temporal sequences $\sigma = \sigma(1)\sigma(2)...$ which fulfill this specification, written as $\sigma \vDash \phi$. At each discrete time $t \in \mathbb{T}$ with $\mathbb{T} \subseteq \mathbb{N}$, a set of propositions $\sigma(t) \subseteq \Pi$ is true, i.e., a sequence is defined as $\sigma \colon \mathbb{T} \to 2^{\Pi}$. A sequence is called finite if it is bounded by a maximum time T and then we say it has length T. Finite LTL is a variant of LTL that can be interpreted over finite sequences. Classes of finite LTL are formulas which are insensitive to infiniteness [5], for example co-safe LTL [12] or LTL_f [4].

The semantics of LTL over a finite sequence $\sigma(1)\dots\sigma(T)$ are defined as follows: **(1)** $\sigma(t) \vDash \pi$ iff $\pi \in \sigma(t)$; **(2)** $\sigma(t) \vDash \neg\varphi$ iff $\sigma(t) \nvDash \varphi$; **(3)** $\sigma(t) \vDash \varphi_1 \wedge \varphi_2$ iff $\sigma(t) \vDash \varphi_1$ and $\sigma(t) \vDash \varphi_2$; **(4)** $\sigma(t) \vDash \varphi_1 \vee \varphi_2$ iff $\sigma(t) \vDash \varphi_1$ or $\sigma(t) \vDash \varphi_2$; **(5)** $\sigma(t) \vDash \circ\varphi$ iff $\sigma(t+1) \vDash \varphi$; **(6)** $\sigma(t) \vDash \varphi_1 \mathscr{U} \varphi_2$ iff there exists $t_2 \in [t,T]$ such that $\sigma(t_2) \vDash \varphi_2$ and $\sigma(t_1) \vDash \varphi_1$ for all $t_1 \in [t, t_2-1]$; **(7)** $\sigma(t) \vDash \varphi_1 \mathscr{R} \varphi_2$ iff for all $t_2 \in [t,T]$ either $\sigma(t_2) \vDash \varphi_2$ or there exists a $t_1 \in [t, t_2-1]$ such that $\sigma(t_1) \vDash \varphi_1$.

Furthermore, we derive the operators *eventually* $\diamond\varphi = \top \mathscr{U} \varphi$ and *always* $\Box\varphi = \Omega \mathscr{R} \varphi$ with $\Omega = \bot$ for all $t \leq T$, where $\top$ denotes *true* and $\bot$ denotes *false*. Note that, as usual for finite LTL [5], the scope of *always* is limited to the range $t \in \{1,\dots,T\}$, not considering an infinite future.

In the case of finite LTL specifications, a finite automaton can be constructed from the given LTL formula ϕ [2].

Definition 1 (NFA). A nondeterministic finite automaton (NFA) is given as the tuple $\mathscr{F} = (Q, Q_0, \alpha, \delta, F)$ consisting of **(1)** a set of states Q, **(2)** a set of initial states $Q_0 \subseteq Q$, **(3)** an alphabet α of Boolean formulas over $\pi \in \Pi$, **(4)** a set of transition conditions $\delta \colon Q \times Q \to \alpha$, **(5)** a set of accepting (final) states $F \subseteq Q$.

Note that we define the set of transitions as Boolean conditions which need to be fulfilled for taking a transition. Especially, the absence of a transition is denoted by the formula $\delta(q_1, q_2) = \bot$, which can never be fulfilled. A finite sequence of states $q \in Q$ is called a run $\rho \colon \{0,1,...,T\} \to Q$ with $\delta(\rho(t-1), \rho(t)) \neq \bot$ for all $t \in \{1,...,T\}$. We say that a sequence σ describes ρ if $\sigma(t) \vDash \delta(\rho(t-1), \rho(t))$ for all $t \in \{1,...,T\}$. In the case that a run ρ leads from an initial state $\rho(0) \in Q_0$ to an accepting state $\rho(T) \in F$, this run is called accepting. If $\mathscr{F}$ is constructed appropriately [22], a finite sequence σ fulfills a finite LTL formula ϕ if and only if σ describes an accepting run ρ in the NFA $\mathscr{F}$ constructed from ϕ.

In general, constructing an NFA from an LTL formula has worst-case time and space complexity exponential in the length of the formula $|\phi|$ [2]. The actual complexity might be lower depending on the specific formula. However, this construction complexity is usually still a limiting factor.

2.2 Closure Labeling

In order to decide if a sequence σ fulfills an LTL specification ϕ, Wolper [22] defines a closure labeling $\tau\colon \mathbb{N} \to 2^{cl(\phi)}$ corresponding to σ and constructed as given below. The closure of ϕ is the set of its subformulas, given by $cl(\phi)$ with $\phi \in cl(\phi)$ and recursively for all operations $\circ\varphi_1 \implies \varphi_1 \in cl(\phi)$ and $(\varphi_1 \wedge \varphi_2), (\varphi_1 \vee \varphi_2), (\varphi_1 \mathscr{U} \varphi_2), (\varphi_1 \mathscr{R} \varphi_2) \in cl(\phi) \implies \varphi_1, \varphi_2 \in cl(\phi)$.

The closure labeling enables to formally reason about requirements imposed by the part of a sequence σ from 1 to t, in the following denoted by $\sigma(1,...,t)$. Following the intuition of [7], we divide the construction of τ into two parts, one defining *expectations* from the previous step τ^e and one consequently required *observations* τ^o. Then, τ is given by $\tau(t) = \tau^e(t) \cup \tau^o(t)$.

Definition 2 (Expectations). Expectations $\tau^e(t+1)$ on the next time step are constructed such that:

- if $\circ\varphi_1 \in \tau(t)$ then $\varphi_1 \in \tau^e(t+1)$
- if $\varphi_1 \mathscr{U} \varphi_2 \in \tau(t)$ and $\varphi_2 \notin \tau(t)$, then $\varphi_1 \mathscr{U} \varphi_2 \in \tau^e(t+1)$
- if $\varphi_1 \mathscr{R} \varphi_2 \in \tau(t)$ and $\varphi_1 \notin \tau(t)$, then $\varphi_1 \mathscr{R} \varphi_2 \in \tau^e(t+1)$.

