Abstract

In this report from the field we describe jArgSemSAT, a Java re-implementation of ArgSemSAT. We show that jArgSemSAT can be easily integrated in existing argumentation systems (1) as an off-the-shelf, standalone library; (2) as a Tweety compatible library; and (3) as a fast and robust web service freely available on the Web. The performance section shows that—despite being written in Java—jArgSemSAT is very efficient w.r.t. preferred semantics, which has associated problems with high computational complexity.

Introduction

Dung’s theory of abstract argumentation (Dung 1995) is a unifying framework able to encompass a large variety of specific formalisms in the areas of nonmonotonic reasoning, logic programming and computational argumentation. It is based on the notion of argumentation framework (AF), that consists of a set of arguments and a binary attack relation between them. Different argumentation semantics introduce in a declarative way the criteria to determine which arguments emerge as “justified” from the conflict, by identifying a number of extensions, i.e. sets of arguments that can “survive the conflict together”. In (Dung 1995) three “traditional” semantics are introduced, namely grounded, stable, and preferred semantics, as well as the auxiliary notion of complete extension.

The preferred semantics represents one of the main contributions in Dung’s theory (Dung 1995) and is widely adopted—among other areas—in decision support systems, e.g. (Tamani et al. 2015), and in critical thinking support systems, e.g. (Toniolo et al. 2015), as it allows multiple extensions (differently from grounded semantics), the existence of extensions is always guaranteed (differently from stable semantics), and no extension is a proper subset of another extension. For an introduction on alternative semantics, see (Baroni, Caminada, and Giacomin 2011).

Many problems associated to preferred, but also to stable, semantics turn to be at the high levels of the polynomial hierarchy (Dunne and Wooldridge 2009). In this paper we will focus on the problem of enumerating all preferred extensions given an argumentation framework, which is empirically the most computationally expensive tasks for Dung’s AFs, according to the results of the last competition ICCMA2015.1

ArgSemSAT scored second at ICCMA2015 for enumerating preferred extensions, but the first best solver, CegarTix, did not address other semantics. Moreover, ArgSemSAT is constantly either first or second placed in each track associated to the most expensive problems for stable and preferred semantics—except one due to an implementation bug discovered after the competition.2 Despite this bug, ArgSemSAT scored second at ICCMA2015.

Building on top of the success of ArgSemSAT, in this report from the field we introduce jArgSemSAT that is specifically designed for being easily integrated within existing argumentation systems. Most of the current tools using argumentation technology are based on existing Java approaches such as Dung-O-Matic (Snaith et al. 2010), adopted e.g. in (Toniolo et al. 2015), or on the Tweety libraries for knowledge representation and reasoning (Thimm 2014), or use a web-service interface as ArgTech (Bex et al. 2013). We developed jArgSemSAT in Java, with a specific focus on being compatible with Dung-O-Matic, Tweety, and with a web-service interface compatible with ArgTech. A complete description of jArgSemSAT, together with an empirical evaluation, is provided immediately after a background section on abstract argumentation and its computational problems.

Finally, we conclude the paper with a summary, a qualitative discussion of the benefits of employing jArgSemSAT within CISpaces application (Toniolo et al. 2015), and a discussion on future work.

Background

An argumentation framework (Dung 1995) consists of a set of arguments3 and a binary attack relation between them.

---

1http://argumentationcompetition.org/
2Details in http://downloads.sourceforge.net/project/argsemsat/ArgSemSAT-1.0rc3/ArgSemSAT_1.0rc3.zip
3In this paper we consider only finite sets of arguments: see (Baroni et al. 2013) for a discussion on infinite sets of arguments.
**Definition 1.** An argumentation framework ($AF$) is a pair $\Gamma = \langle A, R \rangle$ where $A$ is a set of arguments and $R \subseteq A \times A$. We say that $b$ attacks $a$ if $(b, a) \in R$, also denoted as $b \rightarrow a$.

Each argumentation framework has an associated directed graph where the vertices are the arguments, and the edges are the attacks.

The basic properties of conflict-freeness, acceptability, and admissibility of a set of arguments are fundamental for the definition of argumentation semantics.

