A Formal Framework for Multi-Agent Task Planning

Anatoli A. Tziola and Savvas G. Loizou, *Member, IEEE*

***Abstract*—This paper presents a solution to the task planning problem for multi-agent systems. A formal framework is developed based on the Nondeterministic Finite Automata with ϵ-transitions, where given the capabilities, constraints and failure modes of the agents involved, an initial state of the system and a task specification, an optimal solution is generated that satisfies the system constraints and the task specification. The resulting solution is guaranteed to be complete and optimal; moreover, a heuristic solution that offers significant reduction of the computational requirements while relaxing the completeness and optimality requirements is proposed. The constructed system model is independent from the initial condition and the task specification, alleviating the need to repeat the costly pre-processing step for solving other scenarios, while allowing the incorporation of failure modes on-the-fly. An analysis of the completeness and optimality properties as well as the class of addressable problems is provided.**

***Index Terms*—Discrete event systems, heterogeneous systems, multi-agent systems, task planning.**

I. Introduction

Multi-agent systems operating autonomously in dynamical environments to perform complicated tasks have been one of the major areas of research interest during the last decade. High-level task planning using formal methods to define the system requirements is one of the promising approaches. Typical objectives arise from the multi-agent systems' behavior and requirements, such as sequential or reactive tasks, control, coordination and motion and task planning. The main motivation for this work comes from the field of manufacturing logistics automation and the need to automate intra-factory logistics operations. Real world problems that can benefit from automatic task planning appear in multiple sectors, including logistics and supply chain management (e.g. vehicle routing, inventory management, warehouse operations), manufacturing and production (e.g. production scheduling, supply chain planning, maintenance scheduling, recycling planning), healthcare (e.g. patient scheduling, disaster response, drug development), transportation (e.g. air traffic control, public transportation, autonomous vehicle navigation), etc.

Manuscript submitted November 28, 2023. Portions of this work appeared in [1], [2]. A scaled down version of the software developed for this work has been made publicly available at [3]. This work was partially supported by EU programs under grant agreements 951813 (Better Factory, H2020) and 101092295 (CIRCULOOS, Horizon Europe).

The authors are with the Mechanical Engineering and Materials Science and Engineering Department of Cyprus University of Technology, Limassol, CYPRUS (e-mails: {anatoli.tziola, savvas.loizou}@cut.ac.cy).

This work introduces a new framework for high-level task planning problems with respect to agent capabilities, constraints and failure modes. The initial concepts for this work were presented in [1], whereas the current provides all the missing analysis (proofs, class of addressable problems, extended literature analysis). This work builds on top of the work on motion task planning in [4], where module composition was used to automatically synthesize motion controllers defined by LTL specifications. While the current work generalizes the concept of motion tasks to general tasks and develops the formal machinery for their automatic synthesis, leveraging the agents' capabilities and constraints, it significantly departs from [4] by addressing a more general problem in a unified manner without the need to resort to the formalism of hybrid systems and LTL. The problem formulation uses the global system model as the environment model composed by individual agents' capabilities and constraints, considering the individual agent's failure modes to determine the optimal task plan. The problem is posed as a special case of Module Composition Problem (MCP) [5] and then reduced to a combinatorial optimization problem of shortest dipath, which can be solved in polynomial time. The optimal module chain under a supervisory control framework enables the use of the generated module chain as a model system for building supervisory controllers, as demonstrated in [4]. In the sequel, the module chain with the supervisory controllers can e.g. be combined with Navigation Function based controllers [6] with time abstracting properties (i.e. guarantees on the required time for a task - see e.g. [7]), to synthesize control laws with performance guarantees that ensure compositionality for accomplishing individual and global tasks. The developed methodology provides the capability for on-the-fly incorporation of failure modes, however the supervisory control and navigation framework for reactive activation of failure modes during execution, and on-the-fly plan reconfiguration, are beyond the scope of the current work.

Several proposed methodologies address the high-level task planning problem using formal languages to express autonomous systems behavior and synthesize the supervisory control architecture to achieve the given specifications [8], [9]. Some of the most common approaches include Linear Temporal Logic (LTL), Capability Temporal Logic (CaTL) [10], Petri nets [11], sampling-based approaches [12], [13] and domain definition languages. Many of the existing works use LTL formulas to develop bottom-up [14]–[17] and distributed [18] approaches, where local LTL expressions are assigned to robots, or top-down approaches [12], [19]–[22], where a global task is decomposed into independent sub-tasks that are treated separately by each agent. A descriptive overview of the literature using formal methods to accomplish high-level specifications can be found in [23].

There are several works addressing the (heterogeneous) multi-robot task allocation problem using LTL [24]–[26]. These approaches do not directly address the issue of inclusion of failure modes or on-the-fly re-planning. Other works tackle the task planning problem of heterogeneous multi-robot systems using CaTL [10], [27], [28], providing failure handling capabilities and on-the-fly re-planning. However partial agent failures (i.e. failure modes where an agent retains some functionality) are not directly addressed. Petri nets are a widely used modeling formalism for Discrete Event Systems (DES) [11], [29]. In [13], a sampling-based approach is proposed using directed trees to approximate the state space and transitions of synchronous product automata, providing probabilistic completeness and asymptotic optimality guarantees.

Common among the works presented in the literature survey is the adoption of formal verification techniques for motion planning and controller synthesis. Several gaps in the existing literature are been addressed by our approach: (Gap. 1) To the best of our knowledge, none of the existing methodologies provides an effective modeling and execution framework capturing a heterogeneous multi-agents' system individual and collective capabilities, constraints and failure modes. Planning Domain Definition Language (PDDL) implementations have exponential complexity and are not particularly suited to applications requiring on-line re-planning or reconfiguration. (Gap. 2) In LTL-based planners, the solution domain depends on the task specification. Every time the specification is changed a new solution domain needs to be created, incurring increased computational overhead in the case of re-tasking e.g. in the case of failures, modified objectives[1] or initial conditions. (Gap. 3) Increasing the objectives in a task specification, increases the computational complexity, particularly in LTL-based planners where the worst-case computational complexity is double exponential with respect to the length of the formula. (Gap. 4) Sampling-based approaches relax the properties of optimality and completeness in finite time. Some, like in [13], provide asymptotic guarantees as the computational resources asymptotically expand to capture the full state space.

Motivated by the identified gaps in the literature, this work provides the following contributions:

1) A new system modeling framework that combines the agents' capabilities and constraints both at the individual and the group levels, including failure modes (Gap. 1).
2) Determination of optimal task plans that satisfy any possible task specification from any initial condition, without the need to repeat the pre-processing step (Gap. 4).
3) Capability of incorporation of multiple objectives in the task specification, decreasing the time complexity while maintaining the space complexity (Gap. 3).
4) Determination of reduced complexity sub-optimal solutions for any task specification, while maintaining its space complexity (Gap. 2).
5) Capability of incorporation of failure modes on-the-fly for re-planning, without the need to repeat the costly pre-processing step (Gap. 1, Gap. 2).

[1]Note that in the current work we make a distinction between task specification and objective in the sense that the objective denotes the final state of a single agent while the task specification may encapsulate any number of objectives. Compared with the terminology used in the LTL literature, the evolution of the system is not part of the task specification but is encoded in the system's capabilities and constraints.

