Investigation and Specification of Path Finding Algorithms

1710282 Yang Zi

We know that in the field of artificial intelligence, the automatic pathfinding algorithm has been widely concerned as an effective way to improve the robot's efficiency. At present, in the field of artificial automation, automatic pathfinding algorithm has been studied by many scholars, and new automatic pathfinding algorithms emerge in endlessly. However, after the author's research and investigation found that researchers in this field are currently limited to the calculation and research of automatic algorithms, but Whether the automatic algorithm itself is feasible and reliable is not studied by scholars.

However, it is worth noting that only researching the automatic pathfinding algorithm and related robots cannot completely solve the damage problem of mobile robots. We should strengthen the detection of the reliability of the automatic pathfinding algorithm. Because only the high reliability of the automatic pathfinding algorithm can effectively reduce the damage of the robot, making the robot move more stable and efficient.

Through this report's research on automatic pathfinding algorithm, The author wants to achieve the detection of the effectiveness of various automatic pathfinding algorithms, in order to find and develop a reliable automatic pathfinding algorithm, the best detection means, the manual automation is more stable, more good service to humans.

An automatic driving car is a smart car that senses the road environment through an in-vehicle sensing system, automatically plans driving routes, and controls the vehicle to reach a predetermined target.

It uses the onboard sensor to sense the surrounding environment of the vehicle and controls the steering and speed of the vehicle based on the road, vehicle position and obstacle information obtained by the perception so that the vehicle can travel safely and reliably on the road.

One of the most important technologies for automatic driving cars is the automatic pathfinding algorithm, which decides the path the car is going, we can use some diagrams to represent the real world situation, which has some nodes to represent some cross of the road, using cost to represent the distance of the road, and then we have a start node and a goal, using pathfinding algorithm, we can know the shortest path from the start node to the goal. There are many different pathfinding algorithms, one of which is the Dijkstra shortest path algorithm.

To verify the reliability of these algorithms, model checking is required. To check if they can enjoy some desired properties. model checking is giving a model of a system, automatically check whether this model enjoys some properties. Typically, one has hardware or software systems in mind, whereas the specification contains safety requirements such as the absence of deadlocks and similar critical states that can cause the system to crash. Model checking is a technique for automatically verifying the correctness properties of finite-state systems.

This report show a case of pathfinding algorithms Dijkstra, after formalizing Dijkstra to state machine and feed into Maude, we can see the state transition, and then we can do model checking using some desired properties.

Maude sets out to solve a different set of problems than ordinary imperative languages like C, Java. It is a formal reasoning tool, which can help us verify that things are "as they should", and show us why they are not if this is the case. Which means we can formalize some existing algorithms into Maude language, and find some properties and do model checking.

We can use Maude search command to search all the reachable states, to look for some states that we want to.

In this report, we have described the importance of pathfinding algorithms of autonomous robots/cars, and we have described state machines in a formal specification by Maude, we described state machine, invariant, Kripke structure and LTL, and then we described Dijkstra and Maude in a mathematical method, we built a model similar to the real world situation, which is easy for us to concentrate on the algorithm, and then we wrote Dijkstra's algorithm with Maude, and we did model checking, after using a search command, we got a counterexample, which is the shortest path from the start node to the goal.

Since we got the shortest path by using Dijkstra, but we need to verify some properties of Dijkstra, so I found two properties, one is whether Dijkstra always halts, the second property is whether the path found by Dijkstra is one of the shortest paths, in order to prove these two properties, we need to know all possible path from all nodes, so I built a new model that can find all paths of each node, but when I am using the new model to find all paths in a graph which has 64 nodes, it will take a very long time, so I decrease the node number to 32, and then after about 3 minutes, it will find all possible path for all nodes, and then we need to manually find the shortest paths, there may be one or more paths which cost is the same, then we use all shortest path feed into another module to do model checking, and then we can know if Dijkstra enjoys both properties.

To prove this, I did multiple tests by using more different graphs to do model checking, after finding all paths and model checking, the results are all true, which means Dijkstra's algorithm enjoys two properties.

There are different pathfinding algorithms nowadays, not only Dijkstra, but also other algorithms like A*, and more properties, some of the algorithm may not enjoy some properties, and we can test it after model checking, by model checking more and more pathfinding algorithms, we can prove more and more pathfinding algorithms to see whether it is trustworthy.