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

- a set $S \subseteq A$ is a conflict-free set of $\Gamma$ if $\not\exists a, b \in S$ s.t. $a \rightarrow b$;
- an argument $a \in A$ is acceptable with respect to a set $S \subseteq A$ of $\Gamma$ if $\forall b \in A$ s.t. $b \rightarrow a$, $\exists c \in S$ s.t. $c \rightarrow b$;
- a set $S \subseteq A$ is an admissible set of $\Gamma$ if $S$ is a conflict-free set of $\Gamma$ and every element of $S$ is acceptable with respect to $S$.

An argumentation semantics $\sigma$ prescribes for any $AF$ $\Gamma$ a set of extensions, denoted as $E_\sigma(\Gamma)$, namely a set of sets of arguments satisfying the conditions dictated by $\sigma$. A preferred extension is a maximal (w.r.t. set inclusion) admissible set.

Enumerating the set of preferred extensions lies at the second level of the polynomial hierarchy, $\Pi_2^p$.

**System design and cases-study**

$jArgSemSAT$ and ArgSemSAT enumerate preferred extensions by multiple calls to a SAT solver (Cerutti et al. 2014a). A propositional formula over a set of Boolean variables is satisfiable iff there exists a truth assignment of the variables such that the formula evaluates to True. Checking whether such an assignment exists is the satisfiability (SAT) problem. The solvers exploit an encoding of complete extensions as a propositional formula in Conjunctive Normal Form (CNF) and apply a filtering procedure over the space of complete extensions to select the maximal ones, i.e. the preferred extensions.

$jArgSemSAT$ is a mature application that now exists in four different versions:

1. **Stand-alone application** $jArgSemSAT$ exports the same command line interface of ArgSemSAT, which is a superset of the Probo interface. In addition to the options discussed in (Cerutti et al. 2014b), $jArgSemSAT$ allows the user to choose (1) the SAT solver to be used; and (2) the encoding to use.

As for the SAT solver, $jArgSemSAT$ allows the user to choose any desired SAT solver—whose full path must be provided—that supports the DIMACS format, accepts a CNF from the STDIN, and returns a model to the STDOUT. In order to provide an off-the-shelf solver, $jArgSemSAT$ also integrates as a library Sat4j (Le Berre and Parrain 2010). Sat4j is an open source library which allows Java programmers to access cross-platform SAT-based solvers.

With respect to the encoding to be used for generating the SAT formulae, $jArgSemSAT$ provides as default the same encoding that has been proven to be the best for ArgSemSAT.

2. **Dung-O-Matic (DoM) compatible library** $jArgSemSAT$ exports methods whose signature are compatible with Dung-O-Matic (Snaith et al. 2010): those methods encapsulate the code for calling $jArgSemSAT$ with the default configurations, and on data-structures that reside on memory instead on a file.

Therefore, the following snippet code:

```java
Vector<String> args
    = new Vector<String>();
args.add("a");
args.add("b");
Vector<String> [] atts
    = new Vector<String []>();
atts.add(new String []{"a", "b"});
```


---

1. Stand-alone application: this guarantees compatibility with the Probo interface for the International Competition on Computational Models of Argumentation (ICCMA) (Cerutti et al. 2014b);
2. Dung-O-Matic (DoM) (Snaith et al. 2010) compatible library: this ensures compatibility for works already using DoM such as CISpaces (Toniolo et al. 2015);
3. Tweety (Thimm 2014) compatible library: we proudly support the Tweety project whose aim is to provide a general framework for implementing and testing knowledge representation formalisms;
4. ArgTech (Bex et al. 2013) compatible web-service: we created a Tomcat web-service exporting $jArgSemSAT$ with ArgTech-compatible RESTful interfaces.

$jArgSemSAT$ is freely (MIT licence) available on SourceForge and as Maven projects directly accessible from the central repository. It is composed by two jar files and a war file.

