Argumentation-Based Explanations for Answer Sets Using ADF

Lena Rolf[1], Gabriele Kern-Isberner[1], and Gerhard Brewka[2(✉)]

[1] TU Dortmund, Dortmund, Germany
[2] Leipzig University, Leipzig, Germany
brewka@informatik.uni-leipzig.de

Abstract. This paper presents so-called *asl-explanation graphs* for answer set programming based on a translation of extended logic programs to abstract dialectical frameworks (ADF). The graphs show how a literal can be derived from the program, and they evaluate in an argumentative way why necessary assumptions about literals not contained in an answer set hold. With the set of all asl-explanation graphs for a literal and an answer set, it is possible to explain and justify thoroughly why the literal is or is not contained in that answer set. Additionally, we provide a criterion to improve the clarity of explanations by pruning nodes without loss of information and selecting most significant asl-explanation graphs.

Keywords: Answer set programming · Abstract dialectical frameworks · Argumentation · Explanation

1 Introduction

Explainable AI is a highly relevant topic of current research, see e.g. the DARPA XAI initiative (www.darpa.mil/program/explainable-artificial-intelligence). In this paper we focus on explanations in answer set programming (ASP). ASP has been applied to various problems of academic research and industry (cf. [4]). An example for the use in industry is the generation of teams of employees for the seaport of Gioia Tauro [7]. The utilization of ASP for decision support is examined for a lot of additional fields, e.g. physician-advisory systems [3] or logistics [13], in which ASP-based systems support users that are not familiar with logic programming. Because logic programs are nonmonotonic, it is difficult to retrace the results of a solver, i.e. even ASP-knowledgeable persons can sometimes hardly reconstruct why a literal is contained in an answer set. To improve a user's acceptance of suggestions by ASP-based decision support systems, it is helpful to explain why the suggested decision is chosen and alternative solutions, possibly expected by the user, are not.

This research has been supported by DFG (Research Unit 1513 and BR 1817/7-2).

M. Balduccini et al. (Eds.): LPNMR 2019, LNAI 11481, pp. 89–102, 2019.
https://doi.org/10.1007/978-3-030-20528-7_8

To provide helpful explanations, in this paper we focus on answering the question why a literal is or is not in a given answer set with abstract dialectical frameworks (ADFs), a generalization of Dung argumentation frameworks that allows for flexible modelling (cf. [1]). We develop a translation from extended logic programs with constraints to ADFs and show that there is a 1-to-1 relation between answer sets and stable models of the ADF. The transformation is used to construct argumentative *answer set literal-explanation (asl-explanation) graphs* based on the characterization of stable models by Sylwia Polberg in [10]. Every asl-explanation graph for a literal in an answer set contains a possible derivation of the literal based on the program and an explanation for why this derivation is not restrained. Moreover, for a literal not present in an answer set, asl-explanation graphs reveal why its derivation is inhibited. The set of all asl-explanation graphs can thus be used to explain why a literal is (not) contained in the given answer set. Additionally, we propose a criterion based on specificity to reduce the size and number of explanations. The main contributions of this paper which is based on [12] are:

- Translation from extended logic programs with constraints to ADF
- Definition of asl-explanation graphs based on positive dependency evaluations [10]
- Construction of asl-explanations from asl-explanation graphs

The rest of the paper is organized as follows: Sects. 2 and 3 contain background information on ASP and ADFs and an overview over related work. The translation from logic programs to ADFs and the relation between answer sets and stable models are described in Sect. 4. Based on that, the construction of asl-explanations and the reducing criterion are described in Sect. 5. Section 6 concludes and points out future work.

2 Preliminaries on ASP and ADF

Answer Set Programming. A *literal* L is an atom A or a strictly-negated atom $\neg A$ where A is an atomic formula of propositional logic. An *extended logic program* P is a set of rules of the form $H \leftarrow B_1, \ldots, B_n, not\, B_{n+1}, \ldots, not\, B_{n+m}.$ with $m, n \geq 0$ where $H, \ldots, B_1, \ldots, B_{n+m}$ are literals and $B_{n+1}, \ldots, B_{n+m}$ *Negation-as-Failure (NAF)* literals. $Lit(P)$ denotes the set of all literals in P. For a rule r, the set $\{H\}$ is denoted by $head(r)$, the sets $\{B_1, \ldots, B_n\}$ resp. $\{B_{n+1}, \ldots, B_{n+m}\}$ are denoted with $body^+(r)$ resp. $body^-(r)$. Rules with $n+m=0$ are called *facts* and denoted with H for short, rules with $head(r) = \emptyset$ are called *constraints*. For the remainder of this paper, id is a bijective mapping that maps a rule of P to an identifier. If not stated otherwise, the function maps every rule $r \in P$ to an identifier of $\{r_1, \ldots, r_{|P|}\}$ according to its appearance in P. As defined in [6], the *reduct* of P w.r.t. $S \subseteq Lit(P)$ is obtained from P by (1) deleting every rule $H \leftarrow B_1, \ldots, B_n, not\ B_{n+1}, \ldots,\ not\ B_{n+m}. \in P$ with $\{B_{n+1}, \ldots, B_{n+m}\} \cap S \neq \emptyset$ and (2) deleting all NAF literals in the remaining rules. A set $S \subseteq Lit(P)$ is an *answer set* of P iff S is the smallest subset of $Lit(P)$ that does not contain any complementary literals $A, \neg A$ so that $\{B_1, \ldots, B_n\} \subseteq S \Rightarrow \{H\} \cap S \neq \emptyset$ holds for every rule $H \leftarrow B_1, \ldots, B_n. \in P^S$

