Introduction to Game Theory

B. Nebel, R. Mattmüller, S. Wölfl
T. Schulte, D. Speck
Summer semester 2016

University of Freiburg
Department of Computer Science

Exercise Sheet P1

Due: Thursday, June 14, 2016

Exercise P1.1 (Naive Algorithm for solving LCPs, 2 + 4 + 2 points)
In this exercise you will implement the naive algorithm for solving LCPs in an open source programming language of your choice[1]. Submit your solution as a compressed archive (zip, gzip, rar, etc.) containing your source code and all the files necessary to compile and run your program via email to `schultet@informatik.uni-freiburg.de`. Don't forget to mention all group members and your tutor's name in the email.

(a) Implement a program that reads in an arbitrary strategic game from an external file and represents it internally. As an input format use our json format for specifying strategic games. See `http://gki.informatik.uni-freiburg.de/teaching/ss16/gametheory/matching-pennies.json` for an example of the Matching Pennies game. Note that most programming languages provide modules or libraries for parsing json files. We recommend using them.

(b) Implement a function that, for a given strategic *two player* game and a pair of support sets, decides whether a solution to the corresponding linear program exists. Use lp_solve[2] to solve the linear program, and print the solution (if one exists) to the console. Your program should be callable from the command line with two parameters (1) a game file and (2) a string specifying the support sets for each player. Consider the following example:

```
> python solvegame.py matching-pennies.json "[H,T][T,H]"
player1: (H=0.5, T=0.5)
player2: (H=0.5, T=0.5)
```

The program solvegame.py outputs the solution to matching-pennies.json for the support sets [H,T] and [T,H] for player1 and player2 respectively.

(c) Extend the functionality of (b) such that if no second argument is provided, the program outputs **all** solutions to the strategic game. Start by writing a routine that generates all possible combinations of support sets. For each combination print the solution (if one exists) to the console.

The exercise sheets may and should be worked on in groups of two to three students.

[1] Python is a good choice.
[2] `http://lpsolve.sourceforge.net/5.5/`