In the rest of the paper section II presents the preliminaries, section III the problem formulation, section IV casts the problem as a Module Composition one, section V analyses the performance of the methodology and section VI concludes the paper.

II. Preliminaries

A. Definitions

In this section, we introduce the necessary terminology and definitions for the development of our methodology.

If A and B are sets, the cardinality of set A is denoted as $|A|$. The union and intersection of sets are denoted as $A \cup B$ and $A \cap B$ whereas set subtraction of set B from set A is denoted as $A \backslash B$. We use the $\wedge$ operator to denote conjunction.

We will use the Deterministic Finite Automata (DFA) represented as the tuple $G \triangleq (X_G, E_G, f_G, \Gamma_G, x_{0,G}, X_{m,G})$ (see [30] for detailed definition) and the cost function $g_G(e)$ that maps each event to a corresponding cost defined as $g_G : E_G \rightarrow \mathbb{R}_{>0}$. Unless otherwise stated, subscripts of an (DFA or NFA) automaton's tuple elements will refer to the corresponding automaton as above. We need to introduce a new framework of ϵ_0-NFAs that allows the use of ϵ_0-transitions by enabling transitions on an empty event only from the initial state. The introduction of these automata became necessary to enable closure under the newly introduced operations in Section II-B. In practical terms, this allows us to consistently setup our system to handle any initial condition.

Definition 1 (ϵ_0-NFA): An ϵ_0-NFA is a Nondeterministic Finite Automaton with ϵ-transitions only from the initial state, defined as a six-tuple ${}^{\epsilon_0}G \triangleq (X_G \cup \{x_0\}, E_G \cup \{\epsilon\}, {}^{\epsilon_0}f_G, \Gamma_G, x_0, X_{m,G})$, where ${}^{\epsilon_0}f_G : X_G \cup \{x_0\} \times E_G \cup \{\epsilon\} \rightarrow 2^{X_G}$ such that $\forall (x,e) \in X_G \times E_G : {}^{\epsilon_0}f_G(x,e) := f_G(x,e)$, where $x_0 \notin X_G$ is the initial state with $x_0 \neq x_{0,G}$.

Corollary 1: If ${}^{\epsilon_0}G$ is an ϵ_0-NFA then it can be converted to the DFA G by removing x_0 along with the associated ϵ transitions and assigning an $x_{0,G} \in X_G$ as an initial state.

Proof: This can be trivially shown by observing that the six-tuple obtained by the operation is as the one in the definition of DFA. ■

Based on the Corollary 1, we can construct the operator:

Definition 2 (ϵ_0-NFA Determinization): Let ${}^{\epsilon_0}G$ be an ϵ_0-NFA and $x_{0,G} \in X_G$ a state in X_G that we want to assign as an initial state. Then, $\Delta({}^{\epsilon_0}G, x_{0,G}) \triangleq G$ where G is the DFA and the determinization operator Δ performs the conversion as described in Corollary 1.

In this case, the determinization operator converts the ${}^{\epsilon_0}G$ to the DFA G by removing the "virtual" starting state x_0 and the ϵ event, and assigning an element of X_G as the new starting state

$x_{0,G}$ of the constructed DFA G. Compared with the standard determinization in automata theory, the determinization operator in our case, results in a state space equivalent to the state space of ϵ_0-NFA, while the accommodation of ϵ transitions in the resulting DFA is not required.

We define the inverse transition function as follows:

Definition 3 (Inverse Transition Function): Let G be a DFA. Define $f_G^{-1} : X_G \times E_G \to X_G$ to be the inverse transition function such that $f_G(x_G, e) = y_G \wedge f_G^{-1}(y_G, e) = x_G$ for some $x_G : e \in \Gamma_G(x_G),\ y_G : e \in \Gamma_G(f_G^{-1}(y_G, e))$.
Note that an inverse transition function as in Definition 3 cannot always be defined for DFAs. For DFAs where such inverse transition function can be defined, we introduce the following concept:

Definition 4 (Compatible ϵ_0-NFAs): Let ${}^{\epsilon_0}G$ and ${}^{\epsilon_0}B$ be ϵ_0-NFAs. Let $E_C := E_G \cap E_B$. Then, the automata ${}^{\epsilon_0}G$ and ${}^{\epsilon_0}B$ are compatible iff $\forall e \in E_C$, it holds that[2], $f_G(x_G, e) = f_B(x_B, e)$ for some $x_G : e \in \Gamma_G(x_G), x_B : e \in \Gamma_B(x_B)$ and $f_G^{-1}(y_G, e) = f_B^{-1}(y_B, e)$ for some $y_G : e \in \Gamma_G(f_G^{-1}(y_G, e)),\ y_B : e \in \Gamma_B(f_B^{-1}(y_B, e))$. In such case, it will also be true that $x_G = x_B$ and $y_G = y_B$. We denote such a compatibility relation as ${}^{\epsilon_0}G \asymp {}^{\epsilon_0}B$.
Note that the above definition effectively forces the endpoints of common events to be common states.

B. Operations on compatible automata

Here we introduce a custom set of the basic operations on compatible ϵ_0-NFAs: union, subtraction and concatenation. The introduced operations differ from the ones found in the automata literature and provide the necessary functionality for the subsequent developments.

Definition 5 (Union of Compatible Automata): For the ϵ_0-NFAs ${}^{\epsilon_0}G$ and ${}^{\epsilon_0}B$, assume ${}^{\epsilon_0}G \asymp {}^{\epsilon_0}B$. Define the compatible ϵ_0-NFAs union ${}^{\epsilon_0}\mathcal{P} \triangleq {}^{\epsilon_0}G \cup_{\asymp} {}^{\epsilon_0}B$ to be the six-tuple ${}^{\epsilon_0}\mathcal{P} \triangleq (X_{\mathcal{P}} \cup \{x_0\}, E_{\mathcal{P}} \cup \{\epsilon\}, {}^{\epsilon_0}f_{\mathcal{P}}, \Gamma_{\mathcal{P}}, x_0, X_{m,\mathcal{P}})$, where $X_{\mathcal{P}} := X_G \cup X_B$, $E_{\mathcal{P}} := E_G \cup E_B$, $\Gamma_{\mathcal{P}} : X_{\mathcal{P}} \to 2^{E_{\mathcal{P}}}$, $x_0 \notin X_{\mathcal{P}}$ and $X_{m,\mathcal{P}} := X_{m,G} \cup X_{m,B}$. The transition function ${}^{\epsilon_0}f_{\mathcal{P}} : X_{\mathcal{P}} \cup \{x_0\} \times E_{\mathcal{P}} \cup \{\epsilon\} \to 2^{X_{\mathcal{P}}}$ is such that $\forall (x,e) \in X_G \times E_G : {}^{\epsilon_0}f_{\mathcal{P}}(x,e) := f_G(x,e)$ and $\forall (x,e) \in X_B \times E_B : {}^{\epsilon_0}f_{\mathcal{P}}(x,e) := f_B(x,e)$.

Compared with the standard union operation on automata presented in the literature, the union operation of Definition 5 produces the union instead of the cartesian product state space.

