Argumentation-Based Answer Set Justification

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

Abstract

We suggest a method for justifying why a literal is or is not contained in the answer set of a logic program. This method makes use of argumentation theory, more precisely of stable ASPIC+ extensions. We describe a way to translate a logic program into an ASPIC+ argumentation theory and investigate the relation between answer sets of the logic program and stable extensions of the translated ASPIC+ argumentation theory. The structure of ASPIC+ arguments with respect to a stable extension is then used for the justification of literals with respect to an answer set. We also present an implementation of our justification method which displays justifications as graphs.

1 Introduction

Answer Set Programming (ASP) 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). Application areas range from bioinformatics (Baral et al. 2004) over music composition (Boenn et al. 2011) to multi-agent systems (Son, Pontelli, and Sakama 2009). ASP allows a problem to be represented in terms of a logic program with defeasible components, namely negation-as-failure (NAF) literals. The models of such a logic program can be constructed applying the answer set semantics (Gelfond and Lifschitz 1991), where answer sets correspond to the solutions of the encoded problem. Answer sets can be efficiently computed using answer set solvers such as clingo (Gebser et al. 2011), smodels (Niemelä, Simons, and Syrjänen 2000) or DLV (Bihlmeyer et al. 2009).

A possible application of logic programs with NAF literals is the representation of an agent's knowledge base, where the answer set corresponds to the agent's beliefs about the world (Watson and Vos 2011). Currently, an agent cannot explain why it has certain beliefs but not others since answer sets come without any justification; they are plain sets of literals. Especially in multi-agent settings involving negotiation, it is desirable that agents can explain their beliefs about the world to other agents. These also include interactions between humans and artificial agents, where the human could be a doctor and the artificial agent a decision making system for patient treatment. In such a situation it would be useful for the doctor to receive an explanation from the decision making agent why it believes that a certain treatment should (or should not) be applied.

Having an explanation of why a literal is or is not contained in an answer set is not only desirable with respect to agent beliefs. ASP is frequently used as a technique for solving complex problems, e.g. (Dimopoulos, Nebel, and Koehler 1997), (Baral et al. 2004), (Son, Pontelli, and Sakama 2009). If the solution to a problem, i.e. the answer set, is not the expected result it is helpful to have an explanation for the unexpected parts of the answer set.

We present a method for justifying literals with respect to an answer set, thereby providing agents with a basis for explaining their beliefs. Furthermore, literals which were not expected as part of a solution can be justified.

Our justification approach applies another influential technique in nonmonotonic reasoning, namely argumentation theory. Here we use the ASPIC+ framework (Prakken 2010), a structured argumentation framework building arguments from rules, facts, assumptions etc. One of the semantics of argumentation frameworks is given by stable extensions (Dung 1995). This 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 use this connection to justify literals with respect to an answer set by means of ASPIC+ arguments with respect to the corresponding stable extension.

We call our method Argumentation-Based Answer Set Justification. The justification of a literal is a set containing all facts and NAF-literals necessary to derive the literal in question as well as conflicts with other literals. Our implementation of Argumentation-Based Answer Set Justification constructs a graph from this justification set, illustrating both the supporting and conflicting literals of the literal in question. This implementation, called JASA (Justification of Answer Sets via Argumentation), makes use of the answer set solver clingo (Gebser et al. 2011) and the online ASPIC+ tool TOAST (Snaith and Reed 2012).

The paper is organised as follows: After introducing some key definitions about answer sets and argumentation theory in section 2, we explain Argumentation-Based Answer Set Justification in section 3 and present its implementation in Section 4. Section 5 compares Argumentation-Based Answer Set Justification to other approaches and in particular to a method called off-line justification (Pontelli, Son, and

Elkhatib 2009). Section 6 concludes our work and provides a prospect for future work.

2 Background

An `extended logic program` $\mathcal{P}$ is a set of labelled clauses of the form $r : l \leftarrow l_1, \dots, l_m, not\ l_{m+1}, \dots, not\ l_{m+n}$ with $m \geq 0$ and $n \geq 0$, where r is the clause's unique label. l and all l_i are classical literals, i.e. positive atoms a or negated atoms $\neg a$. $not\ l_{m+1}, \dots, not\ l_{m+n}$ are called `negation-as-failure` (NAF) literals. For $r : l \leftarrow l_1, \dots, l_m, not\ l_{m+1}, \dots, not\ l_{m+n}$, $head(r)$ will denote the head l, $body^+(r)$ stands for the set of classical literals $l_1, \dots, l_m$ and $body^-(r)$ for the set of NAF literals $not\ l_{m+1}, \dots, not\ l_{m+n}$ in the clause's body. Let $Lit_{\mathcal{P}}$ be the Herbrand Universe of $\mathcal{P}$.

In the following we will recall the concept of answer sets introduced in (Gelfond and Lifschitz 1991). Let $\mathcal{P}$ be an extended logic program without NAF literals. The `answer set` of $\mathcal{P}$, denoted $\mathcal{AS}(\mathcal{P})$, is the smallest set $S \subseteq Lit_{\mathcal{P}}$ such that:

1. if $l_1, \dots, l_m \in S$ then $l \in S$ for any clause $r : l \leftarrow l_1, \dots, l_m$ in $\mathcal{P}$;
2. $S = Lit_{\mathcal{P}}$ if S contains complementary literals a and $\neg a$.

Given an extended logic program $\mathcal{P}$, possibly containing NAF literals, for any subset $S \subseteq Lit_{\mathcal{P}}$ the `reduct` $\mathcal{P}^S$ is obtained from $\mathcal{P}$ by deleting:

1. all clauses with $not\ l$ in their bodies where $l \in S$,
2. all NAF literals in the remaining clauses.

