Implementing Instantiation of Knowledge Bases in Argumentation Frameworks

Hannes STRASS
Computer Science Institute, Leipzig University, Germany

Abstract We present an implementation of Wyner, Bench-Capon and Dunne's [2013] approach to instantiate knowledge bases in argumentation frameworks. The translation is encoded into answer set programming (ASP); the encoding can be used with ASP-based implementations of argumentation frameworks, such as ASPARTIX or DIAMOND.

Keywords. theory bases, defeasible theories, abstract argumentation frameworks, implementation, answer set programming

1. Introduction: Instantiating Knowledge Bases in Argumentation Frameworks

Wyner et al. [4] have presented a method to instantiate theory bases into abstract argumentation frameworks (AFs). Theory bases are simple logic-inspired formalisms working with inference rules on a set of literals. Inference rules can be strict, in which case the conclusion of the inference (a literal) must necessarily hold whenever all antecedents (also literals) hold. Inference rules can also be defeasible, which means that the conclusion *usually* holds whenever the antecedents hold. Here, the word "usually" suggests that there could be exceptional cases where a defeasible rule has not been applied. For literals $\phi_1, \ldots, \phi_n, \psi$, a *strict rule* is of the form $r : \phi_1, \ldots, \phi_n \to \psi$; a *defeasible rule* is of the form $r : \phi_1, \ldots, \phi_n \Rightarrow \psi$.

A major advantage of the approach of Wyner et al. [4] is that the number of arguments created from a theory base is bounded by the size of the theory base. More precisely, for a theory base with r rules on l literals, the resulting AF has exactly $r + l$ arguments; moreover, the AF can be computed in polynomial time. The approach is also intuitive, simple and briefly explained: They create an argument for each literal in the theory base's language and additionally an argument for each rule. Intuitively, the literal arguments indicate that the literal holds, and the rule arguments indicate that the rule is applicable. Furthermore, the defined conflicts between these arguments are straightforward.

(1) opposite literals attack each other; (2) rules are attacked by the negations of their body literals; (3) defeasible rules are attacked by the negation of their head; (4) all rules attack the negation of their head.

In what follows, we briefly present an implementation of this translation based on answer set programming (ASP) [3]. Due to its use of ASP, our encoding can be employed together with ASP-based implementations of abstract argumentation frameworks such as ASPARTIX [1].

2. Using the Encoding

Instances of theory bases are specified by logic program facts using the binary predicates `head` and `body`, and the unary predicate `def`. An atom `head(R, L)` expresses that rule `R` has as its head the literal `L`; likewise for `body(R, L)`. The atom `def(R)` says that rule `R` is defeasible; all rules are considered strict unless declared defeasible. Literals are used by logic program terms, where negation is expressed by the unary function symbol `neg`. Rules are also represented by terms.

To illustrate this representation, we use Example 4 from [4]. There, the strict rules are $r_1 :\rightarrow x_1,\ r_2 :\rightarrow x_2,\ r_3 :\rightarrow x_3,\ r_4 : x_4, x_5 \rightarrow \neg x_3$; the defeasible rules are $r_5 : x_1 \Rightarrow x_4,\ r_6 : x_2 \Rightarrow x_5$. The ASP representation of the example is given by the text file `exampletb.lp` with the following facts:

```
head(r1, x1).   head(r2, x2).   head(r3, x3).
body(r4, x4).   body(r4, x5).   head(r4, neg(x3)).
def(r5).        body(r5, x1).   head(r5, x4).
def(r6).        body(r6, x2).   head(r6, x5).
```

The main encoding file `theorybase.lp` is available at `http://sourceforge.net/p/diamond-adf/code/ci/master/tree/lib/theorybase.lp`. It contains further rules that implement the translation of a theory base in this syntax into an AF in ASPARTIX syntax. To transform a theory base such as `exampletb.lp` into an argumentation framework, we can use the ASP solver clingo [3]:

```
clingo theorybase.lp exampletb.lp
```

To compute extensions of the translated framework, we can directly use ASPARTIX as usual by adding the semantics-specific encoding (e.g. stable) to the call:

```
clingo stable.dl theorybase.lp exampletb.lp
```

To guarantee that extensions of AF translations in their approach are closed under application of strict inference rules, Wyner et al. [4] devised Constraint 5 in Definition 7, which is also implemented in our encoding as an integrity constraint. The encoding presented in this paper is part of our abstract-argumentation software suite DIAMOND [2]. In the future we will incorporate the encoding further into DIAMOND to make it easier to use.

References

[1] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):147–177, 2010.

[2] Stefan Ellmauthaler and Hannes Strass. The DIAMOND System for Computing with Abstract Dialectical Frameworks. In *COMMA*, 2014. This volume.

[3] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[4] Adam Wyner, Trevor J. M. Bench-Capon, and Paul E. Dunne. On the instantiation of knowledge bases in abstract argumentation frameworks. In *Proceedings of CLIMA XIV*, volume 8143 of *LNAI*, pages 34–50. Springer, 2013.