Mission Planning for Multiple Vehicles with Temporal Specifications using UxAS

Luan V. Nguyen * **Bardh Hoxha** ** **Taylor T. Johnson** *** **Georgios Fainekos** ****

* *University of Texas at Arlington, Arlington, TX 76019, USA.*
** *Southern Illinois University, Carbondale, IL 62901, USA.*
*** *Vanderbilt University, Nashville, TN 37023, USA.*
**** *Arizona State University, Tempe, AZ 85287, USA.*

Abstract: In this paper, we present extensions to Unmanned Systems Autonomy Services (UxAS) to handle mission specifications that require a synchronization of task execution. UxAS uses Process Algebra (PA) as a formal language to specify mission requirements for unmanned aerial vehicle (UAV) operations. However, the current implementation of PA in UxAS utilizes *assigned* semantics which does not guarantee the order of task completion and is unable to provide a mission planning required vehicle-synchronization. To enable the capability of UxAS in operating synchronized mission specifications, we introduce a notion of Synchronized Process Algebra (SPA) which extends PA by adding a synchronized composition operator to the syntax of PA. Such an operator allows us to specify the task's duration and enforce the next task is executed after the previous one has terminated. Moreover, we provide a new service in UxAS, called Temporal Service (TS) to control the flow of the planning process with respect to timing specifications. We apply SPA and TS to specify and operate the mission specification of a forest fire rescue scenario required the synchronized arrivals of multiple UAVs.

Keywords: Motion Planning, Formal Specifications, UxAS, Process Algebra

1. INTRODUCTION

One of the fundamental problems in achieving vehicle autonomy is motion planning. Given a set of tasks, a single or multiple robots should be able to autonomously execute them with little to no human intervention. Due to its significance, this problem has been extensively studied in the past Ryan et al. (2004); LaValle (2006); Shima and Rasmussen (2009). Temporal logic such as Linear Temporal Logic (LTL) is one of the most common formalisms used as a specification language for the motion planing problem of single/multi-agent systems. Formal methods such as model checking and automata-based abstraction can be utilized to address motion planning and controller synthesis problems with respect to specifications given as temporal logic formulas. For example, the authors in Fainekos et al. (2009); Kloetzer and Belta (2007) convert LTL specifications to Büchi automata and create abstraction models for the dynamical systems. Then, they utilize automata-based methods to synthesize controllers for single-vehicle mission planning. In Ding et al. (2014), the authors propose a planning approach that enables receding horizon control for finite deterministic systems with respect to LTL formulas. On the other hand, the motion planning problem of multi-vehicle systems subject to both global LTL specifications for the vehicle team and local LTL specifications for each individual vehicle has been studied in Guo and Dimarogonas (2015); Bhatia et al. (2011); Karaman and Frazzoli (2008).

The computational complexity of the aforementioned methods may pose a challenge for real-world applications of motion planing with multiple vehicles and complex specifications. For instance, the model checking algorithm with respect to LTL specifications is PSPACE-Complete Schnoebelen (2002). The algorithm by Kabanza (1995), in the worst case, is exponential in terms of the size of the input goal and greatest action duration. Similarly, the computational costs of automata-based approaches are rapidly-growing regarding the length of a specification and the level of abstraction of a dynamical system.

Karaman et al. (2009) utilize Process Algebra (PA) as a specification language for unmanned aerial vehicle (UAV) mission planning. The proposed algorithm can guarantee to generate a feasible plan that satisfies a mission specification in polynomial time. This approach was implemented in the Unmanned Systems Autonomy Services (UxAS) framework [1] developed by the Air Force Research Laboratory (AFRL). UxAS is a collection of software modules that communicate on a shared communication channel, and generate task assignments and waypoint paths for one or more UAVs that satisfy a PA mission specification. Notwithstanding the PA language is expressive enough to define many mission specifications, its current implementation within UxAS does not support cooperative missions requiring task/vehicle-synchronization. In fact, the current

⋆ The material presented in this paper is based upon the work performed at the Summer of Innovation organized by AFRL and WBI at Dayton, Ohio, 2017. The authors sincerely appreciate Derek Kingston and Laura Humphrey for their help in understanding the UxAS system and AMASE simulator.

[1] The tutorial, project description and source codes of UxAS are available at: `https://github.com/afrl-rq/OpenUxAS`

framework utilizes *task-assignment semantics*, i.e., a task is taken care of if it is *assigned*. In other words, only the order of task assignment is enforced, not the order of task completion; and this has been done for the sake of finding an optimal plan for a mission composed of independent tasks. However, the current implementation of UxAS makes it difficult to handle either a mission which enforces completion of a part of the mission specification before beginning another, or one requires a synchronized coordination between multiple UAVs.