Then, S is an `answer set` of $\mathcal{P}$ if it is the answer set of $\mathcal{P}^S$, that is if $S = \mathcal{AS}(\mathcal{P}^S)$. A logic program is called `inconsistent` if its only answer set is $Lit_{\mathcal{P}}$, and `consistent` otherwise.

We will now introduce another important technique used in nonmonotonic reasoning, namely argumentation theory. There are various structured argumentation frameworks, e.g. (Dung, Kowalski, and Toni 2009), (Garcia and Simari 2004), (Governatori et al. 2004), but we will here focus on the ASPIC+ framework as introduced in (Prakken 2010). This choice is mainly due to an existing implementation called TOAST which computes extension semantics of ASPIC+ frameworks (Snaith and Reed 2012). The definitions of ASPIC+ given in this paper are abbreviated from (Prakken 2010). We will only introduce those elements of ASPIC+ that are needed for the purpose of answer set justification[1].

An ASPIC+ `argumentation theory` $AT = (AS, \mathcal{K})$ has the following components:

1. an argumentation system $AS = (\mathcal{L}, ^-, \mathcal{R})$ where:
   - $\mathcal{L}$ is a logical language;
   - $^-$ is a contrariness function from $\mathcal{L}$ to $2^{\mathcal{L}}$;
   - $\mathcal{R}$ is a set of labelled[2] rules of the form $r : l_1, ..., l_m \rightarrow l$ with $m \geq 0$;
2. a knowledge base $\mathcal{K} = \mathcal{K}_n \cup \mathcal{K}_a$ containing axioms $\mathcal{K}_n$ and assumptions $\mathcal{K}_a$, where $\mathcal{K} \subseteq \mathcal{L}$ and $\mathcal{K}_n \cap \mathcal{K}_a = \emptyset$.

We will now introduce the notion of argument, where $A, A_1, A_2, \dots$ will be used as unique argument labels. For an argument A, $Prem(A)$ denotes the set of premises used to construct A, whereas $Conc(A)$ is the conclusion of A.

An `argument` in AT has one of the following two forms:

1. $A : l$ is a `base argument` in AT if $l \in \mathcal{K}$. Then,
   - $Prem(A) = \{l\}$
   - $Conc(A) = l$
2. $A : A_1, \dots, A_m \rightarrow l$ is a `rule argument` in AT if $A_1, \dots, A_m$ are arguments in AT and $r : Conc(A_1), \dots, Conc(A_m) \rightarrow l$ is a rule in $\mathcal{R}$. Then,
   - $Prem(A) = Prem(A_1) \cup \ldots \cup Prem(A_m)$
   - $Conc(A) = l$

Note that the set $Prem(A)$ contains only assumptions and facts from $\mathcal{K}$, no other literals. This means that for a rule argument $A : A_1, \dots, A_m \rightarrow l$, where $r : Conc(A_1), \dots, Conc(A_m) \rightarrow l$ is the last rule applied in the construction, $Prem(A)$ will contain $Conc(A_i)$ ($i = 1, \dots, m$) only if A_i is a base argument, since in this case $Prem(A_i) = \{Conc(A_i)\}$. If A_i is a rule argument, $Prem(A_i)$ will be a subset of $Prem(A)$, but $Conc(A_i)$ will not be part of $Prem(A)$. See Example 1 for further clarification.

With an abuse of notation, argument labels will sometimes stand for whole arguments. For example, we may say that argument A is of the form $A : l$.

Example 1. Consider the ASPIC+ argumentation theory AT_1 with:

- $\mathcal{L} = \{a, b, c, d, e, f, g\}$
- $\mathcal{R} = \{r_1 : b, c \rightarrow a;\ \ r_2 : d \rightarrow c\}$
- $\overline{b} = \{e, g\},\ \overline{a} = \{f\}$
- $\mathcal{K}_n = \{d, e\},\ \mathcal{K}_a = \{b\}$

The following arguments can be constructed from AT:

- $A_1 : d$
  $Prem(A_1) = \{d\}$;
- $A_2 : e$
  $Prem(A_2) = \{e\}$;
- $A_3 : b$
  $Prem(A_3) = \{b\}$;
- $A_4 : A_1 \rightarrow c$
  $Prem(A_4) = \{d\}$;
- $A_5 : A_3, A_4 \rightarrow a$
  $Prem(A_5) = \{b, d\}$.

Once arguments are defined in an argumentation theory, attacks and defeats between them can be investigated. We will only consider undermining attacks here, as these are the only kinds of attacks occurring in ASPIC+ argumentation theories constructed from a logic program. Since we omit some elements of an ASPIC+ argumentation theory as compared to the original definition (Prakken 2010), every

[1]The elements of ASPIC+ defined in (Prakken 2010) but omitted here ($\mathcal{R}_d, \leq, \leq', \mathcal{K}_p, \mathcal{K}_i, \preceq$) can be thought of as being empty.

[2]Originally, rules in ASPIC+ (Prakken 2010) do not have labels.

undermining attack succeeds as a defeat[3]. Hence, we will introduce the notion of defeat in terms of the definition of undermining attack.

- An argument A `defeats` an argument B iff $l \in Prem(B) \backslash \mathcal{K}_n$ and $Conc(A) \in \bar{l}$.
- A set of arguments T_1 defeats a set of arguments T_2 iff there are arguments $A \in T_1$ and $B \in T_2$ such that A defeats B.

The semantics of argumentation frameworks are given in terms of extensions, that is sets of arguments. We will here focus on stable extensions, introduced in (Dung 1995) and adapted to ASPIC+ arguments in (Prakken 2010).