$jArgSemSATWeb-VERSION.war$ is a self-contained Tomcat 6 web-service archive compatible with ArgTech specifications. This web-service is also available free-of-charge—with best effort SLA—at http://cicero.cs.cf.ac.uk/jArgSemSATWeb/restapi/argtech/. Its source code is also freely available.

---

4https://sourceforge.net/projects/jargsemsat/
5http://search.maven.org/
6http://tomcat.apache.org/
7http://ws.arg.tech/
8http://www.sat4j.org/
new DungAF(args, atts).getStableExts();
is valid if either DoM or jArgSemSAT library is imported.

3. Tweety compatible library In order to guarantee
the full compatibility with the Tweety libraries (Thimm
2014), and to reduce the burden on programmers al-
day already using them, jArgSemSAT Tweety extends the
net.sf.tweety.arg.dung.GroundReasoner,
net.sf.tweety.arg.dung.PreferredReasoner,
and net.sf.tweety.arg.dung.StableReasoner,
overriding only the method computeExtensions
in each of them. As for the DoM compatible library,
jArgSemSAT uses the default configurations only.

4. ArgTech compatible web-service As presented in http://ws.arg.tech/, the ArgTech (Bex et al. 2013) web-service
solver for abstract argumentation problems requires a POST
message with the following fields:
- arguments, type String Array, e.g.
  ["A", "B", "Arg_1"];
- attacks, type String Array, e.g.
  [(A, B), (B, Arg_1)];
- semantics, type String, one of grounded, preferred, stable, semistable.9

For instance, the following JSON structure
{"arguments": ["a", "b"],
 "attacks": [(a, b)],
 "semantics": "stable"}
is a valid POST request for jArgSemSATWeb.
jArgSemSAT is then invoked with the default configu-
rations only.

Empirical Evaluation

This experimental analysis is focused on comparing the
performance of jArgSemSAT with the performance of
ArgSemSAT (Cerutti, Giacomin, and Vallati 2014), in order
to provide a good overview of the performance gap between
the Java-based proposed system and the more efficient C++
implementation. The experiments were performed on a clus-
ter with computing nodes equipped with 2.4 Ghz processors,
4 GB of RAM and Linux operating system. A cutoff of 900
seconds was imposed for analysing a single AF. We consid-
ered the ICCMA2015 benchmark set, which is a set of
192 randomly generated AFs. Here we focused on the em-
pirically most computationally expensive tasks, according to
the results of ICCMA2015, i.e. preferred extensions enumera-
tion. The Penalised Average Runtime (PAR score) has been
used for comparing solvers’ performance. PAR is a real
number which counts (i) runs that fail to solve the consid-
ered problem as ten times the cutoff time (PAR10) and (ii)
runs that succeed as the actual runtime. PAR scores are com-
monly used in automated algorithm configuration, algorithm

To ensure full compatibility, jArgSemSAT contains an exper-
imental implementation of an algorithm for enumerating semi-
stable extensions, cf. Conclusion.

<table>
<thead>
<tr>
<th></th>
<th>jArgSemSAT</th>
<th>ArgSemSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sat4j</td>
<td>97.9</td>
<td>99.5</td>
</tr>
<tr>
<td></td>
<td>13.5</td>
<td>9.9</td>
</tr>
<tr>
<td>PAR10</td>
<td>256.7</td>
<td>97.2</td>
</tr>
</tbody>
</table>

Table 1: Performance achieved by jArgSemSAT exploiting
either Sat4j or glucose3.0, and ArgSemSAT on the IC-
CMA2015 benchmarks. Results are shown in terms of per-
centages of success, percentages of AFs in which the system
has been the fastest and PAR10. Values in bold indicate the
best results.

We are aware that the exploitation of a C++ software can
pose some strong portability issues, mainly due to compilers
and libraries, but C++ solvers are generally believed to
be faster than corresponding Java-based systems. Therefore,
here we are interested in measuring such performance gap,
in order to make jArgSemSAT users aware of the import-
ance of the solver. However, it should be noted that Sat4j
has been included in the overall jArgSemSAT framework,
while glucose3.0 has to be executed through PIPE commu-
nication system among processes.

