Witnesses for Answer Sets of Basic Logic Programs

Yisong Wang[1,2,3,4] , **Xianglong Wang**[1,3] , **Zhongtao Xie**[1,3] , **Thomas Eiter**[5]
[1] State Key Laboratory of Public Big Data, Guiyang, Guizhou, China
[2] Key Laboratory of Advanced Medical Imaging and Intelligent Computing of Guizhou Province, China
[3] College of Computer Science and Technology, Guizhou University, Guiyang, Guizhou, China
[4] Institute for Artificial Intelligence, Guizhou University, Guiyang, Guizhou, China
[5] Knowledge-based Systems Group, Institute of Logic and Computation, TU Wien, Austria
yswang@gzu.edu.cn, gzu_xlwang@163.com, francisol.net@gmail.com thomas.eiter@tuwien.ac.at

Abstract

Explanation plays an important role in the decisions of both symbolic and neural network-based AI systems. Logic programs under answer set semantics (ASP) have been a typical declarative reasoning and problem-solving paradigm that has extensive applications in various AI domains. In this paper, we consider the issue of explanation for logic programs with abstract constraint atoms (c-atoms) under SPT-answer set semantics. Such c-atoms are general enough to capture complex constructors of logic programs, including aggregates, and the SPT-answer sets exclude circular justifications that other semantics have. We propose a minimal reduct for logic programs with c-atoms that yields a new semantic characterization of SPT-answer sets, and then introduce an extension of resolution for clauses with c-atoms. As we show, every atom in an SPT-answer set enjoys an extended resolution proof from the minimal reduct of its logic program. Finally, we present minimal sufficient subsets of logic programs (witnesses) to structure such an extended resolution proof for an atom in an SPT-answer set. Our results contribute to the justification of answer sets and provide a basis for explainability of ASP-based applications.

1 Introduction

Explanation tasks in AI include justifying the behavior of autonomous agents, debugging machine learning models, explaining medical decision making, and explaining classifier predictions [Burkart and Huber, 2021; Dazeley *et al.*, 2021]. The subject has been attracting growing attention with the success of deep learning neural networks [Marques-Silva and Huang, 2024; Weber *et al.*, 2024]. Notably, explaining symbolic reasoning has been extensively studied in the past in several different contexts, including database query languages such as Datalog [Cheney *et al.*, 2009; Arioua *et al.*, 2015], ontological reasoning under description logics [Bienvenu *et al.*, 2019; Peñaloza, 2020] and existential rules [İsmail İlkan Ceylan *et al.*, 2025].

Logic programming under answer set (stable model) semantics (ASP in short) is a declarative problem solving paradigm [Baral, 2003; Brewka *et al.*, 2011]. It encodes the specifications of a problem in a logic program whose answer sets correspond to the solutions of the problem, which has been widely and successfully applied in planning, diagnosis, scheduling [Gebser *et al.*, 2012; Brewka *et al.*, 2016; Lifschitz, 2019] thanks to the efficient ASP solvers clingo and DLV. An explanation for the answer sets of a logic program amounts to answering "why a set of atoms is an answer set in a well-justified manner, instead of by definition" [Fandinno and Schulz, 2019]. This is quite useful for debugging ASP programs. From the perspective of knowledge representation, an explanation for a solution may be equally or more important than the solution itself [Sosa, 2019]. This challenging task has been extensively investigated from the perspective of off-line justification [Pontelli *et al.*, 2009], causal graph [Cabalar and Fandinno, 2016], minimal assumption set and well-founded derivation [Alviano *et al.*, 2024] and so forth. Various explaining ASP tools and systems are available, such as DiscASP [Li *et al.*, 2021], s(CASP) [Arias *et al.*, 2020], xclingo [Cabalar *et al.*, 2020] and xASP [Alviano *et al.*, 2024].

Abstract constraint atoms (c-atoms) [Marek and Truszczynski, 2004] are a general framework to capture complex constructors in logic programs, such as weight constraints [Simons *et al.*, 2002] and aggregates [Dell'Armi *et al.*, 2003; Faber *et al.*, 2004; Ferraris, 2011]. Such constructors are widely used in practical ASP languages, and they usually complicate the answer set semantics of logic programs [Son *et al.*, 2007; Liu *et al.*, 2010; Shen *et al.*, 2009; Gelfond and Zhang, 2019; Alviano *et al.*, 2023]. Very recently, the explanation issue for answer sets was investigated from the perspective of witness for disjunctive logic programs [Wang *et al.*, 2023], and m-justifications and r-justifications for FLP-answer sets [Faber *et al.*, 2004] of logic programs with abstract constraint atoms [Eiter and Geibinger, 2023]. It is well-known that FLP-answer sets suffer from circular justifications.

This paper follows the idea of the witness approach to the explanation issue of another answer set semantics of logic programs with abstract constraint atoms, namely SPT-answer sets [Son *et al.*, 2007; Shen *et al.*, 2009; Shen *et al.*, 2014]. These answer sets have the property of excluding circular justification in FLP-answer sets, which makes the SPT-semantics particularly attractive. Informally, witnesses are resolution proofs in the context of an answer set that respect dependencies among atoms. The main contributions

of the paper are as follows.

- We present a general minimal reduct for logic programs with c-atoms. Informally, the minimal reduct is to properly remove all the assumed false atoms from a logic program [Wang *et al.*, 2023]. With the help of minimal reduct, a new characterization of SPT-answer sets is established and the minimal model decomposition theorem [Angiulli *et al.*, 2022] is then extended to logic programs without nonmonotonic c-atoms.
- We propose a notion of CA-resolution for rules with c-atoms to resolve a c-atom at one resolution step. We show that every atom in an SPT-answer set of a logic program has a sound CA-resolution proof from the minimal reduct of the logic program *w.r.t.* the SPT-answer set.
- Finally, we present the notions of α-witness and β-witness to explain why an atom is in an SPT-answer set in terms a CA-resolution proof. While α-witnesses are concerned with a set of atoms in each step, β-witness are concerned with a single atom.

Since the current well-known ASP solvers compute FLP-answer sets and SPT-answer sets are non-circular FLP-answer sets, this "resolution proof"-based witness has potential practical applications for ASP, *e.g.*, debugging of ASP programs.

The remainder of the paper is outlined as follows. In the next section, we present the basic notions and notations of logic programs with c-atoms. Section 3 presents minimal reduct and its properties about minimal model decomposition. Section 4 contains the CA-resolution. The α-witness and β-witness are presented in Section 5. We compare the β-witness with r-justification in Section 6. Finally, concluding remarks and future work are discussed in Section 7.