Abstract Dialectical Frameworks. According to [2], an *Abstract Dialectical Framework (ADF)* is a tuple $D = (S, L, C)$ with a set of statements S, a set of links $L \subseteq S \times S$ and a set of acceptance conditions C for statements of S. In this paper, an acceptance condition for a statement $s \in S$ is given as a propositional formula ϕ_s on the parents of s, i.e. the statements with direct link to s. For the rest of the paper, $\phi_s(S')$ for a set $S' \subseteq S$ denotes the formula that can be obtained from ϕ_s by replacing every occurrence of a statement s' with $\top$ (tautology) if $s' \in S'$ and with $\bot$ (contradiction) otherwise. As L is implicitly contained in C, it is not always specified explicitly in the following. $M \subseteq S$ is a *model* of D if $\phi_m(M) \equiv \top \Leftrightarrow m \in M$ holds. M is the *grounded model* of D if it is the least fixpoint of $\Gamma_D(A, R) = (acc(A, R), reb(A, R))$ for $A, R \subseteq S$ with

$$acc(A, R) = \{r \in S \mid \forall S', A \subseteq S' \subseteq (S \setminus R) : \phi_r(S') \equiv \top\}, \tag{1}$$

$$reb(A, R) = \{r \in S \mid \forall S', A \subseteq S' \subseteq (S \setminus R) : \phi_r(S') \equiv \bot\}. \tag{2}$$

M is a *stable model* of D if M is a model of D and the grounded model of the *reduct* D^M *of* D *w.r.t.* M with $D^M = (M, L \cap (M \times M), C^M)$ and $C^M = \{\phi_s[p/\bot : p \notin M] \mid s \in M\}$ where every occurrence of a statement $p \notin M$ in each formula ϕ_s is replaced by $\bot$.

3 Related Work

The approach described in this paper is related to the debugging of ASP programs (e.g. [8,9]) although these approaches reconstruct the computation of answer sets and aim at locating the source of unexpected behavior by a developer. Other related approaches in examining the characteristics of a logic program and understanding its results are based on graph representation of the programs. The approach in [11] also uses graphs to compute *offline-justifications* that illustrate how a literal depends on literals of the answer set for logic programs without strict negation. The Argument-Based Answer Set Justification in [14] and the different justifications in [15] use argumentation to explain why a literal is (not) in an answer set. Both papers use a translation from ASP to an ASPIC$^+$ resp. assumption-based argumentation framework and construct justifications based on the arguments of a stable extension and relations between arguments. In [15], different labels indicate whether literals are facts or NAF-literals and if they are contained in the underlying stable extension. The approaches in [11,14,15] have something in common with the approach presented in this paper as they show different relations between literals, but there are important differences: The graphs constructed in [11] mainly show recursively how a literal can be derived from the rules of a program and which literals must not be in the answer set. The papers [14] and [15] use argumentation frameworks different from ADF for the translations and provide a different relation between answer sets and stable extensions. All three papers do not consider strict negation and constraints explicitly. Similar to the asl-explanation graphs in this paper, the justifications in [14] and [15] do not consider every literal in a derivation, but they do not

give any criteria to compare justifications, or to reduce their size reasonably to improve clarity. The criterion presented in this paper is related to the notion of specificity used in *Defeasible Logic Programming* that is based on the sets of facts and defeasible rules in an argument (cf. [5]) while the specificity defined in this paper takes facts and default negated literals into account.

4 Translations and Relations Between ASP and ADFs

In this section, a translation from a logic program to a corresponding ADF is presented and it is shown that answer sets of the program correspond exactly to the stable models of the ADF. To be able to distinguish strict negation for literals in programs from negation in propositional formulas as used in ADFs, a mapping $^+$ is used that maps a literal to a corresponding statement s.t. an atom is mapped to itself and a strictly-negated atom $\neg A$ to $\overline{A}$ resp. a set of literals $\mathbf{L}$ to $\{L^+ \mid L \in \mathbf{L}\}$.

The statement set of an ADF corresponding to a logic program P contains the identifier of all rules of P, a statement for each literal in $Lit(P)$ and a statement $cmp(A)$ for every pair of complementary literals $A, \neg A \in Lit(P)$. Statements of the form $cmp(A)$ serve the purpose that no model of the ADF contains statements for complementary literals.

Definition 1 (Translation for extended logic programs). *Let P be an extended logic program and $Lit_C(P)$ resp. $Lit_R(P)$ sets of all identifiers of constraints resp. rules with non-empty rule-head. Then $ADF(P) = (S, C)$ is the* ADF corresponding to *P with:*

$$
\begin{aligned}
S =& \{L^+ \mid L \in Lit(P)\} \cup \{cmp(A) \mid A, \neg A \in Lit(P)\} \cup Lit_C(P) \cup Lit_R(P) \\
C =& \{\phi_{id(r)} = B_1^+ \wedge \cdots \wedge B_n^+ \wedge \neg B_{n+1}^+ \wedge \cdots \wedge \neg B_{n+m}^+ \mid \\
& \qquad r = H \leftarrow B_1, \ldots, B_n, not\ B_{n+1}, \ldots, not\ B_{n+m}. \in P\} \\
& \cup \{\phi_{id(c)} = \neg id(c) \wedge B_1^+ \wedge \ldots \wedge B_n^+ \wedge \neg B_{n+1}^+ \wedge \ldots \wedge \neg B_{n+m}^+ \mid \\
& \qquad c = \leftarrow B_1, \ldots, B_n, not\ B_{n+1}, \ldots, not\ B_{n+m}. \in P\} \\
& \cup \{\phi_{H^+} = \bigvee_{\{H\}=head(r), r \in P} id(r) \mid H \in Lit(P)\} \\
& \cup \{\phi_{cmp(A)} = \neg cmp(A) \wedge A \wedge \overline{A} \mid A, \neg A \in Lit(P)\}
\end{aligned}
$$

We illustrate the construction of $ADF(P)$ in the following example.