Contributions. In this work, we extend UxAS to enable more control over timing and task completion for mission specifications that require coordination between vehicles. Indeed, we extend the PA language with a synchronized operator which enforces completion of sub-tasks in UxAS. Based on that, we implement a new service within UxAS to handle a vehicle-synchronization as well as an execution order of complex tasks using multiple UAVs. We demonstrate our contributions with a forest fire rescue mission.

2. BACKGROUND

2.1 UxAS Framework

UxAS is a comprehensive software system architecture that enables autonomous capabilities on-board unmanned systems [1]. UxAS has been developed by the AFRL for more than a decade. The system provides a list of modular services that share a single, broadcast, channel of communication. All messages are specified as xml files. To accomplish a mission, UxAS takes the following inputs:

- Vehicle Configuration: describing operating speed, altitude, and maximum bank angle as well as the sensor configurations of a vehicle,
- Vehicle State: specifying the state information of the vehicles such as position, actual speed, current waypoint, current active task, and the state of the sensors,
- Task Configuration: including task's ID, eligible vehicle assigned to the task and all requirements of the task that need to be satisfied,
- Automation Request: a script initializes UxAS and requests a set of tasks to be completed by a set of eligible vehicles in a particular airspace configuration, and specifies a relationship between tasks defined in a PA string which needs to be satisfied.

The *plan builder* service in UxAS calculates a cost matrix and constructs different mission plans for all of the task assignments. The final plan will then be published in an automation response. UxAS provides a set of complex, automated search tasks that handle common search and surveillance patterns such as point search, line search, area search, pattern search, etc. Each automated task is constructed based on its specific waypoint-based path planning and sensor steering handling that can be customized by changing its parameter values such as a sensor view angle, standoff distance, particular angle of approach (for details see Kingston et al. (2016)).

2.2 Process Algebra

UxAS uses Process Algebra (PA) as a formal language to specify mission requirements for UAV operations Baeten (2004); Fokkink (2013). One of the advantages of PA as a mission specification language is that it allows us to efficiently construct a feasible (and potentially optimal) plan that meets the mission specification in polynomial time Karaman et al. (2009).

Syntax of PA. Let A be a finite set of actions and $\mathcal{P}$ be a set of PA terms defined over A. The syntax of PA is then inductively defined as follows,

- each action $\alpha \in \mathcal{A}$ is in $\mathcal{P}$,
- for every $p, p' \in \mathcal{P}$, we also have $p + p' \in \mathcal{P}$, $p \parallel p' \in \mathcal{P}$, and $p \cdot p' \in \mathcal{P}$.

The operators $(+)$, $(\parallel)$, $(\cdot)$ respectively indicate the *alternative*, *parallel*, and *sequential* relationships between two PA terms p and p'. Intuitively, the composition $p + p'$ specifies that it can either execute a behavior of p or p'. The composition $p \parallel p'$ specifies that both PA terms should execute at the same time. The composition $p \cdot p'$ will first execute p, and then execute p' right after p terminates.

Semantics of PA. A transition $p \xrightarrow{\alpha} p'$ represents that a process p evolves to a process p' by executing α. A special process, denoted as $\triangle$, represents the terminated process that has no action to execute and cannot evolve to any other process. Also, each action can be considered as a PA term; a process p corresponds to a term $\alpha \in \mathcal{A}$ such as $p = \alpha$, can execute α and then terminate which is denoted as $\alpha \xrightarrow{\alpha} \triangle$. A sequence of actions $\lambda = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ is called a trace of a PA term p_0 if there exists a set of processes $p_1, p_2, ..., p_n \in \mathcal{P}$ such that $p_0 \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} p_n$, where $p_n = \triangle$. Intuitively, a trace of a process represents a behavior of the process specified over a sequence of actions leading to termination. We denote Λ_{p_0} as a set of all traces that a process p_0 can execute.

Following, given $\alpha \in \mathcal{A}$ and $p_1, p_1', p_2, p_2' \in \mathcal{P}$, the operational semantics of PA can be defined as follows,

$$\frac{p_1 \xrightarrow{\alpha} \triangle}{p_1 + p_2 \xrightarrow{\alpha} \triangle} \quad \frac{p_1 \xrightarrow{\alpha} p_1'}{p_1 + p_2 \xrightarrow{\alpha} p_1'} \quad \frac{p_2 \xrightarrow{\alpha} \triangle}{p_1 + p_2 \xrightarrow{\alpha} \triangle} \quad \frac{p_2 \xrightarrow{\alpha} p_2'}{p_1 + p_2 \xrightarrow{\alpha} p_2'}$$

