Specification and Planning of UAV Missions: A Process Algebra Approach

2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Specification and Planning of UAV Missions: A Process Algebra Approach

Sertac Karaman Steven Rasmussen Derek Kingston Emilio Frazzoli

Abstract— **Formal languages have recently come under attention as a powerful tool to describe in a precise and rigorous way mission specifications (i.e., mission objectives and constraints) for robotic systems, and to design planning and control algorithms that provably achieve the specifications. In this paper, we consider Process Algebras as a mission specification language for teams of UAVs. The main advantage of the proposed approach is that Process Algebra specifications are amenable to efficient planning algorithms. In particular, a tree-search algorithm is presented that computes a feasible plan in polynomial time. Branch-and-bound techniques are used to compute an optimal plan if computation time is available. Several mission specification examples are presented, and results from a numerical simulation are discussed.**

I. INTRODUCTION

Unmanned Vehicles constitute an emerging technology to be used in civilian and military missions in several domains, where the tasks that need to be handled are dull, dangerous, complex, or even impossible for humans to do. Within the last decade, there has been a considerable amount of research on designing algorithms for high-level reasoning and coordination of multiple Unmanned Aerial Vehicles (UAVs) [1]–[3].

This work focuses on augmenting cooperative control algorithms with a formal language to state mission objectives and constraints (the *mission specifications*), using which human operators can interface effectively with multiple UAVs to execute complex tasks. Recently, temporal logics were utilized for automated controller design from complex specifications [4], [5]. Mainly inspired by this literature, formal languages such as Linear Temporal Logic (LTL) and Metric Temporal Logic (MTL) had been considered for specification purposes in optimal cooperative control, and Mixed Integer Linear Programming (MILP) based algorithms were utilized to generate optimal plans [6], [7].

While these languages are very expressive, they generally require computationally intensive algorithms, especially when utilized for optimal planning purposes. In fact, the model checking problem for, e.g., LTL, is PSPACE-complete [8]. Moreover, MILP-based optimal planning algorithms generally run in exponential time and are not guaranteed to return even a feasible solution within a polynomial time bound. On the other hand, one can consider languages that, at the cost of a more limited expressive power, allow efficient algorithms for, e.g., model checking. For example, in [9], a fragment of LTL was used for specification of temporal properties in reactive robotic motion planning problems, leading to polynomial-time algorithms. In a similar spirit, in this paper we will consider formal languages that may not be as expressive as LTL, but are amenable to efficient planning algorithms, while preserving the ability to describe a broad class of mission specifications of practical interest.

In this paper, we consider Process Algebra (PA) [10]–[12] as a language to describe a class of mission specifications, applicable to UAV operations. Process algebra was first proposed to reason about concurrent software and prove their correctness properties (see also [13] and the references therein), and successively used in several application contexts (see, for example, [14]–[16]). However, to our knowledge, this paper constitutes the first application in UAV cooperative control problems. The main advantage in using PA as a mission specification language is that a feasible plan (i.e., a plan that meets the mission specification) can be constructed efficiently, in polynomial time. In this paper, we extend the *tree search* algorithm in [17] to handle PA specifications.

The contributions of this paper are as follows. Firstly, we propose process algebras for the specification of complex tasks in UAV operations. We also provide examples to motivate the specification language in this application domain. Secondly, we provide a computationally effective algorithm for complex UAV operations specified via process algebra. The computational effectiveness of the algorithm stems from the fact that it returns a feasible plan in polynomial time and is guaranteed to return the optimal solution in finite time.

The rest of the paper is organized as follows. Section II introduces the basics of process algebra. In Section III, the problem definition is given and in Section IV the tree search algorithm is extended to handle complex mission specifications that are expressed in PA. A numerical example is solved in Section V and concluding remarks are given in Section VI.

S. Karaman is with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA sertac@mit.edu

S. Rasmussen is with Miami Valley Aerospace LLC at the Control Science Center of Excellence, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA Steven.Rasmussen@wpafb.af.mil

D. Kingston is with the Control Science Center of Excellence, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA Derek.Kingston@wpafb.af.mil

E. Frazzoli is with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA frazzoli@mit.edu

978-1-4244-4524-0/09/$25.00 ©2009 AACC

II. PROCESS ALGEBRA

Process Algebra is used in computer science to study the *behavior* of software. In this context, the atomic building blocks of behavior, called *action*s, are defined as discrete events which happen instantaneously in time, and represent the output of the software. A behavior is a sequence of actions which have been observed from the software over the course of time.