2 Preliminaries

In this section, we recall some basic notions and notations of logic programs with abstract constraint atoms [Marek and Truszczynski, 2004; Faber *et al.*, 2004; Son *et al.*, 2007; Liu *et al.*, 2010; Shen *et al.*, 2009] and the dependency graph of logic programs [Angiulli *et al.*, 2022].

2.1 Syntax and semantics

We assume an underlying propositional language $\mathcal{L}$ over a finite set $\mathcal{A}$ of atoms. The notions of atoms, literals, interpretations, satisfaction, models, minimal models, least models, and logical consequences are the standard ones in $\mathcal{L}$.

An *abstract constraint atom* (*c-atom* in short) is a tuple $A = (D, C)$, where $D \subseteq \mathcal{A}$ is its *domain* and $C \subseteq 2^D$ is the set of *candidate solutions* of A. For a c-atom $A = (D, C)$, we denote $A_d = D$ and $A_c = C$. We use $\top$ to denote $(D, 2^D)$ and $\bot$ for $(D, \emptyset)$. Any propositional atom a can be expressed by the c-atom $(\{a\}, \{\{a\}\})$. We call the latter *elementary*, and whenever convenient, we identify it with a. The *complement* of a c-atom $A = (D, C)$ is $\overline{A} = (D, \overline{C})$ where $\overline{C} = 2^D \setminus C$. It is evident $A = \overline{\overline{A}}$. A *c-literal* is either a c-atom A (positive c-literal) or its (default) negation *not* A *(negative c-literal)*.

Let $A = (D, C)$ be a c-atom. It is *monotonic* if $S \in C$ implies $S' \in C$ for all S and S' with $S \subseteq S' \subseteq D$. It is *convex* if $S'' \in C$ implies $S' \in C$ for all S, S' and S'' with $S \in C$ and $S \subseteq S' \subseteq S'' \subseteq D$. It is *anti-monotonic* if for every $S \subseteq D$, $S \in C$ implies $S' \in C$ for every $S' \subseteq S$.

A (disjunctive) logic program (with abstract constraint atoms) is a finite set of rules of the form,

$$A_1 \vee \cdots \vee A_l \leftarrow A_{l+1}, \ldots, A_m, \mathit{not}\, A_{m+1}, \ldots, \mathit{not}\, A_n \quad (1)$$

where A_i $(1 \leq i \leq n)$ are c-atoms.

Let r be a rule of the form (1). We denote the *head* of r by $\mathit{hd}(r) = \{A_i \mid 1 \leq i \leq l\}$, the *positive body* of r by $\mathit{bd}^+(r) = \{A_i \mid l+1 \leq i \leq m\}$, the *negative body* of r by $\mathit{bd}^-(r) = \{A_i \mid m+1 \leq i \leq n\}$ and the *body* of r by $\mathit{bd}(r) = \mathit{bd}^+(r) \cup \mathit{not}\, \mathit{bd}^-(r)$, with $\mathit{not}\, S = \{\mathit{not}\, s \mid s \in S\}$. For simplicity, a rule r is usually written as

$$\mathit{hd}(r) \leftarrow \mathit{bd}^+(r) \cup \mathit{not}\, \mathit{bd}^-(r). \quad (2)$$

A rule of form (1) is called *normal* if $l = 1$, a *constraint* if $l = 0$, a *fact* if $l = n = 1$ and A_1 is elementary and in this case the "$\leftarrow$" is usually omitted, *positive* if $n = m$, *definite* if it is normal and $m = n$, *basic* if every A_i $(1 \leq i \leq l)$ is elementary[1]; *convex* if every A_i, for $1 < i \leq n$, is convex. A logic program is *normal (positive, definite, basic and convex, respectively)* if all its rules are normal (positive, definite, basic, and convex respectively). In the present paper, we focus on the fragment of basic logic programs and assume that logic programs are of this form unless explicitly stated otherwise.

Let I be an interpretation (a set of atoms). The satisfaction relation (model) $\models$ is defined below: let A be a c-atom, S be a set of c-literals, r be a rule, P be a logic program,

- $I \models A$ if $I \cap A_d \in A_c$;
- $I \models \mathit{not}\, A$ if I does not satisfy A;
- $I \models S$ if $I \models s$ for every $s \in S$;
- $I \models r$ if $I \models \mathit{bd}(r)$ implies $I \models h$ for some $h \in \mathit{hd}(r)$;
- $I \models P$ if $I \models r$ for every $r \in P$.

Let M and S be two sets of atoms. We say that S *conditionally satisfies* a c-atom A *w.r.t.* M, written $S \models_M A$, if $S \models A$ and $I \in A_c$ for every I with $S \cap A_d \subseteq I \subseteq M \cap A_d$. By $S \models_M T$ for a set T of c-atoms we mean $S \models_M A$ for every $A \in T$.

Let P be a normal basic logic program with c-atoms. A set $M \subseteq \mathcal{A}$ is an *SPT-answer set* of P whenever $M = T^{\infty}_{P^M}(\emptyset, M)$ where

- P^M is obtained from P by
  - removing every rule r from P if $M \models A$ for some $A \in \mathit{bd}^-(r)$, and
  - removing all the remaining *not* A from P where A is a c-atom;
- the operator $T_P : 2^{\mathcal{A}} \times 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ is defined as, for a definite basic logic program P,

$$T_P(S, M) = \{a \mid \exists r \in P \; s.t. \; S \models_M \mathit{bd}(r), \mathit{hd}(r) = \{a\}\}.$$

A set $M \subseteq \mathcal{A}$ is an *FLP-answer set* of a logic program P if M is a minimal model of the FLP-reduct $P^{M,FLP}$ of P *w.r.t.* M, given by $P^{M,FLP} = \{r \in P \mid M \models \mathit{bd}(r)\}$.

[1] This is different from the one in [Shen *et al.*, 2009] in which it requires $l = 1$ additionally. Thus, the notion of "basic" in [Shen *et al.*, 2009] corresponds to "normal basic" in this paper.

Let W and V be two disjoint subsets of $\mathcal{A}$. The *W-prefixed power set* of V is the family $W \uplus V = \{W' \mid W \subseteq W' \subseteq W \cup V\}$ of all subsets W' between W and $W \cup V$; it is *maximal* in a c-atom A if $W \uplus V \subseteq A_c$ and there is no other W' and V' such that $W' \uplus V' \subseteq A_c$ and $W \uplus V \subset W' \uplus V'$. The set of all maximal prefixed power sets of A is denoted by A_c^*. Intuitively, A_c^* contains all maximal intervals $[W, W \cup V]$ of the powerset lattice in A_c; computing A_c^* from A_c is tractable, cf. Theorem 3.5 of [Shen *et al.*, 2009].