Definition 3 (Observations). Observations $\tau^o(t)$ on the current time step are constructed such that:

- $\bot \notin \tau^o(t)$
- if $\varphi_1 \wedge \varphi_2 \in \tau(t)$ then $\varphi_1 \in \tau^o(t)$ and $\varphi_2 \in \tau^o(t)$
- if $\varphi_1 \vee \varphi_2 \in \tau(t)$ then $\varphi_1 \in \tau^o(t)$ or $\varphi_2 \in \tau^o(t)$
- if $\varphi_1 \mathscr{U} \varphi_2 \in \tau(t)$ then either $\varphi_2 \in \tau^o(t)$, or $\varphi_1 \in \tau^o(t)$ and $\varphi_1 \mathscr{U} \varphi_2 \in \tau^e(t+1)$
- if $\varphi_1 \mathscr{R} \varphi_2 \in \tau(t)$ then $\varphi_2 \in \tau^o(t)$, and either $\varphi_1 \in \tau^o(t)$ or $\varphi_1 \mathscr{R} \varphi_2 \in \tau^e(t+1)$.

Requirements on observations $\tau^o(t)$ at time t initially come from the expectations $\tau^e(t)$ on this time step. Thus, starting from $t = 1$, the closure labeling τ can be constructed consecutively. Finally, the following rules regarding propositions need to hold true for all $\pi \in \Pi$ with respect to σ. If $\pi \in \tau(t)$ then $\pi \in \sigma(t)$ and if $\neg\pi \in \tau(t)$ then $\pi \notin \sigma(t)$. Otherwise, we say that σ violates the requirements imposed by the LTL specification ϕ.

By this construction of τ, $\sigma(t,...,T)$ fulfills all $\varphi \in \tau(t)$ if and only if $\tau^e(T+1) \cap cl(\varphi) = \emptyset$. In particular, we can check if $\sigma \vDash \phi$ by constructing τ from the expectation $\tau^e(1) = \{\phi\}$ and then check if $\tau^e(T+1) = \emptyset$. Note that [22] additionally requires that there exists a $t' \geq t$ such that $\varphi_2 \in \tau(t')$ for the until operation $\varphi_1 \mathscr{U} \varphi_2 \in \tau(t)$. However, this requirement is already covered in our extended acceptance condition $\tau^e(T+1) = \emptyset$ and does not need to be explicitly required here.

Illustratively speaking, observations τ^o formally describe what needs to be observed at a given time, while expectations τ^e specify expected future observations. Intuitively, if not all observations are fulfilled at a certain time, the corresponding LTL formula is violated. If this is not the case and at some point, there are no implied expectations anymore, the LTL formula is satisfied.

2.3 System Model

Every agent is represented by a transition system to model its available actions and define which propositions are true consequently. It usually combines a topological map of the environment with discrete actions, which can be executed at certain locations, and is formally defined as follows.

Definition 4 (Agent Model). An agent model is given as the transition system $\mathscr{A} = (S_\mathscr{A}, s_{0,\mathscr{A}}, A_\mathscr{A}, \Pi, \lambda, C_\mathscr{A})$ consisting of **(1)** a set of states $S_\mathscr{A}$, **(2)** an initial state $s_{0,\mathscr{A}} \in S_\mathscr{A}$, **(3)** a set of actions $A_\mathscr{A} \subseteq S_\mathscr{A} \times S_\mathscr{A}$, **(4)** a set of propositions Π, **(5)** a labeling function $\lambda : S_\mathscr{A} \to 2^\Pi$, **(6)** action costs $C_\mathscr{A} : A_\mathscr{A} \to \mathbb{R}$.

Forming a product between the agent model $\mathscr{A}$ and the NFA $\mathscr{F}$ constructed from the LTL specification ϕ creates an automaton that combines properties of $\mathscr{A}$ and $\mathscr{F}$.

Definition 5 (Product Automaton). A product automaton is a tuple $\mathscr{P} = \mathscr{F} \otimes \mathscr{A} = (S_\mathscr{P}, S_{0,\mathscr{P}}, A_\mathscr{P}, C_\mathscr{P})$ consisting of **(1)** a set of states $S_\mathscr{P} = Q \times S_\mathscr{A}$, **(2)** a set of initial states $S_{0,\mathscr{P}} = \{(q, s_{0,\mathscr{A}}) \in S_\mathscr{P} : q \in Q_0\}$, **(3)** a set of actions $A_\mathscr{P} = \{((q_\mathrm{s}, s_\mathrm{s}), (q_\mathrm{t}, s_\mathrm{t})) \in S_\mathscr{P} \times S_\mathscr{P} : (s_\mathrm{s}, s_\mathrm{t}) \in A_\mathscr{A} \wedge \lambda(s_\mathrm{s}) \vDash \delta(q_\mathrm{s}, q_\mathrm{t})\}$, **(4)** action costs $C_\mathscr{P} : A_\mathscr{P} \to \mathbb{R}$ with $C_\mathscr{P}(a_\mathscr{P}) = C_\mathscr{A}(a_\mathscr{A})$.

Consequently, only actions that do not violate ϕ are contained in the model. A run ending in a state $s_n = (q, s_\mathscr{A})$ with $q \in F$ being an accepting state in $\mathscr{F}$ gives an action sequence which fulfills ϕ.

Definition 6 (Action Sequence). An action sequence β is given by $\beta = s_0 a_1 s_1 ... a_n s_n$ with $s_i \in S_\mathscr{P}$, $s_0 \in S_{0,\mathscr{P}}$, and $a_j = (s_{j-1}, s_j) \in A_\mathscr{P}$.

3 LTL Decomposition

Considering a multi-agent system with N agents, we can represent each robotic agent $r \in \{1, \ldots, N\}$ according to the above definitions by an individual product automaton $\mathscr{P}^{(r)}$ created from its agent model $\mathscr{A}^{(r)}$ and the NFA $\mathscr{F}$, obtained from the complete LTL mission specification denoted by $\mathscr{M}$.

