Artificial Intelligence

Assignment 6

Matthias Horbach

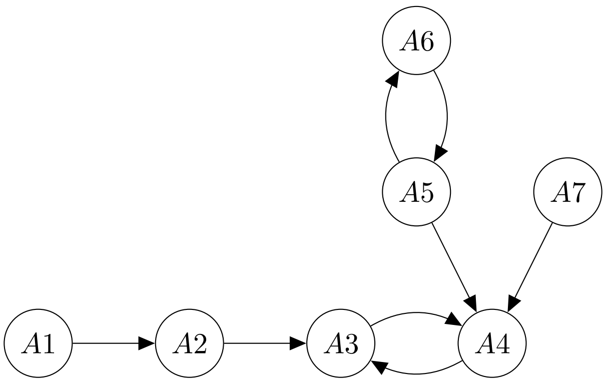
mhorbach@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: 27.06.2020, 23:59
Tutorial on: 28.06.2020

1 Argumentation (8 Points)

Assume the following abstract argumentation framework AF_1:



For each of the following sets, decide whether they are (1) conflict-free and (2) admissible, and whether they are (3) preferred, and/or (4) grounded extensions of AF_1. If they are not, give a short reason why not.

a) $E = \{A_1, A_3, A_7\}$

b) $E = \{A_1, A_4, A_5, A_7\}$

c) $E = \{A_1, A_3, A_6\}$

d) $E = \{A_1, A_3, A_5, A_7\}$

2 Argumentation (5 Points)

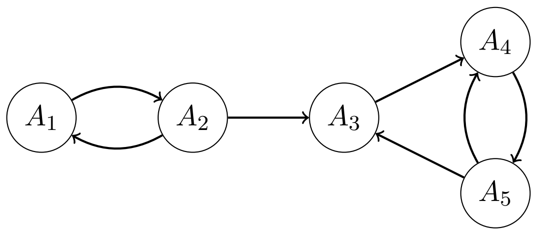
Assume the following abstract argumentation framework AF_2:

$$\begin{tikzcd}
A4 \arrow[r, bend left] & A5 \arrow[l, bend left] \arrow[r] & A6 \\
A2 \arrow[r] & A3 \arrow[u] & \\
A1 \arrow[u, bend left] \arrow[ur, bend right] & &
\end{tikzcd}$$

Find all complete and all stable extensions of AF_2.

3 Encoding Argumentation Semantics using Potassco (35 points)

In this assignment, we compute extensions of an argumentation framework in Potassco, using the example from the lecture:



Your tasks are:

1. Write a Potassco program encoding the complete semantics of the argumentation framework.
2. Write a Potassco program encoding the stable semantics of the argumentation framework.

Use the following steps to create your programs:

- Encode the graph above using two predicates: `argument/1` to indicate that something is an argument and `attacks/2` to indicate that one argument attacks another one.
- In order to generate one model per possible subset $S \subseteq \mathsf{Arg}$, we encode such a set with a predicate `s/1`. The following clauses can be used to make sure that all possible sets are generated:

```
s(A) :- not -s(A), argument(A).
-s(A) :- not s(A), argument(A).
```

- Use a predicate `defeated/1` to denote that a predicate is defeated by S, i.e. that it is attacked by an argument in S.
- Use a predicate `undefended/1` to denote that a predicate is not defended by S. An argument $A' \in \mathsf{Arg}$ is undefended by S if it is attacked by an argument $A \in \mathsf{Arg}$ that is undefeated.
- All extensions must be conflict-free and admissible, i.e. you need constraints stating that no two arguments in S attack each other and that no argument in S is undefended by S.
- Finally, you need constraints that restrict S to complete or stable extensions, respectively. An extension is complete if it contains all arguments that are not undefended. An extension is stable if all arguments that are not in S are defeated.

- Submit both your programs (in individual files) and the output of each program. To limit the output of the ASP solver, use `#show s/1.` to only show the arguments that are contained in S.
- As always, make sure to submit a source code files that Potassco can actually interpret. If your programs produce errors, you may not receive any points for the whole exercise. If you ask Potassco to enumerate all models, you can check your programs' output against the lecture slides 107 and 111.

Important hints

- Always include all names of all group members that helped solving the excercises **on your PDF**. Only those will receive points for solving the excercises.
- By handing in this sheet, you confirm that you solved these excercises yourself. If the situation occurs that two groups have identical solutions, both groups will get zero points.
- Your SVN-Repositories can be accessed via

  `https://svn.uni-koblenz.de/mhorbach/ai23/[yourGroupName]`

  You can use the subfolder `workspace` to share data among your group; this folder's content will not be graded. Submit your solution in the subfolder `solutions` with speaking name such as `assignment6.pdf`. You tutor will upload their notes in the subfolder `comments`.

  Note that you do not have access to the repository's base directory.
- **Format:** All solutions must be contained in PDF documents (including source code). Additionally, source code must be provided as plain files that are readable via a standard text editor.
- Please make sure that all your programs can be run without errors. Comments on your souce code will be in the annotated PDF that we create during excercise corrections.
- Do not use any mutated vowels or special characters in your source code. Also, do not use those or spaces in file names.