Definition 6 (Subtraction of Compatible Automata): For the ϵ_0-NFAs ${}^{\epsilon_0}G$ and ${}^{\epsilon_0}B$, assume ${}^{\epsilon_0}G \asymp {}^{\epsilon_0}B$. Define the compatible ϵ_0-NFAs subtraction ${}^{\epsilon_0}\Theta \triangleq {}^{\epsilon_0}G \setminus_{\asymp} {}^{\epsilon_0}B$ to be the six-tuple ${}^{\epsilon_0}\Theta \triangleq (X_\Theta \cup \{x_0\}, E_\Theta \cup \{\epsilon\}, {}^{\epsilon_0}f_\Theta, \Gamma_\Theta, x_0, X_{m,\Theta})$, where $X_\Theta := X_G$, $E_\Theta := E_G \setminus E_B$, $\Gamma_\Theta : X_\Theta \to 2^{E_\Theta}$ and $x_0 \notin X_\Theta$, $X_{m,\Theta} := X_{m,G} \setminus X_{m,B}$. The transition function ${}^{\epsilon_0}f_\Theta : X_\Theta \cup \{x_0\} \times E_\Theta \cup \{\epsilon\} \to 2^{X_\Theta}$ is such that $\forall (x,e) \in X_\Theta \times E_\Theta : {}^{\epsilon_0}f_\Theta(x,e) := f_\Theta(x,e)$.

Note that the subtraction operation of Definition 6 is more in line with the set difference operator than with the intersection with the language complement used in regular languages.

[2]Note that due to Corollary 1 we have that ${}^{\epsilon_0}f_G(x_G, e) \equiv f_G(x_G, e)$ and ${}^{\epsilon_0}f_B(x_B, e) \equiv f_B(x_B, e)$.

Definition 7 (Concatenation of Compatible Automata): For the ϵ_0-NFAs ${}^{\epsilon_0}G$ and ${}^{\epsilon_0}B$, assume ${}^{\epsilon_0}G \asymp {}^{\epsilon_0}B$. Define the compatible ϵ_0-NFAs concatenation ${}^{\epsilon_0}\Phi \triangleq {}^{\epsilon_0}G \perp\!\!\!\perp_{\asymp} {}^{\epsilon_0}B$ to be the six-tuple ${}^{\epsilon_0}\Phi \triangleq (X_\Phi \cup \{x_0\}, E_\Phi \cup \{\epsilon\}, {}^{\epsilon_0}f_\Phi, \Gamma_\Phi, x_0, X_{m,\Phi})$, where $X_\Phi := \{uv | u \in X_G, v \in X_B\}$, $E_\Phi := E_G \cup E_B$, $\Gamma_\Phi : X_\Phi \to 2^{E_\Phi}$, $x_0 \notin X_\Phi$, and $X_{m,\Phi} := \{uv | u \in X_{m,G} \wedge v \in X_{m,B}\}$. The transition function ${}^{\epsilon_0}f_\Phi : X_\Phi \cup \{x_0\} \times E_\Phi \cup \{\epsilon\} \to 2^{X_\Phi}$ is such that $\forall (x,e) \in X_\Phi \times E_\Phi : {}^{\epsilon_0}f_\Phi(x,e) := f_\Phi(x,e)$.

Note that the operation of concatenation in Definition 7 above, is similar to the well known parallel composition operation but differs in that the event sets undergo a disjoint union in our case to ensure that only one event can be activated at a time (no synchronization on common events needed).

The concept of compatible automata, allows us to recover the properties for the operations of Definitions 5, 6 and 7:

Corollary 2: The automata resulting from the operations defined in Definitions 5, 6 and 7 are ϵ_0-NFAs.

Proof: This is established by first noting that for the operations in Definitions 5, 6 and 7, the resulting automata are six-tuples as in Definition 1. Now we need to show that the resulting transition function f_R is partial in its domain. Assume that R is the resulting automaton from the operation (union, subtraction or concatenation). We have the cases:

1. *Union operation:* Due to the properties of Definition 4, it will hold that $\forall x \in X_R$, then $\forall e \in \Gamma_G(x) \cap \Gamma_B(x)$ it will hold that $f_G(x,e) = f_B(x,e)$.
2. *Subtraction operation:* Due to the properties of Definition 4, it will hold that $\forall x \in X_R$, then $\forall e \in \Gamma_R(x)$ it will hold that $f_R(x,e) = f_G(x,e)$.
3. *Concatenation operation:* Due to the properties of Definition 4, it holds that $\forall (x := uv) \in X_R$, where $u \in X_G$ and $v \in X_B$, then $\forall e \in \Gamma_R(x)$ it holds that: either $f_R(x,e) = f_G(u,e)$, with $e \in E_G$, or $f_R(x,e) = f_B(v,e)$, with $e \in E_B$.

As can be seen, in all three operations, $\forall x \in X_R$ and for each $e \in \Gamma_R(x)$, the state $f_R(x,e)$ is a unique state in X_R and hence, the resulting transition function is partial in its domain. Thus, R will be an ϵ_0-NFA since it fulfills the requirements of Definition 1. ■

We need to introduce the following operator that addresses individual states of the concatenated states of Definition 7.

Definition 8 (Projection Operator): Let $x \in X_\Phi$ be a concatenated state of n elements, where x_i denotes the $i'th$ element of x, $i \in \{1, \ldots, n\}$.

Define the projector b as an n-bit binary and b_i its i'th bit. The projection operator is defined as the ordered set $proj(x,b) \triangleq \{x_i | b_i = 1\}$.

We will need some definitions from MCP literature as well as some new ones for our development. Using the module definition by [5], we state the module in an appropriate form for our development.

Definition 9 (Single Port Module): Let G be a DFA. Consider the finite set of ports $P = P_{in} \cup P_{out}$, where P_{in} be a non-empty finite set of input ports and P_{out} be a non-empty finite set of output ports with $P \subseteq X_G$ and $P_{in} \cap P_{out} \neq \emptyset$. We define the single input/single output port (I/O) module T as a 3-tuple of $T \triangleq \{p, e, q\}$, where $p \in P_{in}$ is the input port, $e \in E_G : f_G(p, e) = q$ and $q \in P_{out}$ is the output port.

The cost associated with module T is defined as $c(T) \triangleq g_G(e)$. We can define the inverted module T^{-1} where its input becomes its output and vice versa such that Definition 3 holds.

III. Problem Formulation

A. Problem statement

Given a set of agents, their capabilities and constraints as individuals, their emerging capabilities and constraints when operating in teams, their failure modes, and a task specification defining the final state of at least one agent, determine (if any) the optimal sequence of actions that brings the system from any initial state, to a state satisfying the task specification.

B. Agent Model

Agents are considered as entities, such as robots, machines, humans, items or anything that it could change its status to act on or react to its surroundings during a process or trigger event. An agent is modeled as an ϵ_0-NFA (see Definition 1) composed by the agent's capabilities and constraints. Agent's capabilities represent the allowed state transitions derived from the combination of individual capabilities and failure mode modeled as ϵ_0-NFAs. Agent's constraints express the forbidden state transitions derived from the union of individual's constraints modeled as ϵ_0-NFAs. The agent model is produced by the subtraction of the agent's constraints from the agent's capabilities. The i^{th} agent, $i \in \{1, \dots, n\}$, is modeled by the ϵ_0-NFA ${}^{\epsilon_0}A_i$, and let ${}^{\epsilon_0}\mathcal{A}$ denote the set of n agents.

