Progress on Model Checking Robot Behaviour

Ryan Kirwan Alice Miller

1 Department Of Computing Science, University Of Glasgow, kirwanr@dcs.gla.ac.uk
2 Department Of Computing Science, University Of Glasgow, alice@dcs.gla.ac.uk

Abstract: We model systems that involve a learning robot which interacts with obstacles in a static environment. Our models are specified in Promela with a view to verifying them using SPIN [2]. Because of the complex nature of these systems our initial models have intractable state-spaces: we aren’t able to model check them. Last year [4] we introduced an abstraction technique to greatly reduce the state-space of our models. One of the challenges we now face is to prove that our abstraction technique is correct i.e., it preserves the properties of concern.

1 Introduction

Traditionally, testing and simulation have been used to validate robot behaviour and to validate algorithms defining robot learning. Model checking [3] robot behaviour allows us to formally check properties and, potentially, to measure the performance of learning algorithms.

The robot behaviour captured in our models is as defined in [5]. Behaviour analysis is the study of how learning is affected by an environment and by the perception of the learner. Through simulations, the ability of a robot to successfully navigate its environment is used to assess the robot’s sensor configuration and the learning algorithm it uses. These assessments are generated by averaging results from sets of simulated experiments. By applying model checking we can derive properties by a single search of the state-space (per property).

Our initial models are specified with a view to verification. We begin our modelling process by generating the Explicit model of the system. This is a highly detailed model with a state-space too large to verify. Our next step is to apply our abstraction technique [4] to the Explicit model to generate a tractable model, the Relative model. The current challenge is to prove the correctness of our abstraction technique.

2 Explicit model

The Explicit model uses polar coordinates to determine position within an environment. This allows us to calculate the turning angles of the robot more accurately than with a standard grid representation. All positions in the environment are stored as an angle and a distance relative to the centre of the environment.

The robot is modelled as the only moving object in an environment. As in the real systems it is composed of two distal and two proximal antennas. In our model we don’t include the robot’s motors or external wheels. The robot is simply modeled to represent its avoidance behaviour.

We use embedded C code, within the Promela specification, to calculate the precise location of the robot as it moves around an environment. The robot’s angle from the centre and it’s distance from the centre are used in the calculation. We also consider the angle that the robot is facing, relative to North.

The robot’s learning is implemented with a simple calculation that involves incrementing a learning factor every time the robot collides into something. If the robot collides with something then it learns to respond more strongly to the type of signals that it received before the collision.

3 Relative model

The Relative model is our abstraction of the Explicit model. Unlike the Explicit model the robot is not represented as a set of coordinates and an orientation, but as the centre of a polar axis. We model an environment from the perspective of the robot, we will herein refer to the area that this perspective covers as the cone of influence.

The cone of influence has 80° of angle and is split into nine units of distance. Each unit of distance is the length of a proximal antenna (the smallest object in the system). The robot’s learning is implemented in the same way as in the Explicit model. The resulting state-space is tractable, allowing us to verify LTL properties using model checking. E.g., the LTL formula “[[ω < 11]” checks that the robot’s learning factor (ω) always remains less than 11.
4 Proving the equivalence of our models

One of the biggest challenges we face is proving the correctness of the Relative model. Our proof is based on one given in [6] where a similar proof is used for abstracted featured networks. We achieve this by converting our models to Guarded Command Form (GCF). We map sets of transitions in the Explicit model to transitions in the Relative model. An example of a Promela specification GCF is given in both sides of Figure 1. Once we have our models in GCF we map transitions, as shown in Figure 1.

![Figure 1: Equating guarded command line from Explicit to Relative model.](image)

With our proof we hope to conclude that there is a simulation relation between the Explicit model (\(M\)) and the Relative model (\(M'\)). This relation implies that any property that holds for \(M'\) also holds for \(M\).

\[
\text{If } M' \text{ simulates } M \text{ then for every \(LTL\) formula } \psi \\
M' \models \psi \text{ implies that } M \models \psi.
\]

5 Future Work

We have begun to use PRISM [1] to create similar system models, but have not verified any quantitative properties yet. Principally, we want to develop a proven abstraction technique for systems that involve robot learning. One further aim is to develop a custom-made tool to automatically abstract and model check this type of system.

References