Definition 1. *Let A be a c-atom and I an interpretation. $W \uplus V \in A_c^*$ is an* abstract satisfiable set *of A w.r.t. I if $I \cap A_d \in W \uplus V$. In this case, W is called a* satisfiable set *of A w.r.t. $I \cap A_d$.*

Given a basic logic program P and an interpretation I, the *generalized Gelfond-Lifschitz reduct* of P *w.r.t.* I, written as $P^{I,SYY}$, is the basic logic program obtained from P by:

(a) removing from P all rules whose bodies contain either a negative c-literal *not* A such that $I \models A$ or a c-atom A such that $I \not\models A$,

(b) removing all negative c-literals from the remaining rules,

(c) replacing each c-atom A in the body of a rule with a special elementary atom θ_A and introduce a new rule

$$\theta_A \leftarrow a_1, ..., a_m \tag{3}$$

for each satisfiable set $W = \{a_1, \ldots, a_m\}$ of A *w.r.t.* $I \cap A_d$,

(d) replacing $hd(r)$ with $hd(r) \cap M$,

Definition 2. *An interpretation $I \subseteq \mathcal{A}$ is a* stable model *of a logic program P if $P^{I,SYY}$ has a minimal model M such that $I = M \cap \mathcal{A}$.*

It has been shown that, for every positive normal logic program P, $M \subseteq \mathcal{A}$ is an SPT-answer set of P according to [Son *et al.*, 2007] if and only if M is a stable model of P by Definition 2, cf. Theorem 6.2 of [Shen *et al.*, 2009]. We thus refer to stable models by Definition 2 also as SPT-answer sets.

Note that if P is a positive logic program without c-atoms, i.e., all atoms are elementary, then the answer sets of P are exactly the minimal models of P. However, if P is a positive logic program with c-atoms, a minimal model of P may not be an FLP-answer set of P, and thus not an SPT-answer set of P[2]. For example, let $P = \{a \leftarrow (\{a, b\}, \{\emptyset, \{a, b\}\})\}$. The set $M = \{a\}$ is a minimal model of P. But M is not an FLP-answer set of P. In fact, P has no FLP-answer set.

2.2 Dependency graph

The *(positive) dependency graph*[3] G_P of a logic program P is the directed graph (V, E), where

- V consists of all the atoms occurring in P and,
- $(q, p) \in E$ if, for some $r \in P$ and $A \in bd^+(r)$, $p \in hd(r)$ and some W exists *s.t.* $q \in W$ and $W \uplus V \in A_c^*$.

[2]For positive basic logic programs, SPT-answer sets are well-justified FLP-answer sets, cf, Theorem 9 of [Shen *et al.*, 2014].

[3]The dependency graph defined below coincides with the one in [Angiulli *et al.*, 2022]. But the direction of edge or arc is opposite to that given in [Lin and You, 2002; Shen *et al.*, 2009].

For a directed graph $G = (V, E)$ and $v \in V$, we denote by $D_G(v)$ the set of all ancestor nodes u of v in G, i.e., G has a path $v_0, \ldots, v_{n+1}$ with $v_0 = u$ and $v_{n+1} = v$; for any set $V' \subseteq V$ we let $D_G(V') = \bigcup_{v' \in V'} D_G(v')$. A vertex v is a *source* of G if $D_G(v) = \emptyset$. If P is clear from its context, *i.e.*, G is the dependency graph of P, we also call v a *source* of P.

The *SCC-graph* of a directed graph $G = (V, E)$ is the directed acyclic graph (DAG) $\mathbb{S}_G = (\mathbb{V}, \mathbb{E})$ [Leone *et al.*, 1997] where

- $\mathbb{V}$ is the set of the strongly connected components (SCCs) of G, i.e., every $C \in \mathbb{V}$ is a maximal subgraph $G' = (V', E')$ of G such that G' has a path from any vertex $v' \in V'$ to every other vertex in V'; we also denote C by V' when it is clear from its context, and
- $\mathbb{E}$ consists of all edges (C, C') such that $E \cap (C \times C') \neq \emptyset$, *i.e.*, there are $p \in C$ and $q \in C'$ for some $(p, q) \in E$.

For a logic program P, we write $\mathbb{S}_P$ instead of $\mathbb{S}_{G_P}$ for convenience. The *atom-clause dependency graph* of a basic logic program P is the directed graph $\mathbb{G}_P = (V, E)$, where

- V consists of the atoms and rules occurring in P;
- for each rule $\delta \in P$, E contains edges $(q, \delta) \in E$ and $(\delta, p) \in E$ for all $p \in hd(\delta)$ and $q \in W$ for some $A \in bd^+(\delta)$ and $W \uplus V \in A_c^*$ for some V, respectively.

Intuitively, if both (A, δ) and (δ, B) are in E, then A depends on B by rule δ. The *super-dependency graph* $\mathbb{SG}_P$ of P is then the collapsed dependency graph of $\mathbb{G}_P$, i.e., $\mathbb{SG}_P = \mathbb{S}_{\mathbb{G}_P}$. We call a source of $\mathbb{SG}_P$ also a source of P.

The example below demonstrates these dependency graphs.

Example 1. *Let P consist of the following rules:*

$$r_1 : a \leftarrow b, \quad r_2 : b \leftarrow a, \quad r_3 : a \vee b, \quad r_4 : c \leftarrow A$$

where $A = (\{a, b\}, \{\{a\}, \{b\}\})$. Then, $A_c^* = \{\{a\} \uplus \emptyset, \{b\} \uplus \emptyset\}$. The various dependency graphs for P are as follows:

