January 2013

Probabilistic Verification of Coordinated Multi-Robot Missions

Probabilistic Verification of Coordinated Multi-Robot Missions[⋆]

Sagar Chaki and Joseph Andrew Giampapa
{chaki,garof}@sei.cmu.edu

Carnegie Mellon Software Engineering Institute

Abstract. Robots are increasingly used to perform a wide variety of tasks, especially those involving dangerous or inaccessible locations. As the complexity of such tasks grow, robots are being deployed in teams, with complex coordination schemes aimed at maximizing the chance of mission success. Such teams operate under inherently uncertain conditions – the robots themselves fail, and have to continuously adapt to changing environmental conditions. A key challenge facing robotic mission designers is therefore to construct a mission – i.e., specify number and type of robots, number and size of teams, coordination and planning mechanisms etc. – so as to maximize some overall utility, such as the probability of mission success. In this paper, we advocate, formalize, and empirically justify an approach to compute quantitative utility of robotic missions using probabilistic model checking. We show how to express a robotic demining mission as a restricted type of discrete time Markov chain (called $\alpha\mathbf{PA}$), and its utility as either a linear temporal logic formula or a reward. We prove a set of compositionality theorems that enable us to compute the utility of a system composed of several $\alpha\mathbf{PA}s$ by combining the utilities of each $\alpha\mathbf{PA}$ in isolation. This ameliorates the statespace explosion problem, even when the system being verified is composed of a large number of robots. We validate our approach empirically, using the probabilistic model checker PRISM.

1 Introduction

Robots are increasingly used to perform a wide variety of tasks. Examples include situations where the task is dangerous (e.g., demining) or involves physically inaccessible localities (e.g., a disaster area). They are often deployed in teams to provide fault tolerance, and to accommodate a wider variety of plans. The tasks consist of both unpredictable and known parts. For example, the operating conditions change unpredictably, and robots might malfunction, become indisposed, or be unable to complete its task due to the lack of capability. These are unknown. On the other hand, there are known parameters, e.g., the number of robots, the capabilities of each robot, the set of plans available to each robot,

[⋆] This is an extended version (with additional proofs in the Appendix) of the paper with the same title in the Proceedings of International SPIN Symposium on Model Checking of Software (SPIN) 2013.

and the coordination algorithms used by the robots, that are within the control of the mission designer. The goal of the designer is to select these parameters so as to increase overall mission utility.

We focus on missions that involve foraging-and-reacting (FAR), where robots have to explore an arena, look for specific objects, and react to them in specific ways. Examples of such missions are demining a minefield [16] where robots attempt to defuse detected mines, and search-and-rescue of a disaster area where robots report the location and status of discovered survivors to authorities.

Designing FAR missions requires assessing two aspects: (a) *success*: estimate the probability of mission success within a certain deadline; and (b) *coverage*: compute the expected amount of terrain covered within a given deadline. Currently, designers rely on their prior knowledge as well as field tests and simulations to solve these two problems. Both have limitations. Relying on prior knowledge is an ad-hoc approach, limited, and typically does not cover unknown and unforeseen situations. Full scale field tests are expensive, time-consuming, and may not be conducted in a way that permits a generalization of the relative impacts of certain parameter settings to similar missions in other contexts.

In this paper, we propose a more systematic, repeatable, and analytic method, based on probabilistic model checking, to solve both success and coverage problems. Specifically, we show how to model a robotic demining mission as a probabilistic automaton (PA). In addition, we show how to express success as a probabilistic LTL [1] formula, and coverage as a cumulative reward over the PA. This is our first contribution. Further details are presented in Section 5.

Our second contribution is tackling the statespace explosion problem during probabilistic model checking of FAR missions. We leverage two types of restrictions commonly found in such missions. First, robots are divided into teams, and each team operates independently on a separate portion of the arena. We call this property independence (**IND**). Second, the PAs for the teams "synchronize" over a common action corresponding to a clock tick since the robot teams operate under the same global clock. We call this property synchronization (**SYNC**). In our approach, these two restrictions are incorporated by modeling each team as a $\alpha\mathbf{PA}$, i.e., a PA with a singleton alphabet $\{\alpha\}$. When $\alpha\mathbf{PA}s$ are composed, they synchronize over the common action α. The result is also a $\alpha\mathbf{PA}$.

Our requirement of synchronization between robotic teams is a purely modeling construct. The teams do not have to possess physically synchronized clocks. However, the time taken by a team for an action must respect the timing constraint on the action used in the model. For example, if the model assumes that sensing a mine requires 40 units of time, and 1 unit equals 1 second, then each team must complete the mine sensing activity within 40 seconds. Otherwise, the predictions made by the model checker will be invalid. Therefore, the model must be constructed based on realistic values for timing constraints and probabilities.

The restricted nature of $\alpha\mathbf{PA}s$ enables us to obtain two compositionality results: (a) probability of satisfying an LTL formula accumulates multiplicatively over $\alpha\mathbf{PA}s$ (cf. Theorem 1 and 2); and (b) expected reward accumulates additively over $\alpha\mathbf{PA}$ (cf. Theorem 3 and 4). Our compositionality theorems hold for an arbitrary (but finite) number of $\alpha\mathbf{PA}s$. Further details are presented in Section 4. These theorems enable us to solve success and coverage for our demining case study in a completely compositional manner by model checking the $\alpha\mathbf{PA}$ for each team individually. Note that these compositionality results do not hold if we remove the restriction to singleton (and identical) alphabets.

Our third and final contribution is an empirical validation of our results by using the state-of-the-art probabilistic model checker PRISM [11] to compute the values of success and coverage for our demining case study using both the compositional approach and the direct non-compositional approach. We show how the non-compositional model checking runs out of resources even for two robotic teams, while the compositional approach scales easily to even thirty teams. Further details are presented in Section 6.

The rest of the paper is structures as follows. In Section 2 we survey related work. In Section 3, we present basic definitions. In Section 4 we present our compositionality theorems. In Section 5, we present our robotic demining scenario and its $\alpha\mathbf{PA}$ model, as well as the properties we want to verify. In Section 6, we present experimental results, and in Section 7, we conclude.

2 Related Work

This paper builds on a wide body of work in modeling and verifying probabilistic systems [14]. In particular, probabilistic model checking has been used to verify systems ranging from pacemakers [3], root contention protocols [13] and biological pathways [8]. Our work explores the application of probabilistic model checking to yet another domain – coordinated multi-robot missions.

The connection between probabilistic systems and compositionality has been studied by a number of researchers. For example, de Alfaro et al. [4] provide a semantic notion of compositionality in the context of probabilistic reactive modules. Our notion of probabilistic automata and parallel composition is borrowed from that proposed by Stoelinga [15] and others. In essence, $\alpha\mathbf{PA}$ are a restricted, yet useful, version of probabilistic automata that admit to strong compositionality results.

A number of projects on compositional verification of probabilistic systems [12] use automated assume-guarantee algorithms that are based on learning [6, 7]. There is also work on learning-based assume-guarantee reasoning for synchronous probabilistic systems [5], assume-guarantee and abstraction refinement for probabilistic systems [9], and on compositional reasoning for probabilistic model checking of hardware designs [10]. Our approach is also compositional, but does not involve assume-guarantee reasoning.

A preliminary version of the demining scenario presented here, its probabilistic model, and experimental results were reported in our previous work [2]. The model was less elaborate, e.g., it did not include uncertainty when moving from cell to cell. Also, it was a DTMC, not $\alpha\mathbf{PA}$, and hence not amenable to the compositionality theorems presented here. Indeed, our prior work [2] did not include any compositionality theorems, nor empirical results showing their effectiveness.

3 Preliminaries

We adopt the formalism of probabilistic automata [15], modifying it in two ways: (a) extending it by labeling states with atomic propositions; and (b) restricting the alphabet to be a singleton. The result is a class of automata we call $\alpha\mathbf{PA}$. Let $Dist(X)$ be the set of all probability distributions over any set X.

