Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets

Sofia Hustiu[1], Dimos V. Dimarogonas[2], Cristian Mahulea[3] and Marius Kloetzer[1]

***Abstract*— This paper proposes a high-level path planning strategy under Time Petri net (TPN) formalism for a multi-agent system, which is subject to Metric Interval Temporal Logic (MITL) specifications. The work aims to design a scalable model with respect to the number of agents, as the MITL formula requires multiple agents to ensure similar tasks. The obtained model is denoted *Composed Time Petri net* and it couples two TPN representations assigned to the motion of the agents, respectively to the MITL specification. The planning approach is based on model-checking methods and the results are evaluated on a case study applied in robotics industry.**

I. Introduction

The field of path planning for multi-agent systems aims to design control laws with respect to (w.r.t.) given requirements, such as: trajectory tracking [1], formation [2] and network control [3]. A particular topic is represented by motion planning control under spatial and/or temporal constraints using formal methods, e.g., "Eventually visit room A and avoid room B" or "Eventually visit room A within 3 time units".

Several specification languages can be used to express these constraints for the multi-agent system, as follows: Linear Temporal Logic (LTL) based on Boolean variables linked by logical and temporal requirements [4], [5], Metric Interval Temporal Logic (MITL) which extends the LTL specifications by imposing time constraints for satisfaction of temporal requirements [6], [7], Signal Temporal Logic (STL) which extends MITL specifications by adding constraints on real continuous variables instead of contrary to logical requirements having only two outputs (true and false) [8], and Time Interval Temporal Logic [9] having a more compact representation when compared with the previous types of specifications. These formalisms express compact, rich and complex mathematical constraints, in a user-friendly manner. The specifications can designate global tasks for the entire team, being further decomposed into local tasks [10], or can be given directly as local tasks, the agents cooperating among them to ensure other requirements [6], [5].

This work was partially supported by Digital Future Smart Construction project and Univ. de Zaragoza, Fundación Bancaria Ibercaja y Fundación CAI - IT 7/22 and by MINECO-FEDERER TED2021-130449B-I00 project.

[1]S. Hustiu and M. Kloetzer are with the Dept. of Automatic Control and Applied Informatics, Technical University "Gheorghe Asachi" of Iasi, Romania {sofia.hustiu, marius.kloetzer}@academic.tuiasi.ro

[2]D. V. Dimarogonas is with Division of Decision and Control System, KTH Royal Institute of Technology, Stockholm, Sweden. dimos@kth.se

[3]C. Mahulea is with the Aragón Institute of Engineering Research (I3A), University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain. cmahulea@unizar.es

MITL can consider both spatial and time constraints. In [6] a decentralized approach is proposed, as each agent receives a local MITL formula, is modeled as a Weighted Transition System, and the dynamics are coupled with its neighbors based on a network graph. Another decentralized solution, but for cooperative tasks, is presented in [7], as the agents communicate local information based on requests. In [11] is proposed a framework that combines both high-level and low-level path planning control, where local MITL tasks are modeled as Timed Automata, then translated to Zone Automata for which the sampling-based RRT* (Rapidly-exploring Random Trees) method is implemented. The mentioned papers provide automata-based solutions for multi-agent systems with local MITL specifications, as the literature is rich for model-checking techniques in this area. Our work is directed to fulfill MITL missions based on a different model denoted Time Petri net, providing also solutions for the scenarios that require multiple agents to perform the same task, e.g., monitoring a region of interest. The planning strategy is based on model-checking methods. A relevant work for this problem description is captured in [12] under the name *CensusSTL*, where the authors express the mission in STL, and the solution is returned by a Particle Swarm Optimization Problem.

One common challenge is finding an adequate discrete representation for multi-agent systems, including time constraints. Related works frequently are based on Weighted Transitions Systems, respectively Timed Automata [6] or Time Petri nets (TPN) [13]. The first two representations capture the individual motion of each agent, returning one product automata for the entire team. Through these works, an overall representation of the team is provided, but the size of the model increases exponentially with the number of agents in the team. Contrary, a more compact model can be obtained via TPN representation, by providing a fixed topology w.r.t. the number of robots. The expressiveness study from [14] highlights that both Time Petri net model and Timed Büchi Automata (TBA) are timed bisimilar. As a result, the provided translation from a TBA model to a TPN one [14] is incorporated in the current work. The benefits of the TPN model in motion planning of multi-agent systems can be enumerated as follows:

- provides a fixed topology for a known environment, which is scalable w.r.t. the number of agents (represented as tokens);
- allows the use of additional constraints such as generalized mutual exclusion constraints [15];

- the use of powerful model-checking algorithms that do not generate the entire full state space, e.g., Romeo tool [16].

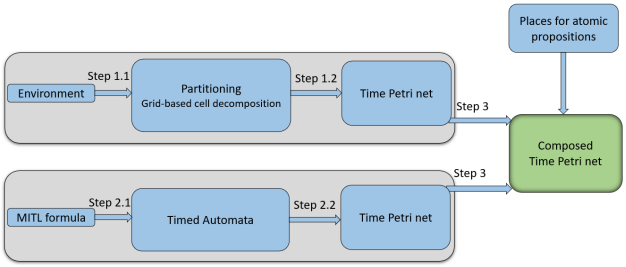

Fig. 1. General Framework treated in the paper

Thus, the current work provides a novel framework for high-level motion planning of the multi-agent systems under time constraints, in which global MITL missions should be ensured by sub-groups of agents with similar capabilities. Fig. 1 illustrates the proposed solution divided into steps, in which two models are coupled: the environment (step 1) and the MITL specification (step 2) into one *Composed Time Petri net*. As a result, one *Composed Time Petri net* model is defined for one MITL specification. With this work, we are interested in minimizing the gap in using TPN models for motion planning, for which the use of structural approaches such as mathematical programming represents an open problem in the literature.

The preliminaries and definitions are given in Section II. The problem statement is formally given in section III, while the proposed solution divided in steps is described in Section IV accompanied by examples. A case study is analyzed in Section V, the conclusions and future directions being captured in Section VI.