1) Individual Agent Capabilities: Let κ be the total number of individual capabilities of agent ${}^{\epsilon_0}A_i$ and let the ϵ_0-NFA ${}^{\epsilon_0}M_{\beta,i}$ denote its β'th individual capability, where $\beta \in \{1, \dots, \kappa\}$.

2) Individual Agent Failure Mode: Consider a state $x' \in X_{M_{\beta,i}}$. We define a failure mode of ${}^{\epsilon_0}A_i$ as the inability to complete the transition from some $x \in X_{M_{\beta,i}}$ to x' with an occurrence of event $e \in E_{M_{\beta,i}}$. This describes a detected transition failure of ${}^{\epsilon_0}A_i$ which renders $f_{M_{\beta,i}}(x, e) = x'$ infeasible. This failure mode is modeled by the ϵ_0-NFA ${}^{\epsilon_0}F_i$ such that $X_{F_i} = \{x, x'\}$ and $E_{F_i} = \{e\}$.

We model the agent's capabilities as the subtraction of the agent failure from the union of individual agent capabilities utilizing the union and the subtraction of compatible automata.

3) Agent Capabilities: Considering κ compatible ϵ_0-NFAs such that ${}^{\epsilon_0}M_{\alpha,i} \asymp {}^{\epsilon_0}M_{\beta,i}$, $\alpha \neq \beta$ with $\alpha, \beta \in \{1, \dots, \kappa\}$ and ${}^{\epsilon_0}F_i \asymp {}^{\epsilon_0}M_{\beta,i}$, the capabilities of ${}^{\epsilon_0}A_i$ are modeled by the ϵ_0-NFA ${}^{\epsilon_0}K_i$ as:

$$ {}^{\epsilon_0}K_i \triangleq \{\bigcup_{\beta=1}^{\kappa}{}_{\asymp}\ {}^{\epsilon_0}M_{\beta,i}\} \setminus_{\asymp} {}^{\epsilon_0}F_i. \tag{1} $$

4) Individual Agent Constraints: Let λ be the total number of individual constraints of agent ${}^{\epsilon_0}A_i$ and let the ϵ_0-NFA ${}^{\epsilon_0}N_{\xi,i}$ denote its ξ'th individual constraint, where $\xi \in \{1, \dots, \lambda\}$.

We model agent's constraints as the union of individual agent constraints utilizing the union of compatible automata.

5) Agent Constraints: Considering λ compatible ϵ_0-NFAs such that ${}^{\epsilon_0}N_{\xi,i} \asymp {}^{\epsilon_0}N_{\eta,i}$, $\xi \neq \eta$ with $\eta \in \{1, \dots, \lambda\}$, the constraints of ${}^{\epsilon_0}A_i$ are modeled by the ϵ_0-NFA ${}^{\epsilon_0}D_i$ as:

$$ {}^{\epsilon_0}D_i \triangleq \{\bigcup_{\xi=1}^{\lambda}{}_{\asymp}\ {}^{\epsilon_0}N_{\xi,i}\}. \tag{2} $$

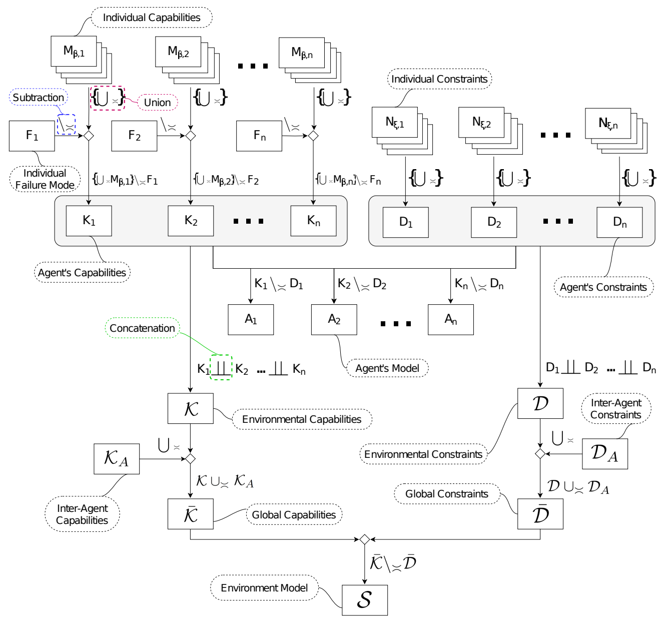

Fig. 1. Flow diagram of the operations to produce models of agents ${}^{\epsilon_0}A_i$ and environment ${}^{\epsilon_0}\mathcal{S}$.

Considering ${}^{\epsilon_0}K_i \asymp {}^{\epsilon_0}D_i$, the i^{th} agent is modeled by the ϵ_0-NFA ${}^{\epsilon_0}A_i$ after subtracting ${}^{\epsilon_0}D_i$ from ${}^{\epsilon_0}K_i$:

$$ {}^{\epsilon_0}A_i \triangleq {}^{\epsilon_0}K_i \setminus_{\asymp} {}^{\epsilon_0}D_i. \tag{3} $$

C. Environment Model

Agents are acting in the environment to reach individual states while being influenced by state capabilities and constraints relating to other agents. Agents who rely on other agents to perform actions or reach goals are grouped into a team. The capabilities and constraints of the team are modeled as a combination of individual states of the members of the team. These capabilities and constraints are defined as inter-agent capabilities and constraints and express the allowed and not-permitted environment state transitions that emerge when the agents of team are present. Given the agents' capabilities and constraints and the inter-agent capabilities and constraints as ϵ_0-NFAs, the environment model expressed as an ϵ_0-NFA captures the full behavior of the system using all the possible combinations of agents' states. The procedure of constructing agents' and environment models is illustrated in Fig. 1. We model the environmental capabilities as the concatenation of agents' capabilities ϵ_0-NFAs, formed as in (1), utilizing the concatenation operation of compatible automata to express the allowed environment state transitions.

1) Environmental Capabilities: For the ϵ_0-NFAs ${}^{\epsilon_0}K_i$ and ${}^{\epsilon_0}K_j$, $i \neq j$ with $i, j \in \{1, \dots, n\}$, let ${}^{\epsilon_0}K_i \asymp {}^{\epsilon_0}K_j$. The environmental capabilities are modeled by the ϵ_0-NFA ${}^{\epsilon_0}\mathcal{K}$ as:

$$ {}^{\epsilon_0}\mathcal{K} \triangleq \coprod_{i=1}^{n}{}_{\asymp}\ {}^{\epsilon_0}K_i. \tag{4} $$

We model the environmental constraints as the concatenation of agents' constraints ϵ_0-NFAs, formed as in (2), utilizing the concatenation operation of compatible automata to express the not-permitted environment state transitions.