- A set T of arguments is `conflict-free` if it does not contain arguments A and B such that A defeats B.
- A set of arguments T is a `stable extension` iff T is conflict-free and it defeats each argument which does not belong to T, or equivalently iff $T = \{A \mid T \textit{ does not defeat } A\}$.

Example 2 (Example 1 cont.)**.** In AT_1, A_2 defeats both A_3 and A_5. This means, that for example the set of arguments $\{A_1, A_2, A_3\}$ is not conflict-free, whereas $\{A_3, A_4\}$ is a conflict-free set of arguments. There is only one stable extension for AT_1, which is $\{A_1, A_2, A_4\}$.

3 Justifying Answer Sets

In order to use ASPIC+ for the justification of literals with respect to an answer set, the respective logic program has to be translated into an ASPIC+ argumentation theory.

Definition 1. Let $\mathcal{P}$ be a consistent extended logic program. Then $AT_\mathcal{P}$ is the `ASPIC+ argumentation theory derived from` $\mathcal{P}$ as follows:

1. $\mathcal{R} = \{r : l_1, \dots l_m, not\_l_{m+1}, \dots, not\_l_{m+n} \rightarrow l \mid r : l \leftarrow l_1, \dots, l_m, not\ l_{m+1}, \dots, not\ l_{m+n} \in \mathcal{P}\}$;
2. $\mathcal{K}_n = \{l \mid r : l \leftarrow\ \in \mathcal{P}\}$;
3. $\mathcal{K}_a = \{not\_l \mid not\ l \in body^-(r), r \in \mathcal{P}\}$;
4. $\overline{not\_l} = \{l\}$ *for all* $not\_l \in \mathcal{K}_a$;
5. $\mathcal{L} = Lit_\mathcal{P} \cup \mathcal{K}_a$.

Note that $AT_\mathcal{P}$ will always be well-formed, where an ASPIC+ argumentation theory is well-formed iff: if ϕ is a contrary of ψ then $\psi \notin \mathcal{K}_n$ and ψ is not the consequent of a rule in $\mathcal{R}$ (Prakken 2010). The reason that $AT_\mathcal{P}$ is well-formed is that the only contraries are of the form $\overline{not\_l} = \{l\}$, and the respective NAF literals $not\ l$ cannot form the head of rules in $\mathcal{P}$.

Example 3. Consider the following consistent extended logic program $\mathcal{P}_{fly}$ about a wounded bird:
$r_1 :\ fly \leftarrow bird,\ not\ abnormalBird$
$r_2 :\ abnormalBird \leftarrow bird,\ wounded$
$r_3 :\ \neg fly \leftarrow\ wounded$
$r_4 :\ wounded \leftarrow$
$r_5 :\ bird \leftarrow$
The ASPIC+ argumentation theory $AT_{\mathcal{P}_{fly}}$ derived from $\mathcal{P}_{fly}$ has the following components:

- $\mathcal{R} = \{r_1 : bird,\ not\_abnormalBird \rightarrow fly;$
  $r_2 : bird, wounded \rightarrow abnormalBird;$
  $r_3 : wounded \rightarrow \neg fly\}$;
- $\mathcal{K}_n = \{bird, wounded\}$;
- $\mathcal{K}_a = \{not\_abnormalBird\}$;
- $\overline{not\_abnormalBird} = \{abnormalBird\}$;
- $\mathcal{L} = \{fly, \neg fly, bird, \neg bird, abnormalBird, \neg abnormalBird,$
  $wounded, \neg wounded, not\_abnormalBird\}$.

In $AT_{\mathcal{P}_{fly}}$, the following ASPIC+ arguments can be constructed:

- $A_1 : not\_abnormalBird$
  $Prem(A_1) = \{not\_abnormalBird\}$;
- $A_2 : bird$
  $Prem(A_2) = \{bird\}$;
- $A_3 : wounded$
  $Prem(A_3) = \{wounded\}$;
- $A_4 : A_3 \rightarrow \neg fly$
  $Prem(A_4) = \{wounded\}$;
- $A_5 : A_2, A_3 \rightarrow abnormalBird$
  $Prem(A_5) = \{bird, wounded\}$;
- $A_6 : A_2, A_1 \rightarrow fly$
  $Prem(A_6) = \{bird, not\_abnormalBird\}$.

The only defeats in $AT_{\mathcal{P}_{fly}}$ are A_5 defeating both A_1 and A_6.

The following proposition is the basis for justifying literals with respect to an answer set using ASPIC+. It states that every answer set has a corresponding stable extension, including a corresponding argument for each literal in the answer set.

Proposition 1. Let $\mathcal{P}$ be a consistent extended logic program. S is an answer set of $\mathcal{P}$ iff $\mathcal{E}$ is a stable extensions of $AT_\mathcal{P}$ such that $S = \{Conc(A) \mid A \in \mathcal{E}, Conc(A) \notin \mathcal{K}_a\}$.

