Project 1
A SNePS-Based Rescue Agent
CSE 663, Advanced Knowledge Representation
Due Thursday, February 18, 2010

Stuart C. Shapiro

January 14, 2010

1 Domain Description

The Rescue Agent World, a domain for agent competitions (*see* `http://web.inf.tu-dresden.de/~mit/LRAPP/`) created by Michael Kandefer, is closely related to the Trinity College Robot Fire-fighting competition (`http://www.trincoll.edu/events/robot/`) and the Wumpus World domain (Russell and Norvig, 1995; Russell and Norvig, 2003; Russell and Norvig, 2010; Shapiro and Kandefer, 2005). You are to use and supplement the base SNePSLOG file for the Rescue Agent defined in the file `/projects/robot/Karel/RescueAgent/RescueAgent.snepslog`.

The Rescue Agent operates in an office building that is burning to the ground with employees trapped inside of it. The building itself is a rectangular world of cells arranged in a grid. Cells are adjacent to each other in the four directions (north, south, east and west) and the whole rectangular grid is surrounded by a wall, with rooms created through a semi-random placement of walls inside the rectangular grid. Each cell is identified by its Cartesian coordinates, with cell(0,0) in the north-west corner, cell(1,0) to its east, and cell(0,1) to its south. The width and length of the building are independently randomly set between 5 and 10 cells. The Rescue Agent starts out in cell(0,0) ("exit") facing east.

Some cells in the building contain fire (represented as a red circle with a smaller orange circle inside it), but there is never a fire in cell(0,0) or any cell containing an employee. Each other cell has a 30% probability of containing fire. One to three employees (represented by pink/white squres with brown/black/yellow circles inside) are positioned randomly, but with no person in cell(0,0), in a cell containing another person, or in a cell containing fire.

The task of the Rescue Agent is to find the employees, pick them up, and take them to the exit (to cell(0,0)), and exit when all the employees have been saved. This all must be done before the building collapses, after 1000 moves.

Though the agent itself is fire resistant (it will not perish from walking through fire), employees are not (they will die if carried through fire). To prevent the accidental death of people in transport, the agent is equipped with a fire-fighting foam spray that can put out fires in front of it. It starts with ten foam charges and can replenish them at the exit should they run out. The agent also has heat sensors that can detect fires when in an adjacent cell and a auditory sensor to detect the screams of the victims (up to 3 cells away). Finally, the agent has a visual sensor that can only see one cell in front of it due to the smoke. This sensor can distinguish between fire, people (employees/victims), walls, corridor (nothing), and the exit.

The Rescue Agent is capable of performing the following primitive acts to aid in its goals. Those noted as a "(move)" count against the total of 1000 moves before the building collapses.

- `senseFor(`*s*`)`: If *s* is `heat`, the agent detects if it can feel any heat from the fire. Performs `believe(Feel(heat))` or `believe(~Feel(heat))` if a fire is/is not in an adjacent cell. If *s* is `scream`, the agent detects the loudest victim's scream, and the intensity of that scream. Performs `believe(ScreamLevel(`*n*`))`, where *n* is a number between 0 and 4. With 0 indicating no scream detected and 4 indicating the person is in the same cell as the agent. For $1 \leq n \leq 4$, a scream level of n indicates that a person is $4 - n$ cells away. If *s* is `foamLevel`, the agent senses how many foam charges it has left. Performs `believe(FoamLevel(`*n*`))` where *n* is a number between 0 and 10.
- `getTimeLeft()`: The agent calculates how much time before the building collapses. Performs `believe(TimeLeft(`*n*`))`, where *n* is a number between 0 and 1000.
- `pickUp()` (move): The agent lifts a victim who is in the same cell as it is. Only one person can be carried at a time.
- `putDown()` (move): If the agent is carrying a person, it puts the person down in the same cell as it is. If it is the exit cell the person is rescued.
- `nothing()`: The agent does nothing.
- `extinguishFire()` (move): The agent shoots a foam spray into the cell it is facing. If there is a fire in this cell, it is put out so long as a wall isn't blocking the spray.
- `replenishFoam()` (move): If the agent is in the exit cell, its foam cartridges are replenished.
- `goFoward()` (move): The agent moves into the cell it is facing if a wall is not blocking its path. It automatically believes what it sees in front of it, performing `believe(AheadIs(`*e*`))`, where *e* is either `fire`, `person`, `exit`, `wall`, or `corridor`.
- `turn(`*d*`)` (move): The agent turns 90 degrees in the direction *d* (either `left` or `right`). It automatically believes what it sees in front of it, performing `believe(AheadIs(`*e*`))`, where *e* is either `fire`, `person`, `exit`, `wall`, or `corridor`.
- `exit()`: The agent leaves the building if it is in the exit cell. All victims left inside are considered lost to the fire.
- `say(`*s*`)`: The agent says (prints) the sentence *s*, which is a string enclosed in quotes.