- *$G_P = (V_1, E_1)$ with $V_1 = \{a, b, c\}$ and $E_1 = \{(b, a), (a, b), (a, c), (b, c)\}$. Notice that this graph has no source, and that the SCCs of G_P are $\{c\}$ and $\{a, b\}$.*
- *Hence, $\mathbb{S}_P = (V_2, E_2)$ with $V_2 = \{v_1, v_2\}$, $v_1 = \{a, b\}$, $v_2 = \{c\}$ and $E_2 = \{(v_1, v_2)\}$.*
- *$\mathbb{G}_P = (V_3, E_3)$ with $V_3 = V_1 \cup \{r_1, r_2, r_3, r_4\}$ and $E_3 = \{(r_1, a), (b, r_1)\} \cup \{(r_2, b), (a, r_2)\} \cup \{(r_3, a), (r_3, b)\} \cup \{(r_4, c), (a, r_4), (b, r_4)\}$. Its SCCs are $\{r_3\}$, $\{r_4\}$, $\{r_1, r_2, a, b\}$, and $\{c\}$.*
- *Hence, $\mathbb{SG}_P = (V_4, E_4)$ with $V_4 = \{v_1, v_2, v_3, v_4\}$, $v_1 = \{a, b, r_1, r_2\}$, $v_2 = \{r_3\}$, $v_3 = \{r_4\}$, $v_4 = \{c\}$ and $E_4 = \{(v_1, v_2), (v_1, v_3), (v_3, v_4)\}$. Here, v_1 is its unique source.*

3 Minimal Reduct and Model Decomposition

In this section, we propose the notion of minimal reduct and prove our minimal model decomposition theorem for basic logic programs.

3.1 Minimal reduct

Let A be a c-atom and $M \subseteq \mathcal{A}$. The *(extended) minimal reduct* of A *w.r.t.* M, written $\mathrm{MR}(A, M)$, is defined as

(a) $\bot$ if $M \not\models A$,

(b) $\top$ if either $M \models A$ and $A_d \cap M = \emptyset$, or $2^{A_d \cap M} = \{S \cap M \mid S \in A_c\}$,

(c) $(A_d \cap M, B_c)$ where $B_c = \{Z \mid \text{there is } W \uplus V \in A_c^* \text{ such that } M \cap A_d \subseteq W \cup V \text{ and } W \subseteq Z \subseteq M \cap A_d\}$ if $M \models A$ and $M \cap A_d \neq \emptyset$.

Intuitively, given the assumption $M \subseteq \mathcal{A}$ that the atoms in M are true and the others are false, the minimal reduct $\mathrm{MR}(A, M)$ of a c-atom A removes from A the assumed false atoms and candidate solutions in which they are true. For instance, let $A = (\{a, b\}, \{\{a\}, \{b\}\})$. Recall that $A_c^* = \{\{a\} \uplus \emptyset, \{b\} \uplus \emptyset\}$. We have $\mathrm{MR}(A, \{a, b\}) = \bot$ since $\{a, b\} \not\models A$ and $\mathrm{MR}(A, \{a\}) = (\{a\}, \{\{a\}\})$.

Note that for an elementary atom a, the corresponding c-atom is $A = (\{a\}, \{\{a\}\})$. Thus, $\mathrm{MR}(A, M) = A$ if $M \models A$ and $\bot$ otherwise. For the c-atom $A = (\{a\}, \{\emptyset\})$, $M \models A$ whenever $M \cap \{a\} = \emptyset$. In this case, $\mathrm{MR}(A, M) = \top$ if $M \models A$ and $\bot$ otherwise.

Let P be a basic logic program, and $M \subseteq \mathcal{A}$. The *minimal reduct* of P *w.r.t.* M, written $\mathrm{MR}(P, M)$, contains for each $r \in P$ with $M \models \mathit{bd}(r)$, the rule

$$M \cap \mathit{hd}(r) \leftarrow \{\mathrm{MR}(B, M) \mid B \in \mathit{bd}^+(r), \mathrm{MR}(B, M) \neq \top\}$$

Please note that $\mathrm{MR}(\{r\}, M) = \emptyset$ if $M \not\models \mathit{bd}(r)$. Thus, this minimal reduct for basic logic programs is indeed a generalization of the same notion for logic programs without c-atoms.

Intuitively, the minimal reduct $\mathrm{MR}(P, M)$ of a logic program P *w.r.t.* M keeps the rules whose body is satisfied by M and applies the minimal reduct to all c-atom in their heads and positive bodies, and it removes the negative bodies; furthermore, simplications (remove $\top$ in bodies, $\bot$ in heads) are made. We note that the minimal reduct for basic logic programs generalizes the one for disjunctive logic programs [Wang *et al.*, 2023]. It is different from the GL-reduct [Gelfond and Lifschitz, 1988], FLP-reduct [Faber *et al.*, 2004], and Ferraris's reduct [Ferraris, 2011].

Proposition 1. *Let r be a basic rule and $M \subseteq \mathcal{A}$. The following statements are equivalent to each other.*

(i) $M \models r$.

(ii) $M \models \mathit{MR}(\{r\}, M)$.

(iii) $M^* \models \{r\}^{M, SYY}$ *where* $M^* = M \cup \{\theta_A \mid A \in \mathit{bd}^+(r), M \models \mathit{bd}(r)\}$.

In terms of the minimal reduct, we obtain a new characterization for SPT-answer sets as follows.

Theorem 1. *Let P be a basic logic program, and $M \subseteq \mathcal{A}$. Then, the following statements are equivalent to each other.*

(i) M *is an SPT-answer set of* P.

(ii) M^* *is the least model of* $P^{M, SYY}$ *where* $M^* = M \cup \{\theta_A \mid r \in P, M \models \mathit{bd}(r), A \in \mathit{bd}^+(r)\}$.

(iii) M *is the least model of* $\mathit{MR}(P, M)$.

The following example confirms the new characterization of SPT-answer sets in terms of the minimal reduct.

Example 2 (Continued from Example 1)**.** *Let $M = \{a, b\}$. One can check that M is the unique minimal model of P and an SPT-answer set of P: $\mathit{MR}(P, M) = \{r_1, r_2, r_3\}$ and the least model of $\mathit{MR}(P, M)$ is M.*

3.2 Minimal model decomposition

In this section, we present a theorem for decomposing minimal models of basic logic programs generalizing previous results.

Let Σ be a basic positive program. We call a source S of $\mathbb{SG}_\Sigma$ *empty*, if $S \cap \mathcal{A} = \emptyset$, and denote by Σ_S the set of rules in Σ that mention only atoms from S.

For a basic positive program Σ and $X \subseteq \mathcal{A}$, the logic program $\mathit{Reduce}(\Sigma, X)$ is obtained from Σ by

(a) removing every rule r if $\mathit{hd}(r) \cap X \neq \emptyset$,

(b) removing every c-atom A from rule bodies if $A_d \setminus X = \emptyset$ and $X \cap A_d \in A_c$,

(c) replacing each remaining c-atom A by A' where $A'_d = A_d \setminus X$ and $A'_c = \{S \setminus X \mid S \in A_c\}$.

