Formal proof of Abstraction for Agent-Based Learning systems

Ryan Kirwan\textsuperscript{1} \hspace{2cm} Alice Miller\textsuperscript{2}

\textsuperscript{1} University of Glasgow, Department of Computing Science, kirwanr@dcs.gla.ac.uk
\textsuperscript{2} University of Glasgow, Department of Computing Science, alice@dcs.gla.ac.uk

Abstract: We describe herein a formal proof of a simulation relation between a model of an Agent-Based Learning (ABL) system and our abstraction of it. Proving that a simulation relation exists means that all LTL properties we verify, via Model checking [4], for our abstraction also hold true in the original model. Formalising this relation allows us to apply the same abstraction technique to other ABL systems safe in the knowledge that a similar simulation relation will exist.

1 Introduction

This work is an incremental step in our research from [1], [5] and [2] where we introduced ABL systems and our abstraction of them. Our abstraction allows us to deal with an entire type of environment and robot specification in one model. It is made possible by the restriction on the type of property we are interested in; we are only interested when the robot is learning to avoid obstacles, or crashing into one. What we find is that our abstraction holds the same properties as the original ABL systems while having the advantages of reasoning about an entire type of system (opposed to just one instance of it) and being less computationally expensive to check. In this paper we prove that our abstraction is sound; showing that it holds a simulation relation (H) with the original ABL model, which it was abstracted from.

We consider two Promela [3] models which represent an ABL system. This system involves one autonomous, learning robot in an environment which contains a well-defined distribution of obstacles. One model is highly detailed, the Explicit model, and the other is our abstraction thereof, the Relative model. We aim to prove that H exists between the models such that the Relative model simulates the Explicit model.

1.1 Explicit model ($M_E$)

The Explicit model ($M_E$) of the system represents a robot in a fixed size of environment with a fixed distribution of obstacles. In $M_E$ the coordinates of the robot are stored so that its explicit position in the environment is known. An example of $M_E$ is shown in Fig 1.

1.2 Relative model ($M_R$)

The Relative model ($M_R$) is the result of our abstraction [5]. This model can be visualised as a cone projecting out from the centre of the robot, shown in Fig 2. Fig 2 shows the size of $M_R$ (this is the area of the cone) and the scale of the robot and an obstacle.

2 Simulation relation (H)

H is necessary for us to be able to reason that every LTL formula that is true in $M_R$ is also true in $M_E$. To prove this relation we must demonstrate that for every state in $M_E$ ($s_{En}$) that is mapped by H to a state in $M_R$ ($s_{Rn}$) such that ($s_{En}$, $s_{Rn}$) $\in$ H, then the following condition must hold: $s_{Rn+1}$ such that $s_{En}$ transitions to $s_{Rn+1}$ in $M_E$, there is a state $s_{Rn+1}$ with the property that $s_{Rn}$ transitions to $s_{Rn+1}$ in $M_R$, and ($s_{En+1}$, $s_{Rn+1}$) $\in$ H.

Figure 3 illustrates our proof of H. We assert that there
is a translation function $T_1$ which maps every state in $M_E$ to a state in $M_R$ and that there is a translation function $T_2$ which maps every state in $M_R$ to a state in $M_E$.

A transition in $M_E (F_E)$ involves going through a transition in $M_R (F_R)$. Hence, we know that the successor state in $M_E$ is part of $H$ because the function $F_E$ is used to calculate its successor state. By defining $F_E$ in this way, we’ve ensured that if $(s_{En}, s_{R0}) \in H$ then $(s_{En+1}, s_{R0+1}) \in H$. The final stage in this proof involves mapping the nondeterministic transitions in $M_R$ to $M_E$.

2.1 Nondeterministic transitions

Figure 4 shows the fully deterministic nature of $M_E$ (on the left), with each state mapping to exactly one successor state. Additionally, we can see the nondeterministic transitions in $M_R$ from the state $s_{R0}$ (on the right): $s_{R0}$ maps to many possible successor states, including itself.

We assert that there is a mapping of every possible successor state from $s_{R0}$ to every successor state from an $s_{En}$ state that is mapped to $s_{R0}$, such that it is in $H$. We can assert this because of what we know about the nature of the environment and about the way the robot moves.

The environment has a fixed density of obstacles and the robot always moves forward at a rate of one unit per time-step. If, in $M_E$, the robot has no obstacles in it then we know that it is in free-space in $M_R$. There are only two types of transition we need to consider here, one where the robot drives forward one unit and encounters an obstacle and one where is drives forward and does not.

Consider the second case where the robot drives forward and there is no obstacle. In this case, all the transitions can be mapped back to the $s_{R0}$ state.

For the first case, where the robot does encounter an obstacle, there are only a finite number of positions that an obstacle can appear in the model. These positions lie only in front of the robot because the robot is driving forward one unit of distance. For $M_R$, we restrict all nondeterministic transitions from the state $s_{R0}$ to ones where an obstacle enters from any coordinate across the large end of the cone (see Fig 2). Therefore, all possible successor states from $s_{R0}$ have to cover all possible successor states in $M_R$ because they cover all possible transitions. From this we can be certain that there is no successor state in $M_E$ that cannot be mapped to a state in $M_R$, hence completing the proof of the simulation relation $H$.

3 Conclusion

We’ve provided an effective abstraction process for dealing with this type of system. We aim to present this technique as a framework which allows the verification of properties in similar, complex models. We maintain the accuracy, while reducing the size of $M_R$ by embedding its functions into C code which runs along side the Promela models without adding to their state-space. Our transition functions allow the abstraction framework to provide an automatic proof of a simulation relation. Hence, using our framework guarantees the soundness of the abstraction as part of its application.

We plan to expand on this work by broadening the parameters which can be passed into the abstraction process. By aiming to deal with multiple robots and an automated model setup, given input parameters, we hope to broaden the scope for the application of our framework.

Acknowledgements

Thanks to Alice Miller, Gethin Norman and Oana Andrei.

References