The First International Competition on Computational Models of Argumentation (ICCMA'15)

Supplementary Notes on probo

Federico Cerutti Nir Oren Hannes Straß Matthias Thimm Mauro Vallati

Thursday 5th March, 2015

This document contains some supplementary notes to probo, the benchmark framework that is used for ICCMA'15. In particular, this document provides some formal background on abstract argumentation, a listing of computational problems that are (currently) addressed by probo, a list of file formats, and a description of the solver interface supported by probo.

1 Abstract Argumentation

An *abstract argumentation framework* AF is a tuple $\mathsf{AF} = (\mathsf{Arg}, \rightarrow)$ where Arg is a set of arguments and $\rightarrow$ is a relation $\rightarrow \subseteq \mathsf{Arg} \times \mathsf{Arg}$. For two arguments $\mathcal{A}, \mathcal{B} \in \mathsf{Arg}$ the relation $\mathcal{A} \rightarrow \mathcal{B}$ means that argument $\mathcal{A}$ attacks argument $\mathcal{B}$. For $\mathcal{A} \in \mathsf{Arg}$ define $\mathcal{F}_{\mathsf{AF}}(\mathcal{A}) = \{\mathcal{B} \mid \mathcal{B} \rightarrow \mathcal{A}\}$.

Semantics are given to abstract argumentation frameworks by means of extensions (Dung, 1995) or labelings (Caminada and Gabbay, 2009; Wu and Caminada, 2010). For what follows, we use the former. An extension E is a set $E \subseteq \mathsf{Arg}$. Define $F : 2^{\mathsf{Arg}} \rightarrow 2^{\mathsf{Arg}}$ via $F(S) = \{\mathcal{A} \in \mathsf{Arg} \mid S \text{ defends } \mathcal{A}\}$ where a set $S \subseteq \mathsf{Arg}$ defends an argument $\mathcal{A}$ if for all arguments $\mathcal{B} \in \mathsf{Arg}$, if $\mathcal{B} \rightarrow \mathcal{A}$ then there is $\mathcal{C} \in E$ with $\mathcal{C} \rightarrow \mathcal{B}$. For a set $E \subseteq \mathsf{Arg}$, define furthermore $E^+ = \{\mathcal{A} \in \mathsf{Arg} \mid \exists \mathcal{B} \in E : \mathcal{B} \rightarrow \mathcal{A}\}$. E is *conflict-free* if and only if there are no $\mathcal{A}, \mathcal{B} \in E$ with $\mathcal{A} \rightarrow \mathcal{B}$. E is *admissible* if and only if $E \subseteq F(E)$.

Let $\mathsf{AF} = (\mathsf{Arg}, \rightarrow)$ be an abstract argumentation framework and $E \subseteq \mathsf{Arg}$ an extension. Then

CO E is *complete* if and only if $E = F(E)$,

GR E is *grounded* if and only if it is complete and minimal,

PR E is *preferred* if and only if it is complete and maximal,

ST E is *stable* if and only if it is complete and attacks every argument in $\mathsf{Arg} \setminus E$,

All statements on minimality/maximality are meant to be with respect to set inclusion. For more discussion on these semantics see (Baroni *et al.*, 2011).

Note that both grounded and ideal extensions are uniquely determined and always exist (Dung, 1995; Dung *et al.*, 2007). However, most semantics are *multi-extension* semantics. That is, there is not always a unique extension induced by the semantics. In order to reason with multi-extension semantics, usually, one takes either a credulous or skeptical perspective. That is, an argument $\mathcal{A}$ is *credulously inferred* with semantics $\mathcal{S} \in \{CO, GR, PR, ST\}$ if there is a

$\mathcal{S}$-extension E with $\mathcal{A} \in E$. An argument $\mathcal{A}$ is *skeptically inferred* with semantics $\mathcal{S}$ if for all $\mathcal{S}$-extensions E it holds that $\mathcal{A} \in E$.

2 Computational problems

Let $\mathcal{S} \in \{CO, GR, PR, ST\}$ be some semantics. Furthermore, let $\mathsf{AF} = (\mathsf{Arg}, \rightarrow)$ be an argumentation framework, E some extension, $A \subseteq \mathsf{Arg}$ a set of arguments, and $\mathcal{A} \in \mathsf{Arg}$ an argument.

We consider the following computational tasks:

DC-$\mathcal{S}$ **Given** AF, $\mathcal{A}$ **decide** whether $\mathcal{A}$ is credulously inferred.

DS-$\mathcal{S}$ **Given** AF, $\mathcal{A}$ **decide** whether $\mathcal{A}$ is skeptically inferred.

EE-$\mathcal{S}$ **Given** AF **enumerate** all sets $E \subseteq \mathsf{Arg}$ that are $\mathcal{S}$ extensions.

SE-$\mathcal{S}$ **Given** AF **return** some set $E \subseteq \mathsf{Arg}$ that is a $\mathcal{S}$ extension.

3 File formats

Here is the current list of file formats supported by probo, each representing this framework: $\mathsf{AF} = (\mathsf{Arg}, \rightarrow)$ where $\mathsf{Arg} = \{a1, a2, a3\}$, and $\rightarrow = \{(a1, a2), (a2, a3), (a3, a1)\}$.