Clearly, if in Step (b) the c-atom A is an elementary atom $a \in X$, then A is removed. Thus, this reduction generalizes $\mathit{Reduce}(\Sigma, X, \emptyset)$ for propositional clause theories Σ [Ben-Eliyahu-Zohary *et al.*, 2017; Angiulli *et al.*, 2022], which replaces every atom p occurring in Σ by **true** if p is in X.

Proposition 2. *Let P be a basic positive program with merely monotonic c-atoms. A set $M \subseteq \mathcal{A}$ is a minimal model of P if and only if M is the least model of $\mathit{MR}(P, M)$.*

We note that the monotonicity condition in the above proposition is necessary and can not be relaxed to convexity:

- Let $P_1 = \{a \leftarrow (\{a\}, \{\emptyset\})\}$. The c-atom in the rule body is convex, and clearly $\{a\}$ is the unique minimal model of P_1. But P_1 has no SPT- and no FLP-answer set.
- Let $P_2 = \{a \leftarrow (\{a, b\}, \{\emptyset, \{b\}\}))$. The c-atom in the rule body is convex and nonmonotonic. The set $\{a\}$ is the unique minimal model of P but neither an SPT- nor an FLP-answer set of P_2.

Theorem 2 (Minimal model decomposition)**.** *Let P be a basic program without nonmonotonic c-atoms, M be a minimal model of P and $Q = \mathit{MR}(P, M)$. Then, for any nonempty source S of Q, $X = M \cap S$ is a minimal model of Q_S, and $M \setminus X$ is a minimal model of $\mathit{Reduce}(Q, X)$.*

Based on Theorem 2, a minimal model verification algorithm is presented in Algorithm 1. It generalizes similar ones for propositional clause theories [Ben-Eliyahu-Zohary *et al.*, 2017; Angiulli *et al.*, 2022] and for disjunctive logic programs without c-atoms [Wang *et al.*, 2023].

Note that a model M of a clause theory T is minimal if and only if the clause theory $T \cup \{\neg p \mid p \in \mathcal{A} \setminus M\} \cup \{\bigvee\{\neg p \mid p \in M\}\}$ is unsatisfiable, where $\bigvee S$ is the disjunction of all elements in S. The above algorithm CheckMinMRCA decomposes the minimal model checking (unsatisfiability) to a series of minimal model checks in terms of modularity *i.e.*, Theorem 2. The following result states its correctness.

Algorithm 1: CheckMinMRCA (Π, N)

Input: A basic positive program Π with merely monotonic c-atoms and a model M of Π
Output: **true** if M is a minimal model of Π, else **false**

$\Sigma \leftarrow \mathrm{MR}(\Pi, M)$; $N \leftarrow M$; $G \leftarrow \mathbb{SG}_\Sigma$;
Recursively delete from G all empty sources;
while G *has some nonempty source* S **do**
  if $S \cap N$ *is not a minimal model of* Σ_S **then break**
  $\Sigma \leftarrow \mathit{Reduce}(\Sigma, N \cap S)$;
  $N \leftarrow N - S$;
  $G \leftarrow \mathbb{SG}_\Sigma$;
  Recursively delete from G all empty sources;
end
if $N = \emptyset$ **then return true else return false**

Theorem 3. *Let P be a basic positive program without non-monotonic c-atoms and M be a model of P. Then M is a minimal model of P iff* CheckMinMRCA(P, M) *returns true.*

4 Resolution of c-atoms

In this section, we present the notion of resolution proof for basic positive rules, which we also call *clauses with c-literals* (CA-clauses in short).

Definition 3 (CA-resolution)**.** *Let $r, r_1, \ldots, r_n$ $(n \geq 1)$ be CA-clauses such that*

- $a_i \in hd(r_i)$ $(1 \leq i \leq n)$,
- $A \in bd(r)$ *and some W exists such that* $W = \{a_1, \ldots, a_n\}$ *and* $W \uplus V \in A_c^*$.

The result of CA-resolution $r, r_1, \ldots, r_n$ *on W of A, written $res(r, r_1, \ldots, r_n, A, W)$, is the following CA-clause:*

$$(hd(r) \cup \textstyle\bigcup_{i=1}^n (hd(r_i) \setminus \{a_i\})) \leftarrow ((bd(r) \setminus \{A\}) \cup \textstyle\bigcup_{i=1}^n bd(r_i)) \quad (4)$$

which we call the resolvent *of CA-resolution $r, r_1, \ldots, r_n$ on W of A. In this case, we call $r, r_1, \ldots, r_n$* resolvable on A *w.r.t. W.*

Please note that if A is elementary, then there are exactly two CA-clauses involved in the CA-resolution. Thus, the CA-resolution is indeed a generalization of the resolution for propositional clauses. The CA-resolution for the resolvent $res(r, r_1, \ldots, r_n, A, W)$ is *sound* if every model of $\{r, r_1, \ldots, r_n\}$ is also a model of $res(r, r_1, \ldots, r_n, A, W)$.

Example 3. *Let us consider the following CA-clauses:*

$$r_1 : a_1 \vee b_1, \qquad r_2 : a_2 \vee b_2, \qquad r : a \leftarrow A, B$$

where $A = (\{a_1, a_2\}, \{\{a_1, a_2\}\})$, $B = (\{b_1, b_2\}, \{\{b_1\}\})$. Note that $\{a_1, a_2\} \uplus \emptyset \in A_c^*$ and $\{b_1\} \uplus \emptyset \in B_c^*$. Then,

$res(r, r_1, r_2, A, \{a_1, a_2\}) = b_1 \vee b_2 \vee a \leftarrow (\{b_1, b_2\}, \{\{b_1\}\})$,
$res(r, r_1, B, \{b_1\}) = a_1 \vee a \leftarrow (\{a_1, a_2\}, \{\{a_1, a_2\}\})$.

We can check that both $res(r, r_1, r_2, A, \{a_1, a_2\})$ and $res(r, r_1, B, \{b_1\})$ are sound. However, CA-resolution is not sound in general E.g., consider the following CA-clauses:

$$r_1 : a, \qquad r : c \leftarrow (\{a, b\}, \{\{a\}\}) (= A).$$

We then have $res(r, r_1, A, \{a\}) = c \leftarrow$. Clearly $M = \{a, b\}$ satisfies both r_1 and r, but $M \not\models c$.