II. Preliminaries and Definitions

Let us consider an environment E including several disjointed regions of interest (ROI) which can be reached and/or avoided by a team of agents (mobile robots) with the same dynamics. Let us denote the set of ROIs with $\mathcal{Y} = \{y_1, y_2, \dots y_{|\mathcal{Y}|}\}$ and the set of mobile robots with $\mathcal{R} = \{r_1, r_2, \dots, r_{|\mathcal{R}|}\}$, where the cardinality of a set S is further denoted with $|S|$. The following definitions are necessary to be introduced w.r.t. time semantics [17]:

- Σ defines an alphabet set over which the actions are triggered in Timed Büchi Automata (Definition 2.3).
- An *atomic proposition* ρ, represents a Boolean variable that can be True ($\top$) or False ($\bot$).
- An *infinite word* over a set $\Sigma = 2^{\mathcal{AP}}$, with $\mathcal{AP}$ set of atomic propositions, is an infinite sequence $w = w_0 w_1 \dots$, where w_i is the i-th element of the sequence and $w_i \in \Sigma, \forall i \geq 0$.
- A *time sequence* is an infinite sequence of time, denoted by $\tau = \tau_0 \tau_1 \dots$, where $\tau_i \in \mathbb{R}_+$ and having the next properties: *monotonicity:* $\tau_i < \tau_{i+1}, \forall i \geq 0$; *progress:* $\forall t \in \mathbb{R}_+, \exists i \geq 1$, such that $\tau_i > t$.
- A *timed word* over a set Σ is defined as an infinite sequence $w^t = (w_0, \tau_0)(w_1, \tau_1) \dots$, where $w_0 w_1 \dots$ is an infinite word and $\tau_0 \tau_1 \dots$ is a time sequence.

Remark 1. From now on, we consider the time in set $\mathbb{Q}_+$. Moreover, in this work, the alphabet set Σ is equivalent with the powerset $2^{\mathcal{Y}}$, where set of atomic proposition $\mathcal{AP}$ is represented by set of regions of interest $\mathcal{Y}$.

***Definition** 2.1:* The syntax of *Metric Interval Temporal Logic (MITL)* specifications over the set of atomic propositions $\mathcal{Y}$ is defined as follows [18]:

$$\varphi := y_k \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc_I \varphi \mid \Diamond_I \varphi \mid \Box_I \varphi \mid \varphi_1 \mathcal{U}_I \varphi_2, \quad (1)$$

where $y_k \in \mathcal{Y}$, I is a non-empty time interval denoted as $[i_1, i_2]$ or (i_1, i_2), with end points $i_1 < i_2$, $i_1 \in \mathbb{N}$ and $i_2 \in \mathbb{N} \cup \{\infty\}$. Further on, the time interval $[0, i_2]$, respectively $[0, i_2)$ is denoted by $\leq i_2$, respectively $< i_2$. In addition to the Boolean operators such as *negation* $\neg$, *and* $\wedge$, the temporal operators accepted by MITL are as follows: *next* $\bigcirc$, *eventually* $\Diamond$, *always* $\Box$, and *until* $\mathcal{U}$.

The tuple (w^t, i) defines the MITL formula $\varphi, i \geq 0$ over the set $\mathcal{Y}$ with the timed word $w^t = (w_0, \tau_0)(w_1, \tau_1) \dots$ is recursively satisfied as follows:

$(w^t, i) \models y_k \Leftrightarrow y_k \in w_i$
$(w^t, i) \models \neg\varphi \Leftrightarrow (w^t, i) \not\models \varphi$
$(w^t, i) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (w^t, i) \models \varphi_1$ and $(w^t, i) \models \varphi_1$
$(w^t, i) \models \bigcirc_I \varphi \Leftrightarrow (w^t, i+1) \models \varphi$ and $\tau_{i+1} - \tau_i \in I$
$(w^t, i) \models \Diamond_I \varphi \Leftrightarrow \exists j \geq i, \text{s.t.} (w^t, j) \models \varphi, \tau_j - \tau_i \in I$
$(w^t, i) \models \Box_I \varphi \Leftrightarrow \forall j \geq i, \tau_j - \tau_i \in I \Rightarrow (w^t, j) \models \varphi$
$(w^t, i) \models \varphi_1 \mathcal{U}_I \varphi_2 \Leftrightarrow \exists j \geq i, \text{s.t.} (w^t, j) \models \varphi_2,$
$\tau_j - \tau_i \in I$ and $(w^t, k) \models \varphi_1, \forall i \leq k < j$

***Example** 2.2:* Both semantics (continuous and pointwise) are suitable for defining an MITL formula φ [19]. Some examples of MITL specifications for one agent can be written as follows, considering the time unit being expressed as seconds or minutes, among others:

- "Visit region y_1 at least a moment from time interval [0,10] and always avoid region y_4 in the time interval $[11, \infty)$": $\varphi = \Diamond_{[0,10]} y_1 \wedge \Box_{[11,\infty)} \neg y_4$;
- "At some point in the time interval [3,6], region y_2 should be visited, until then y_3 will be visited": $\varphi = y_3 \mathcal{U}_{[3,6]} y_2$;

The MITL formula φ over the set of atomic propositions $\mathcal{Y}$ can be translated into a Timed Büchi Automata (TBA) over the alphabet $2^{\mathcal{Y}}$ [18], [20], [21]. This work uses the translation from [18].

Let us define a finite set of clocks $X = \{x_1, x_2, \dots, x_{|X|}\}$. The set of *clock constraints* $\Phi(X)$ is characterized by the grammar:

$$\phi := \top \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid x \bowtie \psi, \quad (2)$$

where $x \in X$ represents a clock, $\psi \in \mathbb{Q}_+$ is a clock constraint and $\bowtie \in \{<, >, \leq, \geq, =\}$. A clock *valuation* (or interpretation) is a function $\nu : X \to \mathbb{Q}_+$ which assigns a value to each clock and $\nu + \delta$ maps every clock x to the value $\nu(x) + \delta$, where $\delta \in \mathbb{N}_+$. The satisfaction of the clock constraint ϕ by the valuation ν is denoted with $\nu \models \phi$.

The Timed Büchi Automata model follows the definition of [17] while using the notations from [14] with the mention that in this work we do not differentiate between the set of final locations and the set of repeated locations.

