CSE 663, Advanced Knowledge Representation
Homework 2
A SNePS-Based Rescue Agent
Due 9:00 AM, Monday, October 10, 2005
Maximum Points: 50

Stuart C. Shapiro & Michael Kandefer

September 23, 2005

1 Domain Description

The Rescue Agent World is a new domain for agent competitions created by Michael Kandefer, but is most 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, pp.153–157)(Russell and Norvig, 2003, pp. 197–200)(Shapiro and Kandefer, 2005). You are to modify the base SNePSLOG file for the Rescue Agent defined in the file `/projects/robot/Karel/RescueAgent/RescueAgent.snepslog`

The Rescue Agent operates in the General Dynamics building (henceforth GDB) that is burning to the ground with employess trapped inside of it. The GDB 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 GDB 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 on top of 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 on top) are also generated in the house and randomly placed, 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 200 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 employees (up to 3 cells away). Finally, the agent has a visual sensor that can only see one cell infront of it due to the smoke. This sensor can distinguish between fire, people (employees), walls, cooridor (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 200 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 employee'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.

- `do(getTimeLeft)`: The agent calculates how much time before the building collapses. Performs `believe(TimeLeft(`*n*`))`, where *n* is a number between 0 and 200.
- `do(pickUp)` (move): The agent lifts an employee who is in the same cell as it is. Only one person can be carried at a time.
- `do(putDown)` (move): The agent puts down an employee in the same cell as it is. If it is the exit cell the employee is rescued.
- `do(nothing)`: The agent does nothing.
- `do(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.
- `do(replenishFoam)` (move): If the agent is in the exit cell, its foam cartridges are replenished.
- `do(goFoward)` (move): The agent moves into the cell it is facing if a wall is not blocking its path. Automatically believes what it sees in front of it. Performs `believe(AheadIs(`*fire/person/exit/wall/cooridor*`))`.
- `do(turnLeft)` (move): The agent turns left 90 degrees. Automatically believes what it sees in front of it. Performs `believe(AheadIs(`*fire/person/exit/wall/cooridor*`))`.
- `do(turnRight)` (move): The agent turns right 90 degrees. Automatically believes what it sees in front of it. Performs `believe(AheadIs(`*fire/person/exit/wall/cooridor*`))`.
- `do(exit)`: The agent leaves the building if it is in the exit cell. All employees 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 employees rescued;
- -1000 $\times$ the number of employees 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

The first thing you should do is copy the file

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

into your own directory. You should then modify/add to this file to create your own rescue agent.

You can operate your agent by following these steps:

1. `cd` into the directory `/projects/robot/Karel/RescueAgent/`

2. Run Common Lisp.
3. Load the file `/projects/shapiro/Sneps/sneps262`
4. Evaluate the Common Lisp form `(snepslog)`
5. Load your Rescue Agent by entering: `demo "`*`path`*`"`, where *`path`* is the complete path to your modified copy of `/projects/robot/Karel/RescueAgent/RescueAgent.snepslog`
6. When the Rescue World GUI appears on your screen, click on the Control button labeled "Start".
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. Exit from the Common Lisp program.

You are to define your own rescue agent **only** by modifying and adding to the SNePSLOG code of the `RescueAgent` file. 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 Microsoft Word or LaTeX, 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.

In addition to the paper, you are to `submit` your program (using `submit_cse663`), so that it can be run and checked if the instructor chooses. Name your file `RescueAgent.snepslog`. You are to submit your file early enough before class on the day it is due, so that you can get to class to hand in the paper.

4 Grading

Syntax/Semantics of every symbol used	25
Not crashing into walls	2
Finding employees to be rescued	6
Bringing a rescued employee to the exit	6
Exiting when all employees have been rescued	3
Not carrying employees through fire	6
Replenishing foam charges	2
Total	50

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.

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.