Problem 1. Given a team of agents $r \in \{1, \ldots, N\}$, each modeled as $\mathscr{A}^{(r)}$, and the finite LTL mission specification $\mathscr{M}$, provide independent action sequences $\beta^{(r)}$ for all agents such that $\mathscr{M}$ is fulfilled in an optimal way for the specified team cost.

In order to utilize the team of agents, it is desirable to decompose the mission $\mathscr{M}$ such that parts of it can be allocated to different agents, i.e., automatically identify independently executable task specifications if there exist some. This decomposition will allow us to distribute $\mathscr{M}$ and solve Problem 1 as if the mission has been specified by one LTL formula for each agent. Based on the above LTL semantics, we define what we accept as a semantically valid decomposition of a finite LTL mission $\mathscr{M}$ following the motivation of specifying $\mathscr{M}$ as a set of independent tasks.

Definition 7 (Finite Decomposition). Let $\mathscr{T}_i$ with $i \in \{1,...,n\}$ be a set of finite LTL task specifications and σ_i denote any sequence such that $\sigma_i \vDash \mathscr{T}_i$. These tasks are called a decomposition of the finite LTL mission specification $\mathscr{M}$ if and only if:

$$\sigma_{j_1}...\sigma_{j_i}...\sigma_{j_n} \vDash \mathscr{M} \tag{1}$$

for all permutations of $j_i \in \{1,...,n\}$ and all respective sequences σ_i.

Note that this decomposition condition (1) includes that each $\mathscr{T}_i$ is a safe, i.e., non-violating, prefix of $\mathscr{M}$. Furthermore, by requiring all permutations of sequences to be feasible, we make sure that no σ_i implies expectations to be respected by other sequences of the decomposition, and that the set of tasks completely covers $\mathscr{M}$. In the following, we discuss how to decompose the mission $\mathscr{M}$ into independently executable tasks $\mathscr{T}_i$ such that $\mathscr{M}$ is fulfilled if the set of tasks is fulfilled.

Example. Consider the LTL mission specification $\mathscr{M} = \diamond a \wedge \diamond b \wedge \square(b \rightarrow c)$. In this simple case, as will be more clear at the end of this section, a possible decomposition of $\mathscr{M}$ is given by $\mathscr{T}_1 = \diamond a \wedge \square(b \rightarrow c)$ and $\mathscr{T}_2 = \diamond b \wedge \square(b \rightarrow c)$. For example, the sequence $\sigma = \sigma_1\sigma_2$ with $\sigma_1 = \{c\}\{a\}$, $\sigma_2 = \{a\}\{b,c\}$ would fulfill $\mathscr{M}$, and also the permutation $\sigma_2\sigma_1 = \{a\}\{b,c\}\{c\}\{a\}$ would be valid. However, note that the modification of the above solution such that $\mathscr{T}_1 = \diamond a$ would not constitute a valid decomposition. In this case, for example, $\sigma_1' = \{b\}\{a\} \vDash \mathscr{T}_1$ and still, $\sigma_2 \vDash \mathscr{T}_2$, but $\sigma_1'\sigma_2 \nvDash \mathscr{M}$.

For more complex LTL formulas, the explicit LTL formulation of a decomposition can be significantly different from simply splitting the mission specification or replicating some parts. However, a boolean conjunction of all tasks $\mathscr{T}_i$ always gives the complete specification $\mathscr{M}$.

Notation Remark. The following notation conventions are used throughout the rest of this paper. $\mathscr{M}$ is the finite LTL mission specification, σ a sequence such that $\sigma \vDash \mathscr{M}$ and τ the closure labeling of σ. A different σ will have a different τ. Furthermore, $\mathscr{T}_i$ is a finite LTL task specification, i.e., a subformula of $\mathscr{M}$, σ_i a sequence such that $\sigma_i \vDash \mathscr{T}_i$ and τ_i the closure labeling of σ_i. Note that τ_i is defined over the closure $cl(\mathscr{T}_i) \subset cl(\mathscr{M})$ while τ is defined over $cl(\mathscr{M})$. Accordingly for τ and τ_i, T and T_i denote the ending times, τ^e and τ_i^e the expectations, and so on.

In order to efficiently determine a valid LTL decomposition as discussed above, note the following observation.

Theorem 1 (Transitivity). *If $\mathscr{T}_1$, $\mathscr{T}_2$ is a decomposition of $\mathscr{M}$ and $\mathscr{T}_3$, $\mathscr{T}_4$ is a decomposition of $\mathscr{T}_2$, then $\mathscr{T}_1$, $\mathscr{T}_3$, $\mathscr{T}_4$ is a decomposition of $\mathscr{M}$.*

Proof. We need to show that all six permutations of the sequences σ_1, σ_3, σ_4 fulfill $\mathscr{M}$. Four of them are rather trivial by substitution of σ_2 with $\sigma_3\sigma_4$ or $\sigma_4\sigma_3$. However, for proving $\sigma_3\sigma_1\sigma_4 \vDash \mathscr{M}$ and $\sigma_4\sigma_1\sigma_3 \vDash \mathscr{M}$, the closure labeling is used. Specifically, we need to show that we can construct a closure labeling τ of $\mathscr{M}$ from the closure labelings τ_i of the tasks $\mathscr{T}_i$, $i \in \{1,3,4\}$, such that $\tau^e(1) = \{\mathscr{M}\}$ and $\tau^e(T+1) = \emptyset$ for completion time $T = T_1 + T_3 + T_4$.

