Introduction to Game Theory

B. Nebel, R. Mattmüller
T. Schulte, D. Bergdoll
Summer semester 2018

University of Freiburg
Department of Computer Science

Exercise Sheet 4

Due: Monday, May 14, 2018

Exercise 4.1 (Linear Complementarity Problem, 1.5 + 1.5 points)
Consider the strategic game given by the following payoff matrix:

		Player 2		
		x	y	z
Player 1	a	$0, 0$	$3, 1$	$3, 3$
	b	$1, 1$	$0, 0$	$1, 3$
	c	$1, 1$	$1, 1$	$0, 0$

(a) For the following pair of support sets formulate the corresponding linear program: $(supp(\alpha), supp(\beta)) = (\{a, b, c\}, \{x, y, z\})$.

(b) Solve the linear program and provide values for each $\alpha(a_1)$ and $\beta(a_2)$, $a_1 \in \{a, b, c\}, a_2 \in \{x, y, z\}$. What is the expected payoff (u, v) of the NE computed above?

Exercise 4.2 (Naive Algorithm for solving LCPs, 1 + 3 + 1 points)
In this exercise you will implement the naive algorithm for solving LCPs in an open source programming language of your choice. **We highly recommend using the Python 3 language**. If your solution consists of multiple files, submit 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 in the email.

(a) Implement a program that reads in a two-player 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/ss18/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 the `linprog` function of the `scipy.optimize` module[1] to solve the linear program and print the solution (if one exists)

[1] If you are not using Python 3 despite our recommendation you may want to have a look at lp_solve: `http://lpsolve.sourceforge.net/5.5/`

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 solve.py matching-pennies.json [H,T][T,H]
player1: (H=0.5, T=0.5)
player2: (H=0.5, T=0.5)
```

The program `solve.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 and handed in in groups of three students. Please indicate all names on your solution.