2) Environmental Constraints: For the ϵ_0-NFAs ${}^{\epsilon_0}D_i$ and ${}^{\epsilon_0}D_j$, $i \neq j$ with $i, j \in \{1, \ldots, n\}$, let ${}^{\epsilon_0}D_i \asymp {}^{\epsilon_0}D_j$. The environmental constraints are modeled by the ϵ_0-NFA ${}^{\epsilon_0}\mathcal{D}$ as:

$$ {}^{\epsilon_0}\mathcal{D} \triangleq \coprod_{i=1}^{n}{}_{\asymp} {}^{\epsilon_0}D_i. \tag{5}$$

Using the agents' models formed as in (3), the inter-agents capabilities ϵ_0-NFA and inter-agents constraints ϵ_0-NFA are modeled as follows:

3) Inter-Agents Capabilities: Let $\text{co} \subseteq {}^{\epsilon_0}\mathcal{A}$ with $|\text{co}| \leq n$ be a set of compatible agent ϵ_0-NFAs. Then, the ϵ_0-NFA ${}^{\epsilon_0}\mathcal{K}_A \subset \coprod_{\asymp i \in \text{co}} {}^{\epsilon_0}A_i$ denotes the inter-agents capabilities[3] between the members of co.

4) Inter-Agents Constraints: Let $\text{co} \subseteq {}^{\epsilon_0}\mathcal{A}$ with $|\text{co}| \leq n$ be a set of compatible agent ϵ_0-NFAs. Then, the ϵ_0-NFA ${}^{\epsilon_0}\mathcal{D}_A \subset \coprod_{\asymp i \in \text{co}} {}^{\epsilon_0}A_i$ denotes the inter-agents constraints between the members of co.

We model the global capabilities as the ϵ_0-NFA derived from the union of environmental capabilities (formed as in (4)) with inter-agent capabilities.

5) Global Capabilities: For the ϵ_0-NFAs ${}^{\epsilon_0}\mathcal{K}$ and ${}^{\epsilon_0}\mathcal{K}_A$, let ${}^{\epsilon_0}\mathcal{K} \asymp {}^{\epsilon_0}\mathcal{K}_A$. The global capabilities are modeled by the ϵ_0-NFA ${}^{\epsilon_0}\widetilde{\mathcal{K}}$ defined as ${}^{\epsilon_0}\widetilde{\mathcal{K}} \triangleq {}^{\epsilon_0}\mathcal{K} \cup_{\asymp} {}^{\epsilon_0}\mathcal{K}_A$.

We model the global constraints as the ϵ_0-NFA derived from the union of environmental constraints (formed as in (5)) with inter-agent constraints.

6) Global Constraints: For the ϵ_0-NFAs ${}^{\epsilon_0}\mathcal{D}$ and ${}^{\epsilon_0}\mathcal{D}_A$, let ${}^{\epsilon_0}\mathcal{D} \asymp {}^{\epsilon_0}\mathcal{D}_A$. The global constraints are modeled by the ϵ_0-NFA ${}^{\epsilon_0}\widetilde{\mathcal{D}}$ defined as ${}^{\epsilon_0}\widetilde{\mathcal{D}} \triangleq {}^{\epsilon_0}\mathcal{D} \cup_{\asymp} {}^{\epsilon_0}\mathcal{D}_A$.

Considering ${}^{\epsilon_0}\widetilde{\mathcal{K}} \asymp {}^{\epsilon_0}\widetilde{\mathcal{D}}$, the environment model is modeled by the ϵ_0-NFA ${}^{\epsilon_0}\mathcal{S}$ after subtracting ${}^{\epsilon_0}\widetilde{\mathcal{D}}$ from ${}^{\epsilon_0}\widetilde{\mathcal{K}}$:

$$ {}^{\epsilon_0}\mathcal{S} \triangleq {}^{\epsilon_0}\widetilde{\mathcal{K}} \setminus_{\asymp} {}^{\epsilon_0}\widetilde{\mathcal{D}}, \tag{6}$$

where the cardinality of $X_{\mathcal{S}}$ is $\theta = \prod_{i=1}^{n} |X_{A_i}|$.

D. Task Specification

Let b be an n-bit binary indicating the states of the agents that we are interested in specifying. We define the task specification γ as the tuple $\gamma \triangleq (\varphi, b)$. Let $x_d \in X_{\mathcal{S}}$. Then, x_d satisfies the task specification γ, if $proj(x_d, b) := \varphi$.

Depending on the cardinality of φ (i.e. how many bits are active in b), single or multiple tasks (objectives) can be concurrently specified in γ.

IV. FORMULATION AS A MODULE COMPOSITION PROBLEM

Let us now consider the environment model ${}^{\epsilon_0}\mathcal{S}$ and assume that we would like to use $x_{0,\mathcal{S}} \in X_{\mathcal{S}}$ as the initial state of our system. Thus, $\mathcal{S} = \Delta({}^{\epsilon_0}\mathcal{S}, x_{0,\mathcal{S}})$ is the DFA description of the environment model. Let $P_{in} \in X_{\mathcal{S}}$ denote the non-empty finite set of input ports and $P_{out} \in X_{\mathcal{S}}$ denote the non-empty finite set of output ports, where $P = P_{in} \cup P_{out}$ and $P_{in} \cap P_{out} \neq \emptyset$. The module T_j is defined as $T_j \triangleq \{p_j, e_j, q_j\}$, where $p_j \in P_{in}$, $e_j \in E_{\mathcal{S}}$, $q_j \in P_{out}$ and $c(T_j) \triangleq g_{E_{\mathcal{S}}}(e_j)$ is the cost of implementing module T_j.

Define the i^{th} task module as $T_{0,i} = \{x_{0,\mathcal{S}}, e_{0,i}, x_{d,i}\}$ where $proj(x_{d,i}, b) := \varphi$, $x_{d,i} \in X_{\mathcal{S}}$ and $e_{0,i}$ a virtual transition from the initial to the final state. Observe that there are up to $|\varphi|!$ potential solutions so $i \in \{1, \ldots, |\varphi|!\}$.

To tackle the problem defined in the Problem Statement, we proceed to formulate our problem as a Module Composition Problem [5]. Since we are using single-port modules, we will have a special case of the MCP that is solvable in polynomial time.

We define $\mathcal{T}_i$ as the finite open module chain describing the task plan for a given task specification γ defined as $\mathcal{T}_i = \{T_{1,i}, \ldots, T_{z,i}\}$, where $z_i = |\mathcal{T}_i|$. We define $\mathring{\mathcal{T}_i}$ as the finite closed module chain, containing $T_{0,i}^{-1}$ and task plan $\mathcal{T}_i$, describing the sequential environment states transitions during the execution defined as $\mathring{\mathcal{T}_i} = \{T_{0,i}^{-1}, \mathcal{T}_i\}$. In the sequel we will drop the index i for notational brevity.

Let t_j denote the number of instances of T_j, $c_j = c(T_j)$, then the discrete optimization problem can be posed as an integer programming problem as follows:

$$ min \sum_{j \in \{1, \ldots |E_{\mathcal{S}}|\}} c_j t_j, \tag{7}$$

subject to:

$$\begin{aligned} & t_0 = 1, t_j \in \mathbb{Z}^+, \\ & \sigma_p = \sum_{q \in P_{out}} w_{p,q}, \forall p \in P_{in}, \\ & \mu_q = \sum_{p \in P_{in}} w_{p,q}, \forall q \in P_{out}, \end{aligned}$$

where $\mathbb{Z}^+$ denotes the non-negative integers, $w_{p,q}$ the number of connections between input port p and output port q, σ_p the number of modules with input port p utilized and μ_q the number of modules with output port q utilized. Since in the current work we have implemented only single port modules, the above integer programming problem can be reduced to the shortest directed path problem [5], that can be solved utilizing the Dijkstra's shortest dipath algorithm. The optimal solution composes the $\mathcal{T}$ that minimizes the cost of states transitions in order to satisfy the task specification γ.

