Appendix D

The games-extension

The games-extension provides a convenient way to define normal-form game-theoretic situations. Optimal points and Nash equilibria are calculated and returned to NetLogo in a well-arranged form. The games-extension is installed by creating a directory named `games` in the `extensions` subdirectory of the `NetLogo` program. All files from

`https://github.com/JZschache/NetLogo-games/tree/master/extensions/games`

have to be downloaded and moved to the newly created directory `extensions/games`. For example:

```
git clone https://github.com/JZschache/NetLogo-games.git
mv NetLogo-games/extensions/games path-to-netlogo/extensions
```

If the games-extension is used in combination with the ql-extension, the jars `games.jar` and `gamut.jar` must be added to the variable `additional-jars` in the file `extensions/ql/application.conf`:

```
netlogo {
  ...
  parallel {
    ...
    # all additional jars that must be loaded by NetLogo
    additional-jars = ["extensions/games/games.jar",
                       "extensions/games/gamut.jar"]
    ...
```

```
9     }
10 }
```

This holds true for every extension that is used with the ql-extension. In the next two sections, features of the games-extension are presented, and some details about the calculation of Nash equilibria and optima are given.

D.1 Defining two-person games

Given the games-extension, a two-person game can be defined manually or by a predefined name. The first way is demonstrated with the help of figure D.1.

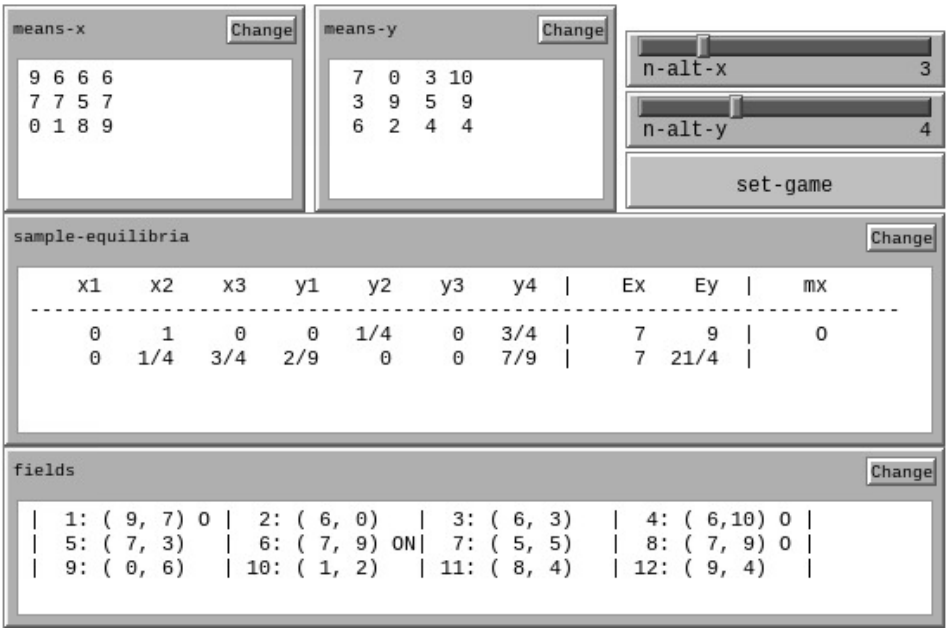

Figure D.1: NetLogo interface of the games-extension

In figure D.1, two NetLogo input fields named `means-x` and `means-y` are seen. Each field contains the mean rewards for player `x` or player `y`, respectively, given the choices of both players. Player `x` is the row-player in both fields. In order to create a game from the two input fields, two *game-matrices* must be created by the reporter `games:matrix-from-row-list`. This function takes a list of lists of numbers as parameter, which is, for example, created by the following reporter:

```
1  to-report read-means-matrix [ nr ]
2    let row-list []
3    let temp means-x
4    if (nr = 2) [set temp means-y]
5    while [temp != ""] [
6      let line-break position "\n" temp
7      ifelse line-break = false [
8        set row-list lput temp row-list
9        set temp ""
10     ] [
11       set row-list lput (substring temp 0 line-break) row-list
12       set temp substring temp (line-break + 1) (length temp)
13     ]
14   ]
15   report (map [read-from-string (word "[ " ? " ]")] row-string-list)
16 end
```

From line 2 until line 14, a list of strings is created and saved to the local variable `row-list`. Each string is a row of the input field. Afterwards, Netlogo's reporter `read-from-string` is deployed to get the required list of lists of numbers.

