ASPARTIX-D: ASP Argumentation Reasoning Tool - Dresden

Sarah Alice Gaggl and Norbert Manthey
Institute of Artificial Intelligence, Technische Universität Dresden, Germany

Abstract. ASPARTIX-D is a system designed to evaluate abstract argumentation frameworks. It consists of collection of answer-set programming (ASP) encodings together with an optimized ASP (resp. SAT) solver configuration for each reasoning problem. The system meets the requirements of the first International Competition on Computational Models of Argumentation (ICCMA 2015).

Keywords: abstract argumentation, ASPARTIX-D, answer-set programming, SAT solving, system

1 Motivation

ASPARTIX-D is the version of ASPARTIX [3,2] which has been further developed in Dresden. In particular, necessary modifications have been performed for the participation in the first International Competition on Computational Models of Argumentation (ICCMA 2015)\(^1\). ASPARTIX-D consists of a collection of answer-set programming (ASP) encodings together with dedicated solvers to compute certain abstract argumentation reasoning tasks. In the following we introduce the necessary background of abstract argumentation frameworks and give an overview of the performed evaluation and final configuration of the system.

2 Semantics and Reasoning Tasks

Abstract argumentation frameworks (AFs) are defined according to [1].

**Definition 1.** An argumentation framework (AF) is a pair \( F = (A, R) \) where \( A \) is a set of arguments and \( R \subseteq A \times A \) is the attack relation. The pair \((a, b) \in R\) means that \( a \) attacks \( b \). An argument \( a \in A \) is defended by a set \( S \subseteq A \) if, for each \( b \in A \) such that \((b, a) \in R\), there exists a \( c \in S \) such that \((c, b) \in R\).

Semantics for argumentation frameworks are given via a function \( \sigma \) which assigns to each AF \( F = (A, R) \) a set \( \sigma(F) \subseteq 2^A \) of extensions. We shall consider here for \( \sigma \) the functions \( ST, CO, PR, \) and \( GR \) which stand for stable, complete, preferred, and grounded semantics respectively.

\(^1\)http://argumentationcompetition.org/.
Definition 2. Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is conflict-free (in $F$), if there are no $a, b \in S$, such that $(a, b) \in R$. $\text{cf}(F)$ denotes the collection of conflict-free sets of $F$. For a conflict-free set $S \in \text{cf}(F)$, it holds that

- $S \in \text{ST}(F)$, if each $a \in A \setminus S$ is attacked by $S$;
- $S \in \text{CO}(F)$, if each $a \in A$ defended by $S$ is contained in $S$;
- $S \in \text{GR}(F)$, if $S \in \text{CO}(F)$ and there is no $T \in \text{CO}(F)$ with $T \subset S$;
- $S \in \text{PR}(F)$, if $S \in \text{CO}(F)$ and there is no $T \in \text{CO}(F)$ with $T \supset S$.

Typical reasoning tasks for any AF $F = (A, R)$ and a semantics $\sigma$ are the following.

- $\text{DC-}\sigma$ credulous reasoning, decide whether $a \in A$ is contained in any $S \in \sigma(F)$;
- $\text{DS-}\sigma$ skeptical reasoning, decide whether $a \in A$ is contained in each $S \in \sigma(F)$;
- $\text{EE-}\sigma$ enumerate all extensions $S \in \sigma(F)$;
- $\text{SE-}\sigma$ return some extension $S \in \sigma(F)$.

2.1 Answer-Set Programming Encodings

ASPARTIX-D is a collection of ASP encodings as described in [3] and optimization encodings which we call in the following metasp encodings as given in [2]. These metasp encodings make use of the metasp optimization front-end for the ASP-package gringo & clasp (see [5] for more details).

The input AF should be specified in the ASPARTIX syntax, i.e. for each argument $a \in A$ one specifies a fact $\text{arg}(a)$ and for each attack $(a, b) \in R$ the fact $\text{att}(a, b)$ should be generated. A typical call of ASPARTIX-D for the reasoning task $\text{DC-}\text{ST}$ looks as follows.

```
./aspartix.sh -p DC-ST -f <file> -fo apx -a <argument>
```

In general ASP encodings are designed to return all (resp. $n$) solutions to a given problem. For credulous and skeptical reasoning we are only interested in a YES or NO decision. As the ASP encodings use predicates $\text{in}/1$ and $\text{out}/1$ to guess the extensions we can perform the following simple modifications. In case of credulous reasoning we just add the argument $a \in A$ in question as the fact $\text{in}(a)$ to the program and check if there is one answer set. If this is the case then, the ASP-solver found one witness extension containing the argument $a$. Otherwise, if the program is unsatisfiable, we know there is no extension which contains $a$. For skeptical reasoning we perform a similar modification, where we add the fact $\text{out}(a)$ to the program. If the program is satisfiable, i.e. an answer set is found, we know that the argument $a$ can not be in each extension of the semantics $\sigma$. However, if the program is unsatisfiable, we obtain that $a$ is skeptically inferred.