For this purpose, we construct a candidate τ with $\tau^e(1) = \{\mathscr{M}\}$ from the given set of τ_i, and then show that this always leads to $\tau^e(T+1) = \emptyset$. Since $\mathscr{T}_1$, $\mathscr{T}_2$ are

a decomposition of $\mathscr{M}$, following decomposition condition (1), $\tau^e(1) = \{\mathscr{M}\}$ is equivalent to $\tau^e(1) = \{\mathscr{T}_1, \mathscr{T}_2\}$, which itself is equivalent to $\tau^e(1) = \{\mathscr{T}_1, \mathscr{T}_3, \mathscr{T}_4\}$ for $\mathscr{T}_3, \mathscr{T}_4$ being a decomposition of $\mathscr{T}_2$. Consequently, we have $\tau(1) = \tau^e(1) \cup \tau^o(1)$ with $\tau^o(1)$ following Definition 3.

We start by considering the permutation $\sigma_3\sigma_1\sigma_4$. For the first part, $t \in [2, T_3]$, we construct the candidate τ from τ_3 such that $\tau(t) = \tau_3(t) \cup (\tau(1) \setminus cl(\mathscr{T}_3))$. From using τ_3 we get that this part fulfills $\mathscr{T}_3$, and extend τ_3 by all requirements which are not covered by $\mathscr{T}_3$, i.e., which are not in $cl(\mathscr{T}_3)$. τ is still valid because the tasks are decomposition pairs as stated in the Theorem. Specifically, $\sigma_{2,1} = \sigma_2\sigma_1 \vDash \mathscr{M}$ implies $\mathscr{T}_1 \in \tau_{2,1}(t)$ with $t \in [1, T_2 + 1]$ for the closure labeling $\tau_{2,1}$ of the permutation $\sigma_2\sigma_1$. This means that the requirement $\mathscr{T}_1$ cannot be violated at any time during execution of $\sigma_2 = \sigma_3\sigma_4$, and thus, also during σ_3.

We can repeat this construction for the remaining two parts, continuing with $\tau^e(T_3 + 1) = \{\mathscr{T}_1, \mathscr{T}_4\}$. Finally, this leads to $\tau^e(T_3 + T_1 + T_4 + 1) = \emptyset$, meaning that $\mathscr{M}$ is fulfilled. Thus, we see that the constructed candidate is a valid closure labeling respecting all requirements and consequently, $\sigma_3\sigma_1\sigma_4 \vDash \mathscr{M}$. The proof for the last permutation $\sigma_4\sigma_1\sigma_3 \vDash \mathscr{M}$ follows accordingly. □

Theorem 1 has especially two consequences. First, only n specific permutations instead of all $n!$ permutations need to be checked in order to decide if a set of n tasks $\mathscr{T}_i$ is a valid decomposition of $\mathscr{M}$. This is obtained by forming pairs, each of one task $\mathscr{T}_i$ and combination of the other $n-1$ tasks, for example by a conjunction. Then, it is sufficient to check the decomposition condition (1) only for these n permutations in order to decide if these tasks form a decomposition of $\mathscr{M}$. Illustratively, the specific n permutations individually separate tasks $\mathscr{T}_i$ from the rest to decide whether $\mathscr{T}_i$ is independent.

Second, it is not required to find a complete set of tasks decomposing $\mathscr{M}$ at once. Instead, it is possible to step-wise identify individual parts to be isolated into a separate task $\mathscr{T}_i$ of the final decomposition and continue with the rest of $\mathscr{M}$. This progress can be repeated until no further task is found to be isolated and especially enables automata-based approaches for finding possible decompositions.

3.1 Decomposition Set

In general, different decompositions of $\mathscr{M}$ can exist and a task $\mathscr{T}_i$ does not need to be minimal in the sense that it cannot be further decomposed. Thus, we propose an efficient automata-based approach to identify all possible choices of decomposition as shown in the remainder of this section. First, note the following relation between states $q \in Q$ of the NFA $\mathscr{F}$ constructed from the LTL mission specification $\mathscr{M}$ and the closure $cl(\mathscr{M})$.

Lemma 1 (Subformula Labeling). *Each state $q \in Q$ of the NFA $\mathscr{F}$ constructed from $\mathscr{M}$ can be labeled with subformulas $\Phi_q \in 2^{cl(\mathscr{M})}$ which are required to be true at this particular state.*

We refer the interested reader to [22], Section 4.4, for a detailed proof, covering the more general case of infinite sequences. In summary, $\mathscr{F}$ is explicitly constructed from $\mathscr{M}$ such that its state space Q is given by $2^{cl(\mathscr{M})}$ as discussed in Section 2.1, and there is a transition if and only if the successor state fulfills the requirements of the closure labeling of its predecessor. In particular, note that $\mathscr{M} \in \Phi_q$ for all $q \in Q_0$ and $\Phi_q = \emptyset$ for all $q \in F$. Lemma 1 shows the connection between the NFA $\mathscr{F}$ and the closure labeling τ, since τ is as well defined over the subformulas $2^{cl(\mathscr{M})}$ and construction of $\mathscr{F}$ respects the requirements imposed by τ.

Furthermore, we introduce the following notion of *essential sequences* to generalize over sequences by associating them with runs ρ in the NFA.

Definition 8 (Essential Sequence). A sequence σ is called essential for an NFA $\mathscr{F}$ if and only if it describes a run ρ in $\mathscr{F}$ and $\sigma(t) \setminus \{\pi\} \nvDash \delta(\rho(t-1), \rho(t))$ for all t and propositions $\pi \in \sigma(t)$, i.e., σ only contains required propositions.

This notation is motivated by the closure labeling τ of σ. By restricting σ to satisfy only the conditions explicitly required by τ, we get the following property.

Lemma 2 (Closure Coverage). *Let τ denote the closure labeling of a sequence σ. If σ is an essential sequence, any other τ' satisfied by σ is at most as restrictive as τ, in the sense that $\tau'(t) \cap \Pi \subseteq \tau(t) \cap \Pi$ for every t and the set of propositions Π.*

Proof. Assume there would be a $\pi \in \Pi$ such that $\pi \in \tau'(t)$ and $\pi \notin \tau(t)$. τ' would then require that $\pi \in \sigma(t)$. However, this cannot be the case since σ is essential. □

This property ensures that if an essential sequence describes a run in one part of the NFA corresponding to τ as well as one corresponding to τ', any other non-essential sequence conforming with τ will not violate τ' neither. This can be used to generalize over sequences without explicitly constructing the closure labeling, but instead finding an essential sequence.

