A Counter Abstraction Technique for the Verification of Probabilistic Swarm Systems

Alessio Lomuscio and Edoardo Pirovano

Imperial College London, UK

18 September 2019
Highlights 2019

Based on a paper at AAMAS19

Robot Swarms

Introduction

- Swarms of drones ("agents") follow certain protocols to achieve an overall goal, e.g., search and rescue, maintenance, etc.
- It is desirable to formally verify they function correctly.
- These protocols are sometimes probabilistic.
- Existing research allows us to verify finite probabilistic systems with a fixed number of agents [KDF12].
- Parameterised model checking techniques allow us to verify systems with an unbounded number of agents [KL15].

Contribution: A method to verify probabilistic MAS with an unbounded number of agents against temporal specs.

Our Contribution

1. We present a novel semantics to reason about **probabilistic** systems with an **unbounded** number of agents.
2. We present a *partial* decision procedure, based on counter abstraction [PXZ02], for verification against probabilistic temporal specifications.
3. We present an open-source implementation of this procedure and evaluate it on an example.

Probabilistic Swarm Systems

- Agent-based semantics.
- It extends Parameterised Interleaved Interpreted Systems [KL15] to include probabilities.
- We assume that all agents are behaviourally identical (can be extended to a finite number of different agent behaviours).
- The agents synchronize with each-other in different ways depending on the type of action being performed:
  - *Asynchronous:* Performed by one agent on its own.
  - *Agent-environment:* Performed by *one* agent together with the environment.
  - *Global synchronous:* Performed by *all* agents together with the environment.

Agents and Environments

Definition (Probabilistic Agent Template)

A *probabilistic agent template* is a tuple $T = \langle S, \iota, Act, P, t \rangle$ where:

- The set S represents a finite set of agent local states.
- $\iota \in S$ is a distinguished initial state.
- The non-empty set $Act = A \cup AE \cup GS$ defines the actions that can be performed by the agents.
- The agent's protocol function $P : S \to \mathcal{P}(Act) \setminus \{\emptyset\}$ defines the actions available in each state.
- The agent's transition function $t : S \times Act \times S \to [0, 1]$ is such that for every $s \in S$ and $a \in P(s)$ we have $\sum_{s' \in S} t(s, a, s') = 1$. This defines the probabilistic next state given the current state and action.

- The environment is similarly defined.

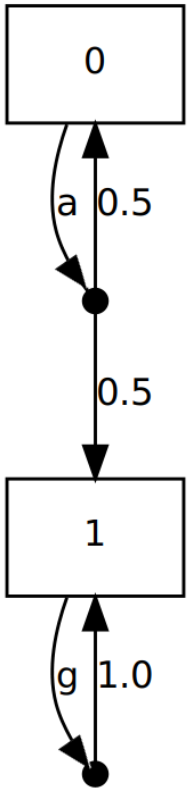
Probabilistic Parmeterised interleaved interpreted systems

Definition (PPIIS)

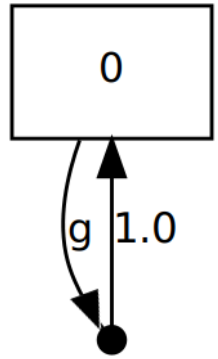
A *probabilistic parameterised interleaved interpreted system* (PPIIS) is a tuple $\mathcal{S} = \langle T, E, L \rangle$, where T is a probabilistic agent template, E is an environment and $L : S \times S_E \to \mathcal{P}(AP)$ is a labelling function for a set of atomic propositions AP.

- Each such parameterised system defines an infinite family of MDPs made by setting a different number of agents.
- We denote by $\mathcal{S}(n)$ the concrete system with n agents; notice each of these is a finite MDP.

