General Game Playing (GGP)
Winter term 2013/2014

4. Design of GDL Games
## Outline

<table>
<thead>
<tr>
<th>Date</th>
<th>What will we do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.10.2013</td>
<td>Introduction, Repetition propositional logic and FOL</td>
</tr>
<tr>
<td>29.10.2013</td>
<td>Repetition FOL / Datalog and Prolog</td>
</tr>
<tr>
<td>05.11.2013</td>
<td>Game Description Language</td>
</tr>
<tr>
<td><strong>12.11.2013</strong></td>
<td><strong>Design of GDL games</strong></td>
</tr>
<tr>
<td>19.11.2013</td>
<td>Search Algorithms 1</td>
</tr>
<tr>
<td><strong>26.11.2013</strong></td>
<td><strong>Please work on your GGP implementation!</strong></td>
</tr>
<tr>
<td>03.12.2013</td>
<td>Search Algorithms 2</td>
</tr>
<tr>
<td>10.12.2013</td>
<td>State-of-the-art Game players</td>
</tr>
<tr>
<td>17.12.2013</td>
<td><strong>Midterm competition</strong></td>
</tr>
<tr>
<td>14.01.2014</td>
<td>Meta-Gaming</td>
</tr>
<tr>
<td>21.01.2014</td>
<td>Game Theory</td>
</tr>
<tr>
<td>28.01.2014</td>
<td>?</td>
</tr>
<tr>
<td>04.02.2014</td>
<td><strong>Final Competition</strong></td>
</tr>
<tr>
<td>11.02.2014</td>
<td><strong>Exam</strong></td>
</tr>
</tbody>
</table>
Recap

• Game Description Language (GDL)
  – Define games using a logic-based approach

• GDL models:
  – All players (roles) with predicate \texttt{role(player)}
  – The initial state with \texttt{init(proposition)}
  – All state transitions are modeled using \texttt{next(proposition)}
  – All legal actions of each player at each state using \texttt{legal(player, action)}
  – A move with \texttt{does(player, action)}
  – The goal with \texttt{goal(proposition)}
  – Terminal states with \texttt{terminal}
A sample game

role(white)
role(black)

init(s)

legal(white,a)
legal(white,b)
legal(white,c)
legal(black,d)

next(p) :- does(white,a) & ~true(p)
next(p) :- ~does(white,a) & true(p)
next(q) :- does(white,b) & true(p)
next(q) :- does(white,c) & true(r)
next(q) :- ~does(white,b) & ~does(white,c) & true(q)
next(r) :- does(white,c) & true(q)
next(r) :- ~does(white,c) & true(r)

goal(white,100) :- terminal
goal(white,0) :- ~terminal
goal(black,100) :- terminal
goal(black,0) :- ~terminal

terminal :- true(p) & true(q) & true(r)
A sample game

role(white)
role(black)

init(s)

legal(white,a)
legal(white,b)
legal(white,c)
legal(black,d)

next(p) :- does(white,a) & ~true(p)
next(p) :- ~does(white,a) & true(p)
next(q) :- does(white,b) & true(p)
next(q) :- does(white,c) & true(r)
next(q) :- ~does(white,b) & ~does(white,c) & true(q)
next(r) :- does(white,c) & true(q)
next(r) :- ~does(white,c) & true(r)

goal(white,100) :- terminal
goal(white,0) :- ~terminal
goal(black,100) :- terminal
goal(black,0) :- ~terminal

terminal :- true(p) & true(q) & true(r)

(a) Does the game always terminate?
(b) Is the game playable?
(c) Is the game strongly winnable for white?
(d) Is the game weakly winnable for white?
(e) Is the game strongly winnable for black?
(f) Is the game strongly winnable for black?
Before we start …

• Few things we need to discuss:
  – Evaluation platform
  – Who uses which programming language?
  – Time for parsing/making a move
  – Use of external code
Evaluation platform

- 1st option: Evaluation platform at department
  - Gruenau
  - Delphi
  - ghost100,
  - ...