Definition 1 (αPA). *A $\alpha\mathbf{PA}$ is a 6-tuple $(S, Init, \Sigma, \delta, AP, \mathcal{L})$ where: (i) S is a countable set of states; (ii) $Init \in S$ is the initial state; (iii) $\Sigma = \{\alpha\}$ is the singleton alphabet; (iv) $\delta : S \mapsto Dist(S)$ is the transition relation; (v) AP is a set of atomic propositions; and (vi) $\mathcal{L} : S \mapsto 2^{AP}$ is a mapping from states to sets of atomic propositions, such that $\mathcal{L}(s)$ is the set of propositions true in s.*

If $M = (S, Init, \Sigma, \delta, AP, \mathcal{L})$ is a $\alpha\mathbf{PA}$, we write $S(M)$, $Init(M)$, $\Sigma(M)$, $\delta(M)$, $AP(M)$, and $\mathcal{L}(M)$ to mean S, $Init$, Σ, δ, AP and $\mathcal{L}$, respectively.

Definition 2 (Execution). *Let M be a $\alpha\mathbf{PA}$. An execution π is a (finite or infinite) sequence of states $s_0, s_1, \ldots$ such that:*

$$\forall i \geq 0 \centerdot \delta(M)(s_i)(s_{i+1}) > 0$$

The execution π starts from s_0. The set of all executions starting from s is denoted by $Ex(s, M)$, and $Ex(M)$ means $Ex(Init(M), M)$. The set of all finite executions starting from s is denoted by $\widehat{Ex}(s, M)$ and $\widehat{Ex}(M)$ means $\widehat{Ex}(Init(M), M)$. We omit M from $Ex(s, M)$ and $\widehat{Ex}(s, M)$ when it is clear from the context.

Given two probability distributions $\mu_1 \in Dist(X_1)$ and $\mu_2 \in Dist(X_2)$, the distribution $(\mu_1 \times \mu_2) \in Dist(X_1 \times X_2)$ is defined as follows:

$$\forall (x_1, x_2) \in X_1 \times X_2 \centerdot (\mu_1 \times \mu_2)(x_1, x_2) = \mu_1(x_1) \times \mu_2(x_2)$$

For any set X and an element $x \in X$, the Dirac distribution $\Delta(x) \in Dist(X)$ maps x to 1 and every other element of X to 0. $\alpha\mathbf{PA}s$ synchronize via the common action α. Let M_1 and M_2 be two $\alpha\mathbf{PA}s$. We write $M_1 \diamond M_2$ to mean $AP(M_1) \cap AP(M_2) = \emptyset$. Formally, the composition of $\alpha\mathbf{PA}$ is defined as follows.

Definition 3. *Let M_1 and M_2 be $\alpha\mathbf{PA}s$ such that $M_1 \diamond M_2$. Their parallel composition $M_1 \parallel M_2$ is the $\alpha\mathbf{PA}$ $(S, Init, \Sigma, \delta, AP, \mathcal{L})$ where:*

$$\begin{gathered} S = S(M_1) \times S(M_2) \qquad Init = (Init(M_1), Init(M_2)) \\ \Sigma = \{\alpha\} \qquad \delta(s_1, s_2) = \delta(M_1)(s_1) \times \delta(M_2)(s_2) \\ AP = AP(M_1) \cup AP(M_2) \qquad \mathcal{L}(s_1, s_2) = \mathcal{L}(M_1)(s_1) \cup \mathcal{L}(M_2)(s_2) \end{gathered}$$

Properties. We assume that properties are specified as LTL [1] formulas. The syntax of a LTL formula Ψ over the set of atomic propositions AP is given by:

$$\Psi := \text{TRUE} \mid a \mid \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\Psi \mid \Psi \mathsf{U} \Psi$$

where $a \in AP$ is an atomic proposition. We write $\pi \models \Psi$ to mean that the infinite execution π satisfies the formula Ψ. Consider a PA M. We write $Ex(s, \Psi)$ to mean the infinite executions starting from s that satisfy Ψ, i.e.,

$$Ex(s, \Psi) = \{\pi \in Ex(s) \mid \pi \models \Psi\}$$

Cylinders. Every finite execution $\widehat{\pi}$ induces a set of infinite executions for which $\widehat{\pi}$ is a prefix. This is known as the cylinder of $\widehat{\pi}$, or $\mathsf{Cyl}(\widehat{\pi})$. A finite execution $\widehat{\pi}$ satisfies Ψ, denoted $\widehat{\pi} \models \Psi$, if $\forall \pi \in \mathsf{Cyl}(\widehat{\pi}) \centerdot \pi \models \Psi$. We write $\widehat{\pi_1} \sqsubseteq \widehat{\pi_2}$ to mean that $\widehat{\pi_1}$ is a prefix of $\widehat{\pi_2}$. A set of finite executions E is *minimal* if it has no two distinct elements $\widehat{\pi_1}$ and $\widehat{\pi_2}$ such that $\widehat{\pi_1} \sqsubseteq \widehat{\pi_2}$. For every LTL formula Ψ and state s, there is a unique minimal subset [17] of $\widehat{Ex}(s)$, denoted $\mathcal{B}(s, \Psi)$, such that:

$$Ex(s, \Psi) = \bigcup_{\widehat{\pi} \in \mathcal{B}(s,\Psi)} \mathsf{Cyl}(\widehat{\pi})$$

Informally, $\mathcal{B}(s, \Psi)$ is a "finite basis" of Ψ whose cylinders generate all (and exactly all) executions from s that satisfy Ψ. Let $\widehat{Ex}(s, k)$ be the subset of $\widehat{Ex}(s)$ containing only executions with $k+1$ states. Let $\widehat{\pi} = s_0, \ldots, s_n \in \widehat{Ex}(s_0, n)$. Let us define $\mathbf{p}(\widehat{\pi})$ as follows:

$$\mathbf{p}(\widehat{\pi}) = 1 \text{ if } n = 0 \text{ and } \mathbf{p}(\widehat{\pi}) = \prod_{0 \le i < n} \delta(M)(s_i)(s_{i+1}) \text{ otherwise}$$

Definition 4. *Given a state s and a LTL formula Ψ, $\mathsf{P}(s, \Psi)$ is the probability that s satisfies Ψ, and is defined as:*

$$\mathsf{P}(s, \Psi) = \sum_{\widehat{\pi} \in \mathcal{B}(s,\Psi)} \mathbf{p}(\widehat{\pi})$$

Rewards. We write $\mathsf{P}(M, \Psi)$ to mean $\mathsf{P}(Init(M), \Psi)$. A reward structure on a $\alpha\mathbf{PA}$ M is a pair (ρ, ι) such that $\rho : S(M) \mapsto \mathbb{R}$ and $\iota : S(M) \times S(M) \mapsto \mathbb{R}$ map states and transitions of M, respectively, to real-valued rewards. Each transition of M corresponds to a discrete unit of time.

Definition 5. *The cumulative reward due to a reward structure $R = (\rho, \iota)$ from state s up to time k (i.e., up to k transitions of M from s), denoted by $C_{\le k}(s, R)$ is defined recursively as follows:*

$$C_{\le 0}(s, R) = 0$$
$$\forall k > 0 \centerdot C_{\le k}(s, R) = \rho(s) + \sum_{s' \in S(M)} \delta(M)(s)(s') \times (\iota(s, s') + C_{\le (k-1)}(s', R))$$

4 Compositional Verification

In this section, we present our compositionality theorems. We begin by defining the "product" of two executions. Let $M_1 \in \alpha\mathbf{PA}$ and $M_2 \in \alpha\mathbf{PA}$. Let $\widehat{\pi_1} =$

$s_0, \ldots, s_n \in \widehat{Ex}(M_1, n)$ and $\widehat{\pi_2} = s'_0, \ldots, s'_n \in \widehat{Ex}(M_2, n)$ be two finite executions. Then, $\widehat{\pi_1} \times \widehat{\pi_2} \in \widehat{Ex}(M_1 \parallel M_2, n)$ is the execution $(s_0, s'_0), \ldots, (s_n, s'_n)$. If $\widehat{\pi} = \widehat{\pi_1} \times \widehat{\pi_2}$, then we write $\pi \downharpoonright 1$ and $\pi \downharpoonright 2$ to mean $\widehat{\pi_1}$ and $\widehat{\pi_2}$, respectively.