Within the context of UAV cooperative control, the actions will refer to atomic individual tasks such as target classification, sector search, or reconnaissance, performed by exactly one UAV. Any mission plan will refer to a temporal ordering of actions, which resembles a behavior. Each process will encode a mission specification, and a plan will be called feasible only if the temporal ordering of the actions is one of the behaviors that this process can exhibit. In order to make the paper self contained, we summarize the description of process algebras given in [12] in this section.

A. Syntax of Process Algebra

An important element of the *Process Algebra* (PA) is the *set of terms*, which is formally defined as follows:

Definition II.1 (Terms of Basic Process Algebra) *Given a finite set A of actions, the set $\mathbb{T}$ of BPA terms is defined inductively as follows:*

- *each action $a \in A$ is in $\mathbb{T}$;*
- *if $p, p' \in \mathbb{T}$, then $p + p' \in \mathbb{T}$, $p \cdot p' \in \mathbb{T}$, and $p \| p' \in \mathbb{T}$.*

Each element of $\mathbb{T}$ is called a PA term.

The compositions $p + p'$, $p \cdot p'$, and $p \| p'$ of processes p and p' are called the alternative, sequential, and parallel compositions, respectively. Intuitively, the process that is associated with the term $a \in \mathbb{T}$, where $a \in A$, can execute a and then terminate. The process $p+p'$, behaves either like the process p or process p', i.e., $p+p'$ executes either a behavior of p or one of p', which introduces a logical coupling between the actions. The process $p \cdot p'$, on the other hand, first executes a behavior of process p and then, after p terminates, executes a behavior of p'; the process $p \cdot p'$ terminates when p' terminates. Finally, $p \| p'$ executes the actions of p and p' in an interleaving manner and terminates when they both terminate. A process which has terminated and can execute no further actions will be denoted by $\surd$.

The size of a term is defined as the number of actions and operators in it. Without loss of any generality, we assume that each action appears at most once in a term.

B. Semantics of Process Algebra

As noted before, each term represents a process, and thus a set of behaviors. A behavior is formalized as a sequence of actions, referred to as a *trace*, that a process can execute. Indeed, each process can be associated with a set of traces, which describes all its behaviors, i.e., all the traces that the process can execute. The function that maps each one of the processes in $\mathbb{T}$ to a set of traces can be defined using a *process graph*.

Definition II.2 (Process Graph) *A* process graph $p_0 \in \mathbb{T}$ *is a labeled transition system $(Q, A, \Rightarrow)$ together with a root state $q_0 \in Q$, and a function $\pi : Q \to \mathbb{T}$, where Q is a set of states, A is a finite set of actions, $\Rightarrow \subset Q \times A \times Q$ is a ternary relation, π associates each state $q \in Q$ with a term $t \in \mathbb{T}$ such that q_0 is mapped to p_0.*

On the process graph of a given process, a transition from a state $q_1 \in Q$ to $q_2 \in Q$ with action $a \in A$, denoted by $q_1 \overset{a}{\Longrightarrow} q_2$, exists if and only if $(q_1, a, q_2) \in \Rightarrow$. If there is a transition $q_1 \overset{a}{\Longrightarrow} q_2$ such that $\pi(q_1) = p_1$, $\pi(q_2) = p_2$, then process p_1 is said to evolve to p_2 after executing Action a, denoted as $p_1 \xrightarrow{a} p_2$. If there is a set of actions $a_1, \ldots, a_k \in A$ and a set of processes $p_0, \ldots, p_k \in \mathbb{T}$ such that $p_{i-1} \xrightarrow{a_i} p_i$ for all $i \in \{1, \ldots, k\}$, then process p_0 is said to evolve to process p_k after executing the sequence $(a_1, a_2, \ldots, a_k)$ of actions. The corresponding sequence of transitions is denoted as $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_k} p_k$. Formally, a *trace* of a PA term p is a sequence $\gamma = (a_1, a_2, \ldots, a_n)$ of actions, for which a corresponding sequence $(p_1, p_2, \ldots, p_n)$ of processes exists such that $p_i \in \mathbb{T}$ and $p_i \xrightarrow{a_i} p_{i+1}$ for all $i \in \{1, 2, \ldots, n\}$. The set of all traces that $p \in \mathbb{T}$ can execute will be denoted as Γ_p. *Operational semantics* can be used to define the process graph of each process via a finite set of *transition rules* defined as follows.