Lemma 1. *Let $N \subseteq \mathcal{A}$, P be a basic positive logic program such that MR$(P, N) = P$ and rules $R = r, r_1, \ldots, r_n$ in P are resolvable on A w.r.t. W. Then*

(i) MR$(res(R, A, W), N) =$ *$res(R, A, W)$.*
(ii) *$\{r, r_1, \ldots, r_n\} \models res(R, A, W)$.*

A *CA-resolution proof* of a CA-clause c from a set Σ of CA-clauses is a sequence

$$res(R^0, A^0, W^0), \ldots, res(R^k, A^k, W^k) \quad (5)$$

where $R^i = r^i, r_1^i, \ldots, r_{n_i}^i$, $0 \leq i \leq k$, of CA-resolutions such that $c = res(R^k, A^k, W^k)$, and for every i, $0 \leq i \leq k$,

- $A^i \in bd(r^i)$, $W_i = \{a_{i,1}, \ldots, a_{i,n_i}\}$,
- $W^i \uplus V^i \in A_c^{i\,*}$ for some V^i and
- r^i, r_j^i, $1 \leq j \leq n_i$, are from $\Sigma \cup \{res(R^t, A^t, W^t) \mid 0 \leq t < i\}$.

The CA-resolution proof (5) is *sound* if each CA-resolution for the resolvent of (5) is sound. By $\Sigma \vdash_{CA} c$ we denote that there is a CA-resolution proof of c from Σ, meaning that Σ can "derive" c (by a CA-resolution proof).

From item (ii) of Lemma 1 and the definition of minimal reduct, we obtain the following result.

Proposition 3. *Let P be a basic logic program and $M \subseteq \mathcal{A}$ such that MR$(P, M) = P$. Then every CA-resolution proof of a clause c from P is sound.*

For convenience and in abuse of notation, we let in what follows $a \vdash_{CA} a$ for atoms $a \in \mathcal{A}$, i.e., any fact can derive itself by a CA-resolution proof.

The following theorem shows that for any SPT-answer set M of a basic logic program P, every atom in M admits a CA-resolution proof from the minimal reduct of P *w.r.t.* M. In this sense, such proofs can be taken as the justification for SPT-answer sets.

Theorem 4. *Let M be an SPT-answer set of a basic logic program P. Then there is a sound CA-resolution proof for every $a \in M$ from MR(P, M).*

5 Witnesses

We are now in the position to present the notion of witnesses for SPT-answer sets of basic logic programs.

We first present the notion of minimal witness to capture the "minimal sufficient subprogram" (under set inclusion) for the CA-resolution proof.

Definition 4 ((minimal) Witness)**.** *Let $M \subseteq \mathcal{A}$ and B, S be disjoint subsets of M. A basic logic program P is a* witness *of B under S w.r.t. M if MR$(P, M) \cup S \vdash_{CA} B$. Moreover, P is minimal if no $P' \subset P$ exists such that MR$(P', M) \cup S \vdash_{CA} B$.*

The set of all minimal witnesses $P' \subseteq P$ of B under S *w.r.t.* M is denoted by $\mathrm{MW}(B, P, S, M)$.

In other words, under the assumption M (to be justified as an SPT-answer set of a basic logic program P) and the justified subset S of M, the (minimal) witness of B under S *w.r.t.* M is

Algorithm 2: MinWitness(Π, B, S, M)

Input: A basic logic program Π, a nonempty SPT-answer set M of Π, and two disjoint subsets B, S of M
Output: A minimal witness $\Pi' \subseteq \Pi$ of B under S *w.r.t.* M

```
Π' ← MR(Π, M);
Π'' ← Π;
foreach r ∈ Π' do
    if Π' − {r} ∪ S ⊢_CA B then
        Π' ← Π' − {r};
        Π'' ← Π'' − {r' ∈ Π | {r'} = MR({r}, M)};
    end
end
return Π'' ;
```

a (minimal) subprogram P' of P such that every atom in B has a CA-resolution proof from the minimal reduct $\mathrm{MR}(P', M)$ and the as justified regarded atoms in S. Algorithm 2 can be used to compute such a minimal witness.

Example 4. *Let P consist of the following rules:*

$$r_1 : a \leftarrow, \qquad r_2 : b \leftarrow not\ c, \qquad r_3 : c \leftarrow A$$

where $A = (\{a, b\}, \{\{a\}\})$ and $A^*_c = \{\{a\} \uplus \emptyset\}$. Then $M = \{a, c\}$ is an SPT-answer set of P.

The CA-resolution $res(r_3, r_1, A, a)$ is $c \leftarrow$, and $P'_c = \{r_1, r_3\}$ is the only minimal witness of $\{c\}$ under $\emptyset$ w.r.t. M. Under M and the justified a, the minimal sufficient subprogram to "justify" c is $\{r_3\}$, i.e., $\mathrm{MW}(\{c\}, P, \{a\}, M) = \{\{r_3\}\}$.

5.1 α-witness and $\alpha^\star$-witness

The following notions of α-witness and $\alpha^\star$-witness capture the structural witness for SPT-answer sets. They serve to explain why a group of atoms are in an SPT-answer set.

Definition 5 (α^*-witness and α-witness)**.** *Let P be a basic logic program, $M \neq \emptyset$ be an SPT-answer set of P, B and S be disjoint subsets of M. An $\alpha^\star$-witness of B under P and S w.r.t. M is a directed acyclic graph $G = (\{(S_i, P_i) | 1 \leq i \leq n\}, E)$ where*

- *$\{S_i \mid 1 \leq i \leq n\}$ is a partitioning of B (i.e., $\bigcup_{i=1}^{n} S_i = B$, $S_i \cap S_j = \emptyset$ $(i \neq j)$ and $S_i \neq \emptyset$), and*
- *for every i, $1 \leq i \leq n$,*
  - *(i) $P_i \subseteq P$ is a witness of S_i under $S \cup X_i$ w.r.t. M,*
  - *(ii) P_i is not a witness of S_j under $S \cup X_i$ w.r.t. M, for every $1 \leq j \neq i \leq n$ and $(S_j, P_j) \notin D_G((S_i, P_i))$[4], where for each k $(1 \leq k \leq n)$,*

$$X_k = \bigcup \{S' \mid (S', P') \in D_G((S_k, P_k))\}.$$

If G induces a total order $(S_1, P_1) < (S_2, P_2) < \cdots < (S_n, P_n)$, i.e., $E = \{((S_i, P_i), (S_{i+1}, P_{i+1})) \mid 1 \leq i < n\}$, then we call it an α-witness of B under P and S w.r.t. M, and denote it as

$$W = [(S_1, P_1), \ldots, (S_n, P_n)].$$

[4]If $(S_j, P_j) \in D_G((S_i, P_i))$ then $S_j \subseteq X_i$ and $Q \cup X_i \vdash_{CA} S_j$ for any basic logic program Q.