Finally, we can associate a pair of tasks $\mathscr{T}_1^q$, $\mathscr{T}_2^q$ with a state $q \in Q$ of $\mathscr{F}$. Every sequence σ_1 describing a run ρ_1 from an initial state $q_0 \in Q_0$ to q satisfies $\mathscr{T}_1^q$, specified by the set of fulfilled subformulas $\Phi_{q_0} \setminus \Phi_q$. $\mathscr{T}_2^q$ is given accordingly by $\Phi_q \setminus \Phi_{q_F} = \Phi_q$ with $q_F \in F$ and represents the rest of $\mathscr{M}$ not fulfilled by $\mathscr{T}_1^q$.

It remains to decide if the pair $\mathscr{T}_1^q$, $\mathscr{T}_2^q$ resulting from a split forms a valid decomposition of $\mathscr{M}$, and we define the *decomposition set* of $\mathscr{F}$ as follows.

Definition 9 (Decomposition Set). The decomposition set $D \subseteq Q$ of the NFA $\mathscr{F}$ constructed from $\mathscr{M}$ contains all states q for which the pair of tasks $\mathscr{T}_1^q$, $\mathscr{T}_2^q$ defines a valid decomposition of $\mathscr{M}$ according to Definition 7.

This decomposition set can then be constructed as follows, giving all possible decomposition choices of the finite LTL mission specification $\mathscr{M}$.

Theorem 2 (Decomposability). *Let $q \in Q$ be a state in the NFA $\mathscr{F}$ constructed from $\mathscr{M}$, and $\sigma = \sigma_1\sigma_2$ be an essential sequence such that σ_1 describes a run from an initial state to q and σ_2 describes a run from q to an accepting state of $\mathscr{F}$. Then, $q \in D$ if and only if $\widehat{\sigma} = \sigma_2\sigma_1$ describes an accepting run in $\mathscr{F}$.*

Proof. The "only if"-part follows directly from the decomposition condition in Definition 7. For the "if"-part, it remains to show that σ generalizes over all possible $\sigma_1' \vDash \mathscr{T}_1^{\mathrm{q}}$ and $\sigma_2' \vDash \mathscr{T}_2^{\mathrm{q}}$, i.e., all pairs of sequences describing a run through q. Note that, given that $\sigma = \sigma_1\sigma_2$ is an essential sequence, also σ_1 and σ_2 are essential.

First, we show that the essential sequence σ_1, generalizes over $\sigma_1' \vDash \mathscr{T}_1^{\mathrm{q}}$. This means, if $\widehat{\sigma} = \sigma_2\sigma_1$ describes an accepting run, then also any other $\sigma_2\sigma_1'$ describes an accepting run. According to Lemma 2, the closure labeling $\widehat{\tau}(t), t \in [T_2+1, T_2+T_1]$ of $\widehat{\sigma}$ is at most as restrictive as $\tau(t), t \in [1, T_1]$ of σ. This means that no sequence can violate $\widehat{\tau}$ if it conforms with τ.

Next, following Lemma 1, we can retrieve the closure labeling τ' of a sequence σ_1' from a run ρ' described by σ_1', given by $\tau'(t) = \Phi_q$ for $q = \rho'(t)$. By construction of the NFA and $\mathscr{T}_1^{\mathrm{q}}$, all sequences leading to the respective last state $\rho'(T)$ fulfill all requirements imposed by $\rho'(0)$. Although these sequences may have a different closure labeling τ', this always satisfies the same requirements as τ, given by $\sigma_1' \vDash \mathscr{T}_1^{\mathrm{q}} = \Phi_{\mathrm{q}_0} \setminus \Phi_{\mathrm{q}}$ with $q_0 = \rho'(0) = \rho(0)$ and $q = \rho'(T_1) = \rho(T_1)$ where ρ is described by σ_1. Consequently, σ_1' cannot violate $\widehat{\tau}$ as shown by Lemma 2.

Finally for the permutation $\sigma_2\sigma_1'$, this gives that any $\sigma_1' \vDash \mathscr{T}_1^{\mathrm{q}}$ applied to the same state as σ_1 leads to an accepting state and thus, $\sigma_2\sigma_1'$ describes an accepting run if $\widehat{\sigma} = \sigma_2\sigma_1$ does, i.e., the essential sequence σ_1 indeed generalizes over possible different realizations of $\mathscr{T}_1^{\mathrm{q}}$. The same then holds true accordingly for σ_2 and thus, we get $\sigma_2'\sigma_1' \vDash \mathscr{M}$ if and only if $\sigma_2\sigma_1 \vDash \mathscr{M}$, given that σ_1 and σ_2 are essential. □

Note that Theorem 2 only requires to check one essential sequence, which is much more efficient than the requirement to check every single possible sequence. Furthermore, an essential sequence σ to a specific state q can be easily constructed from an NFA $\mathscr{F}$, for example by representing the set of transition conditions α of $\mathscr{F}$ in disjunctive normal form (DNF). Then, the essential sequence to q is given by the propositions which are true in one of the conjunctive clauses along the path to q. By step-wise constructing these sequences σ for all states first, all essential sequences can be found in linear time with respect to $|Q|$, which is non-critical compared to constructing $\mathscr{F}$ as discussed earlier in Section 2.1.

4 Team Model Construction

Based on the results of the previous section, a team model can be constructed as follows in order to solve Problem 1. First, the mission specification $\mathscr{M}$ is translated to an equivalent NFA $\mathscr{F}$. Next, we form a local product automaton $\mathscr{P}^{(r)} = \mathscr{F} \otimes \mathscr{A}^{(r)}$ for each agent $r \in \{1, ..., N\}$. Unlike previous task allocation approaches, we do not explicitly calculate the costs for each subset of tasks resulting from a possible decomposition choice in each of the local $\mathscr{P}^{(r)}$. Instead, we combine these local product automata into a team model of tractable complexity in which the optimal task allocation can be calculated much more efficiently.