Example PPIIS



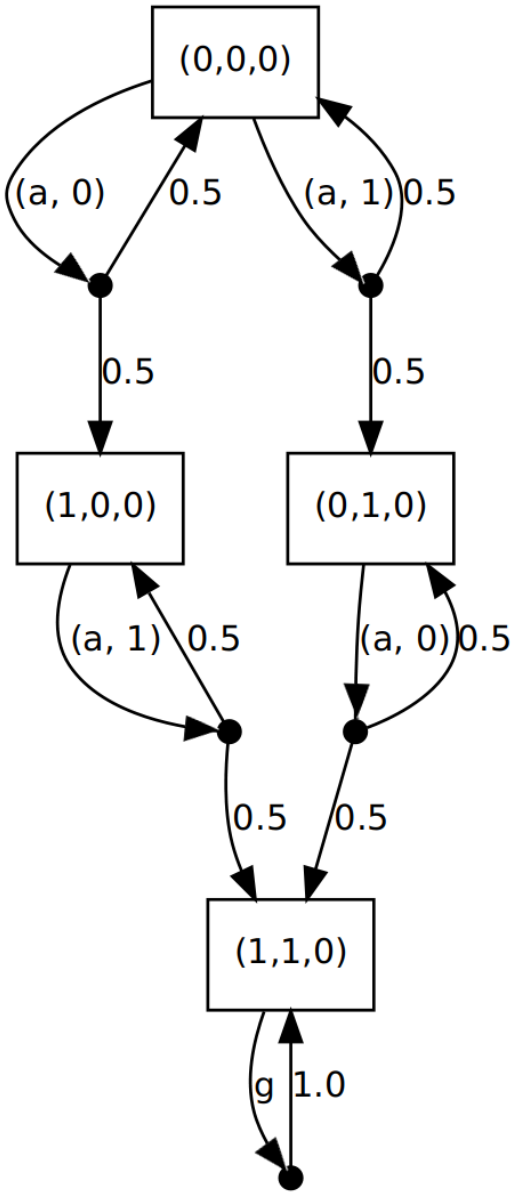
(a) An agent template.



(b) An environment.

Figure: An example PPIIS. The a action is asynchronous, while the g one is global synchronous.

Example Concrete System



Probabilistic LTL

Definition (PLTL)

For $a \in AP$ and $i \in \mathbb{N}$, the probabilistic LTL logic is the set of formulas ϕ defined by the following BNF:

$$\phi ::= P^{\max}_{\bowtie x}[\psi] \mid P^{\min}_{\bowtie x}[\psi] \text{ for } x \in [0,1] \text{ and } \bowtie \in \{\leq, <, \geq, >\}$$
$$\psi ::= \top \mid (a,i) \mid \neg\psi \mid \psi \wedge \psi \mid X\psi \mid \psi \; U \; \psi$$

- The formula $P^{\max}_{\leq x}[\psi]$ is read as "with a scheduler (choice of action for each state) that maximises the probability of ψ occurring, this probability is $\leq x$."

Definition

We say a formula is m-**indexed** if it refers to agents with index at most m (ie. all atomic propositions in the formula are of the form (a,i) for $i \leq m$).

PLTL Examples

Example

The PLTL formula $P^{\max}_{\leq 0.3}[F(\text{win}, 1)]$ is a 1-indexed formula representing that, no matter what choices the scheduler makes, the probability of agent 1 reaching a state where win holds does not exceed 0.3.

Example

The PLTL formula $P^{\min}_{< 0.9}[G(\text{alive}, 2)]$ is a 2-indexed formula stating that, even if the scheduler tries to minimise the probability that agent 2 is always in a state where alive holds, then this probability remains below 0.9.

Parameterised Model Checking

Definition (Parameterised Model Checking)