Example 1. Let P be the logic program containing exactly the following rules associated with their rule identifiers:

r_1 : $bike \leftarrow not\ hurt, not\ \neg bike.$ r_2 : $\neg bike \leftarrow far, exhausting.$
r_3 : $\neg bike \leftarrow far, not\ \neg steep, not\ ebike.$ r_4 : $\neg bike \leftarrow heat, badWeather.$
r_5 : $car \leftarrow not\ bike, not\ \neg car.$ r_6 : $\neg car \leftarrow broke, not\ children.$
r_7 : $exhausting \leftarrow not\ \neg steep.$ r_8 : $badWeather \leftarrow winter.$
r_9 : $badWeather \leftarrow rainy, not\ warm.$ r_{10} : $hurried \leftarrow not\ holidays.$
r_{11} : $winter \leftarrow not\ \neg winter.$ r_{12} : $\neg winter \leftarrow not\ winter.$
r_{13} : $\leftarrow heat, winter.$
r_{14} : $warm.$ r_{15} : $far.$ r_{16} : $ebike.$ r_{17} : $\neg steep.$ r_{18} : $heat.$ r_{19} : $broke.$ r_{20} : $rainy.$

The ADF corresponding to P is given below by the set of acceptance conditions of all statements.

$\phi_{r_1} = \neg hurt \wedge \neg\overline{bike}$ $\phi_{r_2} = far \wedge exhausting$ $\phi_{r_3} = far \wedge \neg\overline{steep} \wedge \neg ebike$
$\phi_{r_4} = heat \wedge badWeather$ $\phi_{r_5} = \neg bike \wedge \neg\overline{car}$ $\phi_{r_6} = broke \wedge \neg children$
$\phi_{r_7} = \neg\overline{steep}$ $\phi_{r_8} = winter$ $\phi_{r_9} = rainy \wedge \neg warm$
$\phi_{r_{10}} = \neg holidays$ $\phi_{r_{11}} = \neg\overline{winter}$ $\phi_{r_{12}} = \neg winter$
$\phi_{r_{13}} = \neg r_{13} \wedge heat \wedge winter$ $\phi_{r_{14}} = \cdots = \phi_{r_{20}} = \top$

$\phi_{badWeather} = r_8 \vee r_9$ $\phi_{bike} = r_1$ $\phi_{\overline{bike}} = r_2 \vee r_3 \vee r_4$ $\phi_{broke} = r_{19}$
$\phi_{car} = r_5$ $\phi_{\overline{car}} = r_6$ $\phi_{children} = \bot$ $\phi_{ebike} = r_{16}$
$\phi_{exhausting} = r_7$ $\phi_{far} = r_{15}$ $\phi_{heat} = r_{18}$ $\phi_{holidays} = \bot$
$\phi_{hurried} = r_{10}$ $\phi_{hurt} = \bot$ $\phi_{\overline{steep}} = r_{17}$ $\phi_{rainy} = r_{20}$
$\phi_{warm} = r_{14}$ $\phi_{winter} = r_{11}$ $\phi_{\overline{winter}} = r_{12}$
$\phi_{cmp(bike)} = \neg cmp(bike) \wedge bike \wedge \overline{bike}$ $\phi_{cmp(car)} = \neg cmp(car) \wedge car \wedge \overline{car}$
$\phi_{cmp(winter)} = \neg cmp(winter) \wedge winter \wedge \overline{winter}$

Obviously, for an acceptance condition $\phi_{id(r)}$ for a statement $id(r) \in Lit_R(P)$ and every set $M \subseteq Lit(P)$ it must hold that: $\phi_{id(r)}(M^+) \equiv \top \Leftrightarrow body^+(r) \subseteq M$ and $body^-(r) \cap M = \emptyset$. The acceptance condition for r_3 in Example 1, e.g., is satisfied by every set $S' \subseteq S$ with $far \in S'$ and $\{\overline{steep}, ebike\} \cap S' = \emptyset$. For a constraint c, the cyclic dependency caused by the membership of $\neg id(c)$ in the acceptance condition effects that $id(c) \in S' \Rightarrow \phi_{id(c)}(S') \equiv \bot$ for every statement set $S' \subseteq S$ thus no model M of the ADF can satisfy $body^+(c) \subseteq M$ and $body^-(c) \cap M = \emptyset$. Because of the constraint $c =\leftarrow heat, winter.$ of P in Example 1, the corresponding ADF contains a statement r_{13} with $\phi_{r_{13}} = \neg r_{13} \wedge heat \wedge winter$. For a set $S' \subseteq S$ with $\{heat, winter\} \subseteq S'$, $\phi_{r_{13}}(S') \equiv \top$ holds iff $r_{13} \notin S'$ and $\phi_{r_{13}}(S') \equiv \bot$ holds iff $r_{13} \in S'$. Similarly, no model can contain two statements $A, \overline{A}$ for complementary literals $A, \neg A$.

$ADF(P)$ has been constructed in such a way that membership of a literal in an answer set corresponds to membership of the corresponding statement in a stable model and that answer sets have a 1-to-1 relation to stable models.