$$\frac{p_1 \xrightarrow{\alpha} \triangle}{p_1 \parallel p_2 \xrightarrow{\alpha} p_2} \quad \frac{p_1 \xrightarrow{\alpha} p_1'}{p_1 \parallel p_2 \xrightarrow{\alpha} p_1' \parallel p_2} \quad \frac{p_2 \xrightarrow{\alpha} \triangle}{p_1 \parallel p_2 \xrightarrow{\alpha} p_1} \quad \frac{p_2 \xrightarrow{\alpha} p_2'}{p_1 \parallel p_2 \xrightarrow{\alpha} p_1 \parallel p_2'}$$

$$\frac{p_1 \xrightarrow{\alpha} \triangle}{p_1 \cdot p_2 \xrightarrow{\alpha} p_2} \quad \frac{p_1 \xrightarrow{\alpha} p_1'}{p_1 \cdot p_2 \xrightarrow{\alpha} p_1' \cdot p_2}.$$

For each transition rule, the numerator specifies a set of premises which is a set of possible transitions of a PA term; and the denominator describes a conclusion which is a transition allowed to execute by the above premises.

3. PROCESS ALGEBRA SPECIFICATION IN UXAS

In this section, we review how PA is utilized to specify mission specifications for multiple UAVs within UxAS. Mission specifications are sent to UxAS through an automation request. This automation request includes the relationship between multiple tasks in which each of them is a combination of different *atomic objectives*.

Definition 3.1. An atomic objective is a tuple $\theta \triangleq \{L_\theta^s, L_\theta^e, \mathcal{T}_\theta^e, v_\theta, \mathcal{I}_{v_\theta}, \mathcal{F}_{v_\theta}\}$ such that:

- $L_\theta^s \in \mathbb{R}^2$ is the starting point where a vehicle has to reach in order to start a task execution,
- $L_\theta^e \in \mathbb{R}^2$ is the exit point which represents the location that a vehicle should reach after executing a task,

- $\mathcal{T}^e_\theta \in \mathbb{R}_{\geq 0}$ is the total execution time,
- $v_\theta \in \mathcal{V}$ is the eligible vehicle to accomplish a task,
- $\mathcal{I}_{v_\theta}, \mathcal{F}_{v_\theta} \in \mathcal{Q}_\mathcal{V}$ is the vehicle's configuration and state before and after executing a task, respectively,

where $\mathcal{Q}_\mathcal{V}$ is the set of all configurations and states of $\mathcal{V}$. Examples of atomic objectives in UxAS include point search, area search, classification, attack, object tracking, etc. (see Rasmussen and Kingston (2008)). Moreover, UxAS allows a mission specification that involves logical and temporal constraints to be specified as a combination of individual tasks, where each task is represented by a PA term defined on a set of atomic objectives.

Example 3.2. Consider the following scenario. An avalanche has occurred and we need to search for missing hikers. There are two particular points of interest A, B and an area C where there is a high probability of finding the missing hikers. A set of three UAVs $\{v_1, v_2, v_3\}$ are available to execute the mission in an order such that both of A and B locations can be searched at the same time before doing an area search at C. Also, either v_1 or v_2 can go to A, only v_3 can go to B, and all of UAVs can travel to C. Then, a process algebra string Φ that represents a high-level specification of the mission can be expressed as: $\Phi = ((\theta_{A*v_1} + \theta_{A*v_2}) \parallel \theta_{B*v_3}) \cdot (\theta_{C*v_1} + \theta_{C*v_2} + \theta_{C*v_3})$, where θ_{A*v_1} denotes the atomic objective of searching the location A using v_1 and the rest of atomic objectives can be interpreted in the same fashion.

Remark 3.3. The PA semantics currently implemented in UxAS is *assigned* semantics. Such semantics does not guarantee the order of task completion and is unable to provide a mission planning required vehicle-synchronization. In other words, a task starts immediately whenever there is an eligible vehicle that is ready to execute the task. As an example, the semantic rule $\frac{p_1 \xrightarrow{\alpha} \triangle}{p_1 \cdot p_2 \xrightarrow{\alpha} p_2}$ cannot be applied as p_2 may start before the termination of p_1. This implementation is for the sake of achieving an optimal plan for a mission comprised of independent tasks. In the previous example, there is always a case such as the area search at C is executed before either the point search tasks at A or B are completed. This will be an issue if we strictly want each task to execute following the order specified by a PA string. Moreover, because this semantics does not ensure that a task can be executed in a synchronized way, it limits the capability of UxAS in providing cooperative task planning for multi-UAVs systems. For instance, suppose that there is a heavy snowfall at the area C that significantly reduces the quality of an image captured by a camera attached on one UAV, it is essential that all UAVs can synchronize at C, and then execute the area search together to acquire an adequately high-resolution image. As a result, it motivates the work of applying synchonization and adding a new service into UxAS to handle a vehicle-synchronization as well as an execution order for complex tasks using multiple UAVs.

