TPLP 13 (4-5): Online Supplement, July 2013. © 2013 [CLAUDIA SCHULZ and FRANCESCA TONI]

ABA-Based Answer Set Justification (Technical Communication)

Claudia Schulz and Francesca Toni
Department of Computing, Imperial College London, UK

submitted 10 April 2013; accepted 23 May 2013

Abstract

We propose a method for justifying why a literal is or is not contained in an answer set of a logic program. This method makes use of argumentation theory, more precisely of stable Assumption-Based Argumentation (ABA) extensions. Given that a logic program can be translated into an ABA framework, we use the structure of ABA arguments and attacks between them for the justification of literals with respect to an answer set.

KEYWORDS: Answer Set Programming, Assumption-Based Argumentation, Stable Extension

1 Introduction

Answer Set Programming has become a widely used technique for the representation of knowledge and efficient problem solving in the field of nonmonotonic reasoning (Anger et al. 2005). Representing a problem as a logic program allows to compute solutions applying the answer set semantics (Gelfond and Lifschitz 1991).

An answer set sanctions a set of literals as justified given a logic program, without providing an actual explanation for these literals. If a literal in an answer set is not an expected result, it is useful to have an explanation for why this literal is justified given the logic program. Similarly, if a literal is not part of an answer set, it is desirable to have an explanation for why this is so.

We present a method for explaining, or "justifying", literals using Assumption-Based Argumentation (ABA) (Bondarenko et al. 1997) and applying the stable extension semantics (Dung 1995; Bondarenko et al. 1997). This argumentation semantics has its roots in the stable model semantics for logic programming, which also forms the basis of answer sets. Due to this common root, answer sets and stable extensions constructed from the same logic program correspond.

We make use of this connection to justify literals with respect to an answer set by means of ABA arguments and attacks between them with respect to the corresponding stable extension. ABA arguments are constructed from rules, corresponding to clauses in the logic program, and are supported by assumptions, corresponding to negation-as-failure literals occurring in these clauses. An argument A for a literal l attacks an argument B if l is the contrary of a negation-as-failure assumption *not* l in the support of B. We

$$\begin{aligned} \text{fly} &\leftarrow \text{bird}, \textit{not}\, \text{abnormalBird} \\ \text{abnormalBird} &\leftarrow \text{bird}, \text{wounded} \\ \neg\text{fly} &\leftarrow \text{wounded} \\ \text{wounded} &\leftarrow \\ \text{bird} &\leftarrow \end{aligned}$$

Fig. 1. Logic Program $\mathcal{P}_{\text{fly}}$

demonstrate our approach, called ABA-Based Answer Set Justification, on the basis of an example.

2 ABA-Based Answer Set Justification

Figure 1 shows a `logic program` with one `negation-as-failure` (NAF) literal, namely *not* abnormalBird. This NAF literal expresses that the first clause can only be applied if the corresponding classical literal abnormalBird cannot be proven to hold, i.e. if abnormalBird is not sanctioned as justified by the answer set semantics.

The `answer set semantics` (Gelfond and Lifschitz 1991) takes this defeasibility of NAF literals into account by applying a fixpoint definition. Using an answer set solver (Gebser et al. 2011; Niemelä et al. 2000; Eiter et al. 1997) for the logic program $\mathcal{P}_{\text{fly}}$ yields the single answer set $\{bird, wounded, abnormalBird, \neg\text{fly}\}$. However, even in this simple case it is not straightforward for humans to understand why for example the literal ¬fly is part of the answer set and why fly is not.

2.1 Translating a logic program into an ABA framework

In order to construct an explanation, or a "justification", for literals with respect to an answer set, we use a method for translating a logic program into an `Assumption-Based Argumentation` (ABA) framework (Bondarenko et al. 1997; Dung et al. 2009). The clauses of the logic program become `rules` ($\mathcal{R}$) in the translated ABA framework and NAF literals occurring in the logic program form the set of `assumptions` ($\mathcal{A}$). The `contrary` of a NAF assumption *not l* is defined as the corresponding classical literal *l*. The `translated ABA framework` of $\mathcal{P}_{\text{fly}}$ is:

- $\mathcal{R}_{\text{fly}} = \mathcal{P}_{\text{fly}}$
- $\mathcal{A}_{\text{fly}} = \{\textit{not}\ \text{abnormalBird}\}$
- $\overline{\textit{not}\ \text{abnormalBird}} = \text{abnormalBird}$