This extends to executions of different length as follows. Given a finite execution $\widehat{\pi} = s_0, \ldots, s_m$, and $n \geq m$, the set of n-extensions of $\widehat{\pi}$, denoted by $\widehat{\pi}^{+n}$, is defined as follows:

$$\begin{aligned}
\widehat{\pi}^{+n} &= \{\widehat{\pi'} \in \widehat{Ex}(s_0, n) \mid \widehat{\pi} \sqsubseteq \widehat{\pi'}\} \\
\widehat{\pi_1} \times \widehat{\pi_2} &= \{\widehat{\pi'}_1 \times \widehat{\pi'}_2 \mid \widehat{\pi'}_1 \in \widehat{\pi_1}^{+n} \wedge \widehat{\pi'}_2 \in \widehat{\pi_2}^{+n} \wedge n = \mathsf{max}(|\widehat{\pi_1}|, |\widehat{\pi_2}|)\} \\
E_1 \times E_2 &= \bigcup_{(\widehat{\pi_1}, \widehat{\pi_2}) \in E_1 \times E_2} \widehat{\pi_1} \times \widehat{\pi_2}
\end{aligned}$$

Note that if $\widehat{\pi} \in \widehat{\pi_1} \times \widehat{\pi_2}$, then $\widehat{\pi_1} \sqsubseteq \widehat{\pi} \downharpoonright 1$ and $\widehat{\pi_2} \sqsubseteq \widehat{\pi} \downharpoonright 2$. Next we present two lemmas. The proofs of these lemmas are in the appendix.

Lemma 1. *Let $M_1 \in \alpha\mathbf{PA}$, $M_2 \in \alpha\mathbf{PA}$ be $\alpha\mathbf{PA}$s such that $M_1 \diamond M_2$. Let $s_1 \in S(M_1)$, $s_2 \in S(M_2)$, and $\varPsi_1$ and $\varPsi_2$ be LTL formulas over $AP(M_1)$ and $AP(M_2)$, respectively. Then:*

$$\mathcal{B}((s_1, s_2), \varPsi_1 \wedge \varPsi_2) = \mathcal{B}(s_1, \varPsi_1) \times \mathcal{B}(s_2, \varPsi_2)$$

Lemma 2. *Let E_1 and E_2 be two minimal sets of finite executions. Then:*

$$\sum_{\widehat{\pi} \in E_1 \times E_2} \mathbf{p}(\widehat{\pi}) = \left(\sum_{\widehat{\pi_1} \in E_1} \mathbf{p}(\widehat{\pi_1})\right) \times \left(\sum_{\widehat{\pi_2} \in E_2} \mathbf{p}(\widehat{\pi_2})\right)$$

Now we present and prove our first compositionality theorem.

Theorem 1. *Let $M_1 \in \alpha\mathbf{PA}$, $M_2 \in \alpha\mathbf{PA}$ be $\alpha\mathbf{PA}$s such that $M_1 \diamond M_2$. Let $\varPsi_1$ and $\varPsi_2$ be LTL formulas over $AP(M_1)$ and $AP(M_2)$, respectively. Then:*

$$\mathsf{P}(M_1 \parallel M_2, \varPsi_1 \wedge \varPsi_2) = \mathsf{P}(M_1, \varPsi_1) \times \mathsf{P}(M_2, \varPsi_2)$$

Proof. The proof proceeds as follows:

$$\begin{aligned}
&\rhd \text{ using Definition 4} \\
&\mathsf{P}(M_1 \parallel M_2, \varPsi_1 \wedge \varPsi_2) = \sum_{\widehat{\pi} \in \mathcal{B}((Init_1, Init_2), \varPsi_1 \wedge \varPsi_2)} \mathbf{p}(\widehat{\pi}) \\
&\rhd \text{ using Lemma 1} \\
&= \sum_{\widehat{\pi} \in \mathcal{B}(Init_1, \varPsi_1) \times \mathcal{B}(Init_2, \varPsi_2)} \mathbf{p}(\widehat{\pi}) \\
&\rhd \text{ using Lemma 2} \\
&= \left(\sum_{\widehat{\pi_1} \in \mathcal{B}(Init_1, \varPsi_1)} \mathbf{p}(\widehat{\pi_1})\right) \times \left(\sum_{\widehat{\pi_2} \in \mathcal{B}(Init_2, \varPsi_2)} \mathbf{p}(\widehat{\pi_2})\right) \\
&\rhd \text{ again using Definition 4} \\
&= \mathsf{P}(M_1, \varPsi_1) \times \mathsf{P}(M_2, \varPsi_2)
\end{aligned}$$

□

Theorem 1 generalizes from 2 to n $\alpha\mathbf{PA}s$ as follows.

Theorem 2. *Let $M_1, \dots, M_n$ be $\alpha\mathbf{PA}s$ such that $\forall 1 \leq i < j \leq n \centerdot M_i \diamond M_j$. Let $\Psi_1, \dots, \Psi_n$ be LTL formulas over $AP(M_1), \dots, AP(M_n)$, respectively. Then:*

$$\mathsf{P}(M_1 \parallel \cdots \parallel M_n, \Psi_1 \wedge \cdots \wedge \Psi_n) = \prod_{i=1}^{n} \mathsf{P}(M_i, \Psi_i)$$

We omit the proof of Theorem 2 for brevity, and turn our attention to rewards. Let M_1 and M_2 be $\alpha\mathbf{PA}s$ and let $R_1 = (\rho_1, \iota_1)$ and $R_2 = (\rho_2, \iota_2)$ be reward structures defined on them. The composition of R_1 and R_2, denoted by $R_1 \oplus R_2$, is the reward structure (ρ, ι) on $M_1 \parallel M_2$ defined as follows:

$$\rho(s_1, s_2) = \rho_1(s_1) + \rho_2(s_2) \qquad \iota((s_1, s_2), (s_1', s_2')) = \iota_1(s_1, s_1') + \iota_2(s_2, s_2')$$

Our second compositionality theorem relates to rewards, as stated next.

Theorem 3. *Let $M_1 \in \alpha\mathbf{PA}$, $M_2 \in \alpha\mathbf{PA}$ be $\alpha\mathbf{PA}s$ such that $M_1 \diamond M_2$. Let R_1 and R_2 be reward structures M_1 and M_2, respectively. Then:*

$$\forall k \centerdot C_{\leq k}((s_1, s_2), R_1 \oplus R_2) = C_{\leq k}(s_1, R_1) + C_{\leq k}(s_2, R_2)$$

Proof. The proof is by induction on k. If $k = 0$, then it follows from Definition 5. Let $R_1 \oplus R_2 = (\rho, \iota)$, $\delta(M_1) = \delta_1$, $\delta(M_2) = \delta_2$, $\delta(M_1 \parallel M_2) = \delta$. If $k > 0$, then:

$\rhd$ using Definition 5

$$C_{\leq k}((s_1, s_2), R_1 \oplus R_2) = \rho(s_1, s_2) +$$
$$\sum_{(s_1', s_2')} \delta(s_1, s_2)(s_1', s_2') \times (\iota((s_1, s_2), (s_1', s_2')) + C_{\leq (k-1)}((s_1', s_2'), R_1 \oplus R_2))$$

$\rhd$ expanding ρ and ι and applying inductive hypothesis

$$= \rho_1(s_1) + \rho_2(s_2) +$$
$$\left(\sum_{s_1' \in S(M_1)} \sum_{s_2' \in S(M_2)} \delta_1(s_1, s_1') \times \delta_2(s_2, s_2') \times (\iota_1(s_1, s_1') + C_{\leq (k-1)}(s_1', R_1)) \right) +$$
$$\left(\sum_{s_1' \in S(M_1)} \sum_{s_2' \in S(M_2)} \delta_1(s_1, s_1') \times \delta_2(s_2, s_2') \times (\iota_2(s_2, s_2')) + C_{\leq (k-1)}(s_2', R_2)) \right)$$

$\triangleright$ rewriting

$$= \rho_1(s_1) + \rho_2(s_2) +$$

$$\underbrace{\left(\sum_{s_2' \in S(M_2)} \delta_2(s_2, s_2')\right)}_{=1} \times \left(\sum_{s_1' \in S(M_1)} \delta_1(s_1, s_1') \times (\iota_1(s_1, s_1') + C_{\leq(k-1)}(s_1', R_1))\right) +$$

$$\underbrace{\left(\sum_{s_1' \in S(M_1)} \delta_1(s_1, s_1')\right)}_{=1} \times \left(\sum_{s_2' \in S(M_2)} \delta_2(s_2, s_2') \times (\iota_2(s_2, s_2')) + C_{\leq(k-1)}(s_2', R_2))\right)$$

$$= \rho_1(s_1) + \left(\sum_{s_1' \in S(M_1)} \delta_1(s_1, s_1') \times (\iota_1(s_1, s_1') + C_{\leq(k-1)}(s_1', R_1))\right) +$$

$$\rho_2(s_2) + \left(\sum_{s_2' \in S(M_2)} \delta_2(s_2, s_2') \times (\iota_2(s_2, s_2')) + C_{\leq(k-1)}(s_2', R_2))\right)$$

$\triangleright$ using Definition 5

$$= C_{\leq k}(s_1, R_1) + C_{\leq k}(s_2, R_2)$$

□

Theorem 3 generalizes from 2 to n $\alpha\mathbf{PA}s$ as follows.