The drawback of using singe port modules in the current work is that we are only limiting transitions to be performed by one agent at a time. Multi-port modules are currently being considered as a further research topic, to enable multiple agents to perform concurrent transitions and is beyond the scope of the current work. We have to also note here that the additional effort in casting the problem as a module composition one, enables us to seamlessly use the generated module chain as a model system for building supervisory controllers as in [4], and is currently under active research.

V. ANALYSIS

A. Pre-processing Analysis

The pre-processing step includes the construction of the agents models and the environment model (see Fig.1) using

[3]Inter-Agents Failure Modes can be defined so as to restrict Inter-Agents Capabilities

the automata operations as described in sections III-B and III-C. A polynomial amount of time on the number of agent states is required to construct the models of all agents and the environment model. More specifically, the model of agent ${}^{\epsilon_0}A_i$ is derived from the subtraction of agent's constraints from agent's capabilities. Both operations, the union and the subtraction, require a total of $O(|X_{A_i}|^2)$ time to be performed. Thus, the complexity of the agent model construction is $O(|X_{A_i}|^2)$. On the other hand, the concatenation operation requires $O(|X_\mathcal{S}|)$ time, where $|X_\mathcal{S}| = \prod_{i=1}^{n} |X_{A_i}|$, while both the union and subtraction operations require $O(|X_\mathcal{S}|^2)$. Hence, the time complexity for the construction of the environment model is $O(|X_\mathcal{S}|^2)$ in total.

It is important to note here that the environment model ${}^{\epsilon_0}\mathcal{S}$ is only needed to be constructed once and can be use thereafter for all possible task specifications and initial states. ${}^{\epsilon_0}\mathcal{S}$ is converted to a weighted directed graph $\mathcal{H}_\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$, where the set of nodes $\mathcal{V}_\mathcal{S}$ corresponds to the set of states $X_\mathcal{S}$, the set of edges $\mathcal{E}_\mathcal{S}$ is defined by $f_\mathcal{S}$ associated with its cost $g_\mathcal{S}$. $\mathcal{H}_\mathcal{S}$ is related to the directed graph produced by the module composition procedure, since by the definition of module T_j, edges in $\mathcal{E}_\mathcal{S}$ serve as vertices of the module composition graph. In addition, due to the compatibility relation requirements, the vertices of each edge in $\mathcal{E}_\mathcal{S}$ are unique, hence the directed graph produced by the module composition procedure is the Line graph $L(\mathcal{H}_\mathcal{S})$ of $\mathcal{H}_\mathcal{S}$. Any directed path, in $\mathcal{H}_\mathcal{S}$ with its associated cost, has an equivalent path of the same cost and the same event sequence in $L(\mathcal{H}_\mathcal{S})$. To see this consider that a path $v_1 e_1 v_2 e_2 \dots v_n e_n v_{n+1}$ in $\mathcal{H}_\mathcal{S}$ will necessarily correspond to the path $e_1(q_1 p_2)e_2 \dots (q_{n-1} p_n)e_n$ in $L(\mathcal{H}_\mathcal{S})$ where $(q_i p_{i+1})$ are the edges in $L(\mathcal{H}_\mathcal{S})$ corresponding to the connection between output port q_i and input port p_{i+1}. Hence, solving a shortest directed path problem on $\mathcal{H}_\mathcal{S}$ is equivalent to solving the module composition problem.

Regarding complexity, the adjacency matrix representation of $\mathcal{H}_\mathcal{S}$ is a 2-dimensional array $X_\mathcal{S} \times X_\mathcal{S}$. Each element in the array stores the cost $g_\mathcal{S}$ related to the edge $f_\mathcal{S}(x \in X_\mathcal{S}, e \in E_\mathcal{S})$. The amount of space required to store the array is $O(|\mathcal{V}_\mathcal{S}|^2)$ in worst case.

B. Analysis of the Complete Solution

The problem solving phase encapsulates the synthesis of the optimal task plan and the integration of individual agent failure mode. The Dijkstra's algorithm is implemented over the weighted graph $\mathcal{H}_\mathcal{S}$ (Algorithm 1, lines 8-15) to find the optimal task plan $\mathcal{T}$ as given in Algorithm 1 in lines 21-23. In the case where a fault is detected for the transition from state x_i to x_j of $\mathcal{S}$ (e.g. by some fault identification system) that affects agent ν, we can disable all the affected transitions by finding the set of states X_i', X_j', where $proj(x_i, b_\nu) \equiv proj(x_i' \in X_i', b_\nu)$ and $proj(x_j, b_\nu) \equiv proj(x_j' \in X_j', b_\nu)$. Then, we eliminate the transitions from all $x_i' \in X_i'$ to all $x_j' \in X_j'$. New failure modes can be incorporated on-the-fly into ${}^{\epsilon_0}\mathcal{S}$ without the need to repeat the costly pre-processing step. The computational time required for this modification is $\theta_f = \prod_{i=1,\ i \neq \nu}^{n} |X_{A_i}|$ in the worst case.

The task planning problem of minimizing the length of the plan is NP-complete. Let $\mathcal{P}$ be the solution to this problem found after running Dijkstra's algorithm and let $\mathcal{P}_i$ denote it's i'th element, $i \in \{1, \dots, |\mathcal{P}|\}$ (Algorithm 1, lines 16-20). In Algorithm 2, the optimal solution can then be found by running the algorithm for all states that satisfy the task specification (Algorithm 2, lines 3-16), that is $\theta' = \prod_{i=1,\ i \notin \sigma}^{n} |X_{A_i}|$ times in the worst case scenario, where σ denotes the set of agents that were used for the task specification γ. The running time for implementing Algorithm 1 with the "Complete" solution type (line 16-17), is $O(\theta' |\mathcal{E}_\mathcal{S}|\ log(|\mathcal{V}_\mathcal{S}|))$, considering the complexity of Dijkstra's algorithm. Considering that $|\mathcal{E}_\mathcal{S}| < |X_\mathcal{S}||E_\mathcal{S}|$, that the event set is linear on the event sets of the individual agents, and that $\theta' < |X_\mathcal{S}|$, the worst case computational complexity of Algorithm 1 is bounded above by $O(|E_\mathcal{S}||X_\mathcal{S}|^2\ log(|X_\mathcal{S}|))$.