3 Evaluation

The main goal of the evaluation was to find the most suitable encodings & solver configuration. As the potassco ASP solvers\(^2\) showed to perform very well for our purpose

\(^2\)http://potassco.sourceforge.net
we decided to test several options from the ASP solver Clingo 4.4 for the original encodings. Furthermore, we considered the gringo3.0.5 & clasp3.1.1 grounder & solver combination for the metasp encodings and the lp2sat & riss SAT Solver [4,7,6], for $DC$-$\{ST, CO, GR\}$ and $DS$-$\{ST, CO, GR\}$.

As benchmarks, we considered a collection of frameworks which have been used by different research groups for testing before consisting of structured and randomly generated AFs, resulting in 5829 frameworks. In particular we used parts of the instances Federico Cerutti provided to us which have been generated towards an increasing number of SCCs [8]. Further benchmarks were used to test the system dynpartix and we included the instances provided by the ICCMA 2015 organizers.

The computation has been performed on an Intel Xeon E5-2670 running at 2.6 GHz. From the 16 available cores we used only every fourth core to allow a better utilization of the CPU’s cache. We applied a 15 minutes timeout and a maximum of 6.5 GB of main memory.

The tests showed in case of $EE$-$PR$ and $SE$-$PR$ the metasp encodings outperform all other options. Also surprisingly the metasp encodings for grounded semantics gave the best results, even though they are not adequate from a complexity point of view (see [2]). This might be due to the fact that the original encodings for grounded semantics use a certain loop construction which have a bad influence on the performance. For the decision tasks $EE$ and $SE$ of stable and complete semantics the clingo option --project returned the best results. Moreover, the lp2sat & riss combination gave better results for the reasoning tasks $DC$-$CO$ and $DS$-$CO$ on real problem instances.

The results of the evaluation led to the following final configuration.

<table>
<thead>
<tr>
<th>Task</th>
<th>Used configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR:</td>
<td>metasp encodings for all reasoning tasks</td>
</tr>
<tr>
<td>DC-ST</td>
<td>original</td>
</tr>
<tr>
<td>DC-CO</td>
<td>$lp2sat$ &amp; riss</td>
</tr>
<tr>
<td>DC-PR</td>
<td>--configuration=auto</td>
</tr>
<tr>
<td>DS-ST</td>
<td>original</td>
</tr>
<tr>
<td>DS-CO</td>
<td>$lp2sat$ &amp; riss</td>
</tr>
<tr>
<td>DS-PR</td>
<td>original</td>
</tr>
<tr>
<td>EE-ST</td>
<td>--project</td>
</tr>
<tr>
<td>EE-CO</td>
<td>--project</td>
</tr>
<tr>
<td>EE-PR</td>
<td>metasp</td>
</tr>
<tr>
<td>SE-ST</td>
<td>--project</td>
</tr>
<tr>
<td>SE-CO</td>
<td>--project</td>
</tr>
<tr>
<td>SE-PR</td>
<td>metasp</td>
</tr>
</tbody>
</table>

The system as well as the benchmarks used for the evaluation are available at https://ddll.inf.tu-dresden.de/web/Sarah_Alice_Gaggl/ASPARTIX-D.
4 Conclusion

The evaluation of different ASP-solver configurations and encoding types clearly showed that with optimized encodings one can obtain better results than only with specific solver options. However, the clingo option \texttt{--project} showed good results for several reasoning tasks. Furthermore, on real problem instances the SAT solver performed better for credulous and skeptical reasoning of complete semantics.

For future work we plan to expand the evaluation also for other argumentation semantics, as for nearly all of them ASP encodings exist (see \url{http://www.dbai.tuwien.ac.at/research/project/argumentation/systempage/#download}).

Acknowledgments: We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous capacity of parallel computation resources. Parts of this project were funded by the Deutsche Forschungsgemeinschaft (DFG) under the number HO 1294/11-1.

References