Theorem 4. *Let $M_1, \ldots, M_n$ be $\alpha\mathbf{PA}s$ such that $\forall 1 \leq i < j \leq n \mathbin{\bullet} M_i \diamond M_j$. Let $R_1, \ldots, R_n$ be reward structures over $M_1, \ldots, M_n$, respectively. Then:*

$$\forall k \mathbin{\bullet} C_{\leq k}(M_1 \parallel \cdots \parallel M_n, R_1 \oplus \ldots \oplus R_n) = \sum_{1 \leq i \leq n} C_{\leq k}(M_i, R_i)$$

We omit the proof of Theorem 4 for brevity. The power of Theorems 2 and 4 is that they enable compositional verification of $\alpha\mathbf{PA}s$. Specifically, Theorem 2 enables us to compute probabilities satisfying a conjunctive LTL formula on the composition of several $\alpha\mathbf{PA}s$ from the probabilities of satisfying individual conjuncts on each component $\alpha\mathbf{PA}$. Similarly, Theorem 4 enables us to compute rewards on the composition of several $\alpha\mathbf{PA}s$ from the individual rewards on each component $\alpha\mathbf{PA}$. This avoids having to computes the reachable statespace of the composed $\alpha\mathbf{PA}$, and therefore the statespace explosion.

In the next section, we present an example that is compositionally verifiable using Theorem 2 and Theorem 4. After that, in Section 6, we present empirical evidence about the improvement in verification due to the compositionality enabled by Theorem 2 and Theorem 4.

5 The Scenario: Robotic Demining

We consider a two-dimensional area (modeled as a grid of cells with Row rows and Col columns) randomly seeded with mines. Robots are organized into T teams, each comprising of N robots. The teams sweep the area, detect each mine, and either defuse it or (failing which) mark it. The mission succeeds if all mines are detected and defused (or marked) within a specified deadline D. The mission is parameterized not only by Row, Col, T, N, and D, but also the capabilities of each robot, the terrain, and coordination algorithm used by the robots. We first describe how each team is modeled as a $\alpha\mathbf{PA}$.

5.1 Modeling a Team

Each team has a pre-defined initial cell *cInit*, final cell *cFinal*, and a path plan P that dictates how to move cell-to-cell from *cInit* to *cFinal*. At any point, the team has a leader, and zero or more followers. In each cell, the team (specifically, the leader) first attempts to sense a mine. If a mine is detected, the leader attempts to defuse it. On successfully defusing, the team moves on to the next cell according to its path plan P. If defusing fails, then the cell is first marked as being mined, and then the team moves on to the next cell according to its path plan P. If the mine explodes (thereby destroying the leader) the followers elect a new leader using a pre-defined leader election algorithm. We are concerned with several sources of uncertainty in this scenario:

1. Due to the terrain and the quality of the leader's sensing capability, it fails to detect a mine.
2. Due to the terrain, the time required to defuse a mine varies.
3. Due to the quality of the leader's defusing capability, the mine explodes while it is being defused.
4. Due to the quality of the leader's marking capability, the mine explodes while the cell is being marked.
5. Due to communication problems, the leader election algorithm fails.
6. Due to the terrain and the team's locomotion capability, the team fails to move to the next cell in its path plan.

To express these uncertainties as part of the team's behavior, we model each team as a $\alpha\mathbf{PA}$. The $\alpha\mathbf{PA}$ is composed of two sub-$\alpha\mathbf{PA}s$ – M_{cell} corresponding to the team's behavior within a cell, and M_{step} corresponding to the team's locomotion from the current cell to the next. Figure 1(a) shows the overall $\alpha\mathbf{PA}$ for a team, and its decomposition into the two sub-$\alpha\mathbf{PA}s$ M_{cell} and M_{step}. The initial state is INIT, and the $\alpha\mathbf{PA}$ ends up in one of three possible end-states – DONE indicates that the team has covered all cells; STUCK indicates that the team is unable to move to its next cell; $\mathsf{BLOWNUP}$ indicates that the team has been destroyed by exploding mines.

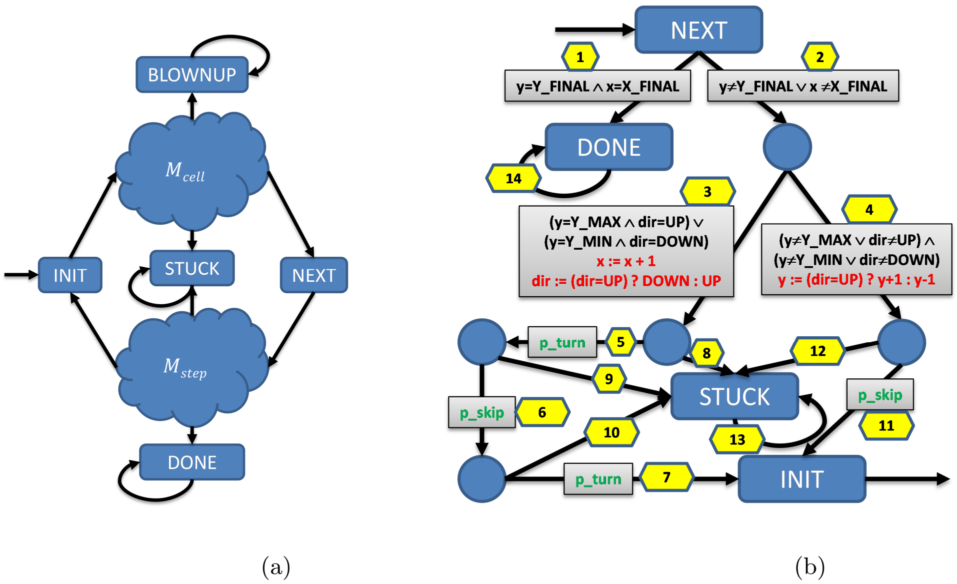

Fig. 1. (a) α**PA** for a team, and its decomposition into sub-α**PA***s* M_{cell} and M_{step}; (b) α**PA** M_{step}; transitions are numbered for ease of reference, and labeled by associated probabilities (green), guards (black) and commands (red); t_k = transition number k; TRUE guards and implied probabilities are omitted for brevity, e.g., the probability of t_1 is 1.0, the guard of t_5 is TRUE, and the probability of t_{12} is $(1 - \mathsf{p\_skip})$. Note that Y_MIN=0 and Y_MAX=Row-1. All transitions are labeled by action tick, i.e., α = tick.

αPA M_{step}. We assume that the teams follow a pre-determined path through the grid. Specifically, if there is a single team (i.e., $T = 1$), then it follows the path shown in Fig. 2(a). If $T > 1$, then each team operates independently on a distinct fragment of the path that is pre-allocated to it. For example, if $T = 4$, the starting and ending cells, and the path of each team is shown in Fig. 2(b).

Figure 1(b) shows the α**PA** M_{step}. The team maintains: (a) its current position in the grid – using variables x and y which are initialized to values (X_INIT and Y_INIT, respectively) corresponding to *cInit*; and (b) the direction of movement – using variable dir which is initialized according to P and takes two possible values UP and DOWN. All transitions are labeled by the action tick.