We call G minimal*, if every P_i is minimal and G is* compact*, if in addition to minimality $P_i \cap P_j = \emptyset$ for all $1 \leq i < j \leq n$. If $B = M$ and $S = \emptyset$, we call G a (minimal, compact) $\alpha^\star$-witness resp. α-witness of M w.r.t. P.*

Note that Definition 5, requires $M \neq \emptyset$. This means that for an empty SPT-answer set (= $\emptyset$) of a logic program, no witness is defined; however, in this case no atom needs to be justified.

While $\alpha^\star$-witnesses are any directed acyclic graphs, α-witnesses are chains. Intuitively, for each node (S_i, P_i) of a witness, the atoms in S_i can be "derived" from P_i with the help of those "derived" atoms in X_i, that are on the paths of G to (S_i, P_i). In this sense P_i is a justification for why the atoms in S_i are in an SPT-answer set, *i.e.*, P_i is minimal such that $\mathrm{MR}(P_i, M) \cup X_i$ can "derive" the atoms in S_i.

Example 5. *Let us consider the logic program P in Example 4. The SPT-answer set $M = \{a, c\}$ of P has two $\alpha^\star$-witnesses:*

- *$G_1 = (V_1, \emptyset)$ with $V_1 = \{(\{a, c\}, \{r_1, r_3\})\}$;*
- *$G_2 = (\{v_1, v_2\}, \{(v_1, v_2)\})$ with $v_1 = (\{a\}, \{r_1\})$ and $v_2 = (\{c\}, \{r_3\})$.*

Notice that both G_1 and G_2 are α-witnesses since they can be written as $W_1 = [(\{a, c\}, \{r_1, r_3\})]$ and $W_2 = [(\{a\}, \{r_1\}), (\{c\}, \{r_3\})]$, respectively. It is easy to check that both G_1 and G_2 are compact.

The next proposition shows that every nonempty SPT-answer set of a basic logic program has a compact $\alpha^\star$-witness.

Proposition 4. *Let P be a basic program, $M \neq \emptyset$ be an SPT-answer set of P and $Q = MR(P, M)$. Then, the DAG $G = (\{(S_i, Q_i) \mid 1 \leq i \leq n\}, E')$ obtained from $\mathbb{S}_Q = (\{S_1, \ldots, S_n\}, E)$ such that*

- *$Q_i \in \mathrm{MW}(S_i, P, S_{\rhd i}, M)$, where $S_{\rhd i} = \bigcup D_{\mathbb{S}_Q}(S_i)$, $1 \leq i \leq n$, and*
- *$((S_i, Q_i), (S_j, Q_j)) \in E'$ if $(S_i, S_j) \in E$*

is a compact $\alpha^\star$-witness of M w.r.t. P.

The compact $\alpha^\star$-witness constructed in Proposition 4 is called the *full-split* $\alpha^\star$ ($\alpha^\star_{fs}$)-witness of M *w.r.t.* P. The proposition also implies that $D_{\mathbb{S}_G}(S_i)$ can be utilized to form different compact $\alpha^\star$-witnesses for M *w.r.t.* P. For example, if there is a path $((S_i, Q_i), (S_{i+1}, Q_{i+1}, \ldots, (S_k, Q_k))$ in the DAG $G = (V, E')$ of Proposition 4, then the DAG $G' = (V'', E'')$ with

- $V'' = V' \setminus \{(S_{i'}, Q_{i'}), \ldots, (S_{k'}, Q_{k'})\} \cup (\bigcup_{j=i'}^{k'} S_j, \bigcup_{j=i'}^{k'} Q_j)$ where $i \leq i' \leq k' \leq k$, and
- $((S_v, Q_v), (S_u, Q_u)) \in E''$ if $((S_v, Q_v), (S_u, Q_u)) \in E'$ and at most one of (S_u, Q_u) and (S_v, Q_v) is combined in V'', *i.e.*, $|\{u, v\} \cap \{i', \ldots, k'\}| \leq 1$

is a compact $\alpha^\star$-witnesses for M *w.r.t.* P.

5.2 β-witness and $\beta^\star$-witness

The notion of α-witness addresses the justifications for a group of atoms simultaneously. However, in many situations, including debugging, engineers are more concerned with the justification of individual atoms. This leads to the following more refined testimony.

Algorithm 3: MinBetaWitness(Π, B, S, M)

Input: A basic logic program Π, a nonempty SPT-answer set M of Π, and two disjoint subsets B, S of M
Output: A minimal β-witness of B under Π and S *w.r.t.* M

1 $T \leftarrow S$; $\Pi' \leftarrow [\,]$; $\Sigma \leftarrow \mathrm{MR}(\Pi, M)$;
2 **while** $B - T \neq \emptyset$ **do**
3 **while** $\exists r \in \Sigma$ *s.t.* $bd^+(r) \subseteq T$, $hd(r) = \{p\}$, $p \in M - T$ **do**
4 $T \leftarrow T \cup hd(r)$;
5 $\Pi'.append((p, rm(\{r\}, \Pi, M)))$;
6 **end**
7 **if** $M = T$ **then break** Let $u \in B - T$;
8 $\Sigma_u \leftarrow$ MinWitness$(\Sigma, \{u\}, T, M)$;
9 **while** $\Sigma_u \cup T \vdash v$ *for some* $v \in M - (\{u\} \cup T)$ **do**
10 $\Sigma_v \leftarrow$ MinWitness$(\Sigma_u, \{v\}, T, M)$;
11 $u \leftarrow v$;
12 $\Sigma_u \leftarrow \Sigma_v$;
13 **end**
14 $\Pi'.append((u, rm(\Sigma_u, \Pi, M)))$;
15 $T \leftarrow T \cup \{u\}$;
16 **end**
17 **return** Π';

Definition 6 ((minimal, compact) β^*- and β-witness)**.** *Let P be a basic logic program, $M \neq \emptyset$ an SPT-answer set of P, and B, S be two disjoint subsets of M.*

- *Every $\alpha^\star$-witness $G = (\{(S_i, P_i) | 1 \leq i \leq n\}, E)$ of B under P and S w.r.t. M such that $S_i = \{p_i\}$ for some atom p_i, $1 \leq i \leq n$, is a $\beta^\star$-witness of B under P and S w.r.t. M, also written as $G = (\{(p_i, P_i) | 1 \leq i \leq n\}, E)$.*
- *The notions of β-witness and minimal/compact $\beta^\star$ resp. β-witness of B under P and S w.r.t. M are defined analogously to α-witness and minimal/compact $\alpha^\star$- resp. α-witness of B under P and S w.r.t. M.*

In particular, if $B = M$ and $S = \emptyset$, then we call G a (minimal, compact) $\beta^\star$-witness, respectively, β-witness of M w.r.t. P.

