The ASPrMin Solver – Enumerating Preferred Extensions Using ASP Domain Heuristics

Wolfgang Faber[1], Mauro Vallati[1] Federico Cerutti[2], and Massimiliano Giacomin[3]

[1] University of Huddersfield, Huddersfield, UK,
n.surname@hud.ac.uk
[2] Cardiff University, Cardiff, UK, ceruttif@cardiff.ac.uk
[3] Università degli Studi di Brescia, Brescia, Italy,
massimiliano.giacominb@unibs.it

Abstract. This paper briefly describes the solver ASPrMin, which enumerates preferred extensions. It achieves this by running the ASP solver `clingo` on an encoding for admissible extensions and setting the heuristics in a way such that a subset maximal answer set is found first. It then uses solution recording to find all subset maximal answer sets.

1 Abstract Argumentation and Preferred Extensions

We recall some basic notions in abstract argumentation (cf. [1]).

Definition 1. *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* $\boldsymbol{b}$ attacks $\boldsymbol{a}$ *iff* $\langle \boldsymbol{b}, \boldsymbol{a} \rangle \in \mathcal{R}$*, also denoted as* $\boldsymbol{b} \to \boldsymbol{a}$*. The set of attackers of an argument* $\boldsymbol{a}$ *will be denoted as* $\boldsymbol{a}^- \triangleq \{\boldsymbol{b} : \boldsymbol{b} \to \boldsymbol{a}\}$*, the set of arguments attacked by* $\boldsymbol{a}$ *will be denoted as* $\boldsymbol{a}^+ \triangleq \{\boldsymbol{b} : \boldsymbol{a} \to \boldsymbol{b}\}$*.*

Definition 2. *Given an AF* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$*:*

- *a set* $S \subseteq \mathcal{A}$ *is a* conflict–free *set of* Γ *if* $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ *s.t.* $\boldsymbol{a} \to \boldsymbol{b}$*;*
- *an argument* $\boldsymbol{a} \in \mathcal{A}$ *is* acceptable *with respect to a set* $S \subseteq \mathcal{A}$ *of* Γ *if* $\forall \boldsymbol{b} \in \mathcal{A}$ *s.t.* $\boldsymbol{b} \to \boldsymbol{a}$*,* $\exists\, \boldsymbol{c} \in S$ *s.t.* $\boldsymbol{c} \to \boldsymbol{b}$*;*
- *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 [2, 3].

Definition 3. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, for each $a \in \mathcal{A}$ a fact*

$$\texttt{arg(a).}$$

is created and for 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).
:- in(X), in(Y), att(X, Y).
defeated(X) :- in(Y), att(Y, X).
not_defended(X) :- att(Y, X), not defeated(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` [4]. Following [5, 6], 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 #`showin/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 set with respect to predicate `in`.

The system `clingo` also allows for solution recording, see [6], 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`.

The full command line therefore is:

`clingo` $admasp_\Gamma$ `--heuristic=Domain --dom-mod=3,16 --enum-mod=domRec`

ASPrMin essentially makes this call and does some minor post-processing using a shell sript. ASPrMIN version 1.0 can be downloaded from:
https://helios.hud.ac.uk/scommv/storage/ASPrMin-v1.0.tar.gz.

Acknowledgements

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

References

1. Dung, P.M.: On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. **77**(2) (1995) 321–357
2. Egly, U., Alice Gaggl, S., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument & Computation **1**(2) (2010) 147–177
3. Charwat, G., Dvorák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation – A survey. Artificial Intelligence **220** (2015) 28–63
4. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Clingo* = ASP + control: Preliminary report. In: Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP 2014). (2014)
5. Gebser, M., Kaufmann, B., Romero, J., Otero, R., Schaub, T., Wanko, P.: Domain-specific heuristics in answer set programming. In desJardins, M., Littman, M.L., eds.: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA., AAAI Press (2013)
6. Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S.: Potassco user guide. (2015) Available at https://sourceforge.net/projects/potassco/files/guide/.