3.1 Optimal Planning for Non-synchonized Process Algebra Specifications in UxAS

Next, we review the algorithm used to find a feasible (may be optimal for some extant) planning for non-synchronized mission specifications by Karaman et al. (2009), which provides the whole basis for our extensions to UxAS presented in Section 4 to handle mission specifications that requires the synchronized arrivals of multiple UAVs.

Definition 3.4. Let Θ be a set of atomic objectives and $\mathcal{V}$ be a list of UAVs, a *single schedule* s_v of a UAV $v \in \mathcal{V}$ is a sequence of atomic objective and instance of execution time pairs, written as $s_v = ((\theta_1, t_{\theta_1}), ..., (\theta_k, t_{\theta_k}))$ such that

- $\theta_i \in \Theta$, $t_{\theta_i} \in \mathbb{R}_{\geq 0}$ for $i \in \{1, 2, \ldots, k\}$, and
- $t_{\theta_{i+1}} - t_{\theta_i} \geq \mathcal{T}^e_{\theta_i} + \mathcal{T}^t_{\theta_{i+1}}$ for $i \in \{1, 2, \ldots, k-1\}$,

where $\mathcal{T}^t_{\theta_{i+1}}$ denotes a time for a UAV v travels from the exit point of θ_i to the starting point of θ_{i+1}. From the above definition, an atomic objective $\theta_i \in \Theta$ is scheduled to be executed at time instance $t_i \in \mathbb{R}_{\geq 0}$ by vehicle v if $(\theta_i, t_{\theta_i}) \in s_v$. Also, we note that $t_{\theta_{i+1}}$ must be computed and chosen so that a UAV has enough time to complete the previous atomic objective θ_i and travel to the starting point of θ_{i+1}. Each single schedule s_v can be accomplished with a completion time $\tau^c_{s_v} = t_{\theta_k} + \mathcal{T}^e_{\theta_k}$, where θ_k is the last atomic objective assigned to v.

We call ζ as a *complete schedule*, which is a set of single vehicle schedules that contains exactly one single vehicle schedule for each UAV. We can associate ζ with different types of cost functions to design some optimal mission planning. One instance of that is to find the minimum value of the total completion time of all UAVs:

$$\Omega(\zeta) = \min \sum_{v \in \mathcal{V}} \tau^c_{s_v} \tag{1}$$

Definition 3.5. An *observation* δ of a complete schedule ζ is a sequence of atomic objectives, i.e., $\delta = (\theta_1, \theta_2, ..., \theta_k)$ which satisfies the following conditions,

- $\forall \theta_i \in \delta$, θ_i is scheduled in ζ,
- $\exists \bar{t_i} \in \mathbb{R}_{\geq 0}$ so that $t_{\theta_i} \leq \bar{t_i} \leq t_{\theta_i} + \mathcal{T}^e_{\theta_i}$,
- $\bar{t_i} < \bar{t_j}$ if θ_i precedes θ_j in δ,

where $\bar{t_i}$, $\bar{t_j}$ are time instances associated with θ_i and θ_j, respectively. We note that the observation of a complete schedule may not be unique; and for all atomic objective θ_i in δ, there exists one or more time instance $\bar{t_i}$ within the execution interval of θ_i such that the ordering of those time instances is the same as that of their corresponding atomic objectives in δ. We denote the set of all observations of a given complete schedule ζ as Π_ζ.

Problem 3.6. Given a set of atomic objectives Θ, a list of UAVs $\mathcal{V}$, the problem of finding an optimal planning that satisfies a PA specification $\hat{p}$ is to find a complete schedule ζ such that:

- any observation δ of ζ is a trace of $\hat{p}$, *i.e.*, $\Pi_\zeta \subseteq \Lambda_{\hat{p}}$,
- the cost function $\Omega(\zeta)$ is optimized.

Branch-and-bound Algorithm. To solve the optimal planning for PA specifications using multiple UAVs, UxAS exploits the branch-and-bound tree search algorithm proposed by Rasmussen and Shima (2006). Briefly, in this algorithm, a PA specification $\hat{p}$ is constructed as a tree structure where each node represents a pair of atomic objective and its associated UAV. Each node of the tree except *leaf* nodes may contain one or more *child* nodes that branch the tree into the more specific sub-trees. A path of the tree which represents a possible assignment

(i.e., a trace of $\hat{p}$) is a sequence of connected nodes starting from the *trunk* node to a leaf node. Hence, the two-phases algorithm incorporating a greedy, best-first search heuristic with branch-and-bound mechanism can be used to efficiently determine the optimal path.