Using the reporter of the previous listing, a game is defined by the commands `games:matrix-from-row-list` and `games:two-persons-game`:

```
1 to set-game
2   let m1 games:matrix-from-row-list (read-means-matrix 1)
3   let m2 games:matrix-from-row-list (read-means-matrix 2)
4   let game games:two-persons-game m1 m2
5 end
```

The second way of creating a two-person game requires only a name and, occasionally, the numbers of alternatives for both players:

```
1 let game games:two-persons-gamut-game game-name n-alt-x n-alt-y
```

The reporter `games:two-persons-gamut-game` is based on the Gamut library (`http://gamut.stanford.edu`). Gamut makes over thirty games, which are commonly found in the economic literature, available (for details see the documentation: `http://gamut.stanford.edu/userdoc.pdf`). The games-extension currently supports the following parameters as name of a game:

- "BattleOfTheSexes"
- "Chicken"
- "CollaborationGame"
- "CoordinationGame"
- "DispersionGame" (considers first number of alternatives)
- "GrabTheDollar" (considers first number of alternatives)
- "GuessTwoThirdsAve" (considers first number of alternatives)
- "HawkAndDove"
- "MajorityVoting" (considers first number of alternatives)
- "MatchingPennies"
- "PrisonersDilemma"
- "RandomGame" (considers both numbers of alternatives)
- "RandomZeroSum" (considers both numbers of alternatives)
- "RockPaperScissors"
- "ShapleysGame"

It should be noted that some of these names do not generate the commonly expected game. For example, the structure of a "HawkAndDove" game resembles a prisoner's dilemma instead of a game of chicken.

By default, the minimum and maximum payoff is set to zero and ten, respectively. The values are specified in the configuration file `application.conf`. Some of the games take the numbers of alternatives into account. They are given as additional parameters to `games:two-persons-gamut-game`. Only the "RandomGame" and the "RandomZeroSum" do not require that these numbers

match. The other games consider only the first of the two parameters. Finally, the games-extension uses integers for the reward matrices. The values are scaled by changing the parameter `int-mult` in `application.conf` (see also `http://gamut.stanford.edu/userdoc.pdf`, p. 3).

After a game has been defined via Gamut, the input fields and sliders of the NetLogo interface can be updated as demonstrated in the following listing:

```
1  to-report write-means-matrix [ matrix ]
2    let strings games:matrix-as-pretty-strings matrix
3    let result ""
4    foreach strings [
5      set result (word result (reduce [(word ?1 " " ?2 )] ?) "\n")
6    ]
7    report result
8  end
9
10 to set-game
11   let game games:two-persons-gamut-game game-name n-alt-x n-alt-y
12   let m1 games:game-matrix game 1
13   let m2 games:game-matrix game 2
14   let m-strings games:matrix-as-pretty-strings m1 "  "
15   set n-alt-x length m-strings
16   set n-alt-y length first m-strings
17   set means-x write-means-matrix m1
18   set means-y write-means-matrix m2
19   set sample-equilibria games:get-solutions-string game "  "
20   set fields games:get-fields-string game "  "
21 end
```

The two game-matrices are obtained by `games:game-matrix` (lines 12 and 13). The first matrix is used to update the numbers of alternatives `n-alt-x` and `n-alt-y`. Since these values are not directly available, the matrix is converted into a list of lists of strings by `games:matrix-as-pretty-strings`. The strings are "pretty" because it is accounted for differences in length of the numbers by inserting offsets. In the example, two spaces (`" "`) are inserted before every number with only one character. Consequently, the entries of each column are displayed right-aligned in `means-x` or `means-y`. The reporter `write-means-matrix` maps the

list of lists of strings into one string. The NetLogo globals `sample-equilibria` and `fields` (figure D.1) are updated similarly by special commands of the games-extension. Some available commands are described in the following list:

- `games:matrix-transpose` takes a games-matrix as parameter and returns the transpose of this matrix. This reporter assists when defining symmetric games. The input matrix must be quadratic.

- `games:get-reward` returns an entry of a games-matrix. Therefore, three parameters are required: the matrix, a row index, and a column index.