Theorem 1. *Let P be an extended logic program, $ADF(P) = (S, C)$ the ADF corresponding to P and $\overline{S} \subseteq S$ the set of statements corresponding to strictly negated literals of $Lit(P)$.*

1. *M is an answer set of P iff $M' = M^+ \cup \{id(r) \in Lit_R(P) \mid body^+(r) \subseteq M,\ body^-(r) \cap M = \emptyset\}$ is a stable model of $ADF(P)$.*
2. *M' is a stable model of $ADF(P)$ iff $M = \{A \in Lit(P) \mid A \in M' \setminus \overline{S}\} \cup \{\neg A \in Lit(P) \mid \overline{A} \in M' \cap \overline{S}\}$ is an answer set of P.*

Proof sketch. (1) If M is an answer set of P, M' is a model of $ADF(P)$ by construction of the acceptance conditions. It can be shown by induction over the steps of the Γ-operator that M' is the grounded model of $ADF(P)^{M'}$. If M' is a stable model of $ADF(P)$, M' does not contain statements corresponding to complementary literals $A, \neg A \in Lit(P)$ because of the cyclic dependency in $\phi_{cmp(A)}$. It can be shown that M satisfies every rule of P^M and that, if

M were not minimal, the Γ-operator for $ADF(P)^{M'}$ would have accepted a statement $s \in S \setminus M'$. (2) Every stable model of $ADF(P)$ is a subset of $\{l^+ \mid l \in Lit(P)\} \cup \{id(r) \in Lit_R\}$, because statements of the form $cmp(A)$ or $id(c)$ for a constraint c cannot be contained in a model of $ADF(P)$ by construction. The relation thus follows from (1).

Example 2. Program P of Example 1 has one answer set $M = \{warm, far, heat, ebike, broke, rainy, \neg steep, \neg winter, bike, \neg car, hurried\}$ thus the corresponding ADF has the corresponding stable model $M' = \{warm, far, heat, ebike, broke, rainy, \overline{steep}, \overline{winter}, bike, \overline{car}, hurried, r_1, r_6, r_{10}, r_{12}, r_{14}, \ldots, r_{20}\}$. M' contains statements for literals in M and the identifiers of those rules whose bodies are satisfied by M.

5 Construction of Explanations

The construction of asl-explanations is based on the translation presented in Sect. 4 and the characterization of stable models of ADFs in [10]. Useful definitions from [10] are recalled in the following.

Definition 2 (pdf, cf. [10]). *Let $D = (S, C)$ be an ADF, $E \subseteq S$ and $\mathcal{N}$ a fixed symbol. A* positive dependency function (pdf) *on E is a function pd that maps every $s \in E$ to a tuple (A_s, R_s) so that (1) $A_s \subseteq E, R_s \subseteq S$, (2) $\phi_s(S') \equiv \top$ for all S' with $A_s \subseteq S' \subseteq S \setminus R_s$, and (3) $pd(s)$ is minimal[1] among all tuples (A, R) satisfying (1) and (2); or to the symbol $\mathcal{N}$ if no tuple (A_s, R_s) exists.*

Definition 3 (pde, [10]). *Let $D = (S, C)$ be an ADF, $E \subseteq S$ and pd a pdf on S. An* (acyclic) positive dependency evaluation (pde) *on E for a statement $e \in E$ is a tuple $\big((a_0, \ldots, a_n), B\big)$ with $a_n = e$ and (1) $(a_0, \ldots, a_n)$ is a sequence of distinct elements of E with $\forall a_i \in \{a_0, \ldots, a_n\} : pd(a_i) \neq \mathcal{N}$, (2) $A_i \subseteq \{a_0, \ldots, a_{i-1}\}$ for every $i \in \{1, \ldots, n\}$ with $pd(a_i) = (A_i, R_i)$, $A_0 = \emptyset$ for $pd(a_0) = (A_0, R_0)$, (3) $B = \bigcup_{i=0}^{n} R_i$ with (A_i, R_i) as defined in (2). $(a_0, \ldots, a_n)$ is called* sequence *and B* blocking set *of $\big((a_0, \ldots, a_n), B\big)$.*

In this paper, only acyclic pdes are used so a pde is assumed to be acyclic if not stated otherwise. A pde for a statement s can be interpreted as a part of an evaluation of the ADF in which s is accepted whenever every statement of B is rebutted, or in which operator Γ_D inserts the statements of the sequence, including s, in set (1) and the statements of B in set (2) during the recursion.

Definition 4 (Blocking, [10]). *Let $D = (S, C)$ be an ADF, $E \subseteq S$ and $\big((a_0, \ldots, a_n), B\big)$ a pde on E for a statement $e \in E$. A set $X \subseteq S$* blocks *$\big((a_0, \ldots, a_n), B\big)$ iff $\exists b \in B : b \in X$ or $\exists s \in \{a_0, \ldots, a_n\} : s \notin X$.*

Theorem 2 (cf. [10, Theorem 6.12]). *Let $D = (S, C)$ be an ADF and $E \subseteq S$ a model of D. Then E is a stable model of D iff there exists a pde on E for every $e \in E$ that is not blocked by E.*

[1] I.e., there is no other such tuple (A'_s, R'_s) with $A'_s \subseteq A_s$ and $R'_s \subseteq R_s$.

Proposition 1. *Let $D = (S, C)$ be an ADF, $\big((a_0, \ldots, a_n), B\big)$ a pde on S and $M \subseteq S$ a model of D. Then $\{a_0, \ldots, a_n\} \subseteq M$ if $M \cap B = \emptyset$.*

Proof. Let $M \cap B = \emptyset$, and assume $\{a_0, \ldots, a_n\} \not\subseteq M$. Then there is a statement $a_i \in \{a_0, \ldots, a_n\}$ with $a_i \notin M$ and $\{a_0, \ldots, a_{i-1}\} \subseteq M$. Then $\phi_{a_i}(S') \equiv \top$ holds for every $\{a_0, \ldots, a_{i-1}\} \subseteq S' \subseteq S \setminus B$ by Definition 3. Since $B \cap M = \emptyset$, $\phi_{a_i}(M) \equiv \top$ and that causes a contradiction because M is a model of D.