The search is started from the trunk node of the tree. For each node, the shortest Euclidean distances between it and the set of atomic objectives associated with different UAVs that can be possibly assigned in the next layer of the tree are calculated. The best node in the current layer of the tree which yields the smallest distance, i.e., the minimum time complete schedule will be chosen. The process repeats until it reaches a leaf node representing a *candidate optimal plan*. Such a solution will be served as an upper bound of the cost of a PA specification and used to perform a branch and bound evaluation. The best-first search procedure is then reversibly applied starting from the leaf node associated with the candidate optimal solution to obtain a lower bound on the cost of the same specification using a different UAV in the group. These upper and lower bounding search procedures iteratively traverse upwards and downwards of the tree to prune out all infeasible solutions and improve the candidate solution. The algorithm is terminated with the actual optimal solution, i.e., the most efficient task ordering after all nodes of the tree have either been pruned or evaluated Rasmussen and Shima (2006).

The main advantage of using this branch-and-bound algorithm is that it can provide a feasible plan within a polynomial time bound corresponding to the length of a PA specification. However, the current implementation of the algorithm in UxAS does not support a specification that requires vehicle-synchronization. To acquire a feasible plan for a synchronized mission, we must take account of a loitering time required for each UAV to be synchronized with others before executing the task. As a result, the planning algorithm should be computed with respect to the constraint such that the assigned UAV has enough time to complete its previous atomic objective, fly to the entry point of the next one, and wait there until other UAVs are synchronized. Thus, adding the capability to determine a feasible plan with respect to the synchronized arrivals of multiple UAVs is a contribution of this paper.

4. SYNCHRONIZED PROCESS ALGEBRA SPECIFICATION IN UXAS

In this section, we present an extension of PA specification language and a new service to enable the definition of more complex tasks with respect to timing.

4.1 Synchronized Process Algebra

We introduce a notion of Synchronized Process Algebra (SPA), which is an extension of PA to specify the duration of a task and enforce that the next task is executed after the previous task has terminated. The general idea of SPA is to enable a more expressive language to define synchronized mission specifications. In fact, SPA is extend from PA with a synchronization functionality, which defines the point where the vehicles should synchronize, i.e., complete all their previous tasks so that they can initialize the next task together. We define a new sequential composition operator $\mathcal{S}$ (besides the parallel composition) in SPA, which enforces synchronous execution of the parallel processes. The semantic rule of $\mathcal{S}$ is defined as follows,

$$\frac{p_1 \xrightarrow{\alpha} p_1' \quad p_2 \xrightarrow{\alpha} p_2'}{\mathcal{S}(p_1 \parallel p_2) \xrightarrow{\alpha} p_1' \parallel p_2'},$$

where α is a synchronized action. Also, the PA string $p_1 \mathcal{S} p_2$ in UxAS enforces the sequential relationship between p_1 and p_2 specified in Section 2.2, which means p_1 should completely terminate before p_2 can start. We note that in UxAS, beside $\mathcal{S}$ operator, we can use a notation of $\mathcal{S}_\delta$ to specify a synchronized mission which all of its parallel atomic objectives will execute simultaneously with a delay time δ. Such a notation allows us to modify task duration, enables more control over timing and task completion.

Consider a mission specification similar to Example 3.2 which requires all of the UAVs to be synchronized at C before executing the area search together, such a specification can be written as: $\Phi = ((\theta_{A*v_1} + \theta_{A*v_2}) \parallel \theta_{B*v_3}) \mathcal{S} (\theta_{C*v_1} \parallel \theta_{C*v_2} \parallel \theta_{C*v_3})$. Here, the operator $\mathcal{S}$ enforces that both the point search tasks at A and B are accomplished completely before the next tasks are initialized. Then, the area search tasks at C with all vehicles v_1, v_2, and v_3 are executed in a synchronous manner. Next, we will present how to compute a loitering time for each UAV when executing a synchronized task.

4.2 Synchronized Process Algebra Specification Planning

Definition 4.1. (Vehicle Loitering Action). A vehicle loitering action of v_θ is an atomic objective θ such that $L^s_\theta \equiv L^e_\theta \wedge \mathcal{I}_{v_\theta} \equiv \mathcal{F}_{v_\theta}$.

Intuitively, a vehicle loitering action is a maneuver added to the flightpath of a vehicle in order to enable more complex mission specifications that require synchronization. Here, the starting point where a vehicle has to come to start a loitering task and the exit point where it should reach after executing a loitering task must be the same. Moreover, the vehicle's state and configuration are not changed before and after the loitering task execution.