***Definition** 2.3:* A *Timed Büchi Automata (TBA)* is defined as a tuple $\mathcal{A} = \langle Q, q_0, X, \Phi(X), \Sigma, E, Inv, F \rangle$, where Q represents a finite set of locations; $q_0 \in Q$ is the initial location; X is the finite set of clocks; $\Phi(X)$ represents the clock constraints; $Inv : Q \to \Phi(X)$ is the invariant; Σ is the alphabet set, $E \subseteq Q \times \Phi(X) \times \Sigma \times 2^X \times Q$ is the set of edges between locations, where an edge from location q to q' is denoted as $e = (q, \gamma, \lambda, R, q')$ with guard $\gamma \in \Phi(X)$, label $\lambda \in \Sigma$ and $R \subseteq X$ the reset set; $F \subseteq Q$ is the set of accepting (final) locations.

A state of $\mathcal{A}$ is characterized as the pair (q, ν) with $q \in Q$ and $\nu \models Inv(q)$ (ν satisfies the clock constraint in the invariant of q). The initial states of $\mathcal{A}$ is defined by the pair $(q_0, \mathbf{0})$ with q_0 the initial location and $\mathbf{0}$ is the valuation which maps every clock of the automata to 0. The automata has two types of transitions. The first one is a *discrete transition* $(q, \nu) \xrightarrow{e} (q', \nu')$ with the edge $e = (q, \gamma, \lambda, R, q')$, states $(q, \nu), (q', \nu')$, $\lambda \in \Sigma$ and it exists iff $\nu \models \gamma, \nu' \models Inv(q')$. R is the *reset set* that resets the clocks to 0, i.e., $\nu'_i = 0, \forall c_i \in R$ and $\nu'_i = \nu_i, \forall c_i \notin R$. The second transition is a *time transition* $(q, \nu) \xrightarrow{\delta} (q', \nu')$ for a given $\delta \in \mathbb{Q}_+$, where δ is a summed component-wise, iff $q = q', \nu' = \nu + \delta, \nu' \models Inv(q)$. As a result of the time-additivity property [22], there exist q', ν' such that $(q, \nu) \xrightarrow{\delta} (q', \nu')$ and $(q', \nu') \xrightarrow{\mathrm{e}} (q'', \nu'')$, with $q = q''$. The previous sequence is denoted by $(q, \nu) \xrightarrow{\delta} \xrightarrow{\mathrm{e}} (q'', \nu'')$.

An *infinite accepted run* in automata $\mathcal{A}$ represents a path from an initial to a final state $q_i \in F, i \geq 1$ that is visited infinitely often. Such run is defined as an infinite sequence of time and discrete transitions: $(q_0, \nu_0) \xrightarrow{\delta_0} (q'_0, \nu'_0) \xrightarrow{e_0} (q_1, \nu_1) \xrightarrow{\delta_1} (q'_1, \nu'_1) \ldots$, with the initial state (q_0, ν_0).

***Example** 2.4:* Fig. 2 illustrates an example of a TBA $\mathcal{A}$ which models the MITL specification $\varphi = \Diamond_{\leq c} y_1$, with $y_1 \in \Sigma$, $c \in \mathbb{Q}_+$ representing the clock constraint for the clock x. The reset of the clock is illustrated with a green color. For this example, the reset of the clock is not mandatory, as it is related with one instance of the temporal operator *eventually*. The reset of the clock becomes trivial in nested MITL specifications [23].

The initial location q_0 is indicated by the initial input arc, the final location q_1 is visualized by the double edge and q_2 represents the sink location (error). When the MITL formula cannot be satisfied, the automata reaches the error location. The set of edges are as follows: $E = \{(q_0, \neg y_1, x \leq c, \emptyset, q_0), (q_0, y_1, x \leq c, x := 0, q_1), (q_0, \top, x \geq c, \emptyset, q_2), (q_1, \top, T, x := 0, q_1), (q_2, \top, T, x := 0, q_2)\}$, where $T \in \mathbb{Q}_+$ represents any time and $\top$ is *True*, meaning that the edge can be triggered by any symbol in the alphabet Σ.

***Definition** 2.5:* A *Time Petri net (TPN)* model [24], [25] is defined as a tuple $\mathcal{TPN} = \langle \mathcal{N}, \boldsymbol{m}_0, I \rangle$, where:

- $\mathcal{N} = \langle P, T, Pre, Post \rangle$ represents a Petri net (PN) model, with P - finite set of places; T - finite set of transitions; $Pre : P \times T \to \mathbb{N}$ - input function defining the arc weights from places to transitions, e.g., $Pre(p, t) = b$ if place $p \in P$ is connected with $t \in T$ with an arc of weight $b \in \mathbb{N}_{>0}$, otherwise $Pre(p, t) = 0$; $Post : P \times T \to \mathbb{N}$ - output function defining the arc weights from transitions to places, e.g., $Post(p, t) = b$ if transition $t \in T$ is connected with $p \in P$ with an arc of weight b, otherwise $Post(p, t) = 0$.
- $m : P \to \mathbb{N}$ represents the marking function, where $m(p)$ is the number of tokens in place p, $\forall p \in P$. The initial marking is denoted m_0.
- $I : T \to [\mathbb{Q}_+ \to \mathbb{Q}_+ \cup \{\infty\}]$ is the function that maps a static interval to each transition. The time interval for one transition is represented by a tuple $I(t) = [\alpha, \beta], \forall t \in T$, with $0 \leq \alpha < \infty$ denoting the earliest firing time, $0 \leq \beta \leq \infty$ denoting the latest firing time and $\alpha \leq \beta$ if $\beta \neq \infty$ or $\alpha < \beta$ if $\beta = \infty$.

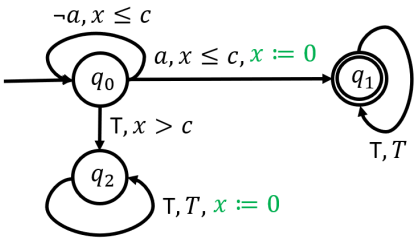

Fig. 2. Example of a Timed Büchi Automata accepting satisfying runs for the MITL formula $\varphi = \Diamond_{\leq c} y_1$

We introduce a labeling function $\Lambda : T \to \Sigma'_\epsilon$ which assigns to each transition $t \in T$ a label with values from alphabet set $\Sigma'_\epsilon = T \cup \{\epsilon\}$. The only repeated label is ϵ while the rest of them are unique for every $t \in T$. The objective of this labeling function is further described in the next section.

