LabSAT-Solver: Utilizing Caminada's Labelling Approach as a Boolean Satisfiability Problem

Florian Brons

Knowledge-Based Systems, Faculty of Mathematics
and Computer Science, University of Hagen, Germany
`florian.brons@live.de`

Abstract. LabSAT is a solver for computing several reasoning tasks in abstract argumentation frameworks. It enumerates extensions of the complete, preferred, stable and grounded semantics. Further, LabSAT solves the problem of deciding credulously and skeptically. The solver utilizes the labelling approach by Caminada and translates it into a boolean satisfiability problem (SAT).

1 Description

LabSAT [1] is a solver for computing several reasoning tasks in abstract argumentation frameworks [2]. It utilizes the labelling approach by CAMINADA [3] and its encoding as a boolean satisfiability problem (SAT) by CERUTTI et al. [4] to compute several reasoning tasks for the complete, preferred, stable and grounded semantics. To solve the boolean satisfiability problem, the SAT solver lingeling (ayv-86bf266-140429) [5] is used[1].

LabSAT supports all 16 combinations of problems (enumerate, enumerate some, decide credulously and decide skeptically) and semantics (complete, preferred, stable and grounded). The supported file format is the Aspartix file format (apx).

For the implementation Java 7 is used. The connection to the SAT solver, which is implemented in C, is realized with the Java Native Interface (JNI). Every reasoning task is a combination of the type `Problem` and the type `Reasoner`. The abstract class `Reasoner` contains the encoding for the complete extensions. The encoding is adjusted or replaced by concrete classes, which extends the abstract class `Reasoner`. In addition, the abstract class `Reasoner` implements the interface `Iterator`, which allows iterative calls of the SAT solver. Concrete classes, which extend the abstract class `Problem`, use the `Iterator` and handle the results with regard to the problem. The computation is started by the method `solve(reasoner: Reasoner)` in the abstract class `Problem`.

[1] I thank Prof. Dr. Armin Biere for granting permission to use lingeling during the ICCMA'15 contest.

1.1 Complete Extensions

The following definition describes the encoding of complete extensions of an abstract argumentation framework as given by CERUTTI et al. [4] that is used in LabSAT.

Definition 1 (Encoding of complete extensions (cf. [4])). *Given $AF = (\mathcal{A}, \hookrightarrow)$, with $|\mathcal{A}| = k$ and $\phi : \{1, \ldots, k\} \to \mathcal{A}$ an indexing of $\mathcal{A}$. The encoding of complete extensions defined on the variables in $\mathcal{V}(AF)$, is given by the conjunction of the clauses (1)-(5):*

$$\bigwedge_{i \in \{1,\ldots,k\}} ((I_i \vee O_i \vee U_i) \wedge (\neg I_i \vee \neg O_i) \wedge (\neg I_i \vee \neg U_i) \wedge (\neg O_i \vee \neg U_i)) \tag{1}$$

$$\bigwedge_{\{i|\phi(i)^- = \emptyset\}} (I_i \wedge \neg O_i \wedge \neg U_i) \tag{2}$$

$$\bigwedge_{\{i|\phi(i)^- \neq \emptyset\}} \left(\bigvee_{j|\phi(j) \to \phi(i)} \neg I_i \vee O_j \right) \tag{3}$$

$$\bigwedge_{\{i|\phi(i)^- \neq \emptyset\}} \left(\neg O_i \vee \left(\bigvee_{j|\phi(j) \to \phi(i)} I_j \right) \right) \tag{4}$$

$$\bigwedge_{\{i|\phi(i)^- \neq \emptyset\}} \left(\left(\bigwedge_{j|\phi(j) \to \phi(i)} (\neg U_i \vee \neg I_j) \right) \wedge \left(\neg U_i \vee \left(\bigwedge_{j|\phi(j) \to \phi(i)} U_j \right) \right) \right) \tag{5}$$

To determine all complete extensions, LabSAT iterates over all existing extensions and – after displaying the set of arguments that was retrieved – excludes the solution that resulted in satisfiable. Some extension is found by using the same mechanism, in this case the iterator is only called once.

The problem of deciding credulously is solved by adding a clause (I_i) to the SAT solver. The clause ensures that the argument of search belongs to the result, if applicable. If some extension exists, the argument is credulously inferred.

To prove that an argument is in every complete extension, the solver uses the grounded extension. If the argument of search is in the minimal extension wrt. set inclusion, the argument is skeptically inferred.

1.2 Stable Extensions

To compute the stable extensions, additional clauses are added to the SAT solver. For every argument the label `undec` is excluded $(\neg U_i)$. The problems *enumerate* and *enumerate some* are computed in the same way as for the complete extensions. The same applies to the problem *decide credulously*.

In the case of deciding skeptically the iterator is called repeatedly until a counterexample – a set without the argument of search – is found. Otherwise, the argument is skeptically inferred.

1.3 Preferred Extensions

The preferred extensions are computed by using the PrefSAT algorithm published by Cerutti et al. [4]. The algorithm maximizes complete extensions wrt. set inclusion. The solver handles the problems *enumerate*, *enumerate some* and *decide credulously* in the same way as for the complete extensions. The problem *decide skeptically* is solved in the same way as for the stable extensions.

1.4 Grounded Extension

The grounded extension is computed without the use of a SAT solver. The algorithm used for the grounded extension is provided by Modgil/Caminada [6]. Since the grounded extension is unique, the problems *enumerate* and *enumerate some* are the same problem. The problems *decide credulously* and *decide skeptically* are identical problems as well. The grounded extension is computed directly and displayed or checked for the argument of search.

References

1. LabSAT GitHub Archive, https://github.com/fbrns/LabSATSolver
2. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence 77(2), 321–358 (1995)
3. Caminada, M.: On the Issue of Reinstatement in Argumentation in: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.), Logics in Artificial Intelligence. LNCS, vol. 4160, pp. 111–123. Springer, Heidelberg (2006)
4. Cerutti, F., Dunne P., Giacomin, M., Vallati, M.: A SAT-based Approach for Computing Extensions in Abstract Argumentation, in: Black, E., Modgil, S., Oren, N. (eds.), Theory and Applications of Formal Argumentation – Second International Workshop (TAFA 2013), Lecture Notes in Computer Science, vol. 8306, pp. 176–193. Springer, Heidelberg (2014)
5. Biere, A.: Yet another Local Search Solver and Lingeling and Friends Entering the SAT Competition 2014, in: Belov, A., Diepold, D., Heule, M., Järvisalo, M. (eds.), Proceedings of SAT Competition 2014: Solver and Benchmark Descriptions, University of Helsinki Department of Computer Science Series of Publications, vol. B-2014-2, pp. 39–40. Helsinki (2014)
6. Modgil, S., Caminada, M.: Proof Theories and Algorithms for Abstract Argumentation Frameworks, in: Rahwan, I., Simari, G.R. (eds.): Argumentation in Artificial Intelligence, pp. 105–129. Springer, Heidelberg (2009)