Definition II.3 (Transition Rule) *A* transition rule ρ *is an expression $\frac{H}{\pi}$, where H is a set of transitions $p \xrightarrow{a} p'$, called the* positive premises *of ρ, and π is a transition $p \xrightarrow{a} p'$, called the conclusion of ρ, where $p, p' \in \mathbb{T}$.*

Intuitively, if the set H of premises are possible transitions, then so is π. A set of transition rules will be referred to as a *transition system specification*.

Definition II.4 (Operational Semantics of PA) *The operational semantics of the process algebra is given by the following transition system specification:*

$$\frac{}{a \xrightarrow{a} \surd}$$

$$\frac{p_1 \xrightarrow{a} \surd}{p_1 + p_2 \xrightarrow{a} \surd} \quad \frac{p_1 \xrightarrow{a} p_1'}{p_1 + p_2 \xrightarrow{a} p_1'} \quad \frac{p_2 \xrightarrow{a} \surd}{p_1 + p_2 \xrightarrow{a} \surd} \quad \frac{p_2 \xrightarrow{a} p_2'}{p_1 + p_2 \xrightarrow{a} p_2'}$$

$$\frac{p_1 \xrightarrow{a} \surd}{p_1 \cdot p_2 \xrightarrow{a} p_2} \quad \frac{p_1 \xrightarrow{a} p_1'}{p_1 \cdot p_2 \xrightarrow{a} p_1' \cdot p_2}$$

$$\frac{p_1 \xrightarrow{a} \surd}{p_1 \| p_2 \xrightarrow{a} p_2} \quad \frac{p_1 \xrightarrow{a} p_1'}{p_1 \| p_2 \xrightarrow{a} p_1' \| p_2} \quad \frac{p_2 \xrightarrow{a} \surd}{p_1 \| p_2 \xrightarrow{a} p_1} \quad \frac{p_2 \xrightarrow{a} p_2'}{p_1 \| p_2 \xrightarrow{a} p_1 \| p_2'}$$

where $a \in A$ and $p_1, p_1', p_2, p_2' \in \mathbb{T}$

The definition above yields a process graph for each process in $\mathbb{T}$. Notice that, by the first rule, the process $a \in \mathbb{T}$ where a is an action, i..e, $a \in A$, can only execute the action a and then terminate. The rest of the rules similarly define the semantics of alternative, sequential, and parallel composition operators.

C. Parse Trees of Process Algebra Terms

Process algebra terms can either be single actions, or be composed of other terms, called *subterms*. Thus, a term can be represented as a tree, called a *parse tree*, where each branch of the tree represents a term. Given two branches which correspond to terms p_1 and p_2, one of the operators from $+$, $\cdot$, or $\|$ can be used to bind these two branches to build up the process $p_1 + p_2$, $p_1 \cdot p_2$, or $p_1 \| p_2$, respectively. The parse tree of this process can be represented as a tree, which has the binding operator in the root and the two branches corresponding to p_1 and p_2 as the left and right children.

Consider, for example, the term $p = (a + (b \cdot c)) \| d$, where $a, b, c, d \in A$. The corresponding parse tree is given in Figure 1. As better seen from the parse tree, $p_1 = (a + (b \cdot c))$, $p_2 = d$, $p_3 = a$, $p_4 = (b \cdot d)$, $p_5 = b$, and $p_6 = d$ are subterms of p, where p_1 and p_2 are bound by $\|$. Similarly, p_3, p_4, p_5, and p_6 are subterms of p_1, etc.

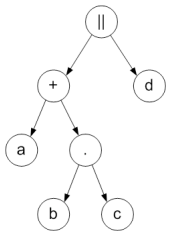

Fig. 1. The parse tree of the term $(a + (b \cdot c)) \| d$.

III. PROCESS ALGEBRA SPECIFICATIONS OF MULTIPLE-UAV MISSIONS

In this section, PA is used to address a class of mission specifications for multiple UAVs. It is also pointed out through examples that complex tasks can be built from simpler ones in an hierarchical manner, thus possibly simplifying the creation and readability of mission specifications by human operators.

