CoQuiAAS v2.0: Taking Benefit from Constraint Programming to Solve Argumentation Problems ⋆

Jean-Marie Lagniez[1], Emmanuel Lonca[1], and Jean-Guy Mailly[2]

[1] CRIL – Univ. Artois, CNRS
{lagniez,lonca}@cril.fr,
[2] LIPADE – Univ. Paris Descartes,
jean-guy.mailly@parisdescartes.fr

Abstract. This paper presents how we have extended the existing software CoQuiAAS to handle the new challenges proposed for the ICCMA 2017. The main idea behind CoQuiAAS is to use different Constraint Programming techniques to develop a software library dedicated to argumentative reasoning. More specifically, we use Boolean Satisfiability (SAT) and Maximal Satisfiable Sets extraction (MSS) to solve all the different computational problems from ICCMA 2017. Our library offers the advantages to be efficient, generic and easily adaptable.

1 Introduction

Computational problems related to abstract argumentation frameworks (AFs) [1] are interesting from a theoretical point of view (since argumentation can be used to encode a wide family of non-monotonic inference relations) and from a practical point of view (applications of argumentation exist in various fields like e-democracy, automated negociation or reasoning from inconsistent knowledge). An AF can be seen as a directed graph where the nodes represent *arguments* and the edges represent *attacks* between these arguments. The meaning of such a graph is determined by an *acceptability semantics*, which indicates how to select sets of arguments which can be jointly accepted; such a set of arguments is then called an *extension*.

We have designed a software library called CoQuiAAS to compute the usual reasoning tasks (credulous or skeptical acceptance of an argument, computation of an extension, enumeration of all the extensions) for the classical semantics (grounded, stable, preferred, complete). CoQuiAAS is based on Constraint Programming (CP) techniques: since this domain already proposes some very efficient solutions to solve high complexity combinatorial problems, it is interesting to take benefit from the advances of the CP community to solve efficiently argumentation problems. We are in particular interested in propositional logic and some formalisms derived from it. More precisely, we use some CNF formulae to solve problems from the first level of the polynomial hierarchy, and some encodings in the *Partial Max-SAT* formalism for higher complexity problems. We take advantage of these encodings to solve these reasoning tasks, using some state-of-the-art approaches and software, which have proven their practical efficiency.

⋆ This work benefited from the support of the project AMANDE ANR-13-BS02-0004 of the French National Research Agency (ANR).

After the development of a first version of CoQuiAAS which can handle the different reasoning tasks for the grounded, stable, preferred and complete semantics [2], we have extended our software library to handle the new challenges proposed in the frame of ICCMA 2017, namely solving the usual reasoning tasks for semi-stable [3], stage [4] and ideal semantics [5], and enumerating the grounded, stable and preferred extensions in a row (a.k.a. Dung's Triathlon). This extension of CoQuiAAS has been permitted by its original conception, which makes our platform easily upgradable. Similarly, CoQuiAAS' conception allows to easily benefit from the advances in CP research. For instance, we have been able to easily add the MSS extractor lbx [6] as a replacement of the built-in MSS extractor which was used in the first version of CoQuiAAS [7]. So now CoQuiAAS can benefit from the best approach for extracting MSSes, and it will be once again easy to update CoQuiAAS if a new technique outperforms lbx in the future. Finally, some other improvements have been made in term of conception, resulting in a better separation between argumentation solvers (*complete semantics solvers, preferred semantics solvers, ...*) and constraint solvers (*SAT solvers, MSS computers, ...*).

CoQuiAAS is distributed as an open source software. The current version of CoQuiAAS is available on-line: http://www.cril.univ-artois.fr/coquiaas.

2 Encoding Range-based Semantics

CoQuiAAS takes advantage of the encodings proposed by [8] to compute the extensions of an AF, and to determine whether a given argument is skeptically or credulously accepted by an AF F. The full description of this technique for "basic" and "max-based" semantics is given in [2]. By basic semantics, we mean an argumentation semantics σ such that, for any AF F, we can build a propositional formula Φ_σ^F such that the models of it correspond to the σ-extensions of F (*e.g.* the stable and complete semantics). By "max-based" semantics, we mean the semantics such that only maximal models of the formula correspond to σ-extensions (*e.g.* the preferred semantics). We recall that we encode the problem of searching maximal models into the search of MSSes. Here we describe how we have adapted this technique for range-based semantics, *i.e.* semantics which yield extensions such that their range[3] is maximal. This is the case, for instance, of the semi-stable semantics.

Our encodings are based on propositional logic, defined with the usual connectives on the set of Boolean variables $V_A = \{x_{a_i}, P_{a_i} \mid a_i \in A\}$. The propositional variable x_{a_i} denotes the fact the argument a_i is in an extension of F, and P_{a_i} means that an attacker of a_i is in the considered extension. Interestingly, the P_{a_i} were already used in the first release of CoQuiAAS to reduce the number of generated clauses, and are now also used to encode the ranges; we can encode the fact that a_i is in the range of the extension using the disjunction $x_{a_i} \vee P_{a_i}$. For a matter a readability, we use in the following a_i (resp. P_i) rather than x_{a_i} (resp. P_{a_i}).

Φ_{co}^F is a propositional formula built on the a_i variables such that its models match the complete extensions of F. Taking advantage of our encoding, we define a Partial

[3] The range of a set of arguments S in an AF F is $R(S, F) = S \cup S^+$, where S^+ is the set of arguments attacked by S in F.