Algorithm 1 Create module chain $\mathcal{T}$.

```
Require: $\epsilon_0$-NFA ${}^{\epsilon_0}\mathcal{S}$, initial state $x_{0,\mathcal{S}}$, task specification $\gamma$, solution type ST ("Complete" or "Heuristic")
Ensure: Module chain $\mathcal{T}$
 1: Initialize $\mathcal{H}$ a zero matrix, $E$ an empty cell array, $\mathcal{T}$ an empty set
 2: $\mathcal{S} \longleftarrow \Delta({}^{\epsilon_0}\mathcal{S}, x_{0,\mathcal{S}})$
 3: if $proj(x_{0,\mathcal{S}}, b) := \varphi \wedge x_{0,\mathcal{S}} \in X_{m,\mathcal{S}}$ then
 4:     return $\mathcal{T}$
 5: else
 6:     for $i \in \{1, \dots, |X_\mathcal{S}|\}$ do
 7:         for $j \in \{1, \dots, |X_\mathcal{S}|\}$ do
 8:             if $\exists e \in E_\mathcal{S} : f_\mathcal{S}(x_i, e) = x_j$ then
 9:                 $\mathcal{H}[i][j] \longleftarrow g_\mathcal{S}(e)$
10:                 $E\{i\}\{j\} \longleftarrow e$
11:             end if
12:         end for
13:     end for
14:     if ST = "Complete" then
15:         [$\mathcal{P}$, cost] $\longleftarrow$ COMPLETE($\mathcal{H}, \mathcal{S}, x_{0,\mathcal{S}}, \gamma$)
16:     else
17:         [$\mathcal{P}$, cost] $\longleftarrow$ HEURISTIC($\mathcal{H}, \mathcal{S}$, E, $x_{0,\mathcal{S}}, \gamma$)
18:     end if
19:     for $i \in \{1, \dots, |\mathcal{P}| - 1\}$ do
20:         $\mathcal{T} \longleftarrow \mathcal{T} \cup \{\mathcal{P}_i,\ E\{\mathcal{P}_i\}\{\mathcal{P}_{i+1}\},\ \mathcal{P}_{i+1}\}$
21:     end for
22:     return $\mathcal{T}$, cost
23: end if
```

Algorithm 2 Complete Function.

```
 1: function COMPLETE($\mathcal{H}, \mathcal{S}, x_{0,\mathcal{S}}, \gamma$)
 2:     Initialize $cost_{min} \longleftarrow \infty$
 3:     for $i \in \{1, \dots, |X_\mathcal{S}|\}$ do
 4:         for $j \in \{1, \dots, |X_\mathcal{S}|\}$ do
 5:             if $proj(x_j, b) := \varphi \wedge x_j \in X_{m,\mathcal{S}}$ then
 6:                 [$\mathcal{P}$, cost] $\longleftarrow$ Dijkstra($\mathcal{H}, x_{0,\mathcal{S}}, x_j$)
 7:                 if $\mathcal{P} = \emptyset$ then
 8:                     return Task infeasible.
 9:                 end if
10:                 if cost $< cost_{min}$ then
11:                     $cost_{min} \longleftarrow$ cost
12:                     $\mathcal{P}_{optimal} \longleftarrow \mathcal{P}$
13:                 end if
14:             end if
15:         end for
16:     end for
17:     return $\mathcal{P}_{optimal}$, $cost_{min}$
18: end function
```

C. Analysis of a Heuristic Solution

In addition to the complete algorithm presented above, a heuristic approach is proposed in Algorithm 3 to reduce the computational time required to find a solution. By implementing Algorithm 1 with the proposed "Heuristic" solution type (line 19), reduces the complexity by only requiring a single execution of Dijkstra's algorithm, while sacrificing optimality and completeness. The computational time requirements of the proposed heuristic is $O(|\mathcal{E}_{\mathcal{S}}|\ log(|\mathcal{V}_{\mathcal{S}}|))$. Following similar arguments as in the complete solution, the worst case computational complexity of Algorithm 1 is bounded above by $O(|E_{\mathcal{S}}||X_{\mathcal{S}}|\ log(|X_{\mathcal{S}}|))$.

The intuition for utilizing the "Heuristic" solution type (Algorithm 3), is to avoid having the agents that are not participating in the task specification to end up in configurations different from their initial ones. Hence, we set the goal state x_d to be such that $proj(x_d, b) := \varphi$ and $proj(x_d, \bar{b}) = proj(x_{0,\mathcal{S}}, \bar{b})$ (Algorithm 3, line 3), where $\bar{b}$ denotes the bitwise negation. To discard unnecessary transitions and reduce the cost (recovering partial optimality), we track the solution to the minimum element k where $proj(\mathcal{P}_k, b) := \varphi$ and use the solution $\mathcal{P}^h = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ (Algorithm 3, line 9-15). While not possessing the completeness property, there are some cases where the proposed heuristic fully recovers optimality.

One such case is when b is only composed of ones, i.e. the full state vector of the goal state is specified. However, a classification of those cases and the conditions for feasibility of solutions is currently under consideration but is not part of the current work.

Algorithm 3 Heuristic Function.

```
1: function HEURISTIC($\mathcal{H}$, $\mathcal{S}$, E, $x_{0,\mathcal{S}}$, $\gamma$)
2:     Initialize $x_d$ to an empty set
3:     Find $x_d \in X_{m,\mathcal{S}} : proj(x_d, b) := \varphi \wedge proj(x_d, \bar{b}) = proj(x_{0,\mathcal{S}}, \bar{b})$
4:     Initialize $c^h \longleftarrow 0$
5:     [$\mathcal{P}$, cost] $\longleftarrow$ Dijkstra($\mathcal{H}, x_{0,\mathcal{S}}, x_d$)
6:     if $\mathcal{P} = \emptyset$ then
7:         return No path to $x_d$.
8:     end if
9:     for $k \in \{2, \ldots, |\mathcal{P}|\}$ do
10:        $c^h \longleftarrow c^h + g_{\mathcal{S}}(E\{\mathcal{P}_{k-1}\}\{\mathcal{P}_k\})$
11:        if $proj(\mathcal{P}_k, b) := \varphi$ then
12:            $\mathcal{P}^h \longleftarrow \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$
13:            return $\mathcal{P}^h$, $c^h$
14:        end if
15:    end for
16: end function
```

D. Completeness and Optimality

The principle of completeness asserts that the algorithm always returns a solution (if one exists), otherwise if there is no solution, the algorithm reports failure. We have the following results regarding the completeness of Algorithms 1, 2:

Proposition 1: For a system constructed based on individual and inter-agent capabilities, constraints and failure modes of the multi-agent system as presented in Section III, the resulting environment model of (6) is complete, in the sense that it represents all and only those state transitions that are dictated by the automata capturing the individual and inter-agent capabilities, constraints and failure modes.

Proof: To demonstrate the above claim we need to show that during the composition of ${}^{\epsilon_0}\mathcal{S}$, no valid transitions or states are being removed from the system and no new states or transitions are being introduced. In particular:

1. No states removed or added: This can be shown by observing that from Corollary 2 the ϵ_0-NFAs is closed under the operations of union, subtraction and concatenation. *Union operation* (Definition 5): This operates only on states defined in its arguments and the resulting states are the union of the argument's states that are part of agents' capabilities and constraints. *Subtraction operation* (Definition 6): This operation does not remove any states from the subtrahend argument. *Concatenation operation* (Definition 7): This operation creates the cross product state space of its argument. None of the operations introduces any state that does not already exist in their arguments.

2. No valid events are removed and no new transitions are introduced. *Union and Concatenation operations*: The resulting event set is the union of the operator argument events. No new events are introduced, that do not already exist in the arguments. *Subtraction operation*: The resulting event set includes only the events that are in the subtrahend and not in the subtractor's event set without introducing any new event. In particular failure mode events are removed from individual agent's capabilities with subtraction operation without affecting any other events. The event set of global constraints is removed from the global capabilities without affecting events that are not included in the global capabilities event set. Any event that is in the individual or inter-agent capabilities and is not in the failure modes, individual constraints or inter-agent constraints event sets will be included in the environment model. Any event that is in the failure modes event set (but not in the inter-agent capabilities) as well as any event in the individual or inter-agent constraint event set, will not appear in the environment model event set. Events that are not included in the individual and inter-agent capabilities will not appear in the environment model event set.