Proof. Let S be an answer set of $\mathcal{P}$, let $\Delta_S = \{not\ l \mid l \in Lit_\mathcal{P}, l \notin S\}$ and $\Delta'_S = \{not\_l \mid l \in Lit_\mathcal{P}, l \notin S\}$. Let $\mathcal{P}' = \{r : l \leftarrow l_1, \dots l_m, not\_l_{m+1}, \dots, not\_l_{m+n} \mid r : l \leftarrow l_1, \dots, l_m, not\ l_{m+1}, \dots, not\ l_{m+n} \in \mathcal{P}\}$. Let $\vdash$ denote derivability using modus ponens for $\leftarrow$ as the only inference rule.

$$\begin{aligned}
S &= \{l \in Lit_\mathcal{P} \mid \mathcal{P}^S \vdash l\} \\
&= \{l \in Lit_\mathcal{P} \mid \mathcal{P} \cup \Delta_S \vdash l\} \\
&= \{l \in Lit_\mathcal{P} \mid \mathcal{P}' \cup \Delta'_S \vdash l\} \\
&= \{l \in \mathcal{L} \mid A \textit{ is an argument in } AT_\mathcal{P} \textit{ with } Conc(A) = l, \\
&\qquad Conc(A) \notin \mathcal{K}_a, Prem(A) \subseteq \Delta'_S \cup \mathcal{K}_n\} \\
&= \{Conc(A) \mid A \textit{ is an argument in } AT_\mathcal{P} \textit{ with} \\
&\qquad Conc(A) \notin \mathcal{K}_a, Prem(A) \subseteq \Delta'_S \cup \mathcal{K}_n\} \\
&= \{Conc(A) \mid A \textit{ is an argument in } AT_\mathcal{P} \textit{ with} \\
&\qquad Conc(A) \notin \mathcal{K}_a \textit{ and no argument } B \textit{ in } AT_\mathcal{P} \textit{ with} \\
&\qquad Conc(B) \in S \textit{ defeats } A\} \\
&= \{Conc(A) \mid A \in \mathcal{E}, Conc(A) \notin \mathcal{K}_a\}
\end{aligned}$$

[3]Since $\mathcal{K} = \mathcal{K}_n \cup\ \mathcal{K}_a$, $Prem(A) \backslash \mathcal{K}_n \subseteq \mathcal{K}_a$ for any A and therefore any undermining attack is a contrary-undermining attack and consequently a defeat.

Example 4 (Example 3 cont.)**.** The answer set of $\mathcal{P}_{\mathit{fly}}$ is $S_{\mathcal{P}_{\mathit{fly}}} = \{bird, wounded, abnormalBird, \neg\mathit{fly}\}$. The ASPIC+ argumentation theory $AT_{\mathcal{P}_{\mathit{fly}}}$ has one stable extension: $\{A_2, A_3, A_4, A_5\}$. As expected, the set of conclusions of arguments in the stable extension is exactly the same as the answer set.

Notation (**Corresponding extension and argument**)**.** Given an answer set S of $\mathcal{P}$ and a stable extension $\mathcal{E}$ of $AT_{\mathcal{P}}$ such that $S = \{Conc(A) \mid A \in \mathcal{E}, Conc(A) \notin \mathcal{K}_a\}$, $\mathcal{E}$ will be called the `corresponding stable extension` of S. Given a literal $l \in S$ and an argument $A \in \mathcal{E}$ with $Conc(A) = l$, A will be called a `corresponding argument` of l.

It is easy to see that for every literal in an answer set there is a corresponding argument in the corresponding stable extension. Note that there might be more than one corresponding argument for a literal.

Example 5. To illustrate the concept of corresponding argument, consider the following logic program $\mathcal{P}_2$:
$r_1 : a \leftarrow \mathit{not}\ d$
$r_2 : a \leftarrow c$
$r_3 : c \leftarrow$
$r_4 : c \leftarrow \mathit{not}\ a$
The answer set is $S_{\mathcal{P}_2} = \{a, c\}$. In the ASPIC+ argumentation theory $AT_{\mathcal{P}_2}$, seven arguments can be constructed:

- $A_1 : \mathit{not\_d}$
  $Prem(A_1) = \{\mathit{not\_d}\}$;
- $A_2 : \mathit{not\_a}$
  $Prem(A_2) = \{\mathit{not\_a}\}$;
- $A_3 : c$
  $Prem(A_3) = \{c\}$;
- $A_4 : A_1 \rightarrow a$
  $Prem(A_4) = \{\mathit{not\_d}\}$;
- $A_5 : A_2 \rightarrow c$
  $Prem(A_5) = \{\mathit{not\_a}\}$;
- $A_6 : A_3 \rightarrow a$
  $Prem(A_6) = \{c\}$;
- $A_7 : A_5 \rightarrow a$
  $Prem(A_7) = \{\mathit{not\_a}\}$.

Arguments A_4, A_6 and A_7 all attack arguments A_2, A_5 and A_7 (this means that A_7 also attacks itself). The stable extension is $\mathcal{E}_{\mathcal{P}_2} = \{A_1, A_3, A_4, A_6\}$. The translated ASPIC+ argumentation theory $AT_{\mathcal{P}_2}$ contains two arguments with conclusion c, but only one of them is a corresponding argument, namely A_3. For literal a there are three arguments in $AT_{\mathcal{P}_2}$, two of which are corresponding arguments, namely A_4 and A_6.

After introducing the link between ASP and ASPIC+, we will now move on to the justification of literals with respect to an answer set. The `Argumentation-Based Answer Set Justification` of why a literal l is or is not part of a chosen answer set is a set which can also be interpreted as a graph. It contains l itself as well as literals which are responsible for l being (or not being) part of the answer set. Furthermore, the justification comprises binary relations expressing defeats or supports between literals. All literals in a justification can be interpreted as nodes in a graph, where binary relations are directed edges between these nodes.

Notation (**Defeat and support relation**)**.** Let $\mathit{Defeat}(A)$ denote the set of all arguments defeating argument A in a given ASPIC+ argumentation theory. $\mathit{def\_rel}(l_1, l_2)$ is a relation expressing that (an argument with conclusion) l_1 defeats (an argument with conclusion) l_2. Similarly, $\mathit{supp\_rel}(l_1, l_2)$ states that l_1 is a premise of (an argument with conclusion) l_2, in other words l_1 supports (an argument with conclusion) l_2.

