jArgSemSAT: An Efficient Off-The-Shelf Solver for Abstract Argumentation Frameworks

Federico Cerutti
Cardiff University
School of Computing Science & Informatics
UK

Mauro Vallati
PARK research group
School of Computing and Engineering
University of Huddersfield, UK

Massimiliano Giacomin
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia, Italy

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 an *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 ICMMA2015 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 arguments[3] and a binary attack relation between them.

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 $\boldsymbol{b} \rightarrow \boldsymbol{a}$.*

[1] http://argumentationcompetition.org/

[2] Details in http://downloads.sourceforge.net/project/argsemsat/ArgSemSAT-1.0rc3/ArgSemSAT_1.0rc3.zip

[3] In this paper we consider only *finite* sets of arguments: see (Baroni et al. 2013) for a discussion on infinite sets of arguments.

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* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$*:*

- *a set* $S \subseteq \mathcal{A}$ *is a* conflict–free *set of* Γ *if* $\nexists \boldsymbol{a}, \boldsymbol{b} \in S$ *s.t.* $\boldsymbol{a} \rightarrow \boldsymbol{b}$*;*
- *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.* $\boldsymbol{b} \rightarrow \boldsymbol{a}$*,* $\exists \boldsymbol{c} \in S$ *s.t.* $\boldsymbol{c} \rightarrow \boldsymbol{b}$*;*
- *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*.*

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

Definition 3. *Given an AF* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$*, a set* $S \subseteq \mathcal{A}$ *is a* preferred extension *of* Γ*, i.e.* $S \in \mathcal{E}_{\mathsf{PR}}(\Gamma)$*, iff* S *is a maximal (w.r.t. set inclusion) admissible set of* Γ*.*

Enumerating the set of preferred extensions lies at the second level of the polynomial hyerarchy, Π_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: 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[4] and as Maven projects directly accessible from the central repository.[5] It is composed by two `jar` files and a `war` file.

`jArgSemSAT-VERSION.jar` provides both the stand-alone application compatible with the Probo interface and the DoM compatible library: we chose not to distribute the library without the Probo interface to facilitate future experiments also from different research groups and to improve the awareness in the community of the ICCMA competition.

`jArgSemSATTweety-VERSION.jar` is a self-contained, Tweety-compatible, library: it includes `jArgSemSAT-VERSION.jar` and provides a Tweety-compatible interface.

`jArgSemSATWeb-VERSION.war` is a self-contained Tomcat [6] web-service archive compatible with ArgTech[7] 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.

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[8] 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:

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

[4] `https://sourceforge.net/projects/jargsemsat/`
[5] `http://search.maven.org/`
[6] `http://tomcat.apache.org/`
[7] `http://ws.arg.tech/`
[8] `http://www.sat4j.org/`

```
atts.add(new String []{"a", "b"});
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 already using them, `jArgSemSATTweety` 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 configurations 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 cluster 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 considered the ICCMA2015 benchmark set, which is a set of 192 randomly generated AFs. Here we focused on the empirically most computationally expensive tasks, according to the results of ICCMA2015, i.e. preferred extensions enumeration.

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 considered problem as ten times the cutoff time (PAR10) and (ii) runs that succeed as the actual runtime. PAR scores are commonly used in automated algorithm configuration, algorithm selection, and portfolio construction, because PAR score allows runtime to be considered while still placing a strong emphasis on coverage.

[9]To ensure full compatibility, jArgSemSAT contains an experimental implementation of an algorithm for enumerating semistable extensions, cf. Conclusion.

	jArgSemSAT		ArgSemSAT
	Sat4j	glucose3.0	
% success	97.9	**99.5**	**99.5**
% best	13.5	9.9	**65.6**
PAR10	256.7	97.2	**81.7**

Table 1: Performance achieved by jArgSemSAT exploiting either Sat4j or glucose3.0, and ArgSemSAT on the ICCMA2015 benchmarks. Results are shown in terms of percentages of success, percentages of AFs in which the system has been the fastest and PAR10. Values in bold indicate the best results.

jArgSemSAT can exploit any SAT solver that supports the DIMACS format. In this analysis we considered the Java-based SAT solver Sat4j—which guarantees high portability and easy usage— and glucose3.0 (Audemard and Simon 2014) that is written in C++.

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 importance 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 communication 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 performance gap between ArgSemSAT and jArgSemSAT is not remarkable, particularly in terms of percentage of successfully analysed frameworks. Beside this, the main conclusion that can be derived from Table 1 is that the use of glucose3.0 does not provide a significant performance improvement. 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 extensions, i.e. there is no direct relation between the number of times the solver is called by jArgSemSAT and the runtime. This means that the overhead added by the PIPE communication is negligible and does not impact on the performance of jArgSemSAT. On the other hand, Sat4j improves 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 beneficial. The ability of glucose3.0 to handle empirically complex 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

Audemard, G., and Simon, L. 2014. Lazy clause exchange policy for parallel sat solvers. In *SAT 2014*. 197–205.

Baroni, P.; Cerutti, F.; Dunne, P. E.; and Giacomin, M. 2013. Automata for Infinite Argumentation Structures. *Artificial Intelligence* 203(0):104–150.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowledge Engineering Review* 26(4):365–410.

Bex, F.; Lawrence, J.; Snaith, M.; and Reed, C. 2013. Implementing the argument web. *Communications of the ACM* 56(10):66.

Caminada, M. 2006. Semi-Stable Semantics. In *COMMA 2006*, 121–130.

Cerutti, F.; Dunne, P. E.; Giacomin, M.; and Vallati, M. 2014a. Computing Preferred Extensions in Abstract Argumentation: A SAT-Based Approach. In *TAFA 2013*, LNCS. 176–193.

Cerutti, F.; Oren, N.; Strass, H.; Thimm, M.; and Vallati, M. 2014b. A Benchmark Framework for a Computational Argumentation Competition. In *COMMA 2014*, 459–460.

Cerutti, F.; Giacomin, M.; and Vallati, M. 2014. ArgSemSAT: Solving Argumentation Problems Using SAT. In *COMMA 2014*, 455–456.

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

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. In *Argumentation in AI*. Springer-Verlag. chapter 5, 85–104.

Le Berre, D., and Parrain, A. 2010. The Sat4j library, release 2.2 system description. *Journal on Satisfiability, Boolean Modeling and Computation* 7(2010):59–64.

Li, H.; Oren, N.; and Norman, T. 2012. Probabilistic Argumentation Frameworks. In *Theorie and Applications of Formal Argumentation*, LNCS. 1–16.

Li, H. 2015. *Probabilistic Argumentation*. Ph.D. Dissertation, U. Aberdeen.

Snaith, M.; Devereux, J.; Lawrence, J.; and Reed, C. 2010. Pipelining argumentation technologies. In *COMMA 2010*, 447–453.

Tamani, N.; Mosse, P.; Croitoru, M.; Buche, P.; Guillard, V.; Guillaume, C.; and Gontard, N. 2015. An argumentation system for eco-efficient packaging material selection. *Computers and Electronics in Agriculture* 113:174–192.

Thimm, M. 2014. Tweety - a comprehensive collection of java libraries for logical aspects of artificial intelligence and knowledge representation. In *KR'14*, 528–537.

Toniolo, A.; Norman, T. J.; Etuk, A.; Cerutti, F.; Ouyang, R. W.; Srivastava, M.; Oren, N.; Dropps, T.; Allen, J. A.; and Sullivan, P. 2015. Agent Support to Reasoning with Different Types of Evidence in Intelligence Analysis. In *AAMAS 2015*, 781—-789.