The basis for this team model is given by a union of all $\mathscr{P}^{(r)}$, resulting in N unconnected partitions. Afterwards, additional *switch transitions* connect these parti-

tions. They represent an option in the planning process to consider a different agent for allocation of the part of the mission which is not yet assigned. Such a switch transition is only present if both parts of the mission form a valid decomposition. Formally, the team automaton is defined as follows.

Definition 10 (Team Automaton). The team automaton $\mathscr{G}$ is a union of N local product automata $\mathscr{P}^{(r)}$ with $r \in \{1,...,N\}$ given by $\mathscr{G} = (S_\mathscr{G}, S_{0,\mathscr{G}}, A_\mathscr{G}, C_\mathscr{G})$ consisting of **(1)** a set of states $S_\mathscr{G} = \{(r,q,s) \colon r \in \{1,...,N\}, (q,s) \in S_\mathscr{P}^{(r)}\}$, **(2)** a set of initial states $S_{0,\mathscr{G}} = \{(r,q,s) \in S_\mathscr{G} \colon r = 1\}$ equivalent to the initial states of one arbitrary agent, **(3)** a set of actions $A_\mathscr{G} = \bigcup_r A_\mathscr{P}^{(r)} \cup \zeta$ where the individual agent actions $A_\mathscr{P}^{(r)}$ are extended by the set of switch transitions ζ as defined below, **(4)** action costs $C_\mathscr{G} \colon A_\mathscr{G} \to \mathbb{R}$ with $C_\mathscr{G}(a_\mathscr{G}) = C_\mathscr{P}^{(r)}(a_\mathscr{P}^{(r)})$ and $C_\mathscr{G}(\varsigma) = 0$ for all $\varsigma \in \zeta$.

Core part of this composition is constructing the set of switch transitions ζ connecting states in the partitions of two different agents and preserving mission progress, restricted to states corresponding to a valid mission decomposition.

Definition 11 (Switch Transition). The set $\zeta \subset S_\mathscr{G} \times S_\mathscr{G}$ denotes switch transitions in the team automaton $\mathscr{G}$ and $\varsigma \in \zeta$ for $\varsigma = ((i,q_\mathrm{s},s_\mathrm{s}),(j,q_\mathrm{t},s_\mathrm{t}))$ if and only if it **(i)** connects different agents: $i \neq j$, **(ii)** preserves the NFA progress: $q_\mathrm{s} = q_\mathrm{t}$, **(iii)** is directed: $r_i \prec r_j$ for an arbitrary ordering of agents $r_1 \prec r_2 \prec ... \prec r_N$, **(iv)** points to an initial agent state: $s_\mathrm{t} = s_{0,\mathscr{A}}^{(j)}$, **(v)** implies a valid decomposition of $\mathscr{F}$: $q_\mathrm{s} \in D$.

While condition (i) is trivial, (ii) characterizes the main purpose of a switch transition, which is transferring the mission progress to another agent. Condition (iv) in combination with (iii) requires to account for the initial state of each agent. Specifically, (iii) ensures that each agent is considered exactly once for participating in solving the mission. Finally, (v) guarantees that any possible decomposition resulting from switch transitions is valid.

The model $\mathscr{G}$ has a much lower state space complexity than the complete product $\mathscr{C}_\mathrm{Prod} = \mathscr{P}^{(1)} \otimes ... \otimes \mathscr{P}^{(N)}$ of all local automata, which would be required if we did not decompose the LTL mission into independent tasks. Specifically, the number of states of $\mathscr{G}$ is linear in the number of agents N and given by $O(N \cdot |Q| \cdot |S_\mathscr{A}|)$. $|S_\mathscr{A}|$ denotes the number of states of the agent model and $|Q|$ the number of states of the NFA $\mathscr{F}$. In contrast, the state space complexity of $\mathscr{C}_\mathrm{Prod}$ would be exponential in the number of agents with $O(|Q| \cdot |S_\mathscr{A}|^N)$.

A team model $\mathscr{G}$ constructed as defined above enables to employ conventional graph-search algorithms for obtaining optimal action sequences $\beta^{(r)}$ for all agents such that the LTL mission specification $\mathscr{M}$ is fulfilled. Consequently, this solves Problem 1 and is summarized by the following properties.

Correctness. $\mathscr{P}^{(r)} = \mathscr{F} \otimes \mathscr{A}$ preserves the acceptance criterion of $\mathscr{F}$. A union of the state space when constructing $\mathscr{G}$ out of all $\mathscr{P}^{(r)}$ does not add any new transitions except ζ and because condition (ii) requires all $\varsigma \in \zeta$ to preserve the NFA component, any accepting run ρ in $\mathscr{G}$ satisfies $\mathscr{M}$.

Independence. Given by switch condition (v) and the construction of the decomposition set D as discussed above, parts of ρ referring to different agents r solve

independent tasks $\mathscr{T}_r$ and thus, do not affect each other. Especially, each $\mathscr{T}_r$ is a safe prefix of all other tasks, i.e., all constraints of $\mathscr{M}$ are covered by $\mathscr{T}_r$.

Completeness. Any set of individual agent action sequences $\beta^{(r)}$ resulting from a run in the complete product automaton $\mathscr{C}_{\text{Prod}}$ is also present in the reduced team model $\mathscr{G}$, since the parts referring to different agents are independent. Especially, also the optimal solution in $\mathscr{C}_{\text{Prod}}$ is as well contained in $\mathscr{G}$.

5 Evaluation

The presented approach has been implemented in ROS and evaluated both in simulation and on a real system. In the following, we discuss our performance evaluation results for a set of simulated scenarios and compare them to the conventional product model approach. For planning the optimal action sequence based on the constructed team model, we used a conventional Bellman-Ford graph-search and minimize the largest individual agent costs, i.e., aim to distribute the mission equally. Note that, although action costs are usually positive, we cannot use a greedy graph-search such as Dijkstra or A* because we aim to distribute the mission equally and not to minimize the sum of all action costs.