Hence all and only transitions that are valid will appear in the environment model's event set. ■

With the completeness property in place we can state the result about the optimality properties of the proposed system:

Proposition 2: For a system constructed based on individual and inter-agent capabilities, constraints and failure modes of the multi-agent system as presented in Section III, and assuming that only one agent is allowed to operate at any time instant, the solution of Algorithm 1 with the "Complete" solution type of Algorithm 2, produces the optimal sequence of actions that brings the system from any initial state to a state that satisfies the task specification γ while minimizing the cost of the task plan $\mathcal{T}$.

Proof: Since according to Proposition 1 the environment model is complete, then all (if any) optimal solutions are encoded in the transition system imposed by the transition function of the environment model. The module composition problem is an integer optimization problem and in our case maps the task planning problem to the shortest directed path - a graph search problem. The implemented Dijkstra's solution is both complete and is guaranteed to find an optimal solution if such a solution exists. ■

E. Class of Addressable Problems

To assess the expressiveness of the proposed framework we have the following result:

Corollary 3: If G is a DFA then it can be converted to the ϵ_0-NFA ${}^{\epsilon_0}G$ by introducing the state x_0 as the initial state of

${}^{\epsilon_0}G$ along with the associated ϵ transition to the starting state $x_{0,G}$ of G.

Proof: This can be shown by observing that the six-tuple obtained by the operation is as the one in Definition 1. ■

Based on Corollary 3, any DFA can be converted to ϵ_0-NFA. Furthermore, g_G that is associated to E_G renders the ϵ_0-NFAs to weighted automata. Hence, the proposed framework inherits the expressiveness of weighted automata.

Remark 1: The LTL-based and weighted automata-based frameworks are complementary [31] since even though they have an intersection, each one addresses problems that the other cannot address.

VI. Conclusions

A framework and a methodology for a novel multi-agent task planner is proposed. Given the capabilities, constraints and failure modes of the agents under the framework of NFAs with ϵ-transitions, the proposed framework produces optimal solutions, providing the sequence of tasks for transporting the state of the environment from any initial to a destination state satisfying the task specification. The developed algorithms can provide a complete solution with optimality guarantees whenever a solution exists. By relaxing the completeness property requirement, a significant reduction in the computational requirements is achieved, that provides sub-optimal solutions through the use of efficient heuristics.

Future research will focus on combining the proposed framework with supervisory control theory to enable reactive execution in dynamic environments, incorporating liveness properties, automatic system reconfiguration in the case of failures, enabling concurrent execution of modules, e.g. in a multi-port module setting, distributing task synthesis and improving the average computational complexity of both the pre-processing and planning phases.

References

[1] A. A. Tziola and S. G. Loizou, “Autonomous task planning for heterogeneous multi-agent systems,” *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3490–3496, 2023.
[2] ——, “Autonomous task planning for heterogeneous multi-agent systems,” *arXiv Preprint arXiv:2209.08611*, 2022.
[3] “Business process optimization (BPO) module,” https://github.com/rcdslabcut/mod.sw.bpo, accessed: November 2023.
[4] S. G. Loizou and K. J. Kyriakopoulos, “Automated planning of motion tasks for multi-robot systems,” in *IEEE International Conference on Decision and Control*, 2005, pp. 78–83.
[5] S. Tripakis, “Automated module composition,” in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2003, pp. 347–362.
[6] S. G. Loizou and E. D. Rimon, “Mobile robot navigation functions tuned by sensor readings in partially known environments,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3803–3810, 2022.
[7] S. G. Loizou, “The navigation transformation,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1516–1523, 2017.
[8] R. C. Hill and S. Lafortune, “Scaling formal synthesis of supervisory control for multiple robot systems,” in *IEEE ACC*, 2017, pp. 3840–3847.
[9] J. Goryca and R. C. Hill, “Formal synthesis of supervisory control software for multiple robot systems,” in *IEEE ACC*, 2013, pp. 125–131.
[10] G. A. Cardona and C.-I. Vasile, “Planning for heterogeneous teams of robots with temporal logic, capability, and resource constraints,” *Int. Journal of Robotics Research*, vol. 43, no. 13, pp. 2089–2111, 2024.
[11] S. Hustiu, C. Mahulea, M. Kloetzer, and J.-J. Lesage, “On multi-robot path planning based on Petri net models and LTL specifications,” *IEEE Transactions on Automatic Control*, pp. 1–8, 2024.
[12] Y. Kantaros and M. M. Zavlanos, “Sampling-based optimal control synthesis for multirobot systems under global temporal tasks,” *IEEE Transactions on Automatic Control*, vol. 64, no. 5, pp. 1916–1931, 2018.
[13] ——, “STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems,” *The International Journal of Robotics Research*, pp. 812–836, 39(7), 2020.
[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, “Where’s waldo? sensor-based temporal logic motion planning,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3116–3121.
[15] ——, “Temporal-logic-based reactive mission and motion planning,” *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
[16] 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 Eng.*, vol. 14(2), 797–808, 2016.
[17] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, “Optimal path planning for surveillance with temporal-logic constraints,” *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.
[18] N. Xu, T. Peng, D. Liu, and J. Li, “Temporal logic control synthesis for distributed multi-agent cooperative tasking,” *Journal of Physics: Conference Series*, vol. 2216, no. 1, p. 012061, 2022.
[19] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, “A formal approach to deployment of robotic teams in urban-like environment,” in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 313–327.
[20] M. Karimadini and H. Lin, “Guaranteed global performance through local coordinations,” *Automatica*, vol. 47, no. 5, pp. 890–898, 2011.
[21] J. Tumova and D. V. Dimarogonas, “Multi-agent planning under local LTL specifications and event-based synchronization,” *Automatica*, vol. 70, pp. 239–248, 2016.
[22] P. Schillinger, M. Bürger, and D. V. Dimarogonas, “Decomposition of finite ltl specifications for efficient multi-agent planning,” in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 253–267.
[23] H. Lin, “Mission accomplished: An introduction to formal methods in mobile robot motion planning and control,” *Unmanned Systems*, vol. 2, no. 02, pp. 201–216, 2014.
[24] P. Schillinger, M. Bürger, and D. V. Dimarogonas, “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems,” *The Int. J. of Robotics Res.*, vol. 37(7), 818–838, 2018.
[25] M. Kloetzer and C. Mahulea, “Multi-robot path planning for syntactically co-safe LTL specifications,” in *13th International Workshop on Discrete Event Systems (WODES)*. IEEE, 2016, pp. 452–458.
[26] X. Luo and M. M. Zavlanos, “Temporal logic task allocation in heterogeneous multirobot systems,” *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3602–3621, 2022.
[27] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, “Scalable and robust algorithms for task-based coordination from high-level specifications (ScRATCHeS),” *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2021.
[28] Z. Chen and Z. Kan, “Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications,” *The International Journal of Robotics Research*, 2024.
[29] S. Lafortune, “Discrete event systems: Modeling, observation, and control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 141–159, 2019.
[30] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.
[31] M. Lahijanian, S. B. Andersson, and C. Belta, “Formal verification and synthesis for discrete-time stochastic systems,” *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2031–2045, 2015.