Autonomous Agent Behaviour Modelled in PRISM – A Case Study

Ruth Hoffmann[1(✉)], Murray Ireland[1], Alice Miller[1], Gethin Norman[1], and Sandor Veres[2]

[1] University of Glasgow, Glasgow G12 8QQ, Scotland
ruth.hoffmann@glasgow.ac.uk
[2] University of Sheffield, Sheffield S1 3JD, UK

Abstract. Formal verification of agents representing robot behaviour is a growing area due to the demand that autonomous systems have to be proven safe. In this paper we present an abstract definition of autonomy which can be used to model autonomous scenarios and propose the use of small-scale simulation models representing abstract actions to infer quantitative data. To demonstrate the applicability of the approach we build and verify a model of an unmanned aerial vehicle (UAV) in an exemplary autonomous scenario, utilising this approach.

1 Introduction

Autonomous systems have the ability to decide at run-time what to do and how to do it. A critical question is how this decision making process is implemented.

Increasingly, autonomous systems are being deployed within the public domain (e.g. driverless cars, delivery drones). Naturally, there is concern that these systems are reliable, efficient and - most of all - safe. Although testing is a necessary part of this process, simulation and formal verification are key tools, especially at the early stages of design where experimental testing is both infeasible and dangerous. Simulation allows us to view the continuous dynamics and monitor behaviour of a system. On the other hand, model checking allows us to formally verify properties of a finite representation. Whereas the simulation model is close to an implementation, simulation runs are necessarily incomplete. Verification models, on the other hand, require us to abstract more coarsely.

The decisions made by an autonomous agent depend on the current state of the environment, specifically in terms of data perceived by the agent from its sensors. If model checking is to be used for the verification of autonomous systems we must reflect the uncertainty associated with the state of the environment by using probabilistic model checking.

We propose a framework for analysing autonomous systems, specifically to investigate decision-making, using probabilistic model checking of an abstract model where quantitative data for abstract actions is derived from small-scale simulation models. We illustrate our approach for an example system composed of a UAV searching for and collecting objects in an arena. The simulation models

D. Bošnački and A. Wijs (Eds.): SPIN 2016, LNCS 9641, pp. 104–110, 2016.
DOI: 10.1007/978-3-319-32582-8_7

for abstract actions are generated using the object-oriented framework Simulink and the abstract models are specified and verified using the probabilistic model checker PRISM. In our example, autonomous decision making involves making a weighted choice between a set of possible actions, and is loosely based on the Belief-Desire-Intention architecture [7].

Previous work in which model checking is used to verify autonomy includes [3] in which the decision making process is verified in isolation, while our aim is to integrate this process with the autonomous agent as a whole. Other research includes an investigation of the cooperative behaviour of robots, where each robot is represented by a hybrid automaton [1], and verification of a consensus algorithm using experimental verification and an external observer [2].

2 Autonomy

In order to formally define autonomous behaviour, we introduce finite state machines which abstract the autonomous actions independent of the agent type.

Before we give the formal definitions we require the following notation. For a finite set of variables V, a valuation of V is a function s mapping each variable in V to a value in its finite domain. Let $val(V)$ be the set of valuations of V. For any $s \in val(V)$, $v \in V$ and value x of V, let $s[v{:=}x]$ and $s[v{\pm}x]$ be the valuations where for any $v' \in V$ we have $s[v{:=}x](v') = x$ and $s[v{\pm}x](v') = s(v'){\pm}x$ if $v'{=}v$ and $s[v{:=}x](v') = s[v{\pm}x](v') = s(v')$ otherwise. For a finite set X, a probability distribution over X is a function $\mu : X \to [0,1]$ such that $\sum_{x \in X} \mu(x) = 1$. Let $Dist(X)$ be the set of distributions over X.

Definition 1. A *probabilistic finite-state machine* is a tuple $\mathcal{M}{=}(V, I, A, T)$ where: V is a finite set of *variables*; $I \subseteq val(V)$ a set of *initial states*; A a finite set of *actions* and $T : val(V){\times}A \to Dist(S)$ a (partial) *transition function*.