Definition 2 (**Justification of answer set literals**)**.** Let $\mathcal{P}$ be a consistent extended logic program, S an answer set of $\mathcal{P}$ and $\mathcal{E}$ the corresponding stable extension of S. Let $l \in S$ and let A be a corresponding argument of l. The `Argumentation-Based Answer Set Justification` of l is a set $just(l, A)$, recursively constructed as follows:

(a) If A is a base argument $A : l$, then
$just(l, A) = \{l\} \cup \bigcup_{B \in \mathit{Defeat}(A)} \{\mathit{def\_rel}(Conc(B), l)\} \cup just(Conc(B), B)$.

(b) If A is a rule argument $A : A_1, \ldots, A_n \rightarrow l$, then
$just(l, A) = \{l\} \cup \bigcup_{\substack{B \text{ base argument}, \\ Conc(B) \in Prem(A)}} \{\mathit{supp\_rel}(Conc(B), l)\} \cup just(Conc(B), B)$.

Note that the base case of this recursive definition occurs if a corresponding argument A is not defeated in condition (a), i.e. if $Defeat(A) = \emptyset$.

Example 6 (Example 4 cont.)**.** The justification of $bird \in S_{\mathcal{P}_{\mathit{fly}}}$ is simple as its corresponding argument A_2 is a base argument, which is not defeated by other arguments. Applying (a) of Definition 2: $just(bird, A_2) = \{bird\}$. This expresses that the literal $bird$ is supported by itself and not "defeated" by other literals.
The justification of a literal whose corresponding argument is not a base argument, like $\neg\mathit{fly} \in S_{\mathcal{P}_{\mathit{fly}}}$, is obtained as follows:
$just(\neg\mathit{fly}, A_4)$
$\quad = \{\neg\mathit{fly}\} \cup \{\{\mathit{supp\_rel}(wounded, \neg\mathit{fly})\}$
$\qquad \cup\ just(wounded, A_3)\}$
$\quad = \{\neg\mathit{fly}, \mathit{supp\_rel}(wounded, \neg\mathit{fly}), wounded\}$
This justification uses both parts of Definition 2 and expresses that $\neg\mathit{fly}$ is in the answer set because it is supported by the fact $wounded$, which is not defeated. The graph corresponding to the justification set is shown in Figure 1.

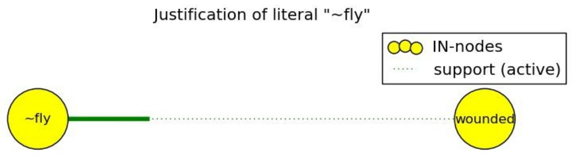

Figure 1: Graphical justification of literal $\neg\mathit{fly}$ (depicted as ~*fly*), corresponding to the justification set in Example 6.

An Argumentation-Based Answer Set Justification only comprises the "lowest" or "base" literals which are needed to derive the literal in question, that is NAF literals and heads of clauses with empty body ("facts"). In terms of argumentation this means that only ASPIC+ premises of arguments are considered; conclusions derived in intermediate steps are omitted. We argue that including all intermediately derived literals in the justification would cause confusion and lack of clarity, especially in the case of large logic programs. However, Argumentation-Based Answer Set Justification could easily be extended to incorporate intermediate supporting literals by adding the conclusions of all subarguments of the corresponding argument to the justification set.

The justification of a literal l which is not contained in an answer set can have two different forms, depending on whether or not there is an argument with conclusion l in $AT_{\mathcal{P}}$. If such an argument exists, the Argumentation-Based Answer Set Justification is similar to the justification of a literal contained in the answer set. If there is no argument with conclusion l in $AT_{\mathcal{P}}$, the justification states why it is impossible to construct such an argument or rather why l cannot be derived from the logic program.

Notation (Unsatisfied body literals). For a clause $r \in \mathcal{P}$, $unsat(r) = body^+(r) \backslash \{l \mid \mathcal{P} \vdash l\}$[4].

Informally, $unsat(r)$ contains all body literals which are responsible for r not being applicable, i.e. all literals which cannot be derived from the logic program.

Definition 3 (Justification of non answer set literals). Let $\mathcal{P}$ be a consistent extended logic program and $AT_{\mathcal{P}}$ the ASPIC+ argumentation theory derived from $\mathcal{P}$. Let S be an answer set of $\mathcal{P}$, $\mathcal{E}$ the corresponding stable extension of S and $l \notin S$. The `Argumentation-Based Answer Set Justification` of l is a set of sets, denoted $just(l)$, constructed as follows:
Let $A_1, \ldots, A_n$ be all arguments with conclusion l in $AT_{\mathcal{P}}$.

(a) If $n > 0$ then
$just(l) = \{just(l, A_1), \ldots, just(l, A_n)\}$[5].

(b) If $n = 0$ then let $r_1, \ldots, r_m$ be all clauses in $\mathcal{P}$ with head l.
   (i) If $m > 0$ then
   $just(l) = \{just(l, r_1), \ldots, just(l, r_m)\}$, where
   $just(l, r_i) = \{l\} \cup \bigcup_{h \in unsat(r_i)} \{h, responsible(h, l)\}$.
   (ii) If $m = 0$ then
   $just(l) = \{\{l, \bot, responsible(\bot, l)\}\}$.

$responsible(l_1, l_2)$ expresses that l_1 is one of the literals responsible that l_2 cannot be derived. $l_1 = \bot$ denotes a non-existing body literal. Obviously, there are no clauses with non-existing body-literals in a logic program, so that case (b)(ii) in Definition 3 simply expresses that there is no clause with head l_2 in $\mathcal{P}$.