***Example** 2.6:* Let us consider the Time Petri net model from Fig. 3 with the initial marking $\boldsymbol{m}_0[p_2^E] = 1$. The token can be consumed and produced in place p_1^E when the time is $\geq \delta_{min}^{2,1}$ but no later than $\delta_{max}^{2,1}$. The figure captures the transition's labels from alphabet Σ'_ϵ, as previously described.

III. Problem Statement

Let us consider a multi-agent system with $r = |\mathcal{R}|$ agents evolving in environment E. The goal of this system is to automatically compute paths that satisfy tasks under *space and time requirements*. One can refer to space requirements as imposed locations in the environment that should be visited (sequentially or simultaneously), while time requirements express deadlines by which those locations should be reached. Thus, the team receives a global MITL (Metric Interval Temporal Logic) φ which specifies the visit and/or avoidance of several locations (ROIs), while a Time Petri net model captures the movement of the agents in the environment.

The current work ensures time and space constraints for multi-agent systems, under the assumption that the robots

include local controllers which can provide suitable inputs to follow the desired actuation. If one location should be visited by multiple agents with similar capabilities, then one MITL mission is given for each of these sub-groups of agents.

Remark 2. The solution to this problem consists in defining one *Composed Time Petri net* model for each MITL formula, as visualized in Fig. 1 and further explained in the next section. A similar idea was proposed in a previous work [26], which couples a Petri net (PN) model with an LTL formula. The results of the current work bring two main contributions when compared with [26]: inclusion of time constraints for the multi-agent system and providing planning strategy for scenarios that require multiple agents to be present in the same region of interest. While in [26] structural approaches based on mathematical programming are used, here the path planning is based on model checking.

IV. Proposed solution

As observed in Fig. 1, the output of the proposed framework is represented by a TPN model denoted *Composed Time Petri net (TPN^C)*. TPN^C incorporates the movement of the robots and the MITL global specification. The superscript "C" is added to each component of TPN^C. One advantage of the Time Petri net representation lies in updating its structure of it to incorporate constraints, e.g., collision avoidance, for example by imposing that the capacity of each place is one [27].

This section describes the necessary steps to achieve the desired model, as depicted in Fig. 1. Let us recall the notations for the sets of agents $\mathcal{R}$, respectively of regions of interest $\mathcal{Y}$.

Step 1.1 This step translates the continuous working space into a discrete representation, which facilitates an easy manipulation of the environment E w.r.t. the motion of the agents. The mapping method used in the current work is based on a cell decomposition technique, e.g., [4], [28]. As a result, the environment is partitioned into regions denoted cells, which can be further labeled as free or critical, i.e., a critical cell belongs to one regions from $\mathcal{Y}$. The partition procedure determine the number of cells which corresponds to a region of interest $y_i \in \mathcal{Y}$. This notion is further taken into account in the building of the Time Petri net model.

The motion of the robots from one cell to an adjacent one is characterized by (i) the deployment of the robots in space (their position in the cell) and (ii) time constraints. The latter constraints are defined as follows: $[\delta_{min}, \delta_{max}]$, where δ_{min} is the minimum motion time, respectively δ_{max} is the maximum motion time to reach the adjacent cell, considering the maximum, respectively the minimum speed of the robot. Note that this notation does not capture the waiting time in the current cell. If the agent has an unknown or unlimited time to wait in its current cell before moving to another adjacent cell, then the upper bound becomes $\delta_{max} = \infty$. The crossing from cell to cell is provided by a low level control strategy that can be user-defined.

Step 1.2 After the environment E is partitioned, the motion of the robots is handled by one Time Petri net (TPN) model, denoted with $\mathcal{TPN}^E$. The superscript "E" is added to each component of $\mathcal{TPN}^E$. An algorithm that builds a PN model based on the set of cells returned by the Step 1.1 is described in the first algorithm from [29]. This method assigns a place to each cell, the adjacency relation between cells is captured by the transitions and each token represents an agent. The desired Time Petri net model is obtained by the addition of motion time constraints from one cell to an adjacent one, as previously described.

The $\mathcal{TPN}^E$ model is tailored according to the space constraints of the MITL mission, which require visiting and/or avoiding several ROIs from $\mathcal{Y}$. Let us consider an observation map function denoted with $h : P^E \to \mathcal{Y} \cup \{\emptyset\}$, which assigns to each place in $\mathcal{TPN}^E$ an atomic proposition from $\mathcal{Y}$. With $\emptyset$ is denoted the free space in the environment. If at least one robot (token) is present in place p_i^E, then the region of interest $h(p_i^E)$ is visited.

Next, we reduce the number of places on TPN^E, by iteratively merging adjacent places p_i^E and p_j^E sharing the same atomic proposition over the set $\mathcal{Y}$, with $h(p_i^E) = h(p_j^E)$. In other words, only the places modeling the free space $h(p_i^E) = \epsilon$ are not merge, to maintain the time information as described previously. Due to this procedure, the output transitions from the merged places will have the upper bound time equal with ∞. The lower bound is updated with the minimum time constraints from different observations, as it is connected to the physical constraints of the robot to move from one cell (place) to another. The ∞ value expresses the fact that the agent can stay unlimited time in the respective region of interest.

Example 4.1: Fig. 3 portrays an illustrative example for the first two sub-steps of the proposed workflow. The left side of the figure depicts an environment E partitioned into 4 cells, from which 2 of them belong to the same region of interest y_1 (color blue). The associated $\mathcal{TPN}^E$ is shown on the right side, being already tailored as previously described: both cells p_3 and p_4, with $h(p_3) = h(p_4) = y_1$, are modeled by a single place $p_{3,4}^E$. The token in p_2^E captures the presence of one agent in cell p_2.

The second phase of the proposed framework designs the mission of the multi-agent system given as an MITL specification.

Step 2.1 As stated in [18], any MITL formula φ given in the normal form (time interval $\leq c$ or $< c$, with $c \in \mathbb{Q}_+$), can be expressed as a Timed Büchi automata $\mathcal{A}$ (Fig. 2). In addition, the authors of [18] proved that any MITL specification can be translated into the normal form, by applying a set of four transformations described therein.