In our approach, `arguments` have the form $Label : (Asms, Facts) \vdash Conc$[1]. *Asms* is a set of assumptions deriving the conclusion *Conc* (a literal) by applying rules from $\mathcal{R}$. *Facts* are all the facts (heads of rules with empty body) used in the derivation of *Conc*. The

[1] For the purpose of providing a justification for literals in an answer set, we introduce unique labels for arguments and add the supporting facts of arguments to the original definition of an ABA argument(Dung et al. 2009).

following arguments can be constructed in the translated ABA framework of $\mathcal{P}_{\text{fly}}$:

$$
\begin{aligned}
A_1 &: (\{\textit{not}\ \text{abnormalBird}\}, \emptyset) \vdash \textit{not}\ \text{abnormalBird}\\
A_2 &: (\emptyset, \{\text{bird}\}) \vdash \text{bird}\\
A_3 &: (\emptyset, \{\text{wounded}\}) \vdash \text{wounded}\\
A_4 &: (\emptyset, \{\text{wounded}\}) \vdash \neg\text{fly}\\
A_5 &: (\emptyset, \{\text{bird}, \text{wounded}\}) \vdash \text{abnormalBird}\\
A_6 &: (\{\textit{not}\ \text{abnormalBird}\}, \{\text{bird}\}) \vdash \text{fly}
\end{aligned}
$$

As an example, A_1 is derived using no rule, supported solely by a NAF assumption. In contrast, A_6 uses the first rule in $\mathcal{R}_{\text{fly}}$, and is supported by a NAF assumption as well as a fact, which is the head of the last rule in $\mathcal{R}_{\text{fly}}$.

An argument `attacks` another argument if the conclusion of the attacker is the contrary of one of the assumptions supporting the attacked one. The only attacks in the translated ABA framework of $\mathcal{P}_{\text{fly}}$ are A_5 attacking both A_1 and A_6, as the conclusion abnormalBird of A_5 is the contrary of the NAF assumption *not* abnormalBird supporting A_1 and A_6.

A `stable extension` (Dung 1995; Bondarenko et al. 1997) of an ABA framework is a set of arguments which does not attack itself but attacks all arguments not contained in it. The translated ABA framework of $\mathcal{P}_{\text{fly}}$ admits a single stable extension: $\{A_2, A_3, A_4, A_5\}$.

When taking a closer look at the conclusions of these arguments, one will notice that they are exactly the same as the literals in the answer set. This result holds in general: For every answer set of a consistent logic program there is a corresponding stable extension, such that for every literal l in the answer set there is a corresponding argument with conclusion l in the corresponding stable extension. Therefore, literals can be justified by means of their corresponding arguments.

2.2 The justification approach

The idea of an `ABA-Based Answer Set (AS) Justification` for a literal l is to find the underlying literals, necessary to derive l, as well as conflicts with other literals which influence whether or not l is part of an answer set. In terms of arguments, the underlying literals are the *Asms* and *Facts* of the corresponding argument of l. Conflicts between l and other literals are attacks on the corresponding argument.

The justification of a literal l is obtained in two steps: first, a tree of attacking arguments, referred to as the `attack tree`, starting with the corresponding argument of l is created; in a second step, the structure of arguments in this tree is used to construct a justification.

The root node of the attack tree is the corresponding argument of the justified literal l. It is labelled '+' if l is in the answer set, and '−' otherwise. A '−' node has exactly one child node, labelled '+' and formed by an argument in the answer set attacking the '−' argument. Conversely, each argument attacking a '+' node forms one of its child nodes and is labelled '−', and so on. The attack tree of fly is shown on the left of Figure 2. The attack tree ends with A_5^+ since no argument attacks A_5. The attack tree of ¬fly (Figure 2, right) contains solely the root node A_4^+ because no argument attacks A_4.

The justification of a literal is obtained from its attack tree as follows: For every

$$\begin{tikzcd}
A_6^{-}: (\{not\ abnormalBird\}, \{bird\}) \vdash fly & \\
& A_4^{+}: (\emptyset, \{wounded\}) \vdash \neg fly \\
A_5^{+}: (\emptyset, \{bird, wounded\}) \vdash abnormalBird \arrow[uu] &
\end{tikzcd}$$