The characterization of stable models by Theorem 2 can be used to evaluate recursively why a pde is (not) blocked by a set of statements. Because of the relation between a logic program and the corresponding ADF in Theorem 1, the set of evaluations for all pdes for a given statement w.r.t. a stable model provides an explanation for why a literal is (not) in the corresponding answer set. Thus Proposition 1 yields the following helpful corollary.

Corollary 1. *Let P be a logic program, $ADF(P)$ its corresponding ADF, M an answer set of P and M' its corresponding stable model of $ADF(P)$. A literal $l \in Lit(P)$ is in M iff there is a pde for the corresponding statement l^+ on S with blocking set B and $M' \cap B = \emptyset$.*

A *justification tree* represents an evaluation for a statement w.r.t. a stable model by recursively checking the pdes of the statements in the blocking set.

Definition 5 (justification tree). *Let $D = (S, C)$ be an ADF, $M' \subseteq S$ a stable model of D, $s \in S$ a statement and $\mathcal{P}$ the set of pdes on S. A* justification tree *for s based on $\mathcal{P}$ w.r.t. M' is a marked tree $T = (V, E)$ such that the following conditions hold:*

- *The root node $v \in V$ is marked with*
  - *$label(v) = \big((a_0, \ldots, s), B, +\big)$ with $\big((a_0, \ldots, s), B\big) \in \mathcal{P}$ if $M' \cap B = \emptyset$ or*
  - *$label(v) = \big((a_0, \ldots, s), B, -\big)$ with $\big((a_0, \ldots, s), B\big) \in \mathcal{P}$ if $M' \cap B \neq \emptyset$.*
- *Every node $v \in V$ with $label(v) = \big((a_0, \ldots, a_n), B, +\big)$ has a child node $v' \in V$ with $label(v') = \big((a'_0, \ldots, a'_n), B', -\big)$ for every pde $\big((a'_0, \ldots, a'_n), B'\big) \in \mathcal{P}$ with $a'_n \in B$. v is a leaf iff no such child exists.*
- *Every node $v \in V$ with $label(v) = \big((a_0, \ldots, a_n), B, -\big)$ has exactly one child node $v' \in V$ for an $a'_n \in M' \cap B$ with $label(v') = \big((a'_0, \ldots, a'_n), B', +\big)$ for $\big((a'_0, \ldots, a'_n), B'\big) \in \mathcal{P}$ with $B' \cap M' = \emptyset$.*

T is positive resp. negative if the root node is marked with $+$ resp. $-$. A node $v \in V$ with $label(v) = \big((a_0, \ldots, s), B, l\big)$ is positive if $l = +$ and negative if $l = -$.

For a positive node, i.e., a pde that is not blocked by the stable model, all pdes for possibly blocking statements are evaluated to show that and why every of them is blocked by the stable model. For a negative node, i.e. a pde that is blocked by the stable model, one unblocked pde for one statement of the blocking set is evaluated. Thus there can be different justification trees for the same pde in the root node that represent different "strategies" how to block the not contained statements. Note that, by definition, every subtree of a justification tree is a justification tree.

Lemma 1. *Let $D = (S, C)$ be an ADF, $s \in S$, $\mathcal{P}$ the set of all pdes on S and M' a stable model of D.*

1. *There is a positive justification tree for s based on $\mathcal{P}$ w.r.t. M' iff $s \in M'$.*
2. *There is a negative justification tree for s based on $\mathcal{P}$ w.r.t. M' with root node v and $label(v) = \big((a_0, \ldots, s), B, -\big)$ for every pde $\big((a_0, \ldots, s), B\big) \in \mathcal{P}$ if $s \notin M'$.*

Lemma 1 states that there is a positive justification tree for a statement s iff it is contained in the stable model M' and that there are otherwise negative justification trees that show why every pde for s is blocked by M'. Although justification trees can be used to explain why a literal is (not) in an answer set in principle, they are unsuitable for practical use. The main reason for this is the large number of different pdes for one and the same statement: Definition 3 does not claim minimality of the sequence so that, for an ADF corresponding to a logic program, the set of rule identifiers in a pde need not be minimal w.r.t. set inclusion. Thus, a corresponding derivation may contain superfluous rules.

Definition 6. *Let P be a logic program, $ADF(P) = (S, C)$ its corresponding ADF and $pde = \big((a_0, \ldots, a_n), B\big)$ a pde on a set $S' \subseteq S$ for $s \in S$. pde is*

- sequence-minimal *iff there is no pde for $a_n = s$ with a sequence $(a'_0, \ldots, a'_m = a_n)$ and $\{a'_0, \ldots, a'_m\} \subset \{a_0, \ldots, a_n\}$.*
- consistent *iff (1) there is no $A \in Lit(P)$ with $A, \overline{A} \in \{a_0, \ldots, a_n\}$, and (2) there is no constraint r with $body^-(r) = \emptyset$ s.t. $body^+(r) \subseteq \{a_0, \ldots, a_n\}$.*

If pdes are interpreted as possible derivations of literals, inconsistent pdes are derivations that contain complementary literals or a set of literals that ensures that a constraint body is satisfied, i.e. they require sets of literals that cannot belong to the same answer set. Because of the construction of the acceptance conditions, an inconsistent pde can be blocked by every model of the ADF thus the evaluation of inconsistent pdes does not provide relevant information. For sake of clarity, a pde for a statement s is transformed into its *set representation* that is a tuple consisting of s, the sets of facts resp. rule identifiers in the sequence and the literals in its blocking set. Other literals which are derived by non-factual rules are omitted.

