Situation Calculus
Assignment II
WS 2015/2016

Gerald Steinbauer
Institute for Software Technology
Organizational Issues

• Dates
  - 10.11.2016 8:45-11:00 (HS i12) lecture and first assignment
  - 17.11.2016 8:45-11:00 (HS i12) lecture and programming assignment
  - 23.11.2016 18:00-18:45 (HS i11) practice
  - 30.11.2016 18:00-18:45 (HS i11) practice and solution for first assignment
  - 28.11.2016 **12:00** submission first assignment
  - 07.12.2016 **23:59** submission programming assignment
Assignment

- help a robot to explore a maze and to find victims
- write a YAGI program that efficiently solves the problem
- evaluate your program in our AI simulator
ASRAEL (Abstract Simulator of Robotics AI for Education and Learning) I

- simulator for teaching and research in AI
- simulator is based on the Unity game engine
- simulates different robot and environments
  - service robot in a kitchen environment
  - the Wumpus world
  - the extended RoboCupJunior rescue setup
- the robot is able to
  - move to move around
  - manipulate objects
  - sense the environment
- the simulator has a simple unified XML-RPC interface to call actions – names and parameters are strings
ASRAEL (Abstract Simulator of Robotics AI for Education and Learning) II

- the robot is able to
  - move forward to next grid cell
  - turn left and right
  - sense for walls, color, and heat
  - ask for score and game state
  - report a found victim
  - leave the maze

- the maze comprises a grid of unknown size
  - a closed wall around the entire grid
  - 4 potential walls around each grid cell
  - victims on walls represented by heat and color
  - fake victims on walls represented by heat or color only
YAGI Interpreter

• yagi is an **interpreter** and interactive **shell** for YAGI
• it provides already a **plugin** to interact with the maze
• action execution in the maze is triggered via action signals
• the **yagi shell** allows
  • importing yagi files, `import("file.y")`; or from command line with option `-f file.y`
  • loading a backend from the command line with option `-b AsraelBackend`
  • interactively **writing** a yagi program, see YAGI language definition
  • **setting** and **querying** of fluents
  • **execution** of procedures and actions
Setup

- **AI simulator**
  - download from course website
  - single archive with binary
  - socket communication to interpreter

- **yagi interpreter**
  - runs on actual Linux systems
  - get the source: git clone http://git.ist.tugraz.at/ais/yagi.git
  - follow the README to compile (remark: special version of libantlr is needed)
  - run the binary ./yagi (stand alone – no real execution of actions)
  - run the binary ./yagi -b AsraelBackend (connection to simulator)
Basic YAGI program

• provided as a starting point, maze.y on the course website
• comprises
  • basic definition for fluents and domains related to rescue domain
  • basic definitions for actions move, left, right, report, and leave
  • basic definition of setting actions which set the local variables
    • senseWall ... $w
    • senseHeat ... $h
    • senseColor ... $c
    • score ... $s
    • state ... $s
• extend it for your own solution
What to Do?

- complete all actions with proper **preconditions** and **effects**
- define additional **fluents** and **actions** if necessary
- implement a procedure **explore()** that
  - explores the maze
  - reports victims if found
  - returns to \([1,1]\) and leaves the maze
- hints
  - do **book keeping** about visited grids and checked walls
  - make use of actions’ **preconditions**, **non-deterministic choices** and search
Submission

• submission per course SVN
  • folder <Repository-URL>/steinbauer
• deadline: 07.12.2016, 23:59 – firm!
• group work of up to 4 students (same as before)
• submit one YAGI file with a program
  • named maze.y
  • the program has to be executable in the YAGI interpreter
  • the file should solve the task in the simulator
  • contains main procedure explore()
• submit one PDF file of 2 to 4 pages
  • contains a description of your solution (fluents, actions, procedures) and a report on performance of your solution (e.g. number of actions needed to solve the task, does it fail sometimes, …)
## Credits

<table>
<thead>
<tr>
<th>Task</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>axiomatization of the domain</td>
<td>25</td>
</tr>
<tr>
<td>regression</td>
<td>25</td>
</tr>
<tr>
<td>programming assignment YAGI</td>
<td>50</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
Questions ?