SAARLAND UNIVERSITY ARTIFICIAL INTELLIGENCE
Dr. Álvaro Torralba and Prof. Wolfgang Wahlster
Dr. Cosmina Croitoru, Daniel Gnad, Marcel Steinmetz
Yannick Körber, Michael Barz
Christian Bohnenberger, Sophian Guidara, Alexander Rath,
El Khansa Rekik, Julia Wichlacz, Anna Wilhelm

**Practical Sheet 2.**
Solutions due Tuesday, **June 5**, 23:59, in the Moodle system.

---

**Exercise 2.** (20 Points)

---

In this sheet your task is to model the following two-player version of *Snake*[1] in GDL[2]. **Use the template file that is provided in Moodle**. More information on the template files is given below.

**Game description.** There are two players: red and blue. Each player controls a snake on a 9-by-9 grid. In each game step, every player needs to move its snake forward to a cell adjacent to the current position of the snake's head. While moving, the snake eats objects that may be located at the new position of the snake's head. There are two kinds of objects: Coins, when eaten, increase the points of the respective player by 3. Apples, when eaten, increase the points of the respective player by 1, and extend the tail of snake by one cell. While apples are consumed by the snakes (are removed from the board after being eaten), coins can be eaten arbitrarily often (remain on the board after being eaten). Moving the snake to a cell is only possible if that cell does not contain a wall. When the snake of player $x$ moves onto a cell which holds this snake or the other snake, then $x$ dies. If red dies, then blue obtains 10 additional points. Vice versa, if blue dies, then red obtains 10 additional points. (If both die, i.e., if they run into each other, both get the additional 10 points.) The game ends when either player died, but after 40 steps at the latest. The player who got most points wins.

**Snakes.** Every snake consists of two parts: a single *head* cell, and possibly multiple *body* cells. Every body cell has a unique successor cell belonging to the snake. The body cell which does not have any predecessor cell is called the *tail*.

[1] https://en.wikipedia.org/wiki/Snake_(video_game_genre)
[2] http://alloyggp.blogspot.de/2013/01/writing-game-with-gdl-sim.html

**Player actions.** There are four possible actions, which allow each player to control its snake: $moveUp$, $moveDown$, $moveLeft$, $moveRight$. Every application of a move action changes the position of the head of the snake depending on the current position of the head, and the move action selected: Let $(x, y)$ denote current position of the head of a snake, and $(x', y')$ the destination location. Depending on the move action, we get

- $moveUp$: $x' = x$ and $y' = y - 1$
- $moveDown$: $x' = x$ and $y' = y + 1$
- $moveLeft$: $x' = x - 1$ and $y' = y$
- $moveRight$: $x' = x + 1$ and $y' = y$

Moving the snake to cell $(x', y')$ is only legal if

(1) the cell at position $(x', y')$ does not contain a wall, and

(2) the move action does not move the snake in the opposite direction of the previous move action, i.e., $(x', y')$ may not be the previous position of the snake's head.

Along the head, every body cell moves forward to its successor cell. If $(x', y')$ contained an apple, the snake is extended at the tail, i.e., making in the next state the cell at the current tail position part of the snake's body; the successor of the new tail cell is defined as the next position of the current tail body cell. For example, consider a snake of size 2 whose head is at position $(3, 3)$ and whose tail is at position $(3, 2)$. Moreover, assume that the cell $(4, 3)$ contains an apple. If the snake moves to the right, the location of the head changes to $(4, 3)$; $(3, 3)$ becomes a body cell since the previous body cell moves forward; the cell $(4, 3)$ becomes the successor of $(3, 3)$; and the cell $(3, 2)$ becomes a new body cell of the snake with successor $(3, 3)$.

**Initial state.** The initial state is shown in Figure 1. You can assume that red already moved its snake from $(4, 2)$ to $(4, 3)$, where it ate an apple. Similarly, blue already moved its snake from $(4, 6)$ to $(4, 5)$, where it also ate an apple. Both players have 1 point.