3.1 Trivial Graph Format

See `http://en.wikipedia.org/wiki/Trivial_Graph_Format`.

```
1
2
3
#
1 2
2 3
3 1
```

3.2 Aspartix Format

See (Egly *et al.*, 2008).

```
arg(a1).
arg(a2).
arg(a3).
att(a1,a2).
att(a2,a3).
att(a3,a1).
```

3.3 CNF Format

This format is basically the CNF format of the SAT solver competition, see `http://logic.pdmi.ras.ru/~basolver/dimacs.html`. Lines beginning with "c" are comment lines and are ignored by the parser. The line beginning with "p af" is the problem definition, it is followed by two positive integers where the first is the number of arguments and the second is the number of attacks in the given framework. After the problem definition there is one line for each attack. The first integer in these lines describe the attacker of the attack and the second the attacked (the attacked is also prefixed by a minus sign "-"). Each line of an attack ends with "0" and a line break.

```
p af 3 3
1 -2 0
2 -3 0
3 -1 0
```

4 Solver Interface

The single executable of a solver should be runnable from a command line and must provide the following behavior (let `solver` be the filename of the executable):

- `solver` (without any parameters)
  Prints author and version information of the solver on standard output.
  Example:

  ```
  user$ solver
  MySolver v1.0
  John Smith
  user$ _
  ```

- `solver --formats`
  Prints the supported formats of the solver in the form

  ```
  [supportedFormat1,supportedFormat2, ..., supportedFormatN]
  ```

  The only possible values for each supported format are `tgf`, `apx`.
  Example:

  ```
  user$ solver --formats
  [tgf,apx]
  user$ _
  ```

- `solver --problems`
  Prints the supported computational problems in the form

  ```
  [supportedProblem1,supportedProblem2, ..., supportedProblemN]
  ```

  The only possible values for each supported problem are `DC-CO`, `DC-GR`, ..., `SE-ST`.
  Example:

  ```
  user$ solver --problems
  [DC-CO,DS-CO,EE-CO,SE-ST]
  user$ _
  ```

- solver -p <problem> -f <file> -fo <fileformat> [-a <additional_parameter>]
  Solves the given problem on the argumentation framework specified by the given file (represented in the given file format) and prints out the result. More specifically:
  - solver -p DC-<semantics> -f <file> -fo <fileformat> -a <additional_parameter>
    Solves the problem of deciding whether an argument (given as additional parameter) is credulously inferred and prints out either YES (if it is credulously inferred) or NO (if it is not credulously inferred).
    Example:
    ```
    user$ solver -p DC-CO -f myFile.apx -fo apx -a Argument1
    YES
    user$ _
    ```
  - solver -p DS-<semantics> -f <file> -fo <fileformat> -a <additional_parameter>
    Solves the problem of deciding whether an argument (given as additional parameter) is skeptically inferred and prints out either YES (if it is credulously inferred) or NO (if it is not credulously inferred).
    Example:
    ```
    user$ solver -p DC-ST -f myFile.txt -fo apx -a Argument2
    NO
    user$ _
    ```
  - solver -p EE-<semantics> -f <file> -fo <fileformat>
    Enumerates all sets that are extensions wrt. the given semantics in the format [[A1,A2,...,AN],[B1,B2,...,BM],..., [Z1,Z2,...,ZN]] (no further parameters needed).
    Example:
    ```
    user$ solver -p EE-PR -f myFile.apx -fo apx
    [[a1,a2],[a5,a8]]
    user$ _
    ```
  - solver -p SE-<semantics> -f <file> -fo <fileformat>
    Returns one extension wrt. the given semantics in the format [A1,A2,...,AN] (no further parameters needed).
    Example:
    ```
    user$ solver -p SE-PR -f myFile.apx -fo apx
    [a1,a2]
    user$ _
    ```
    If there does not exist any extension wrt. the given semantics (which can be the case for stable semantics) then the output NO is expected by the solver.
    Example:
    ```
    user$ solver -p SE-ST -f myFile.apx -fo apx
    NO
    user$ _
    ```

Each solver has to support at least one file format. probo ensures that each solver is only called with file formats and problems he supports (the behavior of a solver when called with unsupported parameters is undefined).

The Java interface net.sf.probo.solver and its descendants AbstractSolver and AbstractDungSolver provide templates for solvers that already partially implement the above interface.

References

P. Baroni, M. Giacomin, and G. Guida. Scc-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168(1–2):162–210, 2005.

Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.

Martin Caminada and Dov M Gabbay. A Logical Account of Formal Argumentation. *Studia Logica (Special issue: new ideas in argumentation theory)*, 93(2–3):109–145, 2009.

Martin Caminada. Semi-stable semantics. In *The First Conference on Computational Models of Argument (COMMA'06)*, pages 121–130, 2006.

P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10–15):642–674, 2007.

P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.

U Egly, S A Gaggl, and S Woltran. ASPARTIX: Implementing Argumentation Frameworks Using Answer-Set Programming. In M de la Banda and E Pontelli, editors, *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, volume 5366 of *Lecture Notes in Computer Science*, pages 734–738. Springer, 2008.

B. Verheij. Deflog: on the logical interpretation of prima facie justified assumptions. *Journal of Logic and Computation*, 13:319–346, 2003.

Y. Wu and M. Caminada. A labelling-based justification status of arguments. *Studies in Logic*, 3(4):12–29, 2010.