When the agent either exits or dies (when the building collapses), it receives a score, which is printed. The total score is the sum of the following.

- +1000 $\times$ the number of people rescued;
- -1000 $\times$ the number of victims left behind or perished in the fire;
- +20 $\times$ the number of fires put out;
- - the number of moves the agent has made;
- -500 if the agent is destroyed (time runs out).

Note: While your goal is to construct an agent that succeeds in its task, you should try to maximize its score.

2 Defining and Operating Your Agent

You are to create a file of `SNePSLOG` inputs to include and supplement the rescue agent already defined. The first line in your file (after identifying comments) must be

```
load /projects/robot/Karel/RescueAgent/RescueAgent.snepslog
```

In the rest of your file, you are to provide any additional frames you need, along with additional domain rules, initial beliefs, etc.

You can operate your agent by following these steps:

1. `cd` into the directory `/projects/robot/Karel/RescueAgent/`
2. Run Common Lisp.
3. Load SNePs by entering the Lisp command

   ```
   :ld /projects/snwiz/bin/sneps
   ```

4. Evaluate the Common Lisp form `(snepslog)`
5. Load your Rescue Agent by entering: `demo` *`path`*, where *`path`* is the complete path to your agent-defining file.
6. When the Rescue World GUI appears on your screen, click on the Control button labeled “Start”. You may adjust the “Speed” slider as you wish. You probably won’t need to adjust the “Zoom” slider.
7. In the SNePSLOG window, enter `perform` *`act`*, where *`act`* is any act that was supplied with the rescue agent, or that you have defined.
8. Select “Exit” from the “File” menu of the GUI.
9. Either exit from the Common Lisp program, or rerun your Rescue Agent, or a modified version of your Rescue Agent, by following these instructions from step 4 or 5.

You are to define your own rescue agent **only** by providing additional SNePSLOG code. Provide Lisp definitions of new primitive acts only with the prior permission of Prof. Shapiro.

3 Deliverables

You are to hand in a paper, produced using a document formatting program such as LaTeX, or Microsoft Word, and printed on 8.5 by 11 inch paper, stapled in the upper left-hand corner. The paper must explain your solution to the Rescue Agent problem, similar to the style in (Shapiro and Kandefer, 2005), although you needn’t explain the problem, nor need you explain SNePS, SNePSLOG, nor SNeRE. The paper is due at the beginning of class on the date given at the beginning of this document.

In addition to the paper, you are to `submit` your program using `submit\_cse663`. Name your file `RescueAgent.snepslog`. The program is due by one-half hour before the start of class on the date given at the beginning of this document.

4 Grading

The project will be given a letter grade. The following table is a guide to the relative importance of various aspects of the project and paper. Remember, just as (Shapiro and Kandefer, 2005) can be read and appreciated by someone without access to the Wumpus World agent program, your performance on this project will be assessed by reading your paper without reference to your program.

	Apx Weight
Implementation	
Not crashing into walls	4
Finding employees to be rescued	12
Bringing a rescued employee to the exit	12
Exiting when all victims have been rescued	6
Not carrying victims through fire	12
Replenishing foam charges	4
Implementation Subtotal	**50**
Paper	
Paper format	5
General project description	5
English description of KB	8
Syntax/Semantics of every symbol used	24
Correct use of quotation and citations	4
Acknowledgments and References	4
Paper Subtotal	**50**
Project Total	**100**

References

Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, second edition.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, third edition.

Shapiro, S. C. and Kandefer, M. (2005). A SNePS approach to the wumpus world agent or Cassie meets the wumpus. In Morgenstern, L. and Pagnucco, M., editors, *IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'05): Working Notes*, pages 96–103. IJCAII, Edinburgh, Scotland. Available as `http://www.cse.buffalo.edu/~shapiro/Papers/shakan05a.pdf`.