**Template.** To model this game, you must start from the `snake.kif` template we provide. In this template, we already define the initial state, provide some useful constants, and list the todos that you need to implement along with some rule skeletons that you may want to use to do so. The todos are meant to assist you in encoding the different rules, but you are not required to follow them strictly. Moreover, you are free to define new constants and relations, update the initial state, add new rules, and so on. Your final model must ensure the following:

1. Your model encodes the game described above.

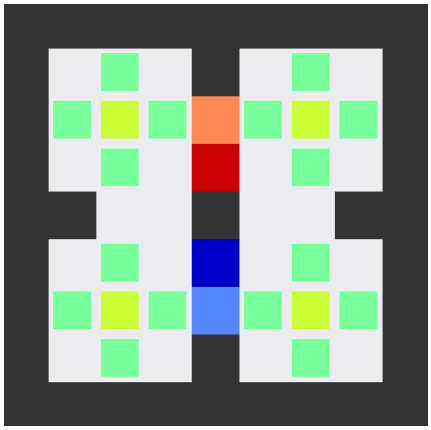

Figure 1: Initial state. Red/blue cells correspond to the snake of the red/blue player, the darker red/blue cells represents the head of the snakes. The green cells represent apples. The yellow cells represent coins. The gray cells represent wall cells impassable by either snake.

2. The actions must be named: *moveUp*, *moveDown*, *moveLeft*, *moveRight*; with the semantics as above.

3. The cell contents must be defined as in the template, i.e., your model must have a relation (*cell* ?*x* ?*y* ?*value*) where

   - $(?x, ?y)$ denotes a position on the board, $?x, ?y \in [0, 8]$ – $(0, 0)$ denotes the top left corner cell of the grid, $(8, 8)$ the bottom right corner cell, and
   - ?*value* is *blueBody* if the cell corresponding to $(?x, ?y)$ contains a body part of the blue snake; ?*value* is *blueHead* if the cell corresponding to $(?x, ?y)$ contains the head of the blue snake; similarly for the red player; ?*value* is *coin* if $(?x, ?y)$ contains a coin; ?*value* is *apple* if $(?x, ?y)$ contains an apple; and ?*value* is *wall* if $(?x, ?y)$ contains a wall.

**Software.** In this project, you need to use two programs that have not been installed in the VM image, initially. Sorry for the inconvenience! You have three options to get the required software.

1. You download the updated VM image from `http://fai.cs.uni-saarland.de/ai18/ai18vm.ova`, and reload it into VirtualBox. Nothing else needs to be done.

2. You keep your current VM image and install the new software using the script that we provide (see details below).

3. If you don't want to use the VM, you can download the tools and install them on your local system.

For the second option, simply copy the install script that is available with this PDF in Moodle to your VM, then run the script (with root privileges) and the required software will be installed automatically. The script is run by typing "sudo sh install_software.sh" in a console in the respective folder. Note that you need to be connected to the university network, otherwise the script will fail.

For the third option, we provided the relevant programs as archives on our webpage:
`http://fai.cs.uni-saarland.de/ai18/eclipse-oxygen-griddle.tar.gz`
`http://fai.cs.uni-saarland.de/ai18/ggp-base-master.zip`

**Usage.** The tools that you should use for this project are **eclipse**, to edit the GDL model and have some syntax checking, and **GGP master**, a general game-playing server that you can use to test your model.

In case you downloaded the new VM image, you don't need to configure anything, but can simply launch both tools from the desktop. You will find the "snake.kif" file in the project that is already loaded in eclipse.

If you installed the tools in your existing VM using the script, you need to import the GDL project in eclipse by clicking *File* → *Import* → *Gradle (Existing Gradle Project)* → *Choose the path "/home/ai/bin/ggp-base-master/games/games/snake" as "Project root directory"* → *Finish*. Then, you should implement the model in the "snake.kif" file.

To launch the GDL server if you are not using the VM, you need to run the "run_server.sh" script in the ggp-base-master folder. The GDL template file is in the following folder:
*ggp-base-master/games/games/snake/snake.kif*

Within the game server, you can start a match with default settings, which takes your current GDL and runs it using the so-called "SampleSearchLight" player. You can also add more players to the server using the "Add" button, for example if you use a RandomGamer player the game should execute a lot faster. To further improve the game speed, you can lower the "Start Clock" and "Play Clock" options. Once a game is started, a new tab will open where you can access the game states and visualizations.

**IMPORTANT** Please add comments to your model. **We will subtract points if you do not put any comments into your model, or your model is difficult to understand with the given comments.**

**Submission instructions** Put your `snake.kif` file into an archive called "name1-name2-name3.zip", where "name1", "name2", "name3" are the family names of all authors. Additionally, **add a file called "authors.txt"** to the archive that contains one line per author, detailing the full name and matriculation number. To upload the archive to the

Moodle system, go to the corresponding assignment, click “Add submission”, and upload it. Note that only one author per group needs to do the submission.