Let t_k mean transition number k in Figure 1(b). From the initial state NEXT, the team first checks if it has reached *cFinal*. In this case (t_1), the team moves to state DONE and stutters (t_{14}). Otherwise, the team attempts to move to the next cell (t_2). This involves two cases: (a) the team moves to the next column (t_3) which involves two turns (t_5, t_7), a skip (t_6), and a change in direction; or (b) the team moves to the next row (t_4) which involves just a skip (t_{11}). Skips and turns succeed with probability p_skip and p_turn, respectively. These probabilities are determined by the terrain and the team's locomotion capability,

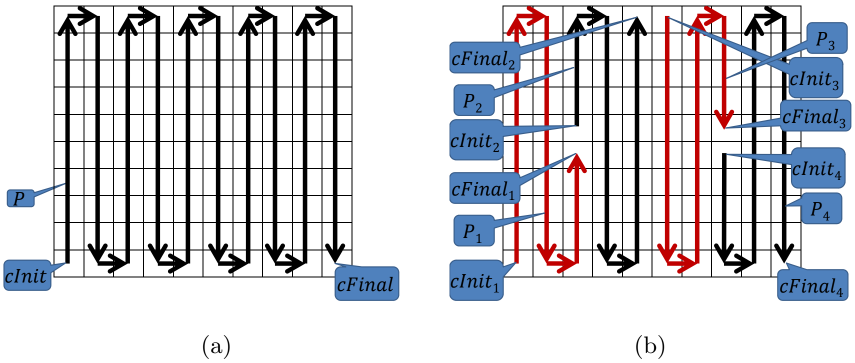

Fig. 2. Path followed by the teams: (a) path with one team; (b) path with four teams; $cInit_i$, $cFinal_i$, and P_i are the starting cell, ending cell, and path plan for i-th team.

as discussed later. If a skip or a turn fails, the team moves to a STUCK state (t_8, t_9, t_{10}, t_{12}) and stutters (t_{13}).

αPA M_{cell}. The α**PA** M_{cell} is shown in Fig. 3. We model whether a mine was missed using variable failed, initialized to FALSE. We also model the number of remaining robots in the team using variable sz, initialized to N. In the following, t_k means the transition labeled k in Fig. 3. The teams begins in state INIT and the leader attempts to detect a mine. The result of mine detection is either an explosion with probability p_explode_detect (t_2), a mine found with probability p_detect_mine (t_1), or no mine found (t_3).

If no mine was detected (state NOT_DETECTED), then we assume that with probability p_false_neg, there is actually a mine. In this case, with equal likelihood, the leader either explodes (t_4) or the team moves to the next cell (t_5). In the latter case, we indicate mission failure (since a mine has been missed) by setting failed to TRUE. Finally, with probability (1 - p_false_neg), the team moves to the next cell (t_6), continuing with its mission. The probability p_false_neg is a function of the leader's detecting capability and the terrain, as discussed later.

If a mine was detected, the leader attempts to defuse it. We assume that the leader is in one of three defusing situations with increasing difficulty – easy, medium and hard. Initially (DEFUSE1), the leader assumes that it is in the easy defusing situation. The result is either an explosion with probability (p_d1 $\times$ p_ed1) (t_8), successful defusing of the mine with probability (p_d1 $\times (1 -$ p_ed1)) (t_7), or a decision to move on to the medium defusing scenario (t_9). Here, p_d1 is the probability that the leader is actually in an easy defusing situation, and p_ed1 is the probability that there is an explosion given that the leader is trying to defuse in an easy situation. As discussed later, while p_d1 is a function of the terrain, p_ed1 is a function of the leader's defusing capability.

In the medium defusing scenario (DEFUSE2), the leader either blows up (t_{11}), successfully defuses the mine (t_{10}), or moves to the hard defusing scenario

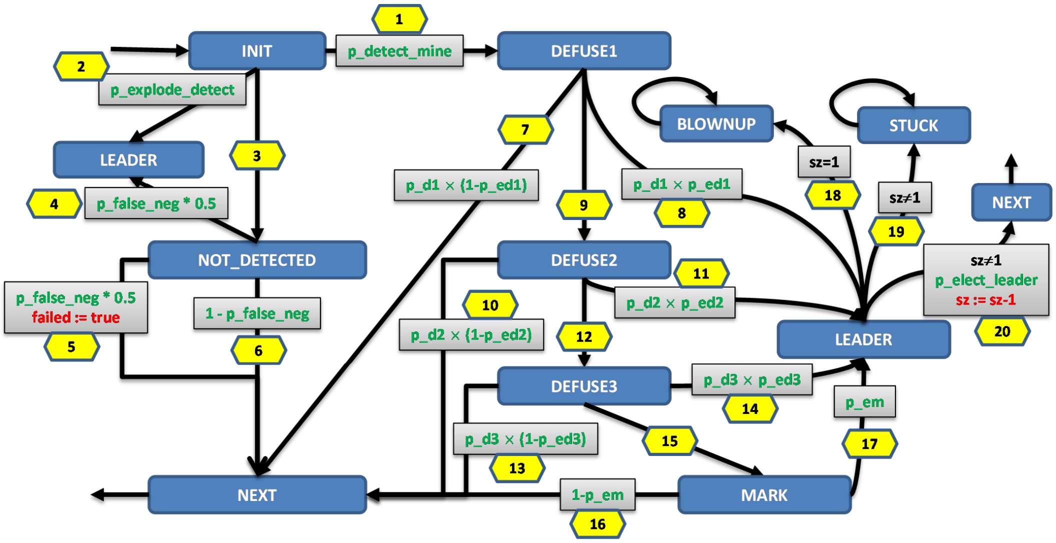

Fig. 3. α**PA** M_{cell}; transitions are numbered and labeled, and guards and probabilities are omitted as in Figure 1(b); states LEADER and NEXT are repeated to reduce clutter; all transitions are labeled by action tick, i.e., $\alpha =$ tick.

(t_{12}). The probabilities involved in this step are: p_d2 – the terrain-dependent probability that the leader is actually in a medium defusing situation, and p_ed2 – the probability (dependent on the leader's defusing capability) that there is an explosion given that it is trying to defuse in a medium situation.

In the hard defusing scenario (DEFUSE3), the leader either blows up (t_{14}), successfully defuses the mine (t_{13}), or attempts to mark the cell (t_{15}) as being mined. The probabilities involved in this step are: p_d3 – the terrain-dependent probability that the leader is actually in a hard defusing situation, and p_ed3 – the probability (dependent on the leader's defusing capability) that there is an explosion given that it is trying to defuse in a hard situation.

Finally, when the leader attempts to mark the cell, it either blows up (t_{17}) with probability p_em, or succeeds (t_{16}) and the team continues to the next cell. The probability p_em of an explosion during the marking operation is a function of the leader's defusing capability, as discussed later.

If the leader blows up, the team elects a new leader from state LEADER. If there are no remaining robots in the team (i.e., sz=0), the team moves to BLOWNUP (t_{18}) and stutters. Otherwise, with probability p_elect_leader, a new leader is elected successfully and the team moves on to the next cell (t_{20}), and with probability (1 - p_elect_leader) leader election fails and the team moves to STUCK (t_{19}) and stutters.

5.2 Team αPA Parameters

The $\alpha\mathbf{PA}$ for a team is parameterized by the following:

1. The number of robots N, and the coordinates for *cInit* and *cFinal*.
2. The probability ($\mathsf{p\_detect\_mine}$) of detecting a mine in a cell.
3. The probability ($\mathsf{p\_elect\_leader}$) of successful leader election.
4. The remaining probabilities were computed from the terrain and the robot's capabilities as discussed next.