Before presenting the details, let us note the following notation. Given a set S, a (finite) sequence σ of (distinct) elements from S is a function which maps $\{1, 2, \ldots, K\}$ to S, where $K \in \mathbb{N}$ and $\sigma(i) = \sigma(j)$ implies that $i = j$. The number K, denoted by $|\sigma|$, is called size of the sequence σ. For convenience, σ is denoted as $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(K))$. An element $s \in S$ is said to appear in the sequence σ, denoted by $s \in \sigma$ with an abuse of notation, if there exists some k such that $\sigma(k) = s$. Element s_1 is said to precede element s_2 in σ if $s_1, s_2 \in \sigma$ and $\sigma^{-1}(s_1) < \sigma^{-1}(s_2)$. Given a sequence $\sigma = (\sigma(1), \ldots, \sigma(K))$, any sequence $\tilde{\sigma} = (\sigma(1), \ldots, \sigma(l))$, where $l < K$, is called a prefix of σ.

A. Atomic Objectives

An atomic objective is an abstraction of a generic individual task, which is very similar to the task abstraction given in [18], which we will use in the following. Every atomic objective is represented by two spatial and one temporal parameter: the entry and exit points, and the duration of the task. Intuitively, the entry point is a point in the two dimensional plane which the UAV has to get to in order to start executing the task. Similarly, the exit point refers to the point where the UAV will be after executing the task. The duration indicates the time required to execute the task. Also, each atomic objective is associated with a set of vehicles which are capable of executing it. More formally, an atomic objective o refers to the tuple

$$\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix}_o, \begin{bmatrix} x_f \\ y_f \end{bmatrix}_o, T_o^e, U_o \right),$$

where first two elements are in $\mathbb{R}^2$, $T_o^e \in \mathbb{R}_{\geq 0}$, $U_o \subseteq \mathcal{U}$.

From a practical point of view, these parameters can be UAV dependent, i.e., each one of the three parameters can vary for different UAV types. This approach can easily be extended to cover such cases.

Examples of atomic objectives include the classification, attack, and verification tasks as well as object tracking, reconnaissance, area search, etc. (see [18]).

B. Complex Objectives

This section presents two examples of hierarchical specification in UAV cooperative control problems via objectives, followed by the formal problem definition.

1) The Target Engagement Scenario: Consider a scenario with n targets and m UAVs, where each target has to be (i) classified, (ii) attacked, and (iii) verified for damage inflicted, in this order. The problem is to assign these tasks to UAVs in such a way that (i) all three tasks are executed on each target and (ii) the cost of the resulting plan is minimized (see for example [17]).

Let c_i, a_i, and v_i be the atomic objectives for classify, attack, and verify tasks on Target i. Then, the set of atomic objectives O is identified as $O = \{c_1, a_1, v_1, c_2, a_2, v_2, \ldots, c_n, a_n, v_n\}$. Even though this scenario is simple enough to write the specification immediately from the atomic objectives, let us consider a middle layer, in which engaging a target is specified as a complex objective, t_i. More precisely, t_i has the set $\{c_i, a_i, v_i\}$ of children objectives and PA term $t_i = c_i \cdot a_i \cdot v_i$. Then, the specification p for this problem can be written as follows: $p = t_1 \| t_2 \| \ldots \| t_n$, which, in turn, can be written in terms of atomic objectives as: $s = c_1 \cdot a_1 \cdot v_1 \| c_2 \cdot a_2 \cdot v_2 \| \ldots \| c_n \cdot a_n \cdot v_n$.

In this approach, at different levels the task specification can be changed rather easily. Consider, for example, the case in which there exists an attack option with a powerful UAV that does not require a verification task. However, assume that this attack option is

available for only some of the targets. Let us denote this attack option with $\bar{a}_i$, which is indeed an atomic objective. Next, introducing a new complex objective $\bar{t}_i$ such that the corresponding term is defined as $t_i = c_i \cdot (a_i \cdot v_i + \bar{a}_i)$, the specification p can be rearranged as: $p = t_1 \,\|\, t_2 \,\|\, \ldots \,\|\, t_k \,\|\, \bar{t}_{k+1} \,\|\, \ldots \,\|\, \bar{t}_n$.

2) The Rescue Mission: A rescue mission scenario, which involves logical and temporal constraints, was presented in [6]. Let us study a similar example using the hierarchical task specification framework.

In this scenario, a friendly infantry unit is pinned down by enemy in a certain location. There are three targets, T1, T2, and T3, that needs to be neutralized for the friendly unit be rescued. Two of the targets, T1 and T2, are protected by missile sites, S1 and S2, respectively. There are two friendly bases: Base B1 and Base B2. There are three types of friendly UAVs: the first type, V1, can only engage the targets, the second type, V2, can only engage the missile sites, and the third one, V3, can engage any of them, but it is a relatively low-speed asset. A map of this scenario can be seen in Figure 2. In order to successfully accomplish this mission, the infantry unit can either escape to Base B1, by the targets T1 and T3 being neutralized, or to Base B2 after the UAVs neutralize targets T2 and T3. Moreover, if the latter option is chosen a V3 type UAV has to head to B2 with necessary medical equipment, even if it does not engage any target along the way.