Fig. 2. Attack trees of the literals fly (left) and ¬fly (right)

argument node A : $(Asms, Facts) \vdash Conc$ in the attack tree, the justification contains a support relation between $Conc$ and each element of $Asms$ and $Facts$, as well as an attack relation between the conclusion of each child node of A and the respective attacked assumption supporting A. Furthermore, the justified literal is part of the justification (that is the conclusion of the root node). The justification of fly, drawn from the attack tree in Figure 2 (left), is:

$$\begin{aligned}
just(\text{fly}) = \{\{&\text{fly}, supp\_rel(not\ \text{abnormalBird}, \text{fly}), supp\_rel(\text{bird}, \text{fly}),\\
&att\_rel(\text{abnormalBird}, not\ \text{abnormalBird}), supp\_rel(\text{bird}, \text{abnormalBird}),\\
&supp\_rel(\text{wounded}, \text{abnormalBird})\}\}
\end{aligned}$$

This justification states that fly is supported by the assumption *not* abnormalBird and the fact bird (drawn from the support of A_6). The supporting literal *not* abnormalBird is attacked by abnormalBird (drawn from A_5 attacking A_6), which is supported by the facts bird and wounded (drawn from the support of A_5). The justification set can also be interpreted as a graph, where the literals occurring in the support- and attack-relations are nodes and the relations themselves are edges between these nodes. Figure 3 illustrates the graphical representation of the justification for fly[2]. This graphical representation is the output of a tool for ABA-Based AS Justification that we developed. It distinguishes supporting facts and assumptions from other literals and illustrates whether or not support- and attack-relations succeed.

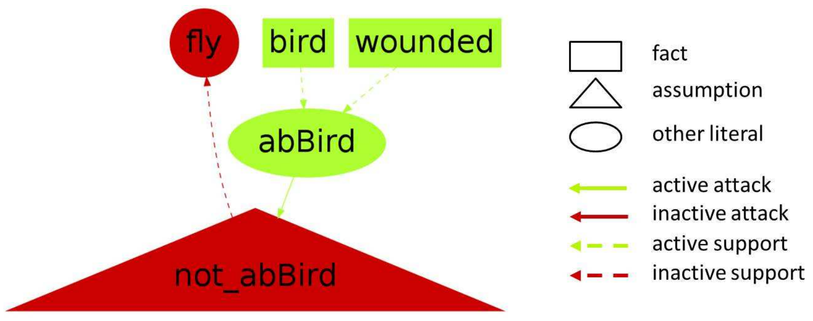

Fig. 3. Graphical justification of literal fly

[2] The full graphical representation of fly should also include an active support relation from bird to fly. This link is omitted from Figure 3 as it is irrelevant for explaining why fly does not belong to the answer set.

3 Conclusion

We present a method for explaining why a literal is or is not contained in an answer set using Assumption-Based Argumentation. To the best of our knowledge, there are only two other justification approaches for answer sets: "off-line justification" (Pontelli et al. 2009) and "Argumentation-Based Answer Set Justification"(Schulz et al. 2013). The strategy used in the first approach is methodologically different from ABA-Based AS Justification in that is applies the well-founded model semantics. The second justification approach uses the ASPIC+ argumentation framework (Prakken 2010).

We plan to use ABA-Based AS Justification in the context of medical treatment decisions. Logic programs will comprise clauses expressing what treatment a patient should get under which circumstances. Using ABA-Based AS Justification will yield a treatment decision (the answer set) along with a justification.

References

Anger, C., Konczak, K., Linke, T., and Schaub, T. 2005. A glimpse of answer set programming. Künstliche Intelligenz 19, 1, 12–19.

Bondarenko, A., Dung, P., Kowalski, R., and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93, 1-2, 63 – 101.

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.

Dung, P. M., Kowalski, R. A., and Toni, F. 2009. Assumption-based argumentation. In Argumentation in Artificial Intelligence, G. Simari and I. Rahwan, Eds. Springer US, 199–218.

Eiter, T., Leone, N., Mateis, C., Pfeifer, G., and Scarcello, F. 1997. A deductive system for non-monotonic reasoning. In Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning. LPNMR'97. 364–375.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Schneider, M. 2011. Potassco: The Potsdam answer set solving collection. AI Communications 24, 2, 105–124.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385.

Niemelä, I., Simons, P., and Syrjänen, T. 2000. Smodels: A system for answer set programming. CoRR cs.AI/0003033.

Pontelli, E., Son, T. C., and Elkhatib, O. 2009. Justifications for logic programs under answer set semantics. Theory and Practice of Logic Programming 9, 1, 1–56.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. Argument and Computation 1, 2, 93–124.

Schulz, C., Sergot, M., and Toni, F. 2013. Argumentation-based answer set justification. In Working notes of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning. Commonsense'13.