Theorem 4.2. Given a set of eligible vehicles $\mathcal{V}$ that can execute ϕ_1 and ϕ_2, if there exists a feasible plan for ϕ_1 with the vehicle's state and configuration before and after executing ϕ_1 given as $\{\mathcal{I}^{\phi_1}_{\mathcal{V}}, \mathcal{F}^{\phi_1}_{\mathcal{V}}\}$ and for ϕ_2 given as $\{\mathcal{I}^{\phi_2}_{\mathcal{V}}, \mathcal{F}^{\phi_2}_{\mathcal{V}}\}$, where $\mathcal{F}^{\phi_1}_{\mathcal{V}} \equiv \mathcal{I}^{\phi_2}_{\mathcal{V}}$, then there exists a feasible plan for $\phi_1 \mathcal{S} \phi_2$ with respect to $\{\mathcal{I}^{\phi_1}_{\mathcal{V}}, \mathcal{F}^{\phi_2}_{\mathcal{V}}\}$.

Proof. The correctness of the above theorem is clear from the semantics of PA. Let p_1, p_2 and p_s be PA terms representing ϕ_1, ϕ_2 and loitering behavior defined by an operator $\mathcal{S}$. Since p_1 is feasible, p_1 can always evolve to p_s after executing some action α_1, i.e., a transition $p_1 \cdot p_s \xrightarrow{\alpha_1} p_s$ is valid. Moreover, because p_2 is feasible and executing p_s does not change the initial configurations and states of UAVs for executing p_2, so the process $p_s \cdot p_2$ can terminate. As a result, the process $p_1 \cdot p_s \cdot p_2$ can also terminate, i.e., $\phi_1 S \phi_2$ is feasible.

Given a mission specification Φ expressed as a sequence of sub-tasks in which some of them require vehicle-synchronization and a list of UAVs, the procedure that computes the loitering time for each UAV before executing a synchronous task together and returns a feasible plan

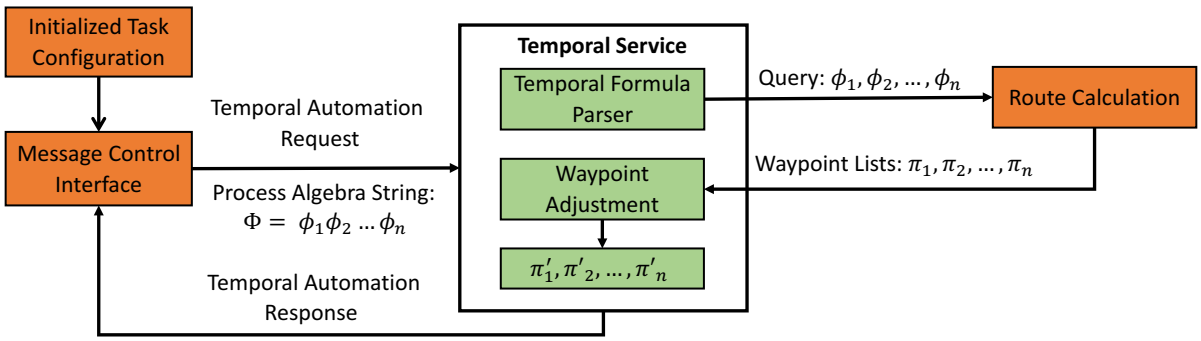

Fig. 1. An overview of the Temporal Service

Algorithm 1 Synchronized Process Algebra Planning

Input: a synchronized PA string Φ, a list of UAVs $\mathcal{V}$,
Output: a feasible planning with a completion time $\Psi(\Phi)$

procedure COMPUTED A SYNCHRONIZED MISSION
  $\Psi(\Phi) \leftarrow 0$
  $\{\phi_1, \phi_2, \ldots, \phi_n\} \leftarrow$ **Partition**(Φ)
  for all ϕ_i **do**
    $\Omega(\zeta_{\phi_i}) \leftarrow$ **Branch-and-bound**(ϕ_i)
    if $i < n \wedge \phi_{i+1}$ requires synchronization **then**
      Determine a set of eligible UAVs $V_{\phi_{i+1}} \subseteq \mathcal{V}$ for executing a synchronized task ϕ_{i+1}
      Compute a loitering times $\mathcal{T}^k_{\phi_{i+1}}$ for each UAV $v_k \in V_{\phi_{i+1}}$
    else
      $\mathcal{T}^k_{\phi_{i+1}} \leftarrow 0$
    end if
    $\Psi(\Phi) \leftarrow \Psi(\Phi) + \Omega(\zeta_{\phi_i}) + \max_{v_k \in V_{\phi_{i+1}}} \mathcal{T}^k_{\phi_{i+1}} + \delta_{i+1}$
  end for
  return $\Psi(\Phi)$
end procedure

associated with Φ is shown in Algorithm 1, and is further interpreted as follows.