Example 7. To illustrate the justification of a literal which is not part of an answer set because there is no argument having this literal as its conclusion, consider the following logic program $\mathcal{P}_3$:
$r_1 : a \leftarrow not\ b$
$r_2 : b \leftarrow c, not\ d$
This logic program has one answer set, $S_{\mathcal{P}_3} = \{a\}$. In the ASPIC+ argumentation theory $AT_{\mathcal{P}_3}$, there are no arguments with conclusion b, c or d, so in order to justify these literals, (b) in Definition 3 has to be used. Literal b has one defining clause, namely r_2, so condition (b)(i) is applied: $unsat(r_2) = \{c\}$ since $c \in body^+(r_2)$ and $c \notin \mathcal{K}_n \cup \{l \mid \mathcal{P} \vdash l\}$. Then,
$just(b) = \{just(b, r_2)\} = \{\{b\} \cup \{c, responsible(c, b)\}\}$
$= \{\{b, c, responsible(c, b)\}\}$.
This justification expresses that c is responsible that literal b cannot be derived. $\mathcal{P}_3$ does not contain a defining clause for either c or d, so that for these two literals condition (b)(ii) is applied: $just(c) = \{\{c, \bot, responsible(\bot, c)\}\}$ and $just(d) = \{\{d, \bot, responsible(\bot, d)\}\}$. These two justifications express that c and d cannot be derived because they do not have a defining clause in the logic program.

Example 8 (Example 4 cont.)**.** The justification of *fly*, which is not part of the answer set of $\mathcal{P}_{fly}$, is shown in Figure 2. Since there is only one argument with conclusion *fly* in $AT_{\mathcal{P}_{fly}}$, the justification is a set containing only one set:

$$\begin{aligned}
&just(fly) = \{just(fly, A_6)\}\\
&= \{\{fly\} \cup \{supp\_rel(bird, fly)\} \cup just(bird, A_2)\\
&\quad \cup \{supp\_rel(not\_abnormalBird, fly)\}\\
&\quad \cup just(not\_abnormalBird, A_1)\}\\
&= \{\{fly, supp\_rel(bird, fly),\\
&\quad supp\_rel(not\_abnormalBird, fly)\} \cup \{bird\}\\
&\quad \cup \{not\_abnormalBird\}\\
&\quad \cup \{def\_rel(abnormalBird, not\_abnormalBird)\}\\
&\quad \cup just(abnormalBird, A_5)\}\\
&= \{\{fly, supp\_rel(bird, fly),\\
&\quad supp\_rel(not\_abnormalBird, fly), bird,\\
&\quad not\_abnormalBird,\\
&\quad def\_rel(abnormalBird, not\_abnormalBird)\}\\
&\quad \cup \{abnormalBird\}\\
&\quad \cup \{supp\_rel(bird, abnormalBird)\}\\
&\quad \cup just(bird, A_2)\\
&\quad \cup \{supp\_rel(wounded, abnormalBird)\}\\
&\quad \cup just(wounded, A_3)\}\\
&= \{\{fly, supp\_rel(bird, fly),\\
&\quad supp\_rel(not\_abnormalBird, fly),\\
&\quad bird, not\_abnormalBird,\\
&\quad def\_rel(abnormalBird, not\_abnormalBird),\\
&\quad abnormalBird, supp\_rel(bird, abnormalBird),\\
&\quad supp\_rel(wounded, abnormalBird)\} \cup \{wounded\}\}\\
&= \{\{fly, supp\_rel(bird, fly),\\
&\quad supp\_rel(not\_abnormalBird, fly),\\
&\quad bird, not\_abnormalBird,\\
&\quad def\_rel(abnormalBird, not\_abnormalBird),\\
&\quad abnormalBird, supp\_rel(bird, abnormalBird),\\
&\quad supp\_rel(wounded, abnormalBird), wounded\}\}
\end{aligned}$$

This justification expresses that *fly* is not contained in the answer set because it is supported by $not\_abnormalBird$, which is defeated by $abnormalBird$. Since the supporting literals of $abnormalBird$, namely $bird$ and $wounded$, are

[4] $\vdash$ denotes modus ponens as in the proof of Proposition 1.
[5] $just(l, A_i)$ refers to Definition 2.

not in conflict with any other literals, $abnormalBird$ is part of the answer set and is able to defeat $not\_abnormalBird$. Consequently, $not\_abnormalBird$ is not contained in the answer set and hence cannot support *fly*.

4 Implementation

As mentioned earlier, an Argumentation-Based Answer Set Justification can be interpreted as a graph. We developed an implementation called JASA (Justification of Answer Sets via Argumentation) which computes a justification set for an answer set literal according to our theory and constructs a graph from it.

The input for JASA is a consistent extended logic program in clingo-format (Gebser et al. 2008). We chose this format since JASA makes use of the answer set solver clingo (Gebser et al. 2011) to compute answer sets. JASA translates a given logic program into the corresponding ASPIC+ argumentation framework (see Definition 1) and utilises the TOAST API (Snaith and Reed 2012) for the construction of ASPIC+ arguments and the computation of attacks between them. With the help of these arguments and attacks, JASA computes the Argumentation-Based Answer Set Justification in the form of a set (Definition 2 and Definition 3) and creates a graph from it. Example graphs constructed by JASA are shown in Figures 1 and 2.

The direction of edges is indicated by the thickening of the lines at one end. Dotted lines stand for supporting relations, whereas solid lines represent attacks. Nodes for literals which are part of the justified answer set are illustrated in green (light grey)[6] and nodes for literals which are not part of the answer set in red (dark grey). Similarly, edges emerging from green nodes are green (light grey), representing active defeats or supports, whereas edges emerging from red nodes are red (dark grey), meaning that they are inactive.