Modeling Terrain and Robot Capabilities. The robot's mine detection capability was modeled by a parameter DET with three possible values – LOW, MEDIUM and HIGH. The robot's mine defusing capability was modeled by a parameter DEF with three possible values – LOW, MEDIUM and HIGH. The robot's locomotion capability was modeled by a parameter LOC with three possible values – LOW, MEDIUM and HIGH. The terrain was modeled by eighteen independent parameters: (i) p_fn_dc0, p_fn_dc1 and p_fn_dc2 are the probabilities of a false negative (i.e., mine present but not detected) given that DET = LOW, MEDIUM and HIGH, respectively; (ii) p_d1, p_d2 and p_d3 are the probabilities of being in an easy, medium, or hard defusing situation, respectively; (iii) p_edet_dc0, p_edet_dc1 and p_edet_dc2 are the probabilities of an explosion during mine detection given that DET = LOW, MEDIUM and HIGH, respectively; (iv) p_edef_dc0, p_edef_dc1 and p_edef_dc2 are the probabilities of an explosion during mine defusing given that DEF = LOW, MEDIUM and HIGH, respectively; (v) p_skip_lc0, p_skip_lc1 and p_skip_lc2 are the probabilities of successful skip given that LOC = LOW, MEDIUM and HIGH, respectively; and (vi) p_turn_lc0, p_turn_lc1 and p_turn_lc2 are the probabilities of successful turn given that LOC = LOW, MEDIUM and HIGH, respectively. For our experiments, all terrain parameters were assigned constant values, but in practice we expect that these atomistic probabilities will be obtained empirically.

Remaining Probabilities. The probability of a false negative in Fig. 3 are computed as follows:

$$\mathsf{p\_false\_neg} = \begin{cases} \mathsf{p\_fn\_dc0} \text{ if } \mathsf{DET} = \mathsf{LOW}, \\ \mathsf{p\_fn\_dc1} \text{ if } \mathsf{DET} = \mathsf{MEDIUM}, \\ \mathsf{p\_fn\_dc2} \text{ if } \mathsf{DET} = \mathsf{HIGH}. \end{cases}$$

The probability of an explosion while detecting a mine is computed as follows:

$$\mathsf{p\_explode\_detect} = \begin{cases} \mathsf{p\_edet\_dc0} \text{ if } \mathsf{DET} = \mathsf{LOW}, \\ \mathsf{p\_edet\_dc1} \text{ if } \mathsf{DET} = \mathsf{MEDIUM}, \\ \mathsf{p\_edet\_dc2} \text{ if } \mathsf{DET} = \mathsf{HIGH}. \end{cases}$$

The probabilities of an explosion while defusing or marking a cell are computed as follows:

$$\mathsf{p\_ed1} = \mathsf{p\_ed2} = \mathsf{p\_ed3} = \mathsf{p\_em} = \begin{cases} \mathsf{p\_edef\_dc0} \text{ if } \mathsf{DEF} = \mathsf{LOW}, \\ \mathsf{p\_edef\_dc1} \text{ if } \mathsf{DEF} = \mathsf{MEDIUM}, \\ \mathsf{p\_edef\_dc2} \text{ if } \mathsf{DEF} = \mathsf{HIGH}. \end{cases}$$

The probability of successful skip is computed as follows:

$$\mathsf{p\_skip} = \begin{cases} \mathsf{p\_skip\_lc0} \text{ if } \mathsf{LOC} = \mathsf{LOW}, \\ \mathsf{p\_skip\_lc1} \text{ if } \mathsf{LOC} = \mathsf{MEDIUM}, \\ \mathsf{p\_skip\_lc2} \text{ if } \mathsf{LOC} = \mathsf{HIGH}. \end{cases}$$

Finally, the probability of successful turn is computed as follows:

$$\mathsf{p\_turn} = \begin{cases} \mathsf{p\_turn\_lc0} \text{ if } \mathsf{LOC} = \mathsf{LOW}, \\ \mathsf{p\_turn\_lc1} \text{ if } \mathsf{LOC} = \mathsf{MEDIUM}, \\ \mathsf{p\_turn\_lc2} \text{ if } \mathsf{LOC} = \mathsf{HIGH}. \end{cases}$$

5.3 Multiple Teams and Properties

Let $M_i = (S_i, Init_i, \Sigma_i, \delta_i, AP_i, \mathcal{L}_i)$ be the $\alpha\mathbf{PA}$ for the i-th team. Recall that all transitions in the $\alpha\mathbf{PA}$ for a team are labeled by the action tick, i.e., $\Sigma_i = \{\mathsf{tick}\}$. A state of M_i is a valuation to the variables x, y, dir, failed, sz and pc, where pc is the *program counter* whose value indicates the position of the $\alpha\mathbf{PA}$ w.r.t. the state machines in Figure 1(b) and Figure 3. For example, $\mathsf{pc} = \mathsf{LEADER}$ means that the team is about to elect a new leader. Note that all variables have a finite domain, hence S_i is finite as well. In addition, M_i has three atomic propositions: (i) $done_i$ which is true in all states where $\mathsf{pc} = \mathsf{DONE}$; (ii) $succ_i$ which is true in all states where $\mathsf{failed} = \textsc{false}$; and (iii) $init_i$ which is true in all states where $\mathsf{pc} = \mathsf{INIT}$. Now consider a scenario with T teams. Clearly, the $\alpha\mathbf{PA}s$ $M_1, \ldots, M_T$ satisfy the conditions of Theorem 2.

Success. The first property we consider is true for all executions where all teams cover all their cells without missing a single mine within a deadline D. Let us write $\mathsf{F}^{\leq k}\Psi$ to mean $\Psi \vee \mathsf{X}\Psi \vee \mathsf{XX}\Psi \vee \cdots \vee \underbrace{\mathsf{XX}\ldots\mathsf{X}}_{k \text{ times}}\Psi$. Then our first property is expressed by the following path formula:

$$success_D \equiv (\mathsf{F}^{\leq D}(done_1 \wedge succ_1)) \wedge \cdots \wedge (\mathsf{F}^{\leq D}(done_T \wedge succ_T))$$

Note that $success_D$ satisfies the conditions of Theorem 2.

Coverage. The second property we consider is coverage. Informally, this is the number of cells processed by all the teams within a deadline D. Formally, it is expressed as the cumulative reward:

$$coverage_D \equiv C_{\leq D}((Init_1, \ldots, Init_T), R_1 \oplus \ldots \oplus R_T)$$

where, for $1 \leq i \leq T$, $R_i = (\rho_i, \iota_i)$ is the reward structure such that:

$$\forall s \in S_i \centerdot \rho_i(s) = 1 \text{ if } Init_i \in \mathcal{L}_i(s) \text{ and } \rho_i(s) = 0 \text{ otherwise}$$
$$\forall (s, s') \in S_i \times S_i \centerdot \iota_i(s, s') = 0$$

In other words, R_i assigns a reward 1 whenever the i-th team enters a new cell.

6 Experiments

We performed a set of experiments using the $\alpha\mathbf{PA}$ model of the robotic demining scenario presented in Section 5. The goal was to demonstrate the suitability of our approach to make appropriate tradeoff decisions when designing robotic missions, and to demonstrate the effectiveness of our compositionality theorem in improving scalability. All our experiments were performed on an Intel Core i7 machine with four cores (each running at 2.7GHz) and 8GB of RAM. We used a timeout of 1800s, and fixed certain parameters as follows:

p_fn_dc0 = 0.05	p_fn_dc1 = 0.01	p_fn_dc2 = 0.005
p_d1 = 0.25	p_d2 = 0.33	p_d3 = 0.5
p_edet_dc0 = 0.05	p_edet_dc1 = 0.04	p_edet_dc2 = 0.03
p_edef_dc0 = 0.05	p_edef_dc1 = 0.04	p_edef_dc2 = 0.03
p_skip_lc0 = 0.9999	p_skip_lc1 = 0.99999	p_skip_lc2 = 0.999999
p_turn_lc0 = 0.9999	p_turn_lc1 = 0.99999	p_turn_lc2 = 0.999999
p_detect_mine = 0.5	$D = 250$	p_elect_leader = 0.9

We also set Row=10, and Col=12. Other parameters were varied based on the experiment. We used PRISM version 4.0.3, which was the latest version available at the start of this project. We modeled $\alpha\mathbf{PA}s$ in PRISM as DTMCs with every transition labeled by an action called tick. Thus, the default synchronization semantics for DTMCs used by PRISM coincided with the semantics of composition of $\alpha\mathbf{PA}s$. All our PRISM models, results, as well as instructions to reproduce them are available at `www.contrib.andrew.cmu.edu/~schaki/discover/spin13.tgz`.

Experiments about success. The first set of experiments were designed to evaluate the impact of DET, DEF, LOC, T and N on $\mathit{success}_D$. The results are summarized in Table 1. We consider eight possible combinations of DET, DEF, and LOC.