Step 2.2 In [14], the authors proposed a translation from a Timed Büchi Automata with invariant clock constraints defined as $x \leq c$ or $x < c$, where $\Phi(X) = \{x\}, c \in \mathbb{Q}_+$ to a Time Petri net model, while maintaining the time bisimilarity property between the two representations. The main idea relies on providing Time Petri net topologies for clock constraints and resets, accounted to each edge and each invariant. Each location of the TBA is represented as a place and each edge is modeled by a time transition, except for the

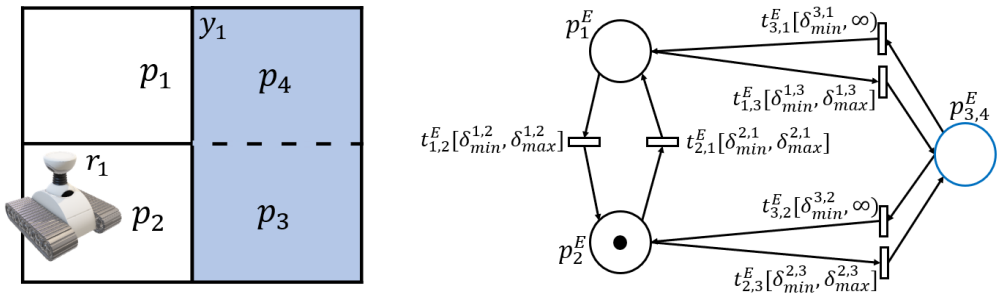

Fig. 3. Example of Time Petri net model $\mathcal{TPN}^E$ (right side) for a partitioned environment E (left side)

self-loops arcs defined as $e = (q, \top, T, \emptyset, q)$, where $q \in Q$. In other words, no transition is added for self-loop that has no clock or logical constraints. Furthermore, the transitions are connected with the TPN topologies assigned to clock constraints (gray blocks) and clock resets (green blocks), as observed in Fig. 4. The full algorithm is described in [14]. This TPN model corresponds to the TBA model indicated in Fig. 2, where the presence of a token in places $p_{fi}^{\varphi}, i = 1, 2$ represents the satisfaction of time requirements.

The corresponding TPN structure for an MITL specification is denoted with $\mathcal{TPN}^{\varphi}$, adding the superscript $\cdot^{\varphi}$ to all model's components. The set of transitions $T^{\varphi} = T_c^{\varphi} \cup T_g^{\varphi}$ with $T_c^{\varphi} \cap T_g^{\varphi} = \emptyset$ is composed from set T_c^{φ} used in the topologies of the clock constraints and the set T_g^{φ} representing the transitions between the places modeling the location $q \in Q$ from automata $\mathcal{A}$. Let us recall the labelling function for transitions as $\Lambda : T \rightarrow \Sigma'_{\epsilon}$. All transitions $t_{c_i} \in T_c^{\varphi}, i \geq 1$, have the label $\Lambda(t_{c_i}) = \epsilon$ to simultaneously validate all clocks connected with transitions $t_{g_i} \in T_g^{\varphi}$, according to [14].

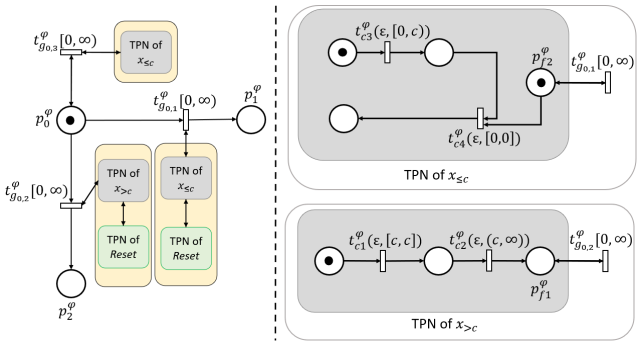

Fig. 4. Example for translation of a TBA model to a TPN model (left side) for the MITL $\varphi = \Diamond_{\leq c} a$, including clock topologies (right side)

Moreover, we introduce the function $\Pi : T_g^{\varphi} \rightarrow 2^{\mathcal{Y}}$ to memorize the symbols $\lambda \in \Sigma$ assigned to the edges of $\mathcal{A}$. The values of function Π are given as a DNF formula (Disjunctive Normal Form) over the set $2^{\mathcal{Y}}$. In Fig. 4, transition $t_{g_{0,1}}^{\varphi}$ models the edge $e = (q_0, y_1, x \leq c, \emptyset, q_1)$, with $q_0, q_1 \in Q, y_1 \in \Sigma, X = \{x\}$, thus $\Pi(t_{g_{0,1}}) = a$.

Step 3. To finalize the proposed model *Composed Timed Petri net* denoted as $\mathcal{TPN}^C$, this step integrates the outputs returned by steps 1.2 and 2.2, together with new inputs modeling the atomic propositions over the set $\mathcal{Y}$. Let us consider the sets of places $P^O = \{p_1^O, p_2^O, \ldots, p_{|\mathcal{Y}|}^O\}$, respectively $P^{\neg O} = \{p_1^{\neg O}, p_2^{\neg O}, \ldots, p_{|\mathcal{Y}|}^{\neg O}\}$ modeling the true, respectively the false value of atomic propositions. The purpose of these places is to provide a snapshot of the movement of the robots with respect to the regions of interest that are reached. The sets $P^O, P^{\neg O}$ are part of $\mathcal{TPN}^C$ and act as an intermediate layer between the two TPN models $\mathcal{TPN}^E, \mathcal{TPN}^{\varphi}$.

The *Composed Time Petri net* model is build based on the next inputs: the TPN assigned to the robotic system $\mathcal{TPN}^E$with the labeling function of the places h, the TPN assigned to the MITL formula $\mathcal{TPN}^{\varphi}$ with the labeling function Π, the aforementioned sets $P^O, P^{\neg O}$, the number of robots required to satisfy the MITL specification φ. and the set of regions of interest $\mathcal{Y}$.