To evaluate applicability in scenarios as motivated in Section 1, we assume a hospital environment, depicted left in Figure 1, and form the agent model $\mathscr{A}$ as the product between this topological map (left) and a transition model of robot actions (right). The set of states $S_{\mathscr{A}}$ is given as the product between the map locations and the robot states. Propositions Π according to the state labels describe specific properties of locations and robot states, e.g., p for pick-up locations, s for station rooms, c for carrying an object. The actions $A_{\mathscr{A}}$ consist of navigation actions according to the undirected edges in the map and further robot actions according to the robot model. These robot actions are limited to certain locations such that an action is only feasible if the respective state contains the propositions listed by the transition label, e.g., the transition from "normal" to "carrying" is only possible at pick-up locations p. Finally, action costs $C_{\mathscr{A}}$ are chosen to approximately represent the execution times of actions.

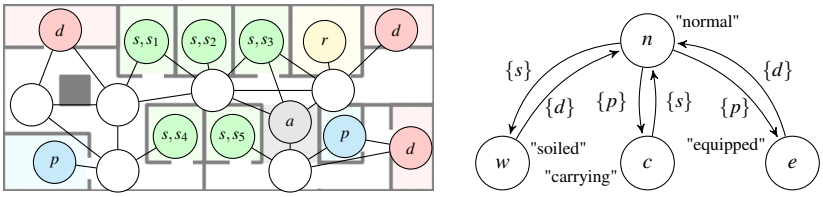


Fig. 1: Map (left) and robot capabilities (right) used in the scenarios. State labels denote propositions which are true at this state, transition labels denote requirements for performing an action.

5.1 Scenarios

Before we present illustrative scenarios in the presented environment, we discuss two mission specifications representing corner cases with respect to the decomposition of the mission, specifically for the size of the decomposition set D. First, requiring the team to visit the five station rooms in any order is essentially a multi-agent traveling salesman problem (TSP) and given by the mission $\mathscr{M}_{TSP} = \diamond s_1 \wedge \diamond s_2 \wedge \diamond s_3 \wedge \diamond s_4 \wedge \diamond s_5$. Consequently, all 32 states of the NFA $\mathscr{F}$ are in the decomposition set: $D = Q$. In this case, construction of $\mathscr{F}$ took approximately 56ms and determination of D around 2.2ms. Note that, although the decomposition set contains the full state space of $\mathscr{F}$, the proposed team automaton $\mathscr{G}$ still has a significantly lower state space complexity than the complete product $\mathscr{C}_{\text{Prod}}$. In fact, the state space complexity is independent of the size of the decomposition set and $|D|$ only determines the density of switch transitions.

In contrast, requiring visits to occur in a specific order is given by the mission $\mathscr{M}_{Seq} = \diamond(s_3 \wedge \diamond(s_4 \wedge \diamond(s_2 \wedge \diamond(s_5 \wedge \diamond s_1))))$. Different robots cannot execute parts of $\mathscr{M}_{Seq}$ independently since the correct order could not be guaranteed. Thus, the decomposition set only contains trivially the initial and accepting states: $D = Q_0 \cup F$, totaling to 2 of 6 states and reducing the mission to an allocation problem of choosing the single best robot to execute the mission alone. Construction of $\mathscr{F}$ took approximately 50ms and determination of D around 0.1ms. For both cases, specialized solutions exist to solve problems of this type. However, most missions in the motivated scenarios usually combine characteristics of both cases.

In the following, we consider three scenarios with different characteristics in the presented hospital environment to represent the most common use cases. Teams consist of three robots, although also the performance for varying team sizes is investigated at the end of this section as well.

Scenario 1 (Station Tour)

$$\mathscr{M}_1 = \diamond s_1 \wedge \diamond s_2 \wedge \diamond s_3 \wedge \diamond s_4 \wedge \diamond s_5 \wedge \square(s \rightarrow e) \wedge \square(e \rightarrow \neg a)$$

The robots are required to visit all five station rooms. In addition, a robot needs to carry medical equipment (proposition e) in order to be at any station room s and should avoid the public area a while being equipped. As presented before, robots know from the agent model $\mathscr{A}$ that equipment can only be picked up at pick-up locations p. This mission is similar to a constrained TSP, but requires robots to perform additional actions before visiting a room, regardless of which room they would choose to visit first.

Scenario 2 (Room Cleaning)

$$\mathscr{M}_2 = \diamond(s_3 \wedge \circ\delta) \wedge \diamond(s_4 \wedge \circ\delta) \wedge \diamond(s_5 \wedge \circ\delta) \wedge \square((\neg s \wedge \circ s) \rightarrow n)$$

with $\delta := w \,\mathscr{U}\, (d \wedge \circ(d \,\mathscr{U}\, \neg w))$. The robots need to pick up waste w at three of the rooms. In this scenario, robots are only required to be in "normal" state n in order to enter a state room s. But in addition, they are required to visit a dispose location d as consequence of visiting a room, given by δ. Again, this combines goal allocation with sequential action planning as consequence of servicing one of the goals.

	$t_{\mathscr{G}}$	$\lvert S_{\mathscr{G}}\rvert$	$t_{\mathscr{C}_{\text{Prod}}}$	$\lvert S_{\mathscr{C}_{\text{Prod}}}\rvert$
$\mathscr{M}_1$	0.965	6,912	1.71×10^4	1.19×10^7
$\mathscr{M}_2$	0.946	9,504	1.52×10^4	1.64×10^7
$\mathscr{M}_3$	2.908	13,824	$>4.32 \times 10^4$	2.39×10^7

Fig. 2: Analysis of the evaluation scenarios for teams of three agents, time given in seconds.

