A4 – CS4300 Assignment A4 Lab Report

Logical Agent using Resolution Theorem Proving

By Haochen Zhang & Tim Wei
10/24/2017

1. Introduction

Four functions are written for this assignment:

- *CS4300_hybrid_agent* will uses the following three functions to determine what its plan(a sequence of actions) will be and execute it one by one.

    1. *CS4300_make_percept_sentence* takes percepts from hybrid agent and create logical sentence.
    2. *CS4300_Tell* uses the sentence that spite out by *CS4300_make_percept_sentence* to update the KB.
    3. *CS4300_Ask* receive the sentence given by *CS4300_hybrid_agent*. It will return true if it is provable and false if it is not certain.

Using *CS4300_RTP* from A3, which is a very costly algorithm, we want to answer the following questions:

- How long does it take for *CS4300_hybrid_agent* to make a move?
- Is *CS4300_hybrid_agent* able to identify where the unsafe cells, such as Wampus, are?
- Is *CS4300_hybrid_agent* able to survive boards that propositional agents cannot otherwise survive?

2. Method

CS4300_hybrid_agent is an implementation of the algorithm from p. 270 of the textbook, which is modified to suit the needs of Wampus World. To be more specific, *t* and *MAKE-ACTION-SENTENCE* were not used in this implementation. There are persistent variables for the utility functions due to the removal of *t*: *agent*, for keeping track of the state of the agent; *board*, for *CS4300_Wampus_A_star*; *safe* and *visited*, for choosing the next unvisited cell to go to; and *have_arrow* and *W_pos*, for finding and killing Wampus. Also, for speed, when the agent is on a previously visited cell, the checks for all safe and unsafe cells are skipped.

In this agent, the size of KB is reduced. The knowledge of there being exactly 1 gold, and pit not being able to coexist with Wampus or gold is removed due to our selection of boards.

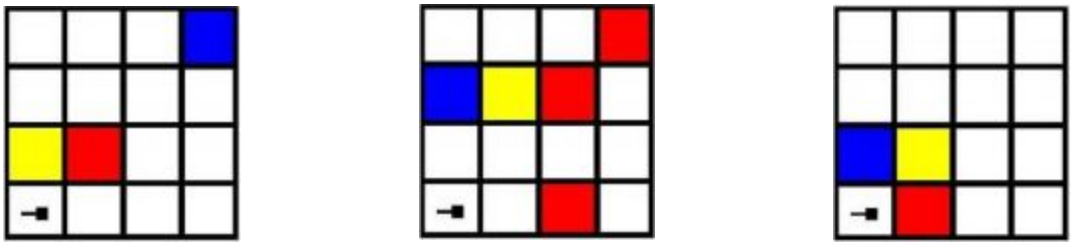
CS4300_make_percept_sentence implements *MAKE-PERCEPT-SENTENCE* by taking the percept and returning a 4-clause sentence, where each clause contains a literal: S_{xy} or $\neg S_{xy}$, B_{xy} or $\neg B_{xy}$, and G_{xy} or $\neg G_{xy}$, and $Scream_{xy}$ or $\neg Scream_{xy}$ (represented by 81 and -81). Bump is ignored since it does not contribute to *KB*.

This sentence is passed to *CS4300_Tell*, and it will add these clauses into *KB*. If *KB* already contains clauses with nothing but the same symbols, *CS4300_Tell* will keep the new clauses and discard the old clauses. If *CS4300_Tell* get $Scream_{xy}$ with any x and y, it will remove all clauses that contain any W_{xy} or S_{xy}, since if the Wampus is dead, all related logic won't be used. *CS4300_Tell* does nothing when it receive $\neg Scream_{xy}$.

CS4300_Ask takes a theorem, and uses *CS4300_RTP* to try to prove it. *CS4300_RTP* is modified so when it has ran for more than a specified time, it returns that it did not finish. One second was the chosen

time frame based on the observation that small theorem proving takes about 0.8 second each. Therefore, if *CS4300_Ask* returns false, it does not necessary means that the theorem is disproven, it may be that the process take too long.

To answer the questions from the introduction, we tested *CS4300_hybrid_agent* against 3 boards (Fig. 1).



(a) (b) (c)

Fig. 1 Boards used for testing.

3. Verification of Program

We pass five precepts at a time every time it finished the previous action to simulate the board. By checking agents' behaviour, we verify if it matches our predicts.

For board (a), the first plan that our agent should produce should be going to one of the two safe cell which is cell (1,2) and cell (2,1). If it goes to cell (2,1), it will sense Breeze and produce a plan to go to cell (1,2). If the agent is in cell (1,2), with or without visiting cell (2,1), it will start a escape sequence from current location to cell (1,1) starting with GRAB and ending with CLIMB.