***Example** 4.2:* Let us consider a team of 5 agents and the following MITL formula $\varphi = \Diamond_{\leq 5}(y_1 \wedge y_2)$ interpreted as visiting simultaneously both y_1 and y_2 in less than 5 time units. In addition, let us require that the regions of interest should be reached by multiple robots, as described in Remark 3. If one region, e.g., y_1 should be reached by 3 agents, while region y_2 should be reached by 2 agents, then the equivalent interpretation of the MITL specification with regards to the $\mathcal{TPN}^C$ model is the following: $\varphi = \Diamond_{\leq 5}((\boldsymbol{m}^O[p_1^O] == 3) \wedge (\boldsymbol{m}^O[p_2^O] == 2))$. Hence, a minimum marking is imposed for the places p_1^O and p_2^O modeling the truth value of the atomic proposition y_1 and y_2.

Remark 3. The current work aims to provide a solution for scenarios in which MITL specifications requires a minimum number of agents such that the atomic propositions have the truth value. This concept is mentioned in literature under the term "census", which imposes multipple agents to ensure the same task (in our case, a task is defined as reaching a region of interest). One example of *census* concept is presented in [12] using STL specifications. In the current work, the MITL formula is handled by an equivalent Time Petri net model which contributes to an easier understanding of the *census* concept represented by imposed markings.

As mentioned previously, a similar idea of building a new model is proposed in the previous work [26], which couples the Petri net (PN) model of the environment with a PN

model assigned to an LTL formula, rather than an MITL one. Therein, he third algorithm describes the procedure of achieving the new Petri net model, capturing the rules of connecting the intermediate layer $P^O, P^{\neg O}$ with the two PN models (for environment and LTL specification). The same basic idea are followed in the current work. But, for being able to create the *Composed Time Petri net*, the following contributions are added to the referred method:

- Adding time constraints for the $\mathcal{TPN}^E$ based on the motion of the multi-agent system, as a result of sub-steps 1.1 and 1.2;
- Representing the assigned TBA model of an MITL formula φ as $\mathcal{TPN}^\varphi$ model, according to [14]. Let us recall that the translation between a TBA model into a TPN model, maintaining the timed bisimilarity property, can be achieved only if certain conditions are fulfilled, as stated in [14] and previously described in this paper along sub-steps 2.1 and 2.2;
- Updating the weight of the arcs from places in set P^O to transitions in T^φ, thus imposing a minimum number of agents required for the true value of the atomic propositions $\mathcal{Y}$ (Example 4.2).

***Example** 4.3:* Fig. 5 illustrates a part of the *Composed Time Petri net* as a result of the general workflow proposed in this paper. The left side of the figure displays a partial Time Petri net which models the motion of the agents in the environment, with $h(p_2^E) = \emptyset$ and $h(p_{3,4}^E) = y_1$, where y_1 is an atomic proposition in $\mathcal{Y}$. The right side of the figure portrays the Time Petri net associated to the MITL formula $\varphi = \Diamond_{\leq c} y_1$. The intermediate layer composed from sets $P^O, P^{\neg O}$ is consistent with the position of the agent in the environment: one agent is in the free space, while no agent is present in the region of interest $y_1 \in \mathcal{Y}$. The linking arcs between the models $\mathcal{TPN}^E, \mathcal{TPN}^\varphi$ and the intermediate layer follow the previously described procedure. In other words, the atomic proposition a has the true value only if a minimum number of ω_i tokens are present in p_1^O, and has the false value if $|\mathcal{R}|$ tokens are in $p_1^{\neg O}$.

The initial marking for each sub-structure of the *Composed Time Petri net* has various implications, as follows: $\boldsymbol{m}_0^E$ of $\mathcal{TPN}^E$ expresses the initial position of the team of agents in the environment, while the tokens model the agents; the marking of sets $P^O, P^{\neg O}$ captures a snapshot of the team w.r.t. their positions regarding the atomic propositions $\mathcal{Y}$ (in other words, this marking defines the number of agents in each region of interest); $\boldsymbol{m}_0^\varphi$ of $\mathcal{TPN}^\varphi$ characterizes the initialization of automata $\mathcal{A}$ assigned to the MITL formula φ, together with the initial state of the clocks.

V. Path Planning and Simulation Results

Within this work, the motion planning strategy for MITL specifications relies on simulations, by solving a reachability problem, known as model-checking approaches. Once the *Composed Petri net model* is computed, the model is implemented in the tool ROMEO [16] which provides a model-checking solution by evaluating if a marking can be reached. If true, the sequence of transitions is returned. The marking to be reached corresponds to the place modeling the final state of the TBA model of the MITL formula. Therefore, the solution reaching the desired marking has the same connotation as satisfying an MITL formula (as shown in Ex. 4.2). The paths of the multi-agent system are conveyed into motions according to the trace run and controller for steering robots between partition cells. One advantage of the tool ROMEO for modeling and analyzing Time Petri net models is represented by the on-the-fly model-checking procedure, which does not create the entire state-class graph when searching for a trace run.

Let us consider the case study tackled in [30], in which a gantry-robot system of R robotic arms should install reinforcement bars (rebars) to create a concrete structure (cage). This scenario envisions two sub-groups of robotic arms, based on their capabilities given by the gripper: the first sub-teams (denoted with r_{pp}) should pick-up the rebars, move them in the construction area, and hold the rebars while they are connected by the second subgroup of robots r_c, with $r_c \cap r_{pp} = \emptyset$, $r_c \cup r_{pp} \subseteq r$. Although [30] proposes a solution for this scenario, the authors mention the lack of flexibility in their approach. Thus, the current work proposes a fixed topology model, w.r.t. the number of agents in the team, e.g., different type of rebars influences the number of robots assigned to each sub-group. In addition, time constraints are integrated into the path-planning strategy. Our work relies on the assumptions introduced in [30], regarding the known data apriori building the cage.

The flexible planning strategy is based on two regions of interest (ROIs) in the configuration space of the robots.

- y_1 - defines the common area designated to pick-up the rebars by the set r_{pp} of robots. It is assumed that the gripping points are assigned to the robots while ensuring collision-free movement.
- y_2 - depicts the construction area in which the rebars are placed by the same set of robots r_{pp} and linked together by the set of robots r_c.

Each sub-group of agents receives one MITL formula as mission. These formulas are iterated for each rebar, until the entire structure is build. The time constraints are expressed as $[0, c_i], \forall I_i, c_i \in \mathbb{N}_+$. Both MITL specifications are related to each other through time condition $c_3 > c_4$ and $c_3 \leq c_4 + c_5$ (the robots r_{pp} should hold still the rebars for a time $c_5 = |I_3|$, while the robots r_c link the rebars together).