The first five rows are the values of $\mathit{success}_D$ for each of these eight combinations using $T = 2$ and different values of N. We observe that changing DET from LOW to HIGH has a much bigger impact on the value of $\mathit{success}_D$ compared to changing DEF or LOC. This suggests that using robots with good mine detection capability should be of high priority during mission design.

The next five rows show the value of $\mathit{success}_D$ with different values of T and N such that $T \times N = 30$, i.e., different team configurations with 30 robots. They indicate that three teams with ten robots each provide optimal values of $\mathit{success}_D$. Note that $\mathit{success}_D$ drops off sharply for $N < 5$ since small teams have a high chance of being blown up completely before mission completion.

The final column shows the average time required to compute $\mathit{success}_D$ over all eight combinations of DET, DEF, and LOC considered. The average is a good indicator since the standard deviation was quite low. These times were measured when we performed our experiments *compositionally*, i.e., computing $\mathit{success}_D$

T	N	$success_D$								Time
		LLL	LLH	LHL	LHH	HLL	HLH	HHL	HHH	seconds
2	2	0.000	0.000	0.000	0.000	0.013	0.014	0.035	0.035	21
2	3	0.001	0.001	0.003	0.004	0.065	0.066	0.129	0.131	26
2	5	0.018	0.018	0.030	0.031	0.256	0.259	0.355	0.359	38
2	10	0.073	0.074	0.086	0.087	0.386	0.391	0.443	0.449	62
2	15	0.076	0.077	0.087	0.089	0.386	0.391	0.443	0.449	87
3	10	0.088	0.090	0.100	0.101	0.435	0.441	0.491	0.498	46
6	5	0.080	0.081	0.094	0.095	0.429	0.434	0.488	0.494	29
10	3	0.046	0.047	0.062	0.063	0.354	0.359	0.435	0.441	35
15	2	0.011	0.012	0.020	0.020	0.175	0.177	0.261	0.264	48
30	1	0.000	0.000	0.000	0.000	0.001	0.001	0.003	0.003	100

Table 1. Results for $success_D$ with different T, N, DET, DEF and LOC; second row entries indicate values of DET, DEF and LOC; e.g., LLL = (DET=LOW, DEF=LOW, LOC=LOW); LHL = (DET=LOW, DEF=HIGH, LOC=LOW), etc.; Time = average time to compute $success_D$ over all combinations of DET, DEF and LOC.

for each team individually, and multiplying the results (in accordance with Theorem 2). When we used the monolithic approach, i.e., all teams composed in the same model, PRISM timed out at 1800 seconds in all cases.

Experiments about coverage. The next set of experiments were designed to evaluate the impact of DET, DEF, LOC, T and N on $coverage_D$. The results are summarized in Table 2. Each cell of the table corresponds to the same values of DET, DEF, LOC, T and N as in the corresponding cell in Table 1.

Not surprisingly, we again observe that changing DET from LOW to HIGH has a much bigger impact on the value of $coverage_D$ compared to changing DEF or LOC. This suggests that using robots with good mine detection capability is a good tradeoff for not only $success_D$, but $coverage_D$ as well.

The results for different values of T (last five rows of Table 1) are somewhat different. The optimal $coverage_D$ is observed for ten teams with three robots each. This reflects a subtle difference between $coverage_D$ and $success_D$ – a cell is covered as soon as the team reaches it, but that does not contribute to success unless the team avoids being blown up as well. In general, the benefit of smaller teams extends further for $coverage_D$ simply because more teams are able to "reach" more cells even if they get blown up. However, for $T > 15$, even $coverage_D$ falls off.

The final column shows the average time required to compute $coverage_D$ over all eight combinations of DET, DEF, and LOC considered. Once again, these times are for the compositional approach, i.e., computing $coverage_D$ for each team individually, and adding the results (in accordance with Theorem 4). For the monolithic approach, PRISM timed out at 1800 seconds in all cases.

T	N	DET:DEF:LOC								Time
		LLL	LLH	LHL	LHH	HLL	HLH	HHL	HHH	seconds
2	2	43.3	43.3	48.3	48.4	61.8	62.0	70.6	70.8	7
2	3	60.1	60.2	66.2	66.4	80.8	81.0	89.1	89.3	7
2	5	82.1	82.4	87.5	87.7	97.8	98.1	102.5	102.8	7
2	10	93.5	93.8	96.2	96.5	101.7	102.0	104.6	105.0	7
2	15	93.6	93.9	96.2	96.5	101.7	102.0	104.6	105.0	7
3	10	101.6	101.8	103.5	103.8	107.5	107.7	109.6	109.8	9
6	5	110.3	110.4	111.4	111.5	113.6	113.8	114.7	114.9	16
10	3	112.9	113.0	113.9	114.0	115.8	115.9	116.6	116.7	25
15	2	112.4	112.4	113.6	113.6	115.8	115.8	116.7	116.8	37
30	1	105.3	105.3	106.9	107.0	110.2	110.3	111.9	111.9	84

Table 2. Results for $coverage_D$ with different T, N, DET, DEF and LOC; second row entries indicate values of DET, DEF and LOC; e.g., LLL = (DET=LOW, DEF=LOW, LOC=LOW); LHL = (DET=LOW, DEF=HIGH, LOC=LOW), etc.; Time = average time to compute $coverage_D$ over all combinations of DET, DEF and LOC.

7 Conclusion

We present an approach to compute quantitative utility of robotic missions using probabilistic model checking. We show how to express a robotic demining mission as a $\alpha\mathbf{PA}$, its success as a LTL formula, and its coverage as a reward. We prove a set of compositionality theorems that enable us to compute the success probability (or, coverage) of a system composed of several $\alpha\mathbf{PA}s$ by combining the success probability (or, coverage) of each $\alpha\mathbf{PA}$ in isolation. This ameliorates the statespace explosion problem, even when the system being verified is composed of many $\alpha\mathbf{PA}s$. We validate our approach empirically, using the probabilistic model checker PRISM for our experiments.

We envision building on this work in several directions. One issue is that our model for the demining mission is based on several atomistic probabilities (e.g., p_fn_dc0). We assume that these probabilities are available with sufficient accuracy. Otherwise, the predictions made via probabilistic model checking will be correspondingly inaccurate. As part of our ongoing work, we are developing ways to estimate these probabilities via field experiments. Another direction is to adapt probabilistic model checking to create a more generative approach – one that constructs an optimal mission – that can handle an expressive range of mission configurations and constraints.

Acknowledgements. We thank Anvesh Komuravelli and Arie Gurfinkel for many helpful discussions and comments that helped shape this work[1].

[1] This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN AS-IS BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREE-

References

1. C. Baier. *On algorithmic verification methods for probabilistic systems.* PhD thesis, University of Mannheim, 1998. Habilitation thesis.
2. S. Chaki, J. M. Dolan, and J. A. Giampapa. Toward A Quantitative Method for Assuring Coordinated Autonomy. In *Proc. of ARMS Workshop*, 2013. to appear.
3. T. Chen, M. Diciolla, M. Z. Kwiatkowska, and A. Mereacre. Quantitative Verification of Implantable Cardiac Pacemakers. In *Proc. of RTSS*, 2012.
4. L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional Methods for Probabilistic Systems. In *Proc. of CONCUR*, 2001.
5. L. Feng, T. Han, M. Z. Kwiatkowska, and D. Parker. Learning-Based Compositional Verification for Synchronous Probabilistic Systems. In *Proc. of ATVA*, 2011.
6. L. Feng, M. Z. Kwiatkowska, and D. Parker. Compositional Verification of Probabilistic Systems Using Learning. In *Proc. of QEST*, 2010.
7. L. Feng, M. Z. Kwiatkowska, and D. Parker. Automated Learning of Probabilistic Assumptions for Compositional Reasoning. In *Proc. of FASE*, 2011.
8. J. Heath, M. Z. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science (TCS)*, 391(3), 2008.
9. A. Komuravelli, C. S. Pasareanu, and E. M. Clarke. Assume-Guarantee Abstraction Refinement for Probabilistic Systems. In *Proc. of CAV*, 2012.
10. J. A. Kumar and S. Vasudevan. Automatic Compositional Reasoning for Probabilistic Model Checking of Hardware Designs. In *Proc. of QEST*, 2010.
11. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Proc. of CAV*, 2011.
12. M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-Guarantee Verification for Probabilistic Systems. In *Proc. of TACAS*, 2010.
13. M. Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. *Formal Aspects of Computing (FACJ)*, 14(3), 2003.
14. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Massachusetts Institute of Technology, 1995. Available as Technical Report MIT/LCS/TR-676.
15. M. Stoelinga. *Alea jacta est: verification of probabilistic, real-time and parametric systems.* PhD thesis, University of Nijmegen, 2002. Available via `http://www.soe.ucsc.edu/~marielle`.
16. G. Sukthankar and K. Sycara. Team-aware Robotic Demining Agents for Military Simulation. `http://www.cs.cmu.edu/~softagents/iaai00/iaai00.html`.
17. M. Y. Vardi. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *Proc. of FOCS*, 1985.