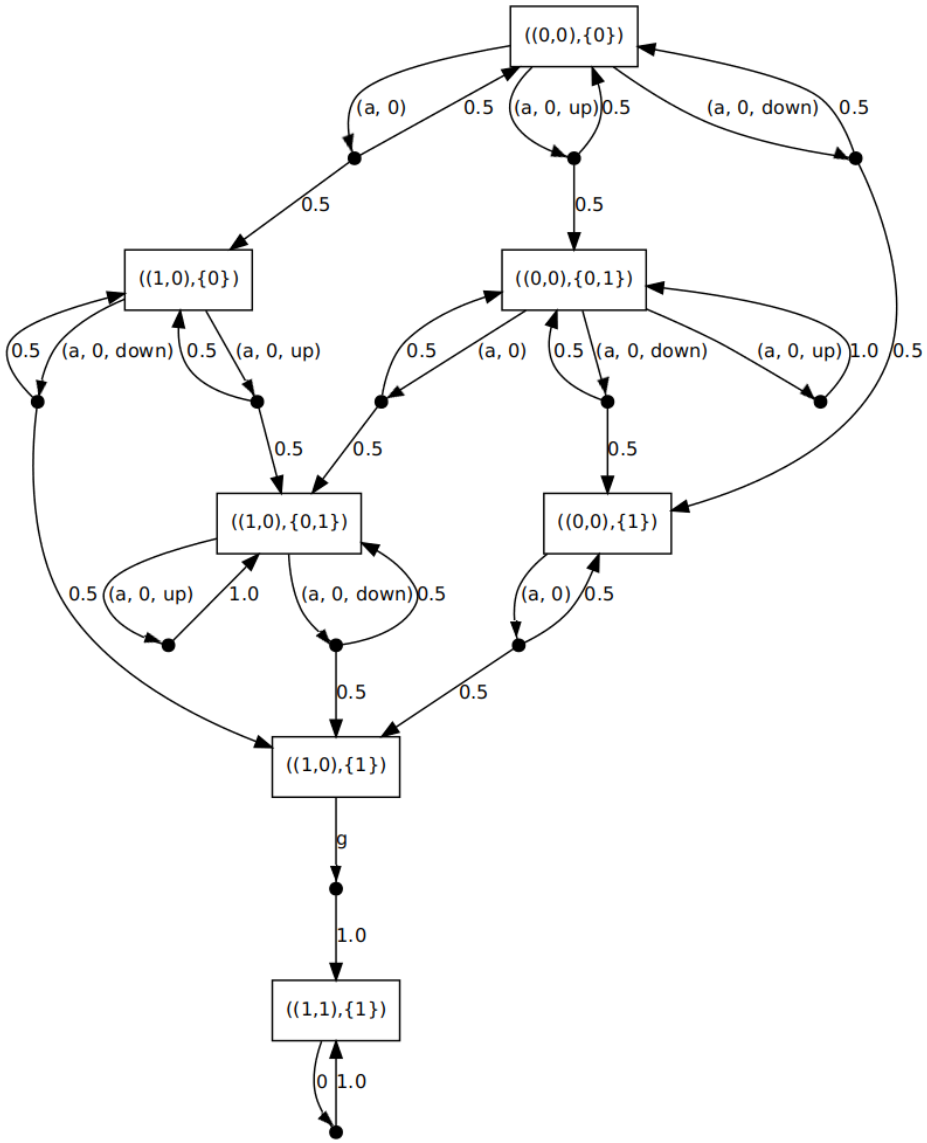
Given a PPIIS $\mathcal{S}$ and an m-indexed PLTL formula ϕ, the parameterised model checking problem involves establishing whether it is the case that $\mathcal{S}(n) \models \phi$ for all $n \geq m$. We write $\mathcal{S} \models \phi$ if this is the case.

- Note that this problem is a generalisation to probabilistic systems of a problem that is known to be undecidable in general [AK86], so it is certainly also undecidable in general.
- Nonetheless, we identify a partial decision procedure.

Abstract Model

- To verify an m-indexed formula, we construct an abstract model.
- The abstract model only records which states have **one or more** agents at them. We still keep track of the exact state of the first m agents, since we need this to evaluate the formula. (Note that other abstractions in the literature instead record *how many* agents are in each state instead).
- Transitions are labelled with whether they were a "shrinking" ($\downarrow$) transition (representing the last agent in this local state performing this action) or a "growing" ($\uparrow$) transition (representing that there were at least two agents in the local state the action was performed from).

Example Abstract System



Results

- We denote by $\hat{\mathcal{S}}(m)$ the abstract system with m agents (see the paper for the formal definition of this).

Theorem

Suppose $\hat{\mathcal{S}}(m) \models P^{max}_{\leq x}[\psi]$ *for some* m*-indexed formula* ψ*. Then,* $\mathcal{S}(n) \models P^{max}_{\leq x}[\psi]$ *for all* $n \in \mathbb{Z}^+$ *with* $n > m$*.*

- Similar results can be obtained for the other PLTL formulas.
- This gives a partial procedure for the PMCP.

