

Multi-Agent Systems

B. Nebel, F. Lindner, T. Engesser
Winter Semester 2018/19

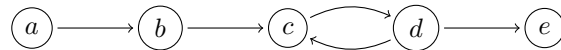
University of Freiburg
Department of Computer Science

Exercise Sheet 11

Due: January 23, 2019, 16:00

Exercise 11.1 (Argumentation Frameworks, 2+1+2)

Consider the following argumentation framework:



- Generate the grounded labeling using the grounded labeling algorithm.
- List all complete labelings. Identify all stable and preferred labelings.
- Use admissible discussions to verify that
 - e is in the **in** set of some preferred labeling,
 - d is in the **in** set of some preferred labeling,
 - c is in the **in** set of some preferred labeling,
 - b is not in the **in** set of any preferred labeling, and
 - a is in the **in** set of some preferred labeling.

Exercise 11.2 (Admissible Discussions, 2+1+1)

We want to write a program that reads a single argumentation framework from a JSON specification file and decides for one given argument whether or not it is part of the **in**-set of some preferred labeling. The JSON object with which we represent an argumentation framework is a single dictionary where the keys are exactly the (names of the) arguments in the framework. The value assigned to each key is a list of exactly the attacked arguments. Both the filename of the JSON specification file and the name of the argument a for which the admissible discussions is to be performed should be passed to your program as command line parameters. The program should then write the following onto the standard output:

- all possible admissible discussions starting with $\text{in}(a)$, each on its own line,
- the winner of each discussion in brackets, at the end of the respective line, as well as
- one final line stating whether a is in for (some|no) preferred labeling.

Consider the following example, where the discussion framework specified in `df.json` is the JSON object `{"a": ["b"], "b": ["c", "d"], "c": ["d", "e"], "d": ["c", "e"], "e": []}` and the argument of interest is d . A call of `python3 discuss.py df.json d` could yield the following output:

```
in(d), out(b), in(a), out(c), in(b) [S]
in(d), out(b), in(a), out(c), in(d) [M]
in(d), out(c), in(b), out(b) [S]
in(d), out(c), in(b), out(a) [S]
in(d), out(c), in(d), out(b), in(a) [M]
d is in for some preferred labeling
```

Exercise 11.3 (CSPs and Admissible Discussions, 2+1)

- Generate a JSON specification file for the argumentation framework representing the constraint satisfaction problem from Exercise 8.1.
- Use your implementation and identify an admissible discussion in which M wins and which contains an assignment for each of the variables. *Hint: You might have to try different initial in-arguments.*