- 2nd option:
  - Use your own hardware

- Which one do you prefer?
Which programming language do you use?

- C++
- Java
- Scala
- ...
Comment on time constraints

- **Time for parsing**
  - You will have enough time to parse the game
  - Amount of time depends on the game

- **Time for making a move**
  - For most games you will not have enough time to compute the whole game tree before/on each move!
  - Amount of time again depends on the game
  - DO NOT assume that you can do a whole A*-search/whatever!

- **To sum up:** Do not make any assumption on the available time at compile time! The values are passed at game initialization only!
Use of external code

• Tricky, since prerequisites for each programming language are quite different
• How to ensure fairness?
• Main message
  – You can use any general framework, that eases
    • Communication
    • Parsing
    • Extraction of states
  – For instance, the GGP base player that makes a legal move
  – You are not allowed to use a fully-fledged game player providing heuristics or search strategies
Today

- We take the textual description of five games and create a GDL representation
- Our games
  - Rock-paper-scissors(-lizard-spock)
  - Diagonal Sudoku
  - 8-puzzle
  - Firesheep
  - Capture the flag
1\textsuperscript{st} game: The original rock-paper-scissors

Nothing beats Rock!

http://acrowbar.files.wordpress.com/2012/11/nothign-beats-rock.png
Rock-paper-scissors: something (un)related

http://www.k2.t.u-tokyo.ac.jp/fusion/Janken/index-e.html
Our game: Rock-paper-scissors-lizard-spock

The rules:

http://www.youtube.com/watch?v=_PUEoDYPuyQ
Rock-paper-scissors-lizard-spock (RPSLS)

It's Simple

https://lh5.googleusercontent.com/-jNInGWhhxIQ/UDDeF_sdYI/AAAAAAAAPu/LUkmThRGqkU/w506-h405/RockPaperScissorsLizardSpock.jpg
GDL for RPSLS

• Number of players?
• Initial state?
• Next relation?
• Legal moves?
• Goal?
• Termination?
RSPLS – One possible solution

role(p1)
role(p2)
init(open)

legal(?x,rock)
legal(?x,paper)
legal(?x,scissors)
legal(?x,lizard)
legal(?x,spock)

(<= terminal (not open))

(<= (next rock1) (does p1 rock))
(<= (next pape1) (does p1 paper))
....
(<= (next rock2) (does p2 rock))
....

(<= (goal p1 100) (and rock1 liza2 ))
(<= (goal p1 100) (and rock1 scis2 ))
...
(<= (goal p1 50) (and rock1 rock2 ))
(<= (goal p1 50) (and pape1 pape2 ))
...
(<= (goal p1 50) (and rock1 rock2 ))
(<= (goal p1 0) (and rock1 pape2 ))
(<= (goal p1 0) (and rock1 spoc2 ))
...
(<= (goal p2 100) (and rock2 liza1 ))
(<= (goal p2 100) (and rock2 liza1 ))
...

2\textsuperscript{nd} game: Diagonal Sudoku

- Each puzzle consists of a 4x4 grid containing given clues in various places. The object is to fill all empty squares so that the numbers 1 to 4 appear exactly once in each row, column and diagonal.
2nd game: Diagonal Sudoku

- Each puzzle consists of a 4x4 grid containing given clues in various places. The object is to fill all empty squares so that the numbers 1 to 4 appear exactly once in each row, column and diagonal.
GDL for Sudoku

- Number of players?
- Initial state?
- Next relation?
- Legal moves?
- Goal?
- Termination?
Sudoku

• One related solution for non-diagonal Sudoku:
3rd game: 8-puzzle

http://www.aiai.ed.ac.uk/~gwickler/images/8-puzzle-states.png
8-puzzle

• What is the problem of 8-puzzle compared to the games we have seen so far, e.g. Tic-Tac-Toe, RPSLS,…?
8-puzzle