The behavior of our agent for board (a) is as follow:

Agent (x, y, direction)	Precept (given)	Plan	Elapsed time (s)
(1,1,0)	(0,0,0,0,0)	3,1	43.501
(1,1,1)	(0,0,0,0,0)	1	0.0146
(1,2,1)	(0,0,1,0,0)	4,2,2,1,6	36.571
(1,2,1)	(0,0,1,0,0)	2,2,1,6	0.020
(1,2,0)	(0,0,1,0,0)	2,1,6	0.017
(1,2,3)	(0,0,1,0,0)	1,6	0.014
(1,1,3)	(0,0,1,0,0)	6	0.018

For board (b), the first plan that our agent should produce should be going to one of the two safe cell which is cell (1,2) and cell (2,1). If it goes to cell (2,1), it will sense Breeze and produce a plan to go to cell (1,2). If the agent is in cell (1,2), it will sense Stench. At this point, our agent will realize that cell (2,2) has neither Wampus or Pit. It will then start a plan to go to cell (2,2). The next plan should be going to one of the two safe cell which is cell (3,2) and cell (3,2). If it goes to cell (3,2), it will sense Breeze and produce a plan to go to cell (2,3). If the agent is in cell (1,2), with or without visiting cell (3,2), it will start a escape sequence from current location to cell(1,1) starting with GRAB and ending with CLIMB.

The behavior of our agent for board (b) is as follow:

Agent (x, y, direction)	Precept (given)	Plan	Elapsed time (s)
(1,1,0)	(0,0,0,0,0)	3,1	43.591
(1,1,1)	(0,0,0,0,0)	1	0.025
(1,2,1)	(1,0,0,0,0)	2,2,1,3,1	39.832
(1,2,0)	(1,0,0,0,0)	2,1,3,1	0.014
(1,2,3)	(0,0,1,0,0)	1,3,1	0.021
(1,1,3)	(0,0,0,0,0)	3,1	0.017
(1,1,0)	(0,0,0,0,0)	1	0.018
(2,1,0)	(0,1,0,0,0)	3,1	38.781
(2,1,1)	(0,1,0,0,0)	1	0.018
(2,2,1)	(1,0,1,0,0)	1	34.973
(2,3,1)	(0,0,1,0,0)	4,3,3,1,1,2,1,6	27.647
(2,3,1)	(0,0,1,0,0)	3,3,1,1,2,1,6	0.017
(2,3,2)	(0,0,1,0,0)	3,1,1,2,1,6	0.014
(2,3,3)	(0,0,1,0,0)	1,1,2,1,6	0.021
(2,2,3)	(0,0,1,0,0)	1.2,1,6	0.016
(2,1,3)	(0,0,1,0,0)	2,1,6	0.014
(2,1,2)	(0,0,1,0,0)	1,6	0.018
(1,1,2)	(0,0,1,0,0)	6	0.013

For board (c), our agent are not able to tell that both cell (1,2) and cell (2,1) is unsafe. Because it cannot tell which cell is Pit and which is Wampus, it will not mark them as unsafe cell. Thus, agent will take a risk and explore one of the two cell.

The behavior of our agent for board (c) is as follow:

Agent (x, y, direction)	Precept (given)	Plan	Elapsed time (s)
(1,1,0)	(1,1,0,0,0)	3,1	46.237
(1,1,1)	(1,1,0,0,0)	1	0.0146

4. Data and Analysis

Using WW3, we ran *CS4300_hybrid_agent* on the boards 3 times each. *CS4300_hybrid_agent* always survives and get the gold on board (a) and (b) and dies on board (c). The time used for each test are listed in the table below:

	Board (a) (s)	Board (b) (s)	Board (c) (s)
Test 1	82.3145	186.3952	46.0969
Test 2	82.0698	186.2435	46.0977
Test 3	82.0365	189.1503	46.1231

5. Interpretation

The *CS4300_hybrid_agent* takes tremendous time each time *CS4300_RTP* is called since it needs to update all uncertain cells. We optimized our function to only update *safe* if agent step into an unvisited cell. This will minimize the amount of time used updating *safe*. Thus, once the agent step into an unvisited cell, it will cause it update *safe*. The more update it did, the more time it will cost. Agent explore 2 cells in board (a), 5 cells in board (b) and 1 cells in board (c). The data matches our prediction where the time *CS4300_hybrid_agent* spend in board (c) is about half of the time it spend in board (a) and about fifth of the time it spend in board (b).

On board (b), a propositional agents would not be not be able to tell that cell (2,2) is safe. On the contrary, hibird agent is able to recognize it as a safe cell and grab the gold in future moves. Furthermore, it is able to tell that cell (1,3) is a Wampus if we add the knowledge of *exact one Wampus* in *KB*.

6. Critique

We used a technique to boil down the most time consuming part in our algorithm, and made a time-out mechanism. This will be useful for future agents. This also sped up the debugging process. However, the testing process was still taking a large amount of time, with each agent call taking almost a minute to finish, making it hard to test more boards.

We did not have a board where the shooting of Wampus is necessary, therefore we did not have the chance to test the effect of that part of code. Further experiments should be done on this matter.

7. Log

Haochen Zhang(Section 1, 3, 5)

A total of 10 hours was spent performing the experiment in Matlab.

A total of 3 hours was spent performing the experiment in writing the report.

Tim Wei (Section 2, 4, 6)

A total of 10 hours was spent performing the experiment in Matlab.

A total of 3 hours was spent performing the experiment in writing the report.