The set of states of $\mathcal{M}{=}(V, I, A, T)$, denoted S, is the set of valuations $val(V)$ of V. Let $A(s)$ denote the actions available from state s, i.e. the actions $a \in A$ for which $T(s, a)$ is defined. In state s an action is chosen non-deterministically from the available actions $A(s)$ and, if action a is chosen, the transition to the next state is made according to the probability distribution $T(s, a)$. A probabilistic finite-state machine describes a system without autonomy, we introduce this through a weight function adding decision making to the finite-state machine.

Definition 2. An *autonomous* probabilistic finite-state machine is a tuple $\mathcal{A} = (V, I, A, T, w)$ where (V, I, A, T) is a probabilistic finite-state machine and w is a *weight function* $w : val(V){\times}A \to [0,1]$ such that for any $s \in val(V)$ and $a \neq b \in A$ we have $w(s, a) \neq w(s, b)$ and $w(s, a){>}0$ implies $a \in A(s)$.

In an autonomous machine the non-determinism in the first step of a transition is removed. More precisely, if a machine $\mathcal{A}{=}(V, I, A, T, w)$ is in state s, then the action performed is that with the largest weight, that is the action:

$$a_{s,w} = \arg\max\{w(s, a) \mid a \in A(s)\}\,.$$

Requiring the weights for distinct actions and the same state to be different ensures this action is always well defined. Having removed the non-determinism through the introduction of a weight function, the semantics of an autonomous finite state machine is a discrete time Markov chain.

3 UAV Example

In this case study we consider a specific example (a simple search and retrieve example, with a UAV in a finite sized arena) to demonstrate our approach.

The UAV first takes off and checks whether the system and sensors are functional. If the UAV detects an issue in the system, then it returns to base. Otherwise it will proceed to search for a given number of objects. When an object is found, the UAV positions itself above the object and descends until the grabber can pick it up. The UAV then ascends to transportation height and transports the object to the deposit site. There is the possibility that the UAV will drop its object along the way and need to retrieve it. Once the UAV is above the deposit site, it releases the object and ascends back to search height. It will then decide whether it continues the search or returns to the base and complete the mission. During operation, the UAV may return to base to recharge if it is low on battery, or conduct an emergency landing, due to an internal system error. If the mission time limit is reached, the UAV abandons the mission and returns to base. Figure 1 represents this scenario, showing the different modes of the UAV and progression between the modes.

We represent this scenario using a autonomous finite-state machine $\mathcal{A}$. The variables V of $\mathcal{A}$ are given by:

- obj the number of objects which have not been found;
- $pos{=}(pos_x, pos_y)$ the position of the UAV in the arena;
- $ret{=}(ret_x, ret_y)$ the return coordinates when search is interrupted;
- m the current mode of the UAV;
- t the mission time;
- b the battery charge level.

Each state $s \in val(V)$ of $\mathcal{A}$ is a valuation of these variables. The transition and weight functions T and w are based on Fig. 1. We focus on the target approach and search modes of the UAV.

In the target approach mode ($m{=}4$), the UAV positions itself above an observed object and we denote this abstract action by $Approach$. Thus, the weight function is $w(s, Approach){=}1$ for all states s such that $s(m){=}2$, and for any such state s we have for any $s' \in S$:

$$T(s, Approach)(s') = \begin{cases} 1 & \text{if } s' = s[m{:=}5][t{+}T_{ap}][b{-}B_{ap}] \\ 0 & \text{otherwise} \end{cases}$$

where T_{ap} and B_{ap} are the time and battery charge used approaching the object, and $m{=}5$ is the mode for descending. The abstract action $Approach$ models