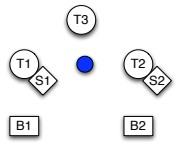

Fig. 2. Map of the Rescue Mission. Infantry unit is shown in blue.

Let p be the mission specification, which states that the infantry unit must be rescued by either escaping to Base B1 or Base B2, i.e., $p = o_{B1} + o_{B2}$, where o_{B1} and o_{B2} are complex objectives. Noting that both targets $T1$ and $T3$ have to be engaged in order to accomplish the objective o_{B1}, we can write $o_{B1} = o_{T1} \,\|\, o_{T3}$, where o_{T1} and o_{T3} denote the objectives of neutralizing targets T1 and T3 respectively. Similarly, $o_{B2} = o_{T2} \,\|\, o_{T3} \,\|\, o_{V3@B2}$, where o_{T2} is the objective of neutralizing Target T2 and $o_{V3@B2}$ is the objective of moving a V3 type vehicle to Base B2 with the necessary health equipment. Even though $o_{V3@B2}$ is an atomic objective, notice that o_{T1}, o_{T2}, and o_{T3} are still complex objectives that need to be identified in terms of atomic objectives. As noted before there are two possible ways of neutralizing Target $T1$: (i) either the missile site $S1$ can be attacked by a vehicle of type $V2$ after the neutralization of $T1$ by a vehicle of type $V1$, (ii) or $T1$ can be neutralized directly using a vehicle of type $V3$. Hence, $o_{T1} = (o_{V2@S1} \cdot o_{V1@T1}) + o_{V3@T1}$. Similarly, $o_{T2} = (o_{V2@S2} \cdot o_{V1@T2}) + o_{V3@T2}$. Noting that Target T3 is not protected by any missile sites, $o_{T3} = o_{V1@T3} + o_{V3@T3}$.

C. Problem Definition

Let us fix a set $O = \{o_1, o_2, \ldots, o_n\}$ of atomic objectives and a set $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ of UAVs. Given a specification p expressed as a PA term, the task assignment problem for PA specifications is, informally, to assign a necessary number of atomic objectives from O to the UAVs in $\mathcal{U}$ such that the "temporal ordering" of atomic objectives in the resulting assignment is a trace of p, and a given cost function is minimized.

More formally, let T^t_{u,o_i,o_j} be time it takes for UAV u to travel from the exit point of o_i to the entry point of o_j. Then, a *single UAV schedule* σ_u for a UAV $u \in \mathcal{U}$ is a sequence of atomic objective and execution time pairs, $\sigma_u = ((o_1, t_{o_1}), (o_2, t_{o_2}), \ldots, (o_l, t_l))$, such that $o_i \in O$, $t_i \in \mathbb{R}_{\geq 0}$ for $i \in \{1, 2, \ldots, l\}$, and $t_{o_i} + T^e_{o_i} + T^t_{u,o_i,o_{i+1}} \leq t_{o_{i+1}}$ for $i \in \{1, 2, \ldots, l-1\}$. Notice that the execution time t_{o_i} of o_i must be chosen such that the UAV has time to complete the previous atomic objective o_{i-1} and fly to the entry point of o_i. Each single UAV schedule σ_u is naturally associated with a completion time, denoted by $\tau^c_{\sigma_u}$, which refers to the completion time of the last atomic objective assigned in σ_u, i.e., $\tau^c_{\sigma_u} = t_{o_l} + T_{o_l}$ where o_l is such that $\sigma_u(|\sigma_u|) = (o_l, t_{o_l})$. A *complete schedule* $\mathcal{S}$ is a set of schedules such that there exists a unique single UAV schedule $\sigma_u \in \mathcal{S}$ for all $u \in \mathcal{U}$. A complete schedule can be associated with a cost function in several ways. Two of the commonly used cost functions are the *minimum mission time* and the *cumulative execution time*, which can be written in terms of completion times respectively as

$$\max_{u_k \in \mathcal{U}} \tau^c_{\sigma_{u_k}}, \quad \text{or} \quad \sum_{u_k \in \mathcal{U}} \tau^c_{\sigma_{u_k}}. \tag{1}$$

