Introduction To Artificial Intelligence Homework Three Report

Andrés Ponce(0616110) 彭思安

May 20, 2020

1 Background

For the second assignment of the course, we were tasked to model the game of Minesweeper as a constraint satisfiability problem, and apply suitable algorithms to solve a board given an initial setup. For the third homework assignment, instead of viewing it in terms of constraints, we model it as a problem of logical inference. Each cell in our board is then treated as a logical proposition which can either be true("safe") or false("mined").

Given this new approach, the task involved finding an assignment of the nodes which lead to a solved board in the end. Thus we treat a set of literals (or their complements) as a *clause*, which we can compare to other clauses to deduce their truth values. While this is not the most efficient way to solve a Minesweeper board, it does provide a useful insight into the area of logical satisfiability as applied to game-playing agents.

2 Program

3 Further Discussion

3.1 How could we use first-order logic here?

First order logic differs from propositional logic in its power at expressing propositions. For example, a sentence such as "Every student will get a perfect score on this homework assignment" cannot be easily written using only propositional logic, unless we number all n students in the class and say the he or she will get a perfect score; something like $p_1 \wedge p_2 \wedge ... \wedge p_n$.

First order logic allows us to use universal or existential *quantifiers* which make a proposition apply to all the elements in the specified domain. Thus, if $g(x)$ represents the proposition "x will receive a perfect score on this assignment", we could model the previous sentence as $\forall x g(x)$, where x are the students in the course.

Since first-order logic already deals with applying predicate symbols to constant symbols, applying this structure to that of a program might prove more straightforward than using propositional logic. We could apply some functions to check the relations between the different objects. If we choose appropriate relations to check, maybe such to check a valid assignment or the number of adjacent objects, may be a more intuitive approach to this problem.

3.2 Is forward chaining or backward chaining applicable to this problem?

3.3 How could we imporve the success rate of "guessing" when you want to proceed from a "stuck" game.

3.4 How could we modify the method of Assignment 2 to solve the current problem?