5 Related Work

Answer Set Justification is closely related to answer set debugging. However, debugging strategies trace the computation of answer sets in order to obtain an explanation, as for example done in (El-Khatib, Pontelli, and Son 2005). In contrast, we present a justification method which is independent of any computation strategy.

To the best of our knowledge, there is only one other justification approach for answer sets which does not trace the computation, namely "off-line justification" (Pontelli, Son, and Elkhatib 2009). Off-line justification constructs the explanation of a literal with respect to an answer set in the form of a graph. In Argumentation-Based Answer Set Justification, in contrast, the justification of a literal is given as a set. This has the advantage that the justification can be easily represented in other forms, as for example done by our implementation JASA which constructs a graph from the justification set.

The strategy used in off-line justification is methodologically completely different from Argumentation-Based Answer Set Justification in that it makes use of the well-founded model semantics for logic programs.

Comparing off-line justification graphs to graphs constructed by JASA, we find similarities as well as differences. Both off-line justification and JASA graphs indicate whether or not a literal is contained in the answer set (using plus/minus signs and green/red colours respectively). The main difference between the graphs lies in the literals which are depicted. Off-line justification graphs only comprise classical literals and omit NAF literals. NAF literals like $not\ l$ are indirectly represented by expressing that l cannot be part of the answer set in the case that $not\ l$ is in the body of a clause that was used in the derivation of the literal in question. In contrast, JASA graphs (as well as Argumentation-Based Answer Set Justification sets) explicitly contain NAF literals, which makes the graphs easier to understand for people who are not familiar with the concept of NAF literals. Another difference between the two justification approaches is that an off-line justification graph contains all intermediate literals between the literal in question and the base literals necessary to derive this literal. In Argumentation-Based Answer Set Justification, we omit all intermediate literals to enhance clarity and to emphasise the fundamental reasons why a literal is (or is not) part of an answer set.

6 Conclusion and Future Work

We present a method for explaining why a literal is or is not contained in an answer set by translating a logic program into an ASPIC+ argumentation theory. This is accompanied by an implementation called JASA (Justification of Answer Sets via Argumentation), which constructs justifications as graphs.

Argumentation-Based Answer Set Justification is based on the correspondence between answer sets in logic programming and stable extension in argumentation theory. More precisely, every literal in an answer set has a corresponding ASPIC+ argument in the corresponding stable extension built from the same logic program. The Argumentation-Based Answer Set Justification of a literal comprises all base literals, i.e. facts and NAF literals, which are necessary to deduce this literal. The justification also includes conflicts between literals which cannot occur in an answer set together.

Our implementation JASA translates a given logic program into an ASPIC+ argumentation theory and lets the user choose an literal he wants to justify. JASA then constructs a justification according to the theory of Argumentation-Based Answer Set Justification and creates a graph from it. The constructed graph includes the colours green and red to indicate whether or not a literal is part of an answer set, which makes it easy to understand.

So far, JASA does not incorporate the graphical justification for literals which are not contained in the answer set because no corresponding argument can be constructed (case (b) in Definition 3). The implementation of this case will be part of future work.

A current focus of future work are logic programs with preferences. Various semantics for logic programs with

[6]For the purpose of this paper, nodes which are usually green are depicted in yellow in order to make them easily distinguishable from red nodes when printing in black and white.