An atomic objective is said to be scheduled in a complete schedule $\mathcal{S}$ if there exists a single UAV schedule $\sigma_u \in \mathcal{S}$ such that (o, t) appears in σ_u for some $t \in \mathbb{R}_{\geq 0}$. Each schedule is associated with a set of temporal orderings through the notion of observation. An *observation* of a complete schedule $\mathcal{S}$ is a sequence $\pi = (o_1, o_2, \ldots, o_l)$ of atomic objectives such that (i) an atomic objective o appears in π if and only if it is scheduled in $\mathcal{S}$, (ii) for all $i = 1, 2, \ldots, l$, there exists $\tau_i \in \mathbb{R}_{\geq 0}$, for which $\tau_i \leq \tau_{i+1}$, $t_i \leq \tau_i \leq t_i + T_{o_i}$, and for any $i, j \in \{1, 2, \ldots, l\}$ if $\tau_i < \tau_j$ then o_i precedes o_j in π. Intuitively, each o that appears in π is assigned a time τ_i, which is within the execution interval of o_i in $\mathcal{S}$ and the atomic objectives o_i are ordered in π according to the natural ordering of their corresponding τ_i.

Notice that the observation of a complete schedule does not have to be unique in general. Let us denote the set of all observations of a given complete schedule $\mathcal{S}$ by $\Pi_{\mathcal{S}}$. Then the task assignment problem for PA specifications can be stated formally as follows:

Problem III.1 *Given a set O of atomic objectives, a set $\mathcal{U}$ of UAVs, and a PA specification p, find a complete schedule $\mathcal{S}$ such that (i) any observation π of $\mathcal{S}$ is a trace of p, i.e., $\Pi_{\mathcal{S}} \subseteq \Gamma_p$, (ii) one of the cost functions in* (1) *is minimized.*

IV. OPTIMAL PLANNING FOR PROCESS ALGEBRA SPECIFICATIONS

The task assignment algorithm proposed in this paper builds the schedule that solves Problem III.1, in stages. Starting with a schedule $\mathcal{S}_0$ in which no atomic objective is assigned, intuitively, at stage $L+1$ the algorithm extends the schedule $\mathcal{S}_L$ of the previous stage by choosing an atomic objective o to be assigned, a UAV u which the task is assigned to, and an execution time t_o for o such that after the assignment (i) any observation of the resulting schedule $\mathcal{S}_{L+1}$ is a prefix of some trace of p, (ii) the completion time of σ_u is minimized in $\mathcal{S}_{L+1}$. In this manner, the algorithm extends the tree search algorithm given in [17] to handle PA specifications.

Each PA specification p represents a different tree structure, every node of which is labeled with an atomic objective and UAV pair. The nodes are connected such that any path on the tree corresponds to a trace of p and for all traces there is a corresponding path on the tree. For example, the tree structure of the rescue mission of Section III-B.2 is shown in Figure 3.