- First, we partition a PA string ϕ into a collection PA sub-strings with respect to synchronization. For instance, a mission specification Φ in Example 3.2 can be truncated into two sub-strings $\phi_1 = (\theta_{A*v_1} + \theta_{A*v_2}) \parallel \theta_{B*v_3}$ and $\phi_2 = \theta_{C*v_1} \parallel \theta_{C*v_2} \parallel \theta_{C*v_3}$.
- For each sub-string ϕ_i, we utilize a branch-and-bound algorithm to calculate the minimum value of the total completion time for ϕ_i. If the next task ϕ_{i+1} requires synchronization, we determine the loitering time for each UAV to wait until all required UAVs are synchronized to execute ϕ_{i+1}. Such computation is based the distance between the current position and the synchronized location, and the actual speed of each UAV.
- Finally, we return a feasible plan with the completion time inductively accumulated by the sum of three quantities, the completion time computed by the branch-and-bound algorithm, the maximum loitering times, and the given delays.

4.3 Temporal Service

To implement the synchronization functionality, we added a new service, called *Temporal Service*, into UxAS to control the flow of the planning process with respect to timing specifications. The Temporal Service (TS) includes two sub-services: the *Temporal Formula Parser* (TFP) and the *Waypoint Adjustment* (WPA). The core service description of TS is shown in Figure 1, and further interpreted as follows.

(1) The TS receives a temporal automation request that includes a list of all tasks and their timing relationships specified by a temporal formula Φ.
(2) Next, the TFP built inside the TS will truncate Φ into a collection of sub-formulas ϕ_i. Each of them will be sent to the existing Route Calculation service in UxAS to retrieve a corresponding waypoint list π_i that will be fed into the WPA later on. Here, π_i is determined based on i) the task completion time of ϕ_i computed by the branch-and-bound algorithm implemented in the RC service, and ii) the configurations and states of UAVs that are eligible to execute ϕ_i.
(3) Depending on the target where the vehicles should synchronize before executing a next task, WPA will first calculate a loitering time required for each vehicle, and then generate new waypoint lists $\pi'_1, \pi'_2, ..., \pi'_n$. Each waypoint list can be represented as $\pi'_i = \pi_i | \ell_i$, where $(|)$ is a list concatenation operator and ℓ_i is a waypoint list that defines a loitering behavior of each vehicle that should wait.
(4) Finally, the TS will publish a temporal automation response including synchronized waypoint lists with respect to the original automation request.

In this paper's scope, each sub-formula ϕ_i represents a PA term. However, we note that TS was not only designed to work with PA, but also set the groundwork for future implementations of temporal specification language such as LTL, Metric Temporal Logic (MTL), etc.

5. CASE STUDY: FOREST FIRE RESCUE MISSION

In this section, we demonstrate our extensions to UxAS in specifying and operating the mission specification of a forest fire scenario involving multiple UAVs[2]. Consider a forest fire rescue mission shown in Figure 2. There are four vehicles (V_1, V_2, V_3, and V_4) and two water supply locations (W_1 and W_2). To stop a fire, the vehicles need to pump the water at W_1 and W_2, synchronize at A, B, C, D, and then simultaneously disperse water in a parallel formation. The mission specification of this scenario is expressed

[2] The proposed extensions as well as the case study presented in this paper are implemented under the mission planning branch of UxAS: `https://github.com/afrl-rq/OpenUxAS/tree/missionplanning`.

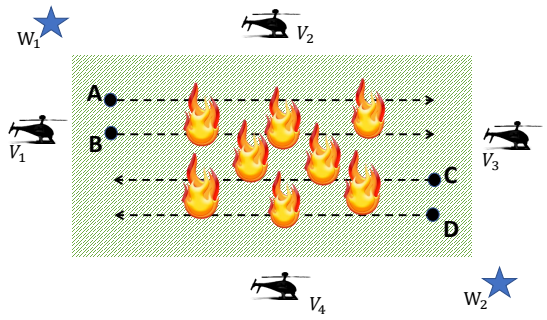

Fig. 2. Forest Fire Rescue Scenario

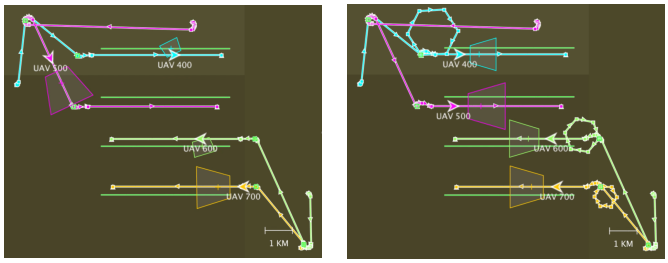

Fig. 3. Simulation of a forest fire rescue mission. Left: without synchronization, Right: with synchronization

