Model Checking Multi-Agent Systems

Ryan Kirwan  
Department Of Computing Science  
University Of Glasgow  
r.kirwan.1@research.gla.ac.uk

Alice Miller  
Department Of Computing Science  
University Of Glasgow  
alice@dcs.gla.ac.uk

Abstract

Model checking [2] has been applied to many areas of software and hardware verification; this includes hybrid systems, which contain both hardware and software components. To model hybrid systems both types of component are included in a single model. In this research we consider hybrid systems in which multiple agents interact in an environment. An agent is a simple robot capable of moving, detecting obstacles, and learning to avoid obstacles. The goal of this research is to discover a practical and standardised way of modelling these systems, and to develop theoretical techniques to measure performance.

1 Introduction

Model checking involves verifying properties of a system and it can be an important tool for system development. The importance of model checking is due to the impracticality of running exhaustive testing to verify properties of real physical systems. Even when exhaustive testing can be used to verify a system’s properties, model checking could be used to verify the same properties faster and more accurately, through automated exhaustive checking on abstracted system models. The goal of this research is to avoid system testing by using model checking to apply verifications to multi-agent systems.

The multi-agent systems we are modelling contain two or more identical robots that are learning to avoid each other and obstacles in a walled environment. These robots have 2 long range “distal” sensors and 2 short range “proximal” sensors; distal sensors are used to avoid collisions, proximal sensors are used to detect collisions.

1.1 Initial Model Checking

To better understand the problem domain, the modelling of existing multi-agent systems was undertaken. The aim was to use mainstream model checkers such as PRISM [4] and SPIN [5]. These model checkers were chosen to model systems used in experiments performed at the University of Glasgow [7]. We work closely with the developers of these experiments; this allows us access to new and interesting systems and allows our results to influence their designs.

The explicit state model checker SPIN is well established in the field of model checking, and allows us to verify properties of systems described in the model specific language Promela [5]. Promela is an intuitive language which allows models to be created quickly and easily. PRISM is a symbolic model checker which allows it to verify much larger state-spaces than SPIN. It also has the benefit of quantitative analysis and probabilistic weightings. The quantitative analysis can provide a performance measure for multi-agent systems, which SPIN cannot. However, PRISM has a less comprehensive modelling language than Promela.

In our models each robot is treated as a process, as is the environment. The environment is aware of the position of all things within it. The robots are able to communicate to the environment via sensors. The environment is abstracted to a grid and the robots can move in either of the 8 compass directions, moving 1 cell at a time. When a robot choses an adjacent cell to move to, it first uses its sensors to check that the grid cell is vacant.

The physical dimensions of the robots are created to an accurate scale based on the resolution of the environment’s grid. The environment is represented as a grid with resolution of 22x22 cells, the outer cells form a surrounding wall, and each robot is 2x2 cells with distal sensor antenna 4 grid cells long and proximal antenna 1 grid cell long.

Instead of static, cell-to-cell movements, the model is designed to better emulate the continuous driving of real robots, by making small adjustments as they move. Each robot reassesses its direction every time it arrives at a new grid position and since the resolution of the grid is relatively high each calculation for a new movement is done at small intervals, hence closely emulating the continuous reassessment of the real system.

The calculations for the robots’ movement are based on the difference between the Manhattan distances [1] of each of the robots’ distal antenna sensors. Distal sensors are long rods protruding at 45 degrees from the robot and are used for

---

1In reality it would be possible to measure the direction a robot is facing to exact degrees of accuracy, but having 8 directions allows the robots to access all adjacent grid cells. This creates an accurate model without overloading the number of directions in which a robot can face.
avoiding, whilst the proximal sensors are impact sensors which are close to the robot. The distal sensors send signals of varying strength depending on how close or far away an obstacle is to the robot. This movement calculation emulates the avoidance behaviour of the real robots.

1.2 Results

Modelling with SPIN allows properties to be checked, such as: is it possible for "Robot1" to collide with "Robot2", or other obstacles? This property can be expressed in Linear Time Temporal Logic (LTL) [6], thus:

\[
[] ( ! ( ((robo1X==robo2X) && (robo1Y==robo2Y)) || ((robo1X==obVal) && (robo1Y==obVal)) ) )
\]

Modelling with PRISM allows us to check Probabilistic Computation Tree Logic (PCTL) [6] properties such as, what is the probability of an erroneous state being reached? Synchronizing on their movements, the Steady-state probability of the system having two agents occupying the same grid cell is 0.0064. This was verified using the PCTL property:

\[
S=? [(robo1X==robo2X) \& (robo1Y==robo2Y)]
\]

To make the PRISM model accurate each robot must be able to move in and out of synchronization with each other robot. To allow this the PRISM code must specify the probability of the agents synchronizing and not. There is not an accurate way of selecting this probability, yet the value chosen for it will affect every subsequent validations’ probability.

These types of property are useful, but are based on assumptions. Assumptions are made so models can be abstracted enough to apply verification. Examples of these assumptions are that sensors were assumed to behave like perfect springs whenever an obstacle or agent touched a point on one of the sensor antenna. It was assumed that the robots moved and turned with consistent accuracy. Another assumption is that the discrepancies in the positioning of the sensors relative to the robots’ facing direction was insignificant —this discrepancy arose from abstracting the environment to a grid. The relevance of the verification becomes unclear once these assumptions are factored in.

1.3 Future Work

We plan to use Hybrid Modelling Languages such as Hytech [3], to create highly detailed system models. We will also continue working closely with current experiments involving multi-agent systems in order to expand our knowledge of how to model such systems. Principally we want to develop various models, classifying the benefits and costs of each approach. One further aim is to develop a custom-made tool to model these types of multi-agent system.

References