Definition 7 (set representation of a pde). *Let P be a logic program and $ADF(P) = (S, C)$ the corresponding ADF. The* set representation *of a pde $\big((a_0, \ldots, a_n), B\big)$ on S with $\{a_0, \ldots, a_n\} = Seq$ is*

$$\langle a_n, Seq \cap \{x^+ \mid x. \in P\}, B, Seq \cap \{id(r) \mid r \in P\}\rangle$$

Example 3. $p_1 = \big((r_{18}, heat, r_{11}, winter, r_8, badWeather, r_4, \overline{bike}), \{\overline{winter}\}\big)$ is a pde for the statement $\overline{bike}$ corresponding to literal $\neg bike$ of program P in Example 1. Because P contains constraint $r = \leftarrow heat, winter.$ with $body^-(r) = \emptyset$ and $\{winter, heat\} = body^+(r)$, p_1 is inconsistent. The set representation of p_1 is $\langle \overline{bike}, \{heat\}, \{\overline{winter}\}, \{r_{18}, r_{11}, r_8, r_4\}\rangle$. $\big((r_{18}, heat, r_{11}, winter, r_8, r_{20}, rainy, r_9, badWeather, r_4, \overline{bike}), \{\overline{winter}, warm\}\big)$ is not sequence-minimal. The derivation uses r_8 and r_9 with the rule head $badWeather$ so one rule is superfluous.

Definition 8 (asl-explanation graph). *Let P be a logic program, $ADF(P) = (S, C)$ the corresponding ADF, M' a stable model of $ADF(P)$, $l \in Lit(P)$, $\mathcal{P}$ the set of all consistent, sequence-minimal pdes on S and $T = (V, E)$ a positive resp. negative justification tree for l^+ based on $\mathcal{P}$ w.r.t. M' and root node w. Let f be a function that maps a node v with $label(v) = \big((a_0, \ldots, a_n), B, m\big)$ to a tuple $\big(\langle a_n, F, B, R\rangle, m\big)$ where $\langle a_n, F, B, R\rangle$ is the set representation of $\big((a_0, \ldots, a_n), B\big)$. A graph $G = (V', E')$ with $V' = \{f(v) \mid v \in V\}$ and $E' = \{(f(v), f(v')) \mid (v, v') \in E\}$ is a* positive *resp.* negative asl-explanation graph for l *w.r.t.* M'.

Analogously to justification trees, a node with $m = -$ is called *negative*, and *positive* for $m = +$. For a positive resp. negative node, the outgoing edges are called *attack edges* resp. *defense edges*. $f(w) \in V'$ is called *goal node of* G where w is the root of T. Every positive node of $V' \setminus \{f(w)\}$ is called *defender in* G.

In asl-explanation graphs, only consistent, sequence-minimal pdes are considered by construction. The use of the set representation reduces the number of nodes, because two pdes with equal blocking sets and sequences that contain exactly the same statements, in a possibly different order, are combined to one node, and provide the most important information in the context of ASP. If the sets contain many statements, the visualization of the pde becomes confusing. In that case, for an implementation and use in practice, the number of showed statements can be limited and the full visualization can be restricted. Despite of these modifications, the existence of asl-explanation graphs is guaranteed in a way that is analogous to the existence of justification trees in Lemma 1.

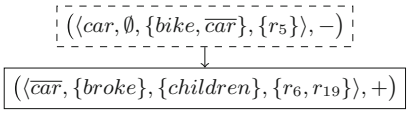

Fig. 1. asl-explanation graph G_1 for *car* (see Example 4), the goal node is surrounded by a dashed line, solid arrows represent defense edges

Example 4. Consider program P and $ADF(P)$ in Example 1 and the stable model M' of $ADF(P)$ (Example 2). Figures 1, 2 and 3 show all asl-explanation graphs G_1, G_2, G_3 for *car* w.r.t. M'. G_2 and G_3 differ only in the additional defending node for the pde for *ebike* that is only contained in G_2.

Definition 9 (asl-explanation). *Let P be a logic program, $ADF(P)$ the corresponding ADF, M an answer set of P, M' its corresponding stable model of $ADF(P)$ and $l \in Lit(P)$.*

- *If $l \in M$, every positive asl-explanation graph for l w.r.t. M' is a* positive asl-explanation for *l w.r.t. M.*
- *If $l \notin M$, the set of all asl-explanation graphs for l w.r.t. M' is the* negative asl-explanation for *l w.r.t. M.*

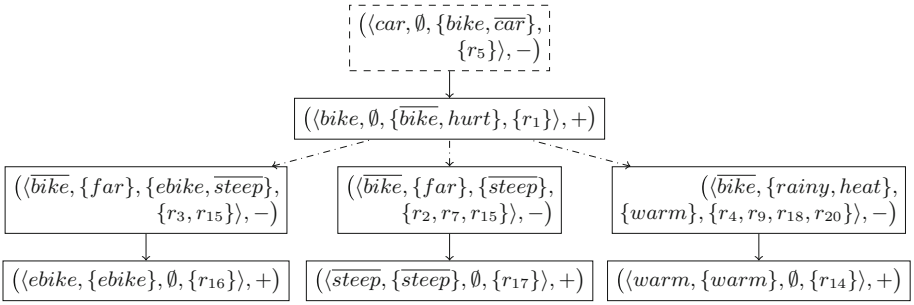

Fig. 2. asl-explanation graph G_2 for *car* (see Example 4), dash-dotted arrows represent attack edges

Example 5. The set of the asl-explanation graphs depicted in Figs. 1, 2 and 3 is a negative asl-explanation for *car* w.r.t. the answer set M in Example 2. For *bike*, there are two positive asl-explanations w.r.t. M that can be obtained from G_2 resp. G_3 in Figs. 2 and 3 by deleting the node $\big(\langle car, \emptyset, \{bike, \overline{car}\}, \{r_5\}\rangle, -\big)$ and the corresponding edge and setting $\big(\langle bike, \emptyset, \{\overline{bike}, hurt\}, \{r_1\}\rangle, +\big)$ as goal node.