A Proof of Lemma 1

We begin with another lemma.

Lemma 3. *Let $E_1 \subseteq \widehat{Ex}(s_1)$ and $E_2 \subseteq \widehat{Ex}(s_2)$ be two minimal sets of finite executions. Then $E_1 \times E_2 \subseteq \widehat{Ex}(s_1, s_2)$ is also minimal.*

Proof. By contradiction. Assume $\pi, \pi' \in E_1 \times E_2$ such that $\pi \sqsubset \pi'$ (i.e., $\pi \sqsubseteq \pi' \wedge \pi \neq \pi'$). Then:

$$\exists \pi_1, \pi_1' \in E_1 \bullet \exists \pi_2, \pi_2' \in E_2 \bullet \pi \in \pi_1 \times \pi_2 \wedge \pi' \in \pi_1' \times \pi_2' \tag{1}$$

Note that:

$$(\pi_1 \neq \pi_1') \vee (\pi_2 \neq \pi_2') \tag{2}$$

otherwise, we cannot have $\pi \sqsubset \pi'$. Without loss of generality, assume $\pi_1 \neq \pi_1'$. Also, from (1) we have:

$$(\pi_1 \sqsubseteq \pi \downharpoonleft 1) \wedge (\pi_1' \sqsubseteq \pi' \downharpoonleft 1) \tag{3}$$

Again since $\pi \sqsubset \pi'$, we have:

$$(\pi \downharpoonleft 1 \sqsubset \pi' \downharpoonleft 1) \tag{4}$$

Combining (3) and (4) we have:

$$(\pi_1 \sqsubseteq \pi' \downharpoonleft 1) \wedge (\pi_1' \sqsubseteq \pi' \downharpoonleft 1) \tag{5}$$

Again, since we assumed $\pi_1 \neq \pi_1'$, from (5) we have:

$$(\pi_1 \sqsubset \pi_1') \vee (\pi_1' \sqsubset \pi_1) \tag{6}$$

which contradicts the assumption that E_1 is minimal. □

Lemma 1. *Let $M_1 \in \alpha\mathbf{PA}$, $M_2 \in \alpha\mathbf{PA}$ be $\alpha\mathbf{PA}$s such that $M_1 \diamond M_2$. Let $s_1 \in S(M_1)$, $s_2 \in S(M_2)$, and Ψ_1 and Ψ_2 be LTL formulas over $AP(M_1)$ and $AP(M_2)$, respectively. Then:*

$$\mathcal{B}((s_1, s_2), \Psi_1 \wedge \Psi_2) = \mathcal{B}(s_1, \Psi_1) \times \mathcal{B}(s_2, \Psi_2)$$

Proof. Since $\mathcal{B}(s_1, \Psi_1)$ and $\mathcal{B}(s_2, \Psi_2)$ are minimal, by Lemma 3, $\mathcal{B}(s_1, \Psi_1) \times \mathcal{B}(s_2, \Psi_2)$ is also minimal. So we only have to prove that:

$$\begin{aligned}\bigcup_{\widehat{\pi} \in \mathcal{B}(s_1, \Psi_1) \times \mathcal{B}(s_2, \Psi_2)} \mathsf{Cyl}(\widehat{\pi}) &= \textstyle\bigcup_{\widehat{\pi} \in \mathcal{B}((s_1, s_2), \Psi_1 \wedge \Psi_2)} \mathsf{Cyl}(\widehat{\pi}) \\ &= \{\pi \in Ex(s_1, s_2) \mid \pi \models \Psi_1 \wedge \Psi_2\}\end{aligned}$$

Let $\pi \in Ex(s_1, s_2)$. Then:

$$\begin{aligned}
&\pi \models \Psi_1 \wedge \Psi_2 \iff (\pi \downharpoonleft 1 \models \Psi_1) \wedge (\pi \downharpoonleft 2 \models \Psi_2) \\
&\iff (\exists \widehat{\pi_1} \in \mathcal{B}(s_1, \Psi_1) \centerdot \pi \downharpoonleft 1 \in \mathsf{Cyl}(\widehat{\pi_1})) \wedge (\exists \widehat{\pi_2} \in \mathcal{B}(s_2, \Psi_2) \centerdot \pi \downharpoonleft 2 \in \mathsf{Cyl}(\widehat{\pi_2})) \\
&\iff \exists \widehat{\pi} \in \mathcal{B}(s_1, \Psi_1) \times \mathcal{B}(s_2, \Psi_2) \centerdot \pi \in \mathsf{Cyl}(\widehat{\pi})
\end{aligned}$$

Therefore:

$$\bigcup_{\widehat{\pi} \in \mathcal{B}(s_1, \Psi_1) \times \mathcal{B}(s_2, \Psi_2)} \mathsf{Cyl}(\widehat{\pi}) = \{\pi \in Ex(s_1, s_2) \mid \pi \models \Psi_1 \wedge \Psi_2\}$$

□

B Proof of Lemma 2

Lemma 2. *Let E_1 and E_2 be two minimal sets of finite executions. Then:*

$$\sum_{\widehat{\pi} \in E_1 \times E_2} \mathbf{p}(\widehat{\pi}) = \left(\sum_{\widehat{\pi_1} \in E_1} \mathbf{p}(\widehat{\pi_1}) \right) \times \left(\sum_{\widehat{\pi_2} \in E_2} \mathbf{p}(\widehat{\pi_2}) \right)$$

Proof.

$\rhd$ since E_1 and E_2 are minimal

$$\sum_{\widehat{\pi} \in E_1 \times E_2} \mathbf{p}(\widehat{\pi}) = \sum_{\widehat{\pi_1} \in E_1} \sum_{\widehat{\pi_2} \in E_2} \sum_{\pi \in \widehat{\pi_1} \times \widehat{\pi_2}} \mathbf{p}(\pi)$$

$\rhd$ let $n = \mathsf{max}(|\widehat{\pi_1}|, |\widehat{\pi_2}|)$

$$= \sum_{\widehat{\pi_1} \in E_1} \sum_{\widehat{\pi_2} \in E_2} \sum_{\widehat{\pi'_1} \in \widehat{\pi_1}^{+n}} \sum_{\widehat{\pi'_2} \in \widehat{\pi_2}^{+n}} \mathbf{p}(\widehat{\pi'_1}) \times \mathbf{p}(\widehat{\pi'_2})$$

$\rhd$ rewriting

$$= \sum_{\widehat{\pi_1} \in E_1} \sum_{\widehat{\pi_2} \in E_2} \left(\left(\sum_{\widehat{\pi'_1} \in \widehat{\pi_1}^{+n}} \mathbf{p}(\widehat{\pi'_1}) \right) \times \left(\sum_{\widehat{\pi'_2} \in \widehat{\pi_2}^{+n}} \mathbf{p}(\widehat{\pi'_2}) \right) \right)$$

$\rhd$ rewriting

$$= \sum_{\widehat{\pi_1} \in E_1} \sum_{\widehat{\pi_2} \in E_2} (\mathbf{p}(\widehat{\pi_1}) \times \mathbf{p}(\widehat{\pi_2}))$$

$\rhd$ rewriting

$$= \left(\sum_{\widehat{\pi_1} \in E_1} \mathbf{p}(\widehat{\pi_1}) \right) \times \left(\sum_{\widehat{\pi_2} \in E_2} \mathbf{p}(\widehat{\pi_2}) \right)$$

□