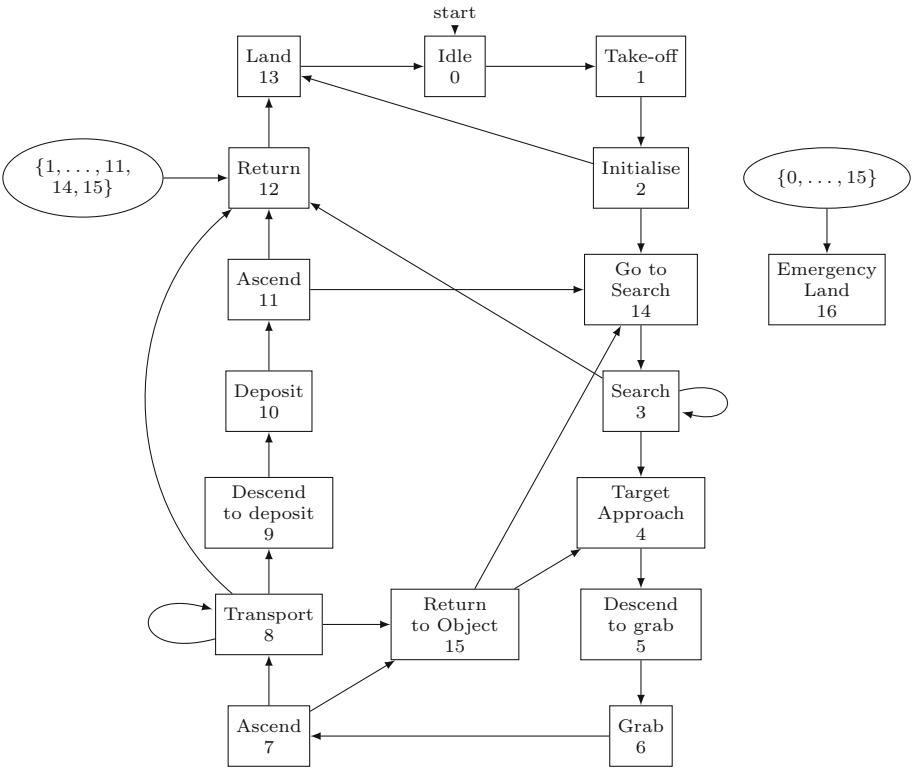


Fig. 1. The finite state machine representing the scenario.

several different operations of the UAV including the use of its camera and navigational system. A small-scale simulation model was built for this abstract action to provide the required quantitative data.

When the UAV is in search mode (m=3) there are two actions that can occur: *Search* and *BatteryLow* with the UAV continuing search if the battery charge level is above a certain threshold, and returning to base otherwise. The weight function for the *Search* action from any state s such that $s(m)$=3 is given by:

$$w(s, Search) = \begin{cases} 0 & \text{if } s(b) \leq B_{low} \\ 1 & \text{if } s(b) > B_{low} \end{cases}$$

and for the *BatteryLow* action we have $w(s, BatteryLow) = 1 - w(s, Search)$. Concerning the transition function we have for any $s' \in S$:

$$T(s, Search)(s') = \begin{cases} 1-\alpha & \text{if } s' = s[pos{:=}\Delta pos][t{+}\Delta t][b{-}\Delta b] \\ \alpha & \text{if } s' = s[pos{:=}\Delta pos][ret{:=}\Delta pos][m{:=}4][t{+}\Delta t][b{-}\Delta b] \\ 0 & \text{otherwise} \end{cases}$$

and $T(s, BatteryLow)(s)=1$ if $s'=s[pos:=\Delta pos][ret:=\Delta pos][m:=12][t+\Delta t][b-\Delta b]$ and 0 otherwise, where Δpos denotes the movement from one discrete square in the arena to the next, Δt and Δb are the time and battery consumption of the UAV while moving one square. The UAV has probability α of finding an object in a given position, if an object is found the UAV changes to mode $m=4$ and *ret* is set to the current coordinates, as the search has been interrupted. If no object is found, the UAV continues searching.