Table 1 shows the results of the comparison between
jArgSemSAT exploiting the mentioned SAT solvers, and
ArgSemSAT. As a first note, we observe that the perfor-
mance gap between ArgSemSAT and jArgSemSAT is not
remarkable, particularly in terms of percentage of success-
fully analysed frameworks. Beside this, the main conclu-
sion that can be derived from Table 1 is that the use of glu-
cose3.0 does not provide a significant performance improve-
ment. The use of the C++ SAT solver allows jArgSemSAT
to successfully analyse a few more benchmark frameworks,
but the exploitation of Sat4j allows to quickly solve a larger
number of AF.

Surprisingly, the performance of considered SAT solvers
are not directly related to the number of preferred exten-
sions, i.e. there is no direct relation between the number of
times the solver is called by jArgSemSAT and the run-
time. This means that the overhead added by the PIPE com-
munication is negligible and does not impact on the per-
f ormance of jArgSemSAT. On the other hand, Sat4j im-
proves the performance of jArgSemSAT on AFs that can be
solved in less than—approximately—50 CPU-time seconds;
on more complex AFs, the use of glucose3.0 is usually ben-
eficial. The ability of glucose3.0 to handle empirically com-
plex AFs, is confirmed by the fact that the use of glucose3.0
allows jArgSemSAT to solve, within the given time, a few
more AFs from the considered ICCMA2015 benchmark.
Conclusion

In this paper we present jArgSemSAT, an efficient off-the-shelf solver for abstract argumentation problems. In the previous sections we give evidence of how jArgSemSAT is compatible with the current off-the-shelf solver, namely Dung-O-Matic (Snaith et al. 2010), and with the Tweety libraries (Thimm 2014); exists in a web-service version compatible with ArgTech technologies (Bex et al. 2013)— and we made it freely available at http://cicero.cs.cf.ac.uk/jArgSemSATWeb/restapi/argtech/—; and it is only slightly less efficient than its ancestor ArgSemSAT, which is written in C++.

Currently, jArgSemSAT is used within CISpaces (Toniolo et al. 2015) that has been our main use-case. CISpaces (Collaborative Intelligence Spaces) is a tool mostly written in Java—only the GUI is written in Python—to help analysts in acquiring, evaluating and interpreting information. Indeed, the aim of intelligence analysis is to make sense of information that is often conflicting or incomplete, and to weigh competing hypotheses that may explain a situation. This imposes a high cognitive load on analysts, and there are few automated tools to aid them in their task. CISpaces assists analysts in reasoning with different types of evidence: analysts are supported in structuring evidence using argumentation schemes, and in identifying plausible hypotheses via the computation of preferred extensions.

By adopting jArgSemSAT CISpaces now computes the preferred extensions of average analysis almost instantaneously. Moreover, jArgSemSAT allows CISpaces to use its probabilistic argumentation engine in real analysis. Indeed, CISpaces includes a probabilistic argumentation engine (Li, Oren, and Norman 2012; Li 2015) that heavily resides on preferred extensions computed on probabilistic manipulation of AFs. Therefore, the preferred extension enumeration solver needs to be invoked an exponential number of times. Dung-O-Matic limited the use of the probabilistic argumentation engine to toy examples: jArgSemSAT makes it available for real analysis.

The future of jArgSemSAT, in our view, lays in supporting all the research community to build argumentation-based tools. That is the reason that motivated us in providing a free-of-charge, but clearly with best-effort only SLA, web-service interface to jArgSemSAT. From a technical perspective, we need to ultimate the technical documentation and we plan to include support for the remaining semantics, notably semi-stable (Caminada 2006)—that is already supported in an experimental, non-optimised version.

Acknowledgments

The authors thank (in alphabetical order) Prof. Chris Reed and Dr. Mark Snaith—University of Dundee—for their support in ensuring compatibility between jArgSemSAT and existing technologies at ArgTech. Moreover, we thank Dr. Matthias Thimm—Universität Koblenz-Landau—for his support in integrating jArgSemSAT with Tweety, and Dr. Alice Toniolo—University of Aberdeen—for integrating jArgSemSAT within CISpaces. The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work.

References