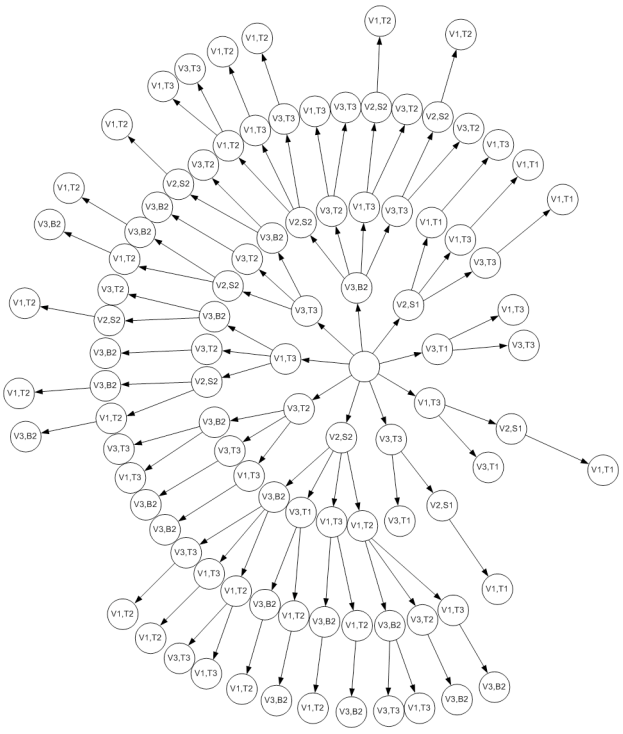

Fig. 3. The tree structure of the rescue mission

For effectiveness purposes, however, the structure of the tree should be constructed "on the fly" while executing the search algorithm. That is, given a node of the tree one has to efficiently determine the set of children nodes, in other words, the set of atomic objectives that can be assigned in the next stage. For the case of PA specifications, an algorithm that runs in time linear with respect to the size of the specification is given in Algorithm 1. This algorithm takes a specification p and returns a set of pairs of PA terms and atomic objectives such that $(p', o) \in Next(p)$ if and only if $p \xrightarrow{o} p'$. The correctness of the algorithm is clear from the semantics of PA.

```
switch p do
    case p_1 + p_2
        return Next(p_1) ∪ Next(p_2)
    case p_1 · p_2
        return {(p'_1 · p_2, o) | (p'_1, o) = Next(p_1)}
    case p_1 || p_2
        return {(p'_1 || p_2, o) | (p'_1, o) = Next(p_1)} ∪
               {(p_1 || p'_2, o) | (p'_2, o) = Next(p_2)}
    otherwise
        return {(√, p)}
    end
end
```

Algorithm 1: Computation of $Next(p)$ for given $p \in \mathbb{T}$

Moreover, at each stage, the newly assigned atomic objective o has to be associated with its execution time t_o. To achieve a minimum time complete schedule, intuitively, o must be assigned an execution time that is as early as possible. However, one must consider two constraints: firstly, the execution time must be such that the UAV it is assigned has time to complete its previous atomic objective and fly to the entry point of o, secondly, all observations of the resulting schedule must be a prefix of a trace of p. Regarding the latter constraint, the t_o is chosen such that it is later than all the assigned *predecessors* of o. Formally speaking, an atomic objective $\bar{o}$ is said be a *predecessor* of an atomic objective o in term p, if (i) there exists a trace γ of p such that $\bar{o}$ precedes o in γ and (ii) there is no trace of p in which o precedes $\bar{o}$. An equivalent characterization is as follows. Consider the parse tree of p. It can be shown that $\bar{o}$ precedes o in p if and only if there is a node in the parse tree that binds two subterms p_1 and p_2 with a sequential operator, i.e., $p_1 \cdot p_2$, where p_1 contains $\bar{o}$ and p_2 contains o. Let $Pred_p(o)$ be the set of all predecessors of o in p. A naive algorithm to compute $Pred_p(o)$ for each $o \in \boldsymbol{O}$ would be to consider every atomic objective and determine whether or not it precedes o via the above characterization. Note that such an algorithm runs in polynomial time; moreover, it can be executed before starting the tree search algorithm.

Following the discussion above, let o be the atomic objective which is to be assigned to a UAV u at stage $L+1$. Let $\mathcal{S}_L$ be the schedule in the previous stage. Then, the execution time of t_i is chosen such that $t_i := \max(\tau^c_{\sigma_u}, \tau^c_{Pred_p(o)})$, where $\tau^c_{\sigma_u}$ is the completion time of σ_u in $\mathcal{S}_L$ and $\tau^c_{Pred_p(o)}$ is maximum of the completion times of atomic objectives which are predecessors of o and are assigned in $\mathcal{S}_L$, i.e., $\tau^c = \max\{\bar{t} \mid \bar{o} \in Pred_p(o), \exists u \in \mathcal{U}.(\bar{o}, \bar{t}) \in \sigma_u\}$. Notice that, the execution time t_i of o_i is chosen to be the earliest possible in the sense that, if $t_i < \tau^c_{\sigma_u}$ then u would not be able to execute o, and if

$t_i < \tau^c_{Pred_p(o)}$ then there would be an observation of $\mathcal{S}_{L+1}$, which is not a prefix of any trace of p.

The schedules constructed in this manner are feasible in the sense that any observation of such schedules are prefixes of some trace of p. Moreover, the tree search algorithm eventually terminates with the optimal solution. The proofs of these statements will not be discussed here for the sake of brevity.

For effectiveness, a best-first search heuristic enhanced with a branch-and-bound scheme is used for the simulation, which will be presented in the next section. The best-first search heuristic returns a feasible solution to the problem in time polynomial with respect to the size of the specification. Using branch-and-bound, the tree structure is explored effectively leading to the optimal solution.