Scenario 3 (Medication Delivery)

$\mathscr{M}_3 = \diamond(s_1 \wedge n) \wedge \diamond(s_2 \wedge n) \wedge \diamond(s_3 \wedge n) \wedge \diamond(s_4 \wedge n) \wedge \diamond(s_5 \wedge n) \wedge \square((\neg s \wedge \circ s) \rightarrow c)$

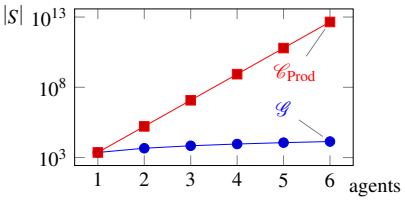
The robots need to deliver medication to all station rooms. They can only enter a room s when carrying medication c and need to deliver it by switching back to their "normal" state n. Consequently, robots need to repeatedly visit pick-up locations.

Figure 2 summarizes our performance results for the three scenarios, each randomly initialized. t is the average planning time in seconds, including model construction, calculation of the decomposition set and planning, and $|S|$ the total number of states in the model. We compare the team model $\mathscr{G}$ of our presented approach with the conventional complete product model $\mathscr{C}_{\text{Prod}}$. Already for the small team of three robots, our approach is much more efficient.

Furthermore, Figure 3 provides an analysis of how both approaches scale with an increasing number of agents, evaluated for scenario $\mathscr{M}_1$. The significantly increasing planning times on the product model $\mathscr{C}_{\text{Prod}}$ reflect the exponential growth of its state space. In contrast, our team model $\mathscr{G}$ scales well with increasing team size and produces reasonable results even for large teams.

6 Conclusion

Motivated by the need for efficient methods for multi-robot team planning, we presented an approach for decomposition of finite LTL mission specifications into independent tasks, resulting in a team model of tractable complexity for increasing team sizes. On this model, graph-search algorithms can efficiently distribute action sequences for the available robots, such that the LTL mission is completed by the team best suitable, which can dynamically change between missions. We illustrated the computational advantages of our approach over the conventional product model in example scenarios, resulting in significantly lower planning times.



agents	$t_{\mathscr{G}}$	$\lvert S_{\mathscr{G}}\rvert$	$t_{\mathscr{C}_{\text{Prod}}}$	$\lvert S_{\mathscr{C}_{\text{Prod}}}\rvert$
1	0.324	2.3×10^3	0.342	2.3×10^3
2	0.525	4.6×10^3	94.75	1.6×10^5
3	0.965	6.9×10^3	1.7×10^4	1.2×10^7
6	2.324	1.4×10^4	n/a	4.5×10^{12}
10	3.896	2.3×10^4	n/a	1.2×10^{19}
100	92.46	2.3×10^5	n/a	1.7×10^{187}

Fig. 3: Complexity analysis with respect to the team size, performed for scenario $\mathscr{M}_1$. Planning time in seconds, missing entries exceeded the maximum time of 8 hours.

References

[1] M. Agarwal, N. Kumar, and L. Vig. Non-additive multi-objective robot coalition formation. *Expert Systems with Applications*, 41(8):3736–3747, 2014.

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

[3] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Trans. on Robotics*, 28(1):158–171, 2012.

[4] G. De Giacomo and M. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Int. Joint Conf. on Art. Intell. (IJCAI)*, pages 854–860. Assoc. for Comp. Machinery, 2013.

[5] G. De Giacomo, R. De Masellis, and M. Montali. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *AAAI*, pages 1027–1033. Citeseer, 2014.

[6] J. Fu, H. Tanner, and J. Heinz. Concurrent multi-agent systems with temporal logic objectives: Game theoretic analysis and planning through negotiation. *IET Control Theory & Applications*, 9(3):465–474, 2015.

[7] R. Gerth, D. Peled, M. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Int. Symp. on Prot. Spec., Testing and Verification*. IFIP, 1995.

[8] M. Guo and D. V. Dimarogonas. Bottom-up motion and task coordination for loosely-coupled multi-agent systems with dependent local tasks. In *CASE*, pages 348–355. IEEE, 2015.

[9] S. Karaman and E. Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Conf. on Decision and Control (CDC)*, pages 3953–3958. IEEE, 2008.

[10] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, 26(1):48–61, 2010.

[11] M. Kloetzer and C. Mahulea. Accomplish multi-robot tasks via Petri net models. In *Int. Conf. on Automation Science and Engineering (CASE)*, pages 304–309. IEEE, 2015.

[12] O. Kupferman and M. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[13] B. Lacerda and P. Lima. LTL-based decentralized supervisory control of multi-robot tasks modelled as Petri nets. In *Int. Conf. on Intell. Robots and Systems (IROS)*, pages 3081–3086. IEEE, 2011.

[14] R. Luna, M. Lahijanian, M. Moll, and L. Kavraki. Asymptotically optimal stochastic motion planning with temporal goals. In *Alg. Found. of Robotics XI*, pages 335–352. Springer, 2015.

[15] V. Raman and H. Kress-Gazit. Synthesis for multi-robot controllers with interleaved motion. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 4316–4321. IEEE, 2014.

[16] V. Raman, C. Finucane, and H. Kress-Gazit. Temporal logic robot mission planning for slow and fast actions. In *Int. Robots and Systems (IROS)*, pages 251–256. IEEE, 2012.

[17] S. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning for surveillance with temporal logic constraints. *Int. Journal of Robotics Research*, 2011.

[18] A. Stefanescu. *Automatic synthesis of distributed transition systems*. PhD thesis, Univ. Stuttgart, 2006.

[19] J. Tumova and D. V. Dimarogonas. Decomposition of Multi-Agent Planning under Distributed Motion and Task LTL Specifications. In *CDC*, pages 1775–1780. IEEE, 2015.

[20] A. Ulusoy, S. Smith, X. C. Ding, and C. Belta. Robust multi-robot optimal path planning with temporal logic constraints. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 4693–4698. IEEE, 2012.

[21] E. Wolff, U. Topcu, and R. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 5319–5325. IEEE, 2014.

[22] P. Wolper. Constructing Automata from Temporal Logic Formulas: A Tutorial. In *Lect. on Form. Methods and Perf. Analysis*, pages 261–277. Springer, 2001.

[23] R. Zlot and A. Stentz. Complex task allocation for multiple robots. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1515–1522. IEEE, 2005.