(i) Formula (3) is assigned to the set r_{pp} and it specifies the sequence of actions to pick-up a rebar from y_1 for time I_1, place it in y_2 in the time I_2 computed w.r.t. the pick-up time, and hold the rebar for I_3 time.

$$\varphi_{r_{pp}} = \Diamond_{I_1} y_1 \wedge (y_1 \rightarrow \Diamond_{I_2} \Box_{I_3} y_2) \tag{3}$$

(ii) Formula (4) requires the robots r_c to reach y_2 w.r.t. I_4, while staying there for I_5 time to link the rebars together.

$$\varphi_{r_c} = \Diamond_{I_4} \Box_{I_5} y_2 \tag{4}$$

The main idea is to compute one *Composed Time Petri net* model for each MITL formula while checking if the

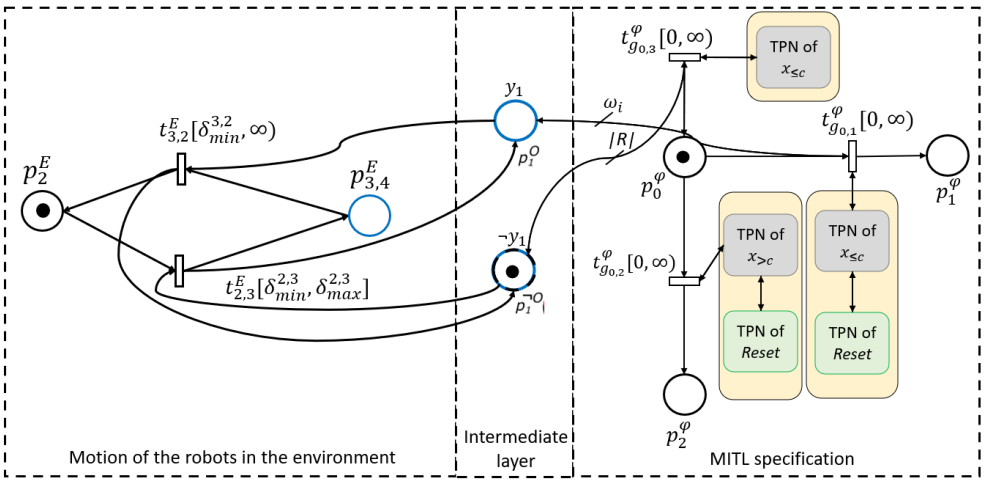

Fig. 5. Part of *Composed Time Petri net*, based on the atomic proposition a used in the MITL formula $\varphi = \Diamond_{\leq c} y_1$

specification can be fulfilled based on a model-checking approach for the $\mathcal{TPN}^C$ representation.

Algorithm 1: Movement of robots

Input : r, $\varphi_{r_{pp}}$, φ_{r_c}, $structure_{fixed}$, E
Output: Strategic movement of the robots

1 All robots r are placed in the home position;
2 $structure = \emptyset$;
3 Build $\mathcal{TPN}^{\varphi_{r_{pp}}}$ based on specification $\varphi_{r_{pp}}$;
4 Build $\mathcal{TPN}^{\varphi_{r_c}}$ based on specification φ_{r_c};
5 **while** $structure \neq structure_{fixed}$ **do**
6 Arrival of a rebar in the pick-up place y_1;
7 Analyze the type of the rebar to compute the number of robots in sub-groups r_{pp} and r_c;
8 Compute and execute paths for r_{pp} and r_c;
9 Update $structure$;
10 Return r_c and r_{pp} to their home position;
11 **end**

Alg. 1 describes the procedure followed in this case study. Initially, all robots are placed in a home position. With every arrival of a rebar in the pick-up area y_1, the number of robots assigned for each subgroups is computed, depending on the type of rebar, as in [30]. One MITL formula is assigned to each sub-group r_{pp} and r_c, for which an individual $\mathcal{TPN}^C$ model is computed. These formulas are ensured when robots r_{pp} satisfy $\varphi_{r_{pp}}$ and r_c satisfy φ_{r_c}. This is equivalent with reaching final markings in the TPN models, with y_i true only if the number of tokens are equal with $|r_{pp}|$, respectively $|r_c|$. Once the sequences of transitions in the TPN models (robot paths) are returned, the robots return to their home position and the structure is updated. The process from lines 6- 13 is repeated until the $structure$ is equal with the intended structure $structure_{fixed}$ given as input.

Let us consider the environment partitioned into cells, modeled by $\mathcal{TPN}^E$ with 20 places. The table I includes the numerical evaluation obtained for MITL specifications (3),(4) considering the size of each *Composed Time Petri net* model $\mathcal{TPN}^C$ denoted with $\mathcal{TPN}^{\varphi_{r_{pp}}}$, respectively $\mathcal{TPN}^{\varphi_{r_c}}$. The simulation results are obtained on a computer with i7 - 8^{th} gen. CPU @ 2.20GHz and 8GB RAM.

Remark 4. As mentioned in [14], both models TBA and TPN are timed bisimilar. If each agent of the robotic system is modeled as a TBA, e.g., [6], then the total number of locations for the entire team increases exponentially as a result of the product automata, incorporating also the number of locations for the TBA assigned to an MITL formula φ. On the other hand, the size of a *Composed Time Petri net* model w.r.t. the number of places P^C depends on the cardinality of sets $P^E, P^O, P^{\neg O}$ and P^{φ}. The total number of places P^C is not influenced by the number of agents for which the $\mathcal{TPN}^C$ model was computed.

Discussion The proposed TPN structure provides clear coordination between the local time constraints related to the motion of the robots (modeled by $\mathcal{TPN}^E$) and the global time constraints given by the MITL mission (modeled by $\mathcal{TPN}^{\varphi}$). In addition, the *Composed Time Petri net* model is suitable to capture the census concept as mentioned in [12], which requires a task to be fulfilled by multiple agents.