V. SIMULATIONS

Let us consider a simple scenario where two UAVs are required to first attack six targets and then search for possible new targets in the same area. Let o_d and o_s denote objectives for the engagement of six targets and the search mission respectively. Then, the specification of the mission is $p = o_d \cdot o_s$. The objective o_d is equal to $o_{d1} \,||\, o_{d2} \,||\, o_{d3} \,||\, o_{d4} \,||\, o_{d5} \,||\, o_{d6}$, where o_{di} denotes the atomic objective of attacking the target i for $i = 1, 2, \ldots, 6$. The objective o_s, on the other hand, requires searching the there regions in parallel, i.e., $o_s = o_{s1} \,||\, o_{s2} \,||\, o_{s3} \,||\, o_{s45}$, where o_{s1}, o_{s2}, o_{s3} denote the atomic objectives for searching the regions $s1$, $s2$ and $s3$, respectively. To make the matters more complicated, let it be the case that searching either area $s4$ or $s5$ is adequate for accomplishing $s3$, but not both, i.e., $o_{s45} = o_{s4} + o_{s5}$, where o_{s4} and o_{s5} are the atomic objectives corresponding to searching the regions $s4$ and $s5$, respectively. A solution of this problem instance is presented in Figure 4, which was generated in less than a second of computation time, after exploring 1000 nodes of the tree.

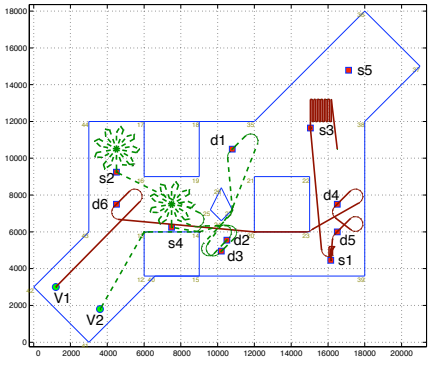

Fig. 4. Scheduling that satisfies $p = o_d \cdot o_s$.

VI. CONCLUSIONS

This paper proposed and studied Process Algebra as a formal language to state mission specifications for multiple-UAV systems. The task assignment problem with PA specifications was proposed and some problem instances of interest were pointed out. The problem was solved using a computationally effective algorithm, which returns a best-first feasible solution to the problem in polynomial time and terminates with the optimal solution in finite time.

Future research directions include addressing similar problems with a timed process algebra using which quantitative properties of time can be expressed.

VII. ACKNOWLEDGMENTS

The authors would like to acknowledge Corey Schumacher for several inspiring discussions. This work was supported in part by the Michigan/AFRL Collaborative Center on Control Sciences, AFOSR Grant #FA9550-07-1-0528. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the supporting organizations.

References

[1] J. Bellingham, M. Tillerson, A. Richards, and J.P. How. Multi-task allocation and path planning for cooperating UAVs. In S. Butenko, R. Murphey, and P.M. Pardalos, editors, *Cooperative Control: Models, Applications and Algorithms*. Kluwer Academic Publishers, 2001.

[2] C. Schumacher, P. Chandler, and S. Rasmussen. Task allocation for wide area search munitions. In *ACC*, 2002.

[3] C. Schumacher, P.R. Chandler, M. Pachter, and L.S. Patcher. Optimization of air vehicles operations using mixed-integer linear programming. *Journal of the Operational Research Society*, 58:516–527, 2007.

[4] P. Tabuada and G.J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[5] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53:287–297, 2007.

[6] S. Karaman and E. Frazzoli. Complex mission optimization for multiple-UAVs using linear temporal logic. In *ACC*, 2008.

[7] S. Karaman and E. Frazzoli. Optimal vehicle routing with metric temporal logic specifications. In *CDC*, 2008.

[8] P. Schnoebelen. The complexity of temporal logic model checking. In *4th Workshop on Advences in Modal Logics*, 2002.

[9] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *IEEE Conf. on Robotics and Automation*, 2007.

[10] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[11] J.M.C. Baeten. *Process Algebra*. Cambridge University Press, 1990.

[12] W. Fokkink. *Introduction to Process Algebra*. Springer, 2000.

[13] J.M.C. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335:131–146, 2005.

[14] J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335(2-3):215–280, 2003.

[15] G. Salaun, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. *International Journal of Business Process Integration and Management*, 1(2):116–128, 2006.

[16] P. Adao and P. Mateus. A process algebra for reasoning about quantum security. *Electronic Notes in Theoretical Computer Science*, 170:3–21, 2007.

[17] S.J. Rasmussen and T. Shima. Tree search algorithm for assigning cooperating UAVs to multiple tasks. *International Journal of Robust and Nonlinear Control*, 18:135–153, 2008.

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