An asl-explanation for a literal with respect to an answer set provides an explanation for why a literal is (not) in the given answer set. A positive asl-explanation for a literal l w.r.t. an answer set M is a positive asl-explanation graph and shows how l can be derived from facts and rules in the goal node and why required NAF-literals can be satisfied with respect to the answer set. A negative asl-explanation for a literal l w.r.t. an answer set M contains an asl-explanation graph with a corresponding goal node for every pde for l. A negative asl-explanation thus provides an explanation for why every possible derivation of l based on P is blocked by M resp. why there is at least one NAF-Literal that is necessary for the derivation but is not satisfied with respect to M. The negative asl-explanation is empty iff there is no pde for l on a set of statements.

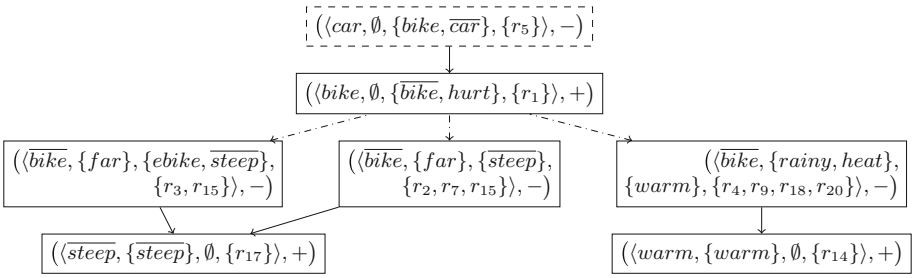

Fig. 3. asl-explanation graph G_3 for *car* (see Example 4)

Example 5 shows that different positive asl-explanations and the set of asl-explanation graphs in negative asl-explanations can contain similar relations between pdes that don't provide additional information. G_1 of Fig. 1 shows how the derivation in the goal node can be inhibited by $\overline{car}$ resp. $\neg car$. G_2 and G_3 in Figs. 2 and 3 show an alternative "defense-strategy" via *bike*. The comparison of G_2 and G_3 shows: The set of defending nodes in G_3 is a proper subset of the set of defending nodes in G_2, G_3 is thus a more compact way to defend the positive node with *bike*. Both graphs contain nodes $n_1 = \big(\langle \overline{bike}, \{far\}, \{ebike, \overline{steep}\}, \{r_3, r_{15}\}\rangle, -\big)$ and $n_2 = \big(\langle \overline{bike}, \{far\}, \{\overline{steep}\}, \{r_2, r_7, r_{15}\}\rangle, -\big)$ to evaluate why the corresponding pdes are blocked by the stable model. Intuitively, the evaluation of n_1 seems to be unnecessary because every explanation for why the pde corresponding to n_2 can be blocked is also an explanation for the pde corresponding to n_1. To improve the benefit of asl-explanations in practice w.r.t. these observations, a criterion to compare pdes is defined.

Definition 10 (specificity). *Let $ADF(P) = (S, C)$ be the ADF corresponding to a logic program P, $p = \big((a_0, \ldots, a_n, s), B\big)$ with set representation $\langle s, F, B, R\rangle$ and $p' = \big((a'_0, \ldots, a'_m, s), B'\big)$ with set representation $\langle s, F', B', R'\rangle$ two pdes for s on S. p* is more specific than *p' ($p \succ p'$) if (1) $F' \subseteq F$ and $B' \subset B$ or (2) $F' \subset F$ and $B' \subseteq B$ hold.*

Because specificity depends on statements for facts and the blocking set only, pdes with the same set-representation behave identically w.r.t. specificity. Specificity is used in Definition 11 to reduce the size of asl-explanation graphs by the restriction to least resp. most specific pdes for negative resp. positive nodes.

Definition 11 (reduced asl-explanation graph). *Let P be a logic program, $ADF(P)$ its corresponding ADF, M' a stable model of $ADF(P)$, $G = (V, E)$ an asl-explanation graph for $l \in Lit(P)$ w.r.t. M' and the goal node t. Let $G' = (V', E')$ be a subgraph of G with $t \in V'$ such that the following conditions hold:*

1. *For every positive node $(\langle s, F, B, R\rangle, +) \in V'$ and pde p corresponding to $\langle s, F, B, R\rangle$ there is no pde p' that is not blocked by M' s.t. $p' \succ p$.*

2. *For every negative node* $(\langle s, F, B, R\rangle, -) \in V'$ *and pde* p *corresponding to* $\langle s, F, B, R\rangle$ *there is no pde* p' *s.t.* $p \succ p'$.

The subgraph of G' *that contains goal node* t*, all nodes reachable from* t *and associated edges is a* reduced asl-explanation graph for l *w.r.t.* M'.

Based on reduced asl-explanation graphs, it is possible to build *reduced asl-explanations* that can be defined analogously to Definition 9 but based on the set of reduced asl-explanation graphs only.