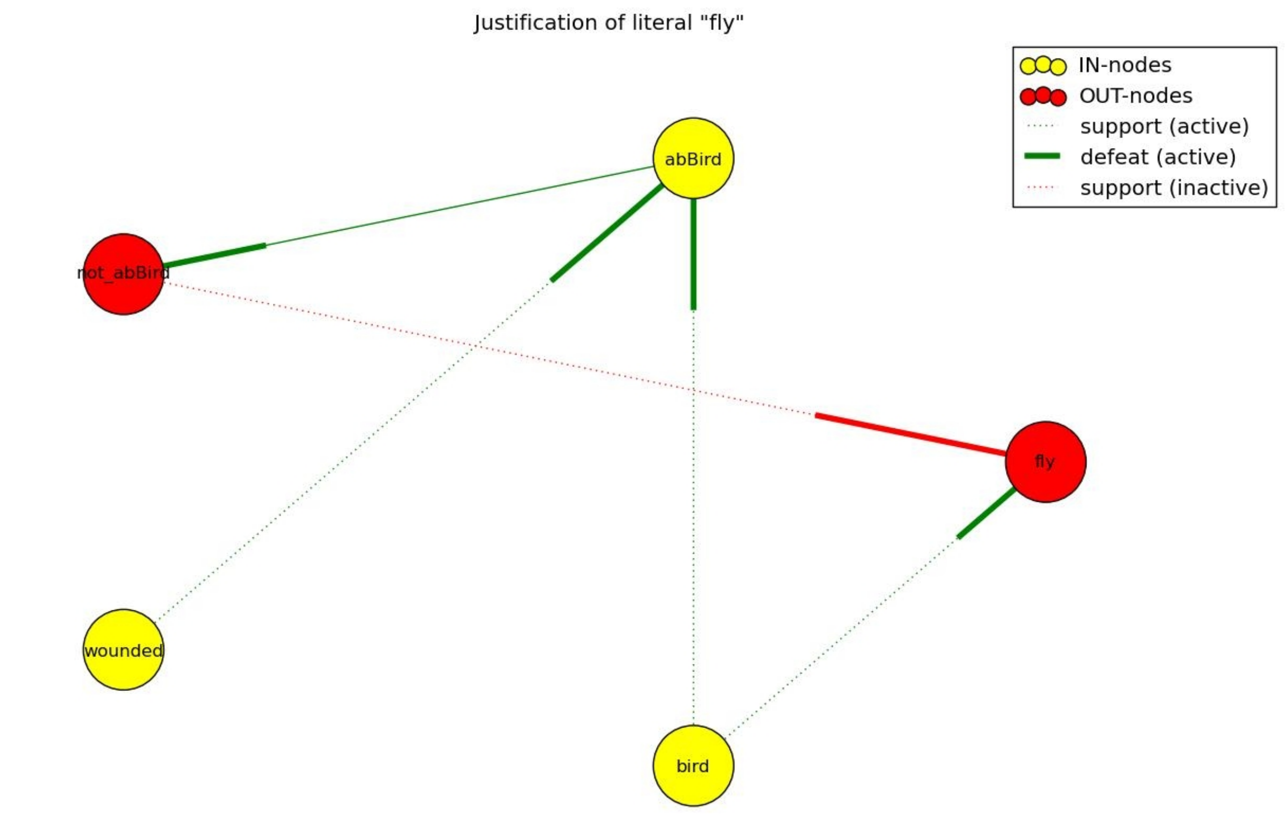

Figure 2: Graphical justification of literal *fly*

preferences have been suggested, for example (Wang, Zhou, and Lin 2000), (Brewka and Eiter 1999) and (Delgrande, Schaub, and Tompits 2000). It will be interesting to investigate how to justify literals according to these different semantics and compare the resulting justifications. Our interest in preferences was one reason for choosing ASPIC+ rather than another argumentation framework for Argumentation-Based Answer Set Justification. ASPIC+ supports preferences between rules, literals and arguments, which could be useful for the justification of logic programs with preferences. The approach of (Delgrande, Schaub, and Tompits 2000), which translates logic programs with preferences into logic programs without preferences, seems especially promising with respect to answer set justification. It could be possible to apply Argumentation-Based Answer Set Justification to the answer sets of the translated logic program without preferences and thereby yield a meaningful justification for the original logic program with preferences. However, such translated logic programs contain a large number of new clauses, making them rather complex. Currently, the ASPIC+ tool TOAST is not able to deal with such complex argumentation theories, meaning that JASA cannot be used for the construction of an Argumentation-Based Answer Set Justification.

Future work includes the use of other argumentation frameworks, for example Assumption-Based Argumentation (ABA) (Bondarenko et al. 1997), to provide a more scalable system for the justification of answer sets. This could also allow for the application of the translation approach for logic programs with preferences (Delgrande, Schaub, and Tompits 2000). It will also be interesting to investigate the relation with Abductive Logic Programming and to see if Argumentation-Based Answer Set Justifications and abductive explanations are connected. Since ABA is a generalisation of Abductive Logic Programming (Toni 1995), we believe that using ABA will pave the way in understanding links between justifications and abductive explanations.

References

Anger, C.; Konczak, K.; Linke, T.; and Schaub, T. 2005. A glimpse of answer set programming. *KI* 19(1):12–19.

Baral, C.; Chancellor, K.; Tran, N.; Tran, N.; Joy, A. M.; and Berens, M. E. 2004. A knowledge based approach for representing and reasoning about signaling networks. In *ISMB/ECCB (Supplement of Bioinformatics)*, 15–22.

Bihlmeyer, R.; Faber, W.; Ielpa, G.; Lio, V.; and Pfeifer, G. 2009. *DLV - User Manual*. Available at http://www.dlvsystem.com/html/DLV_User_Manual.html.

Boenn, G.; Brain, M.; Vos, M. D.; and Fitch, J. 2011. Automatic music composition using answer set programming. *TPLP* 11(2-3):397–427.

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.

Brewka, G., and Eiter, T. 1999. Preferred answer sets for extended logic programs. *Artificial Intelligence* 109(1-2):297–356.

Delgrande, J. P.; Schaub, T.; and Tompits, H. 2000. Logic programs with compiled preferences. In *ECAI*, 464–468.

Dimopoulos, Y.; Nebel, B.; and Koehler, J. 1997. Encoding planning problems in nonmonotonic logic programs. In *Proceedings of the Fourth European Conference on Planning*, 169–181. Springer-Verlag.

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

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.

El-Khatib, O.; Pontelli, E.; and Son, T. C. 2005. Justification and debugging of answer set programs in asp. In *Proceedings of the sixth international symposium on Automated analysis-driven debugging*, 49–58. ACM.

Garcia, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming* 4(2):95–138.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Thiele, S. 2008. *A User's Guide to gringo, clasp, clingo, and iclingo*. Available at http://freefr.dl.sourceforge.net/project/potassco/potassco_guide/2010-10-04/guide.pdf.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The Potsdam answer set solving collection. *aicom* 24(2):105–124.

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

Governatori, G.; Maher, M. J.; Antoniou, G.; and Billington, D. 2004. Argumentation semantics for defeasible logic. *Journal of Logic and Computation* 14:675–702.

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.

Snaith, M., and Reed, C. 2012. Toast: online aspic+ implementation. In *COMMA*, COMMA 2012, 509–510.

Son, T. C.; Pontelli, E.; and Sakama, C. 2009. Logic programming for multiagent planning with negotiation. In *Proceedings of the 25th International Conference on Logic Programming*, 99–114. Springer-Verlag.

Toni, F. 1995. A semantics for the kakas-mancarella procedure for abductive logic programming. In *GULP-PRODE*, 231–244.

Wang, K.; Zhou, L.; and Lin, F. 2000. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the First International Conference on Computational Logic*, CL '00, 164–178. Springer-Verlag.

Watson, R., and Vos, M. 2011. Astrea: Answer sets for a trusted reasoning environment for agents. In Balduccini, M., and Son, T., eds., *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 490–509.