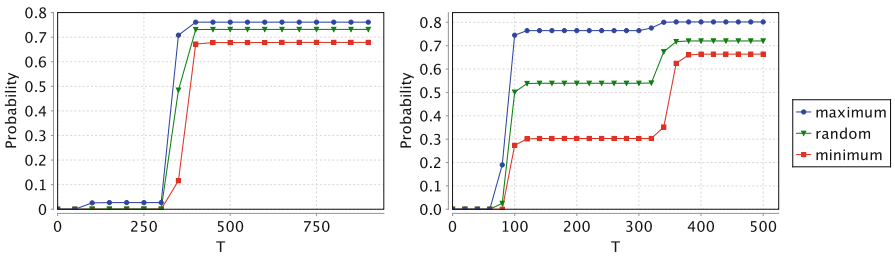
4 Results

We have modelled our scenario in the probabilistic model checker PRISM [5], building small-scale simulation models to determine individual abstract actions to generate probabilistic and timing values. To encode the timing values in PRISM as integer variables we take the floor and ceiling, introducing non-determinism into the model and upper and lower bounds for properties [4]. The experiments were performed on a computer with 16GB RAM and a 2.3 GHz Intel Core i5 processor.

The main properties of interest concern the UAV successfully completing the mission (finding and depositing all objects within the time limit) and failing the mission (either due to an emergency landing, missing some objects or running out of time). We have also considered other properties including how often the UAV drops an object and how often it recharges during a mission, more details can be found in the repository [6].

We have analysed two scenarios where there are 3 objects and a time limit of 900s, and 2 objects and a time limit of 500 s respectively. For the first scenario the model has 116 191 709 states, was built in 488 s and verifying a property varied between 298 s and 813 s. This is far more efficient than running Monte Carlo simulations, as simulating 10 000 runs of the same scenario takes over two weeks. For the second scenario the model has 35 649 503 states and model construction time was 77 s.

For the first scenario, the maximum and minimum probabilities of the UAV completing a mission are 0.7610 and 0.6787 respectively. The maximum and



(a) Scenario 1. (b) Scenario 2.

Fig. 2. Probability of completing the mission successfully by deadline T.

minimum probabilities of running out of time are negligible, searching the arena and missing some objects are 0.2617 and 0.1808, and 0.0628 and 0.0552 for performing an emergency landing. Figure 2 shows the maximum and minimum probability of a successful mission within a time bound as well as the results obtained when the non-determinism is replaced by a uniform random choice. The probability increases after a threshold time as the UAV has to search a proportion of the arena before finding all objects.

5 Conclusions

We have proposed using small-scale simulation models to inform probabilistic models used for verification. The simulation models can be used to provide quantitative data for abstract actions. The probabilistic models can be used for fast property specific verification, that is not possible using simulations alone.

Our approach is highly adaptable; once the initial small-scale simulation and probabilistic models have been set up, different decision algorithms can be easily substituted and analysed. Our example illustrates the use of a weight function for decision making. In a more extensive scenario the weight function would be more complex (e.g. involving current values associated with all sensors and guiding systems). Our use of non-determinism when approximating quantitative data obtained from the small-scale simulation models allows us to provide an range of uncertainty for our results. We aim to formally prove a link between the simulation and the abstract model to allow us to infer results from the abstract model for the actual system. To allow the analysis of more complex scenarios we plan to incorporate abstraction techniques.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/N508792/1].

References

1. Chaimowicz, L., Campos, M.F.M., Kumar, V.: Hybrid systems modeling of cooperative robots. In: Proceedings International Conference Robotics and Automation (ICRA 2003), pp. 4086–4091. IEEE Press, New York (2003)

2. Cook, J., Hu, G.: Experimental verification and algorithm of a multi-robot cooperative control method. In: Proceedings IEEE/ASME International Conference Advanced Intelligent Mechatronics (AIM 2010), pp. 109–114. IEEE Press, New York (2010)
3. Dennis, L., Fisher, M., Lincoln, N., Lisitsa, A., Veres, S.: Practical verification of decision-making in agent-based autonomous systems. Autom. Softw. Eng., pp. 1–55 (2014). http://link.springer.com/article/10.1007/s10515-014-0168-9
4. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: A game-based abstraction-refinement framework for Markov decision processes. Formal Methods Syst. Des. **36**(3), 246–280 (2010)
5. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
6. PRISM model repository (2016). http://dx.doi.org/10.5525/gla.researchdata.274
7. Veres, S., Molnar, L., Lincoln, N., Morice, C.: Autonomous vehicle control systems - a review of decision making. Proc. Inst. Mech. Eng. Part I: J. Syst. Control Eng. **225**(2), 155–195 (2011)