The model-checking approach applied to the present case study supports the effectiveness of the proposed framework, providing a fixed topology model w.r.t. the number of agents which should fulfill an MITL formula. Thus, the current work based on Time Petri nets provides a good baseline to develop scalable models of multi-agent systems. For now, one downside of the current approach prevails outputs for a large number of agents. Thus, although the model-checking of the ROMEO tool does not explore the entire reachability graph, the conducted simulations with more than 3 agents could not yield a solution. Therefore, various approaches can be further exploited, such as structural methods, e.g., mathematical programming, or methods that partially investigate the state space of the model, e.g., distributed methods.

TABLE I
NUMERICAL EVALUATION

MITL Formula	Size of $\mathcal{TPN}^C$	No. of agents	Model-checking run time [sec]
$\varphi_{r_{pp}} = \Diamond_{I_1} y_1 \wedge (y_1 \rightarrow \Diamond_{I_2} \Box_{I_3} y_2)$	$\lvert P^C \rvert = 50, \lvert T^C \rvert = 65$	2 3	2.2 81.3
$\varphi_{r_c} = \Diamond_{I_4} \Box_{I_5} y_2$	$\lvert P^C \rvert = 42, \lvert T^C \rvert = 58$	2 3	1.8 49

VI. CONCLUSION

This paper presents a framework for high-level planning strategy, based on a newly defined *Composed Time Petri net* model coupling space and time requirements included in an MITL formula given for a multi-agent system. The main steps to achieve the proposed model include the representation of the workspace, respectively an MITL formula, into TPN models, which are furthered coupled with sets of places modeling the number of agents required for the truth value of an atomic proposition (region of interes). The benefits of the proposed Time Petri net model include a fixed topology w.r.t. the number of agents evolving in a known environment.

Future work envisions studying of path planning procedures with time constraints including local or collaborative tasks for the robots, depending on the type of rebar. One work handling such tasks without time constraints is [5], in which the authors differentiate between various tasks based on LTL formalism.

REFERENCES

[1] A. Das, Y. Kasemsinsup, and S. Weiland, "Optimal trajectory tracking control for automated guided vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 303–308, 2017.

[2] E. Restrepo, J. Matouš, and K. Y. Pettersen, "Tracking-in-formation of multiple autonomous marine vehicles under proximity and collision-avoidance constraints," in *2022 European Control Conference (ECC)*. IEEE, 2022, pp. 930–937.

[3] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal logic task planning and intermittent connectivity control of mobile robot networks," *IEEE Trans. on Autom. Control*, vol. 64, no. 10, pp. 4105–4120, 2019.

[4] C. Mahulea, M. Kloetzer, and R. González, *Path planning of cooperative mobile robots using discrete event models*. John Wiley & Sons, 2020.

[5] M. Guo and D. V. Dimarogonas, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Trans. on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, 2016.

[6] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "Cooperative planning for coupled multi-agent systems under timed temporal specifications," in *2017 American Control Conference (ACC)*, 2017, pp. 1847–1852.

[7] W. Wang, G. F. Schuppe, and J. Tumova, "Decentralized multi-agent coordination under MITL specifications and communication constraints," 2022.

[8] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[9] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[10] I. Hustiu, M. Kloetzer, and C. Mahulea, "Distributed path planning of mobile robots with LTL specifications," in *24th Int. Conf. on System Theory, Control and Computing (ICSTCC)*, 2020, pp. 60–65.

[11] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Integrated motion planning and control under metric interval temporal logic specifications," in *18th European Control Conference (ECC)*, 2019, pp. 2042–2049.

[12] Z. Xu and A. A. Julius, "Census signal temporal logic inference for multiagent group behavior analysis," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 264–277, 2016.

[13] Z. He, Y. Dong, G. Ren, C. Gu, and Z. Li, "Path planning for automated guided vehicle systems with time constraints using timed Petri nets," *Measurement and Control*, vol. 53, no. 9-10, pp. 2030–2040, 2020.

[14] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux, "Comparison of the expressiveness of timed automata and time Petri nets," in *International conference on formal modeling and analysis of timed systems*. Springer, 2005, pp. 211–225.

[15] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints for Petri nets with uncontrollable transitions," in *IEEE Int. Conf. on Systems, man, and cybernetics*, 1992, pp. 947–949.

[16] G. Gardey, D. Lime, M. Magnin, and O. Roux, "Romeo: A tool for analyzing time Petri nets," in *International Conference on Computer Aided Verification*. Springer, 2005, pp. 418–423.

[17] R. Alur and D. Dill, "The theory of timed automata," in *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*. Springer, 1991, pp. 45–73.

[18] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[19] D. D'Souza and P. Prabhakar, "On the expressiveness of MTL in the pointwise and continuous semantics," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[20] O. Maler, D. Nickovic, and A. Pnueli, "From MITL to timed automata," in *International conference on formal modeling and analysis of timed systems*. Springer, 2006, pp. 274–289.

[21] D. Ničković and N. Piterman, "From MTL to deterministic timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 152–167.

[22] R. Alur, "Timed automata," in *International Conference on Computer Aided Verification*. Springer, 1999, pp. 8–22.

[23] G. E. Fainekos and G. J. Pappas, "Robust sampling for MITL specifications," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2007, pp. 147–162.

[24] P. Merlin and D. Farber, "Recoverability of communication protocols-implications of a theoretical study," *IEEE transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[25] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE transactions on software engineering*, vol. 17, no. 3, p. 259, 1991.

[26] S. Hustiu, C. Mahulea, M. Kloetzer, and J.-J. Lesage, "On multi-robot path planning based on Petri net models and LTL specifications," 2022. [Online]. Available: https://arxiv.org/abs/2211.04230

[27] M. Kloetzer, C. Mahulea, and J.-M. Colom, "Petri net approach for deadlock prevention in robot planning," in *IEEE 18th Conf. on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–4.

[28] M. Lupascu, S. Hustiu, A. Burlacu, and M. Kloetzer, "Path planning for autonomous drones using 3d rectangular cuboid decomposition," in *23rd Int. Conf. on System Theory, Control and Computing (ICSTCC)*, 2019, pp. 119–124.

[29] C. Mahulea and M. Kloetzer, "Robot planning based on boolean specifications using Petri net models," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2218–2225, 2017.

[30] M. Momeni, J. Relefors, A. Khatry, L. Pettersson, A. V. Papadopoulos, and T. Nolte, "Automated fabrication of reinforcement cages using a robotized production cell," *Automation in Construction*, vol. 133, p. 103990, 2022.