Enumerating Preferred Extensions Using ASP Domain Heuristics: The ASPrMin Solver[1]

Wolfgang FABER [a], Mauro VALLATI [b], Federico CERUTTI [c] and Massimiliano GIACOMIN [d]

[a] *Alpen-Adria-Universität Klagenfurt, Austria*
[b] *University of Huddersfield, UK*
[c] *Cardiff University, UK*
[d] *Università degli Studi di Brescia, Italy*

Abstract. This paper briefly describes the solver ASPrMin, which enumerates preferred extensions and scored first in the Extension Enumeration problem—the only one implemented—of the Preferred Semantics Track of the Second International Competition on Computational Models of Argumentation, ICCMA17.

Keywords. argumentation, solver, ASP

1. Abstract Argumentation and Preferred Extensions

We recall some basic notions in abstract argumentation (cf. [2]).

An *argumentation framework* (*AF*) is a pair $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. We say that $\mathbf{b}$ *attacks* $\mathbf{a}$ iff $\langle \mathbf{b}, \mathbf{a} \rangle \in \mathcal{R}$, also denoted as $\mathbf{b} \to \mathbf{a}$. The set of attackers of an argument $\mathbf{a}$ will be denoted as $\mathbf{a}^- \triangleq \{\mathbf{b} : \mathbf{b} \to \mathbf{a}\}$, the set of arguments attacked by $\mathbf{a}$ will be denoted as $\mathbf{a}^+ \triangleq \{\mathbf{b} : \mathbf{a} \to \mathbf{b}\}$.

Given an *AF* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$: a set $S \subseteq \mathcal{A}$ is a *conflict–free* set of Γ if $\nexists\, \mathbf{a}, \mathbf{b} \in S$ s.t. $\mathbf{a} \to \mathbf{b}$; an argument $\mathbf{a} \in \mathcal{A}$ is *acceptable* with respect to a set $S \subseteq \mathcal{A}$ of Γ if $\forall \mathbf{b} \in \mathcal{A}$ s.t. $\mathbf{b} \to \mathbf{a}$, $\exists\, \mathbf{c} \in S$ s.t. $\mathbf{c} \to \mathbf{b}$; a set $S \subseteq \mathcal{A}$ is an *admissible set* of Γ if S is a conflict–free set of Γ and every element of S is acceptable with respect to S of Γ. a set $S \subseteq \mathcal{A}$ is a *preferred extension* of Γ, i.e. $S \in \mathcal{E}_{\mathsf{PR}}(\Gamma)$, if S is a maximal (w.r.t. $\subseteq$) admissible set of Γ.

2. Implementation Using ASP Solver `clingo`

We use a straightforward and well-known encoding for admissible extensions, see [3,1]. Given an *AF* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, for each $a \in \mathcal{A}$ a fact `arg(a).` is created and for

[1] `https://helios.hud.ac.uk/scommv/storage/ASPrMin-v1.0.tar.gz`

each $(a, b) \in \mathcal{R}$ a fact $\texttt{att(a, b)}$. is created (this corresponds to the apx file format in the ICCMA competition). Together with the program

```
in(X) :- not out(X), arg(X).                    out(X) :- not in(X), arg(X).
defeated(X) :- in(Y), att(Y, X).
not_defended(X) :- att(Y, X), not defeated(Y).
:- in(X), in(Y), att(X, Y).                     :- in(X), not_defended(X).
```

we form $admasp_\Gamma$ and there is a one-to-one correspondence between answer sets of $admasp_\Gamma$ and admissible extensions.

We can then exploit domain heuristics in the ASP solver `clasp`, a component of `clingo` [5]. Following [6,4], command line option `--heuristic=Domain` enables domain heuristics, and `--dom-mod=3,16` applies modifier `true` to all atoms that are shown. Since we want to apply the modifier to all atoms with predicate `in`, we augment $admasp_\Gamma$ by the line `#show in/1`. This means that the solver heuristics will prefer atoms with predicate `in` over all other atoms and will choose these atoms as being true first. This will find a subset maximal answer sets with respect to predicate `in`. The system `clingo` also allows for solution recording, see [4], by specifying command line option `--enum-mod=domRec`. Together with the domain heuristic, this will enumerate all subset maximal answer set with respect to predicate `in`.

ASPrMin essentially makes the following call and does some minor post-processing using a shell script:
`clingo` $admasp_\Gamma$ `--heuristic=Domain --dom-mod=3,16 --enum-mod=domRec`

Acknowledgements

We would like to thank Stefan Woltran and Torsten Schaub for pointing us to this way of implementing subset maximality.

References

[1] Günther Charwat, Wolfgang Dvorák, Sarah A. Gaggl, Johannes P. Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation – A survey. *Artificial Intelligence*, 220:28–63, 2015.

[2] Phan M Dung. On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. 77(2):321–357, 1995.

[3] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Marius Lindauer, Max Ostrowski, Javier Romero, Torsten Schaub, and Sven Thiele. Potassco user guide. 2015.

[5] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Clingo* = ASP + control: Preliminary report. In *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP 2014)*, 2014.

[6] Martin Gebser, Benjamin Kaufmann, Javier Romero, Ramón Otero, Torsten Schaub, and Philipp Wanko. Domain-specific heuristics in answer set programming. In Marie desJardins and Michael L. Littman, editors, *Proceedings of AAAI2013*. AAAI Press, 2013.