- `games:get-solutions-string` can be used to update a NetLogo input field (see listing above and figure D.1). It prints (strictly) mixed Nash equilibria (if some are found). It also prints the expected reward of each player and indicates, by an `O` in the last column, whether a solution is (Pareto) optimal compared to the other (pure and mixed) solutions. Similar to `games:matrix-as-pretty-strings`, the second parameter is used to adjust the alignment.

- `games:get-fields-string` can be used to update a NetLogo input field (see listing above and figure D.1). It prints a joint payoff matrix. Each field of the matrix contains an index and the mean rewards as specified by the game. It also indicates the pure Nash equilibria (`N`) and pure (Pareto) optima (`O`). Similar to `games:matrix-as-pretty-strings`, the second parameter is used to adjust the alignment.

- `games:pure-solutions` returns a list of boolean values, one for each field of the joint payoff matrix (as given by `games:get-fields-string`). The boolean value indicates whether this field is pure Nash equilibria.

- `games:pure-optima` returns a list of boolean values, one for each field of the joint payoff matrix (as given by `games:get-fields-string`). The boolean value indicates whether this field is (Pareto) optimal compared to the other (pure and mixed) solutions.

D.2 The calculation of equilibria and optima

While pure Nash equilibria are easily identified, the search for a Nash equilibrium of a normal-form game is, in general, computationally intensive (see e.g. Shoham and Leyton-Brown, 2009, ch. 4). More specifically, the problem of finding a Nash equilibrium of a general-sum finite game with two players is PPAD-complete (Daskalakis et al., 2009). For the class of PPAD-complete problems, it is known that at least one solution (Nash equilibrium) exists. But, to the best of the current knowledge, there exists no algorithm that is guaranteed to find this solution in polynomial time (e.g. Papadimitriou, 2014, p. 15884). Instead, the computation time of known algorithms increases exponentially with the number of alternatives.

Nevertheless, existing algorithms run efficiently in practice (e.g. Codenotti et al., 2008). One of the better known (but not the fastest) one (Shoham and Leyton-Brown, 2009, p. 91) is the Lemke-Howard algorithm (Lemke and Howson, 1964). This algorithm is implemented in the games-extension (as given by Codenotti et al., 2008). Even though the Lemke-Howard algorithm necessarily finds a Nash equilibrium, it is generally not able to find all equilibria (Shoham and Leyton-Brown, 2009, p. 98). The implementation of the games-extension tries to find multiple equilibria by starting the algorithm with every possible variable that can be part of the solution (see the pseudocode in Shoham and Leyton-Brown, 2009, p. 96). This step is repeated for every solution that has already been calculated. Since not all Nash equilibria are found, the input field of the NetLogo interface was named `sample-equilibria`.

Furthermore, the problem of stating whether a Nash equilibrium is also Pareto optimal is NP-hard (Shoham and Leyton-Brown, 2009, p. 102). In other words, this problem is "as hard as it can get". The difficulty of this problem stems from the necessity to search an infinite space of possible outcomes. With a finite set of outcomes, the search for the optimal ones can be completed in polynomial time (and, on average, even in linear time, Godfrey et al., 2007). Consequently, the games-extension inspects only the pure and mixed Nash equilibria that are found directly or by the Lemke-Howard algorithm. The labelling of an outcome by an `O` must, hence, be understood relatively to the outcomes that are shown.

Bibliography

Codenotti, B., S. D. Rossi, and M. Pagan (2008, Nov). An experimental analysis of lemke-howson algorithm. `arXiv:0811.3247`.

Daskalakis, C., P. W. Goldberg, and C. H. Papadimitriou (2009). The complexity of computing a Nash equilibria. *SIAM Journal on Computing 39*(1), 195 – 259.

Godfrey, P., R. Shipley, and J. Gryz (2007). Algorithms and aanalyse for maximal vector computation. *The VLDB Journal 16*, 5–28.

Lemke, C. E. and J. T. Howson (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics 12*(2), 413–423.

Papadimitriou, C. (2014). Algorithms, complexity, and the sciences. *Proceedings of the National Academy of Sciences 111*(45), 15881 – 15887.

Shoham, Y. and K. Leyton-Brown (2009). *Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations*. New York: Cambridge University Press.