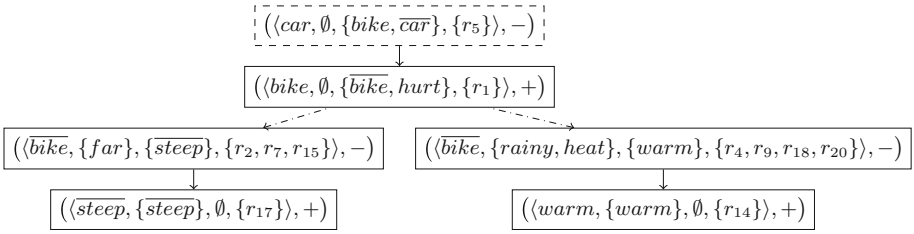

Fig. 4. reduced asl-explanation graph G'_2 for *car* (see Example 6)

Example 6. For program P in Example 1, there are two reduced asl-explanation graphs for *car* w.r.t. the stable model M' (see Example 2). The asl-explanation graph in Fig. 1 is reduced because it does not contain two nodes for the same literal, the second graph is depicted in Fig. 4. The reduced asl-explanation for *car* w.r.t. M thus consists of G_1 and G'_2.

A reduced asl-explanation graph contains no nodes with pdes that are comparable with respect to specificity (disregarding the goal node). Negative nodes in reduced asl-explanation graphs correspond to least specific pdes whose blocking set is minimal w.r.t. set inclusion. According to the definition of specificity and asl-explanation graphs, every node that can be linked to a node corresponding to a least specific pde via a defense edge can also be linked to a node for a more specific pde. The defenders in a reduced asl-explanation graph are thus particularly meaningful as the evaluation of the blocking set of the least specific pde explains why more specific pdes are blocked. Positive nodes in reduced asl-explanation graphs correspond to most specific pdes that are not blocked by the stable model. The set of evaluated outgoing attack edges for a most specific pde is a superset of the evaluated attack edges for nodes with a less specific pde. The evaluation of less specific pdes thus provides no additional information and the size of an asl-explanation graph can be reduced.

Specificity of pdes may also be used as a criterion to filter the set of positive resp. negative asl-explanation graphs w.r.t. their information content by comparing the pdes corresponding to the goal nodes and selecting only the graphs with

the most resp. least specific ones. Another criterion to compare asl-explanation graphs for the same literal and answer set could be the sets of defending nodes. An asl-explanation graph with a set of defending nodes that is minimal w.r.t. set inclusion represents a very compact defending strategy. Further details on these two filtering strategies can be found in [12].

Due to the specificity criterion, the number of considered pdes and asl-explanation graphs can be reduced. Thus, for larger programs, the size and number of asl-explanation graphs for a literal do not necessarily increase.

6 Conclusion and Future Work

In this paper we presented asl-explanation graphs as a possibility to compose argumentative explanations for why a literal is or is not contained in a given answer set. A prototypical implementation allows one to compute (reduced) asl-explanations and the visualization of asl-explanation graphs, depending on user input, and is to provide explanations for logistics applications of ASP [13]. In [12], an extension of the translation shown in Sect. 4 is presented that is able to deal with disjunctive rules and cardinality rules, and provides the base for an adapted definition of asl-explanations. Furthermore, it is shown how the translation, particularly the statements for complementary literals and constraints, and (stable) model semantics of ADFs can be used to explain why a given literal set is not an answer set of the program. Directions for future work can be the development of further mechanisms to improve clarity, e.g. for non-ground input programs, or the consideration of other parts of the input language for solvers as conditional literals or optimization statements.

References

1. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks. An overview. IFCoLog J. Log. Their Appl. **4**(8), 2263–2317 (2017)
2. Brewka, G., Strass, H., Ellmauthaler, S., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks revisited. Proc. IJCAI **2013**, 803–809 (2013)
3. Chen, Z.: Automating disease management using answer set programming programming. In: Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, pp. 22:1–22:10. OASICS (2016)
4. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Mag. **37**(3), 53 (2016)
5. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory Pract. Log. Program. **4**(2), 95–138 (2004)
6. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**(3), 365–385 (1991)
7. Grasso, G., Iiritano, S., Leone, N., Lio, V., Ricca, F., Scalise, F.: An ASP-based system for team-building in the Gioia-Tauro seaport. In: Carro, M., Peña, R. (eds.) PADL 2010. LNCS, vol. 5937, pp. 40–42. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11503-5_5

8. Oetsch, J., Pührer, J., Tompits, H.: Catching the ouroboros: on debugging non-ground answer-set programs. Theory Pract. Log. Program. **10**(4–6), 513–529 (2010)
9. Oetsch, J., Pührer, J., Tompits, H.: Stepping through an answer-set program. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS (LNAI), vol. 6645, pp. 134–147. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20895-9_13
10. Polberg, S.: Extension-based semantics of abstract dialectical frameworks. CoRR abs/1405.0406 (2014)
11. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for logic programs under answer set semantics. Theory Pract. Log. Program. **9**(1), 1–56 (2009)
12. Rolf, L.: Argumentation für Erklärung und Debugging von clingo-ASP-Lösungen (Argumentation for explaining and debugging of Clingo-ASP-solutions), TU Dortmund (2018)
13. Schieweck, S., Kern-Isberner, G., ten Hompel, M.: Various approaches to the application of answer set programming in order-picking systems with intelligent vehicles. In: Proceedings of the 9th International Joint Conference on Computational Intelligence, vol. 1, pp. 25–34 (2017)
14. Schulz, C., Sergot, M., Toni, F.: Argumentation-based answer set justification. In: Working Notes of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning (2013)
15. Schulz, C., Toni, F.: Justifying answer sets using argumentation. Theory Pract. Log. Program. **16**(01), 59–110 (2016)