Max-SAT instance for the semi-stable semantics Φ_{sst}^F as

$$\Phi_{sst}^F = \{(\Phi_{co}^F, +\infty), (a_1 \vee P_1, 1), \ldots, (a_n \vee P_n, 1)\}$$

The set of semi-stable extensions of F corresponds to the set of MSSes of Φ_{sst}^F. The hard constraint $(\Phi_{co}^F, +\infty)$ states that only a MSS which corresponds to a complete extension can be selected. The soft constraints $(a_i \vee P_i, 1)$ for each argument a_i ensure that the range will be maximized: a MSS of Φ_{sst}^F is an affectation of the Boolean variables such that each hard constraint is satisfied, and the set of soft constraints which are satisfied is maximal w.r.t. $\subseteq$. This corresponds to the definition of semi-stable extensions. The Φ_{stg}^F formula, corresponding to the stage semantics is built in the same way, except that the hard constraint part is replaced by conflict freeness constraints.

Concerning the ideal semantics extension, we compute the maximal extension which only involves arguments that are parts of all the preferred extensions. Thus, we first take advantage of the Φ_{pr}^F formula defined in [2], and then add unit clauses in order to prevent arguments b_i which are not parts of all preferred extensions to appear in the resulting ideal extension, which leads to the following formula :

$$\Phi_{id}^F = \{(\Phi_{co}^F, +\infty), (\neg b_1, +\infty), \ldots, (\neg b_p, +\infty), (a_1, 1), \ldots, (a_n, 1)\}.$$

Finally, the computation of Dung's triathlon mainly consists in computing the set of preferred extensions, since the stable extensions may be determined from the set of preferred extension in polynomial time, while the computation of the grounded extension is also polynomial.

3 CoQuiAAS : Design of the Library

We take advantage of the OOP paradigm offered by the C++ programming language to provide a modular, reusable and easily adaptable architecture, shown at Figure 1.

First, CoQuiAAS is built on top of three main components : the option (command line) parser, the instance parser and the solver parts. Since these three parts are independant, one can easily adapt the code in order to provide alternative ways to enter the argumentation framework or the solver options (*i.e.* use CoQuiAAS as a library instead of a standalone application).

Compared to the older versions of CoQuiAAS, argumentation solvers and constraint solvers are now clearly separated, which decreases the code redundency. Before this architecture improvement, it was necessary to write one solver by couple of semantics and constraint solver ; now, when adding a new constraint solver, it can be immediatly used for any semantics depending of this kind of solver (SAT or MSS).

In the future, we plan to build a real library, allowing any user to use CoQuiAAS in another application without needing any code modification.

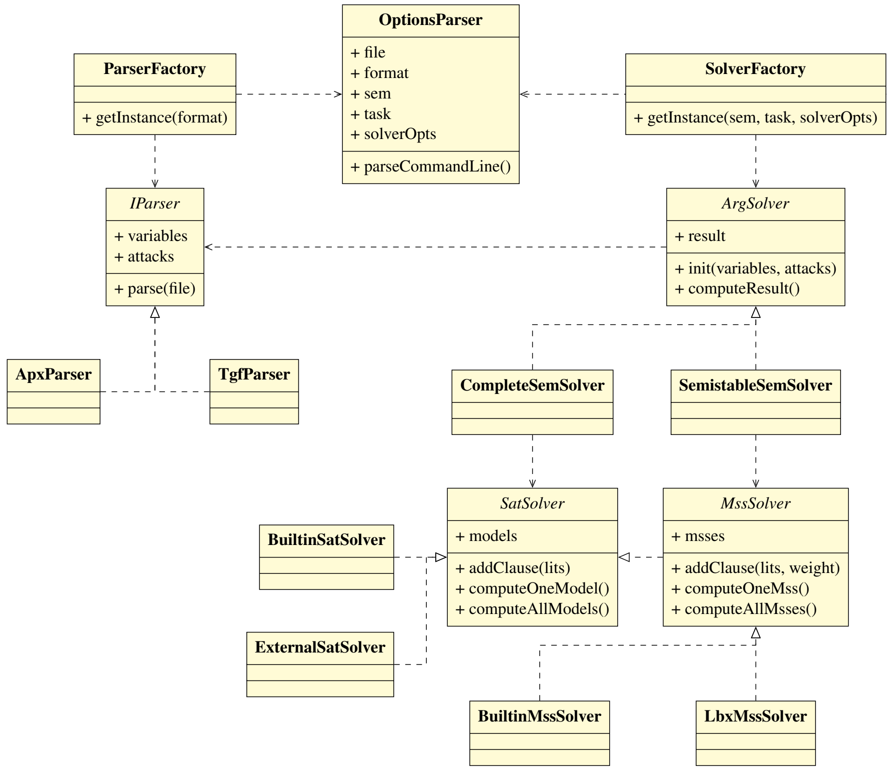

Fig. 1. Simplified class diagram of CoQuiAAS

References

1. 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) (1995) 321–357
2. Lagniez, J.M., Lonca, E., Mailly, J.G.: CoQuiAAS: A constraint-based quick abstract argumentation solver. In: Proc. of ICTAI'15. (2015) 928–935
3. Caminada, M., Carnielli, W.A., Dunne, P.E.: Semi-stable semantics. J. Log. Comput. **22**(5) (2012) 1207–1254
4. Verheij, B.: Two approaches to dialectical argumentation: admissible sets and argumentation stages. In: Proc. of NAIC'96. (1996) 357–368
5. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artif. Intell. **171**(10-15) (2007) 642–674
6. Mencía, C., Previti, A., Marques-Silva, J.: Literal-based MCS extraction. In: Proc. of IJCAI 2015. (2015) 1973–1979
7. Grégoire, É., Lagniez, J.M., Mazure, B.: An experimentally efficient method for (MSS, CoMSS) partitioning. In: Proc. of AAAI'14. (2014) 2666–2673
8. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: Proc. of NMR'04. (2004) 59–64