• What is the problem of 8-puzzle compared to the games we have seen so far, e.g. Tic-Tac-Toe, RPSLS,…?
  – Now we have potentially infinite games!
  – Solution: Artificially restrict the number of steps (to 30)
8-puzzle

- Constraints on our 8-puzzle game
  - Termination
    - At most 30 steps
  - Goals
    - Arranging the tiles in exactly 30 steps gives 100 points
    - Arranging the tiles in less steps gives 99 points
    - Otherwise 0 points
  - Goal state (with score > 0)
    - 1 in upper left, 2 in upper center, 3 in upper right, …
GDL for 8-puzzle

• Number of players?
• Initial state?
• Next relation?
• Legal moves?
• Goal?
• Termination?
8-puzzle – One possible solution

(role player)
(init (cell 1 1 8))
(init (cell 1 2 7))
(init (cell 1 3 6))
(init (cell 2 1 5))
(init (cell 2 2 4))
...

(succ 1 2)
(succ 2 3)
(pred 2 1)
(pred 3 2)

(<= (legal player (move ?x ?y)) (true (cell ?u ?y b)) (or (succ ?x ?u) (pred ?x ?u)))
(<= (legal player (move ?x ?y)) (true (cell ?x ?v b)) (or (succ ?y ?v) (pred ?y ?v)))

(<= (next (cell ?u ?y ?z)) (does player (move ?x ?y)) (true (cell ?u ?y b)) (true (cell ?x ?y ?z)) (distinct ?z b))
(<= (goal player 100) inorder (true (step 30)))
(<= (goal player 99) inorder (true (step ?x)) (distinct ?x 30))

(<= (goal player 0) (not inorder))
(<= terminal inorder)
(<= terminal (true (step 60)))

(<= inorder (true (cell 1 1 1)) (true (cell 1 2 2)) (true (cell 1 3 3))
(true (cell 2 1 4)) ...)

Complete solution: http://130.208.241.192/ggpserver/public/view_game.jsp?name=8puzzle
4th game: Firesheep

- In a desert land, players must herd sheep to feed at their meager grass patches in the middle of the board.
- Players have three powers that influence individual sheep at their disposal: burn, freeze, and a one-time-use kill. Sheep will move in the direction they have previously been moving if they are left alone. A "burn" action will set a sheep on fire. A burning sheep will remain on fire for one turn and turn to the left once. Any sheep that are on the same tile as a burning sheep will themselves be set on fire. A frozen sheep will not move for one turn, but will turn to the right when it thaws.
- Since players make moves simultaneously, it is possible for a sheep to be both frozen and on fire. In this case, the frozen state takes precedence. A burning sheep will burn away grass tiles over which it traverses. A player will gain 10 points by having at least one sheep on one of his grass patches for two consecutive turns.
GDL for Firesheep

- Number of players?
- Initial state?
- Next relation?
- Legal moves?
- Goal?
- Termination?
GDL for Firesheep

• Complete solution:
5th game: Capture the flag (CTF)

- In a maze with 10x10 fields, a player has to find the path to a flag (located at one field). The player can move left, right, up, down, unless he hits a wall, or remain at his position.
- There are two soldiers in the maze, who can shoot the player in case the distance between the soldier and the player is less than 3 and there is no wall between them.
- The game ends after 60 steps
- Scores:
  - 100, if the player has the flag
  - 50 if the player is alive at the end of the game
  - 0 if the player is dead
GDL for CTF

- Number of players?
- Initial state?
- Next relation?
- Legal moves?
- Goal?
- Termination?
GDL for CTF

- Complete solution (of a similar game):
  - http://130.208.241.192/ggpgroup/public/view_game.jsp?name=war_game02
Conclusions

• We have seen quite different games today
• Whether a game can be modeled in GDL basically depends only on the level of abstraction
• Next week:
  – we will discuss strategies for searching (game) graphs