as: $\Phi = (\theta_{W_1 * V_1} \parallel \theta_{W_1 * V_2} \parallel \theta_{W_2 * V_3} \parallel \theta_{W_2 * V_4}) \mathcal{S} (\theta_{A * V_1} \parallel \theta_{B * V_2} \parallel \theta_{C * V_3} \parallel \theta_{D * V_4})$. Figure 3 shows the simulations of the mission without and with vehicle-synchronization using AMASE[3], where UAVs 400, 500, 600, 700 correspondingly represent for V_1, V_2, V_3 and V_4. A loitering path is defined as a hexagon in which the radius is proportional to a waiting time for a vehicle-synchronization. In case with the UAV-synchronization, UAV 400 is the first vehicle that arrives at the synchronized point. As a result, it has a longest waiting time illustrated via the biggest hexagon. UAV 500 does not have a loitering behavior as it is the last vehicle arriving at the synchronized point.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed extensions to UxAS to enable a mission planning that requires vehicle-synchronization as well as enforces the order of task completion. We introduced a notion of SPA to define synchronized mission specifications and added a new service into UxAS to control the flow of the planning process with respect to timing specifications. We demonstrated our extensions to UxAS through the case study of a forest fire rescue scenario that requires the synchronization of multiple UAVs.

Future Work. We plan to extend the TS not only to support vehicle-synchronization using SPA but also to enable mission planning for multiple vehicles with timing constraints specified by temporal logic formulae. Consider the MTL specification "every time a vehicle enters region R, then within 20 seconds v_1 will execute a surveillance task θ", written as $G_{[0,100]}(R \implies F_{[0,20]}\theta_{R-v_1})$. Such a specification can be described in an automation request, and the TS will correspondingly generate waypoint paths satisfying the specification. In real missions, a UAV will not be executing their flight plans within the pre-computed flight times. Thus, we also plan to enable online updates on mission goals during plan execution. For example, we can apply a requirement rewriting technique proposed in Roşu and Havelund (2005) to generate new plans while ensuring that the old ones will be satisfied.

[3] AMASE is simulation-based framework developed the Aerospace Vehicles Technology Assessment & Simulation Branch of AFRL. AMASE can display a mission planning with simulated objects, waypoints path, communication channels, etc. AMASE is available at: `https://github.com/afrl-rq/OpenAMASE`.

REFERENCES

Baeten, J.C. (2004). *Applications of process algebra*, volume 17. Cambridge university press.

Bhatia, A., Maly, M.R., Kavraki, L.E., and Vardi, M.Y. (2011). Motion planning with complex goals. *IEEE Robotics & Automation Magazine*, 18(3), 55–64.

Ding, X., Lazar, M., and Belta, C. (2014). Ltl receding horizon control for finite deterministic systems. *Automatica*, 50(2), 399–408.

Fainekos, G., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343–352.

Fokkink, W. (2013). *Introduction to process algebra*. Springer Science & Business Media.

Guo, M. and Dimarogonas, D.V. (2015). Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2), 218–235.

Kabanza, F. (1995). Synchronizing multiagent plans using temporal logic specifications. In V. Lesser (ed.), *Proceedings of the First International Conference on Multi–Agent Systems*, 217–224. MIT Press, San Francisco, CA.

Karaman, S. and Frazzoli, E. (2008). Complex mission optimization for multiple-uavs using linear temporal logic. In *American Control Conference, 2008*, 2003–2009. IEEE.

Karaman, S., Rasmussen, S., Kingston, D., and Frazzoli, E. (2009). Specification and planning of uav missions: a process algebra approach. In *American Control Conference, 2009. ACC'09.*, 1442–1447. IEEE.

Kingston, D., Rasmussen, S., and Humphrey, L. (2016). Automated uav tasks for search and surveillance. In *Control Applications (CCA), 2016 IEEE Conference on*, 1–8. IEEE.

Kloetzer, M. and Belta, C. (2007). Temporal logic planning and control of robotic swarms by hierarchical abstractions. *Robotics, IEEE Transactions on*, 23(2), 320–330.

LaValle, S.M. (2006). *Planning algorithms*. Cambridge university press.

Rasmussen, S.J. and Kingston, D. (2008). Assignment of heterogeneous tasks to a set of heterogenous unmanned aerial vehicles. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, volume 1447.

Rasmussen, S.J. and Shima, T. (2006). Branch and bound tree search for assigning cooperating uavs to multiple tasks. In *American Control Conference, 2006*, 6–pp. IEEE.

Roşu, G. and Havelund, K. (2005). Rewriting-based techniques for runtime verification. *Automated Software Engineering*, 12(2), 151–197.

Ryan, A., Zennaro, M., Howell, A., Sengupta, R., and Hedrick, J.K. (2004). An overview of emerging results in cooperative uav control. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, 602–607. IEEE.

Schnoebelen, P. (2002). The complexity of temporal logic model checking. *Advances in modal logic*, 4(393-436), 35.

Shima, T. and Rasmussen, S. (2009). *UAV cooperative decision and control: challenges and practical approaches*. SIAM.