Implementation

- Our implementation is based on PRISM [KNP11].
- We used our implementation to model a *foraging* protocol in which robot aim to find food and bring it back to a nest.
- We checked the (0-indexed) property $P^{\max}_{\leq p}[F^{<k}\text{deposited2}]$ which says that for any choice of scheduling, the probability that 2 units of food are deposited within k steps does not exceed p.
- Note that when the property is true, we know it is true in *all concrete systems of any size*. However, when it is false we cannot make any claim as our procedure is partial.

Results

		k				
		6	8	10	12	14
p	0.25	False	False	False	False	False
	0.50	True	False	False	False	False
	0.75	True	True	False	False	False
	0.90	True	True	True	False	False
	0.95	True	True	True	True	False
	0.99	True	True	True	True	True

Table: For different values of k and p, whether the property $P^{\max}_{\leq p}[F^{<k}\text{deposited2}]$ held in the abstract model.

- The abstract model takes around 50 seconds to construct and has 277,593 states and 8,880,150 transitions.
- The properties take a negligible amount of time to check ($\sim$50ms).

Conclusion

- We have developed and implemented a method for verifying probabilistic swarm systems, and used our implementation to check a small example protocol.
- Our procedure targets the novel overlap of checking systems that are both probabilistic and have a possibly unbounded number of agents.
- We plan to continue work in this area by targetting:
  - More tractable verification procedures.
  - Examples of verification of more realistic protocols.
  - Richer specification languages.

References

K.R. Apt and D. C. Kozen. "Limits for automatic verification of finite-state concurrent systems". In: *Information Processing Letters* 22.6 (1986), pp. 307–309.

S. Konur, C. Dixon, and M. Fisher. "Analysing robot swarm behaviour via probabilistic model checking". In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 199–213.

P. Kouvaros and A. Lomuscio. "A Counter Abstraction Technique for the Verification of Robot Swarms". In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*. AAAI Press, 2015, pp. 2081–2088.

M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV11)*. Springer, 2011, pp. 585–591.

A. Pnueli, J. Xu, and L. Zuck. "Liveness with (0, 1,infinity)-counter abstraction". In: *Proceedings of the 14th International Conference on Computer Aided Verification (CAV02)*. Vol. 2404. Lecture Notes in Computer Science. Springer, 2002, pp. 93–111.

Progress since IJCAI18

- The agents in this work synchronise differently based on action types, in our previous work all agents always acted at every time step without any synchronisation.
- Our environments are now probabilistic, our previous work only considered deterministic environments.
- Our specifications can describe unbounded traces, whereas in our previous work we only considered bounded traces.

Global Protocol

Definition (Global protocol)

The global protocol $P_n : S_n \to \mathcal{P}(Act_n)$ is defined by $a \in P_n(g)$ iff:

- *(Asynchronous environment).* (i) $a \in A_E$; (ii) $a \in P_E(g.E)$.
- *(Asynchronous agent).* (i) $a = (a', i) \in A \times \mathcal{A}_n$; (ii) $a' \in P(g.i)$.
- *(Agent-environment).* (i) $a = (a', i) \in AE \times \mathcal{A}_n$; (ii) $a' \in P(g.i)$; (iii) $a' \in P_E(g.E)$.
- *(Global-synchronous).* (i) $a \in GS$; (ii) for all $i \in \mathcal{A}_n$, $a \in P(g.i)$; (iii) $a \in P_E(g.E)$.

Global Transition Function

Definition (Global transition function)

The global transition function $t_n : S_n \times Act_n \times S_n \to [0,1]$ is defined by:

$$(g,a,g') \mapsto \begin{cases} t_E(g.E, a, g'.E) & \text{if } a \in A_E \\ & \quad \text{and } \forall i \in \mathcal{A}_n : g.i = g'.i \\ t(g.i, a', g'.i) & \text{if } a = (a', i) \in A \times \mathcal{A}_n \\ & \quad \text{and } \forall j \in \mathcal{A}_n \setminus \{i\} : g.j = g'.j \\ t_E(g.E, a', g'.E) \cdot t(g.i, a', g'.i) & \text{if } a = (a', i) \in AE \times \mathcal{A}_n \\ & \quad \text{and } \forall j \in \mathcal{A}_n \setminus \{i\} : g.j = g'.j \\ t_E(g.E, a, g'.E) \cdot \prod_{i=1}^{n} t(g.i, a, g'.i) & \text{if } a \in GS \\ 0 & \text{otherwise} \end{cases}$$