Clearly β-witnesses and $\beta^\star$-witnesses are special cases of α-witnesses and $\alpha^\star$-witnesses, respectively. It has been shown that an answer set of a logic program containing merely elementary atoms may not have a compact β-witness, and that deciding whether some compact β-witness exists is Σ_2^p-complete [Wang *et al.*, 2023]. This motivates us to confine computation to minimal β-witnesses, which always exist.

Algorithm 3, MinBetaWitness, computes a minimal β-witness of B under a basic logic program P relative to S *w.r.t.* M. It uses a function $rm(\Sigma, \Pi, M)$ that returns, given basic logic programs Σ, Π and $M \subseteq \mathcal{A}$, a minimal subset Π' of Π such that $\mathrm{MR}(\Pi', M) = \Sigma$. The external **while-loop (line 2-17)** checks whether all atoms in M have a minimal witness. The first inner **while-loop** identifies the atoms by unit propagation, while the second inner **while-loop** searches for an atom u and its minimal witness Σ_u under T such that there is no atom v in $\{u\} \cup T$ that can be observed by $\Sigma_u \cup T$.

The next proposition shows the correctness of Algorithm 3. Thus, every nonempty SPT-answer set of a basic logic program has a minimal β-witness.

Proposition 5. *If M is an SPT-answer set of a basic logic program P then* MinBetaWitness *$(P, M, \emptyset, M)$ returns a minimal β-witness of M w.r.t. P.*

Example 6. *Let us consider the logic program P with aggregates from Example 14 in [Eiter and Geibinger, 2023], which consists of the following rules:*

$$r_1 : a \leftarrow \#sum\{1 : b, 2 : d\} > 2, \#sum\{3 : d, 2 : c\} < 5,$$
$$r_2 : d \leftarrow b, \qquad r_3 : b \vee c \leftarrow .$$

The formulas involving $\#sum$ are aggregate atoms of the form $\#sum\{w_i : l_i \mid 1 \leq i \leq n\} \preceq w$ where w_i and w are numbers, $\preceq$ is an arithmetic comparison operator and l_i is a (possibly negated) atom. An interpretation M satisfies such an aggregate atom if $(\sum_{i=1, M \models l_i}^{n} w_i) \preceq w$.

Hence the aggregate atoms $\#sum\{1 : b, 2 : d\} > 2$ and $\#sum\{3 : d, 2 : c\} < 5$ can be modeled as abstract constraint atoms $A = (\{b, d\}, \{\{b, d\}\})$ and $B = (\{c, d\}, \{\emptyset, \{c\}, \{d\}\})$, respectively. Thus, rule r_1 can be represented as

$$r_1' : a \leftarrow A, B.$$

It is not hard to check that $M = \{a, b, d\}$ is an SPT-answer set of P. We note that $MR(B, M) = \top$ as $2^{\{c,d\} \cap M} = \{S \cap M \mid S \in \{\emptyset, \{c\}, \{d\}\}\} = \{\emptyset, \{d\}\}$. Thus, $MR(\{r_1', r_2, r_3\}, M)$ consists of following rules, where $A' = (\{b, d\}, \{\{b, d\}\})$:

$$r_1'' : a \leftarrow A', \qquad r_2' : d \leftarrow b, \qquad r_3' : b \leftarrow .$$

Furthermore, M has a unique compact β-witness, namely:

$$W = [(b, \{r_3\}), (d, \{r_2\}), (a, \{r_1'\})].$$

Informally, $b \in M$ is justified by rule r_3, as $MR(\{r_3\}, M) = \{r_3'\}$ (minimally) "derives" b. Then, with the help of b, $MR(\{r_2\}, M) = \{r_2'\}$ (minimally) "derives" d. Finally, with the help of b and d, $MR(\{r_1\}, M)$ minimally "derives" a.

6 Related Work

There have been various notions for explaining answer sets of logic programs and some systems have been available. The difference between witness-based explanations, off-line justifications, and causal graph based ones have been discussed in [Wang *et al.*, 2023]. To our best knowledge, only r-justification [Eiter and Geibinger, 2023] can deal with abstract constraint atoms. We firstly recall some basic notations and notions about r-justification.

A *partial interpretation* is a tuple $I = \langle I^+, I^- \rangle$, where $I^+, I^- \subseteq \mathcal{A}$ and $I^+ \cap I^- = \emptyset$; it is *total on* a set S of atoms if $S \subseteq I^+ \cup I^-$. For two partial interpretations J_1, J_2, we denote by $J_1 \leq J_2$ that $J_1^+ \subseteq J_2^+$ and $J_1^- \subseteq J_2^-$, and by $J_1 < J_2$ that $J_1 \leq J_2$ and $J_1 \neq J_2$. A partial interpretation $I = \langle I^+, I^- \rangle$ *satisfies* a c-atom A if $I^+ \cap A_d \in A_c$ and, for each $U \in \overline{A_c}$ with $I^+ \cap A_d \subset U$, $U \cap I^- \neq \emptyset$ holds. The latter intuitively means that I^+ cannot be extended (under the restriction of I^-) to falsify A. In other words, $U \in A_c$ for every U *s.t.* $A_d \cap I^+ \subseteq U \subseteq A_d \setminus I^-$. If I is total (on A_d), then $I \models A$ whenever $I^+ \cap A_d \in A_c$.

Definition 7 (Presumptuous Entailment)**.** *Let P be a program, J be a partial interpretation and A be a c-atom. Then $P \models_J A$ if every total model I of P s.t. $I \geq J$ fulfills $I \models A$.*

Definition 8 (Failed Support)**.** *Let r be a rule and I be a total model of r. Then, r is a* failed support *for atom a w.r.t. I if $I \not\models bd(r)$ and for every I', $I' \models a$ if $I' \models r$ and $I' \models bd(r)$.*

Definition 9 (r-justification)**.** *Let P be a program, I be an answer set of P and a be an atom. Then, a triple (a°, Q, J), where $\circ \in \{+, -\}$, $Q \subseteq P$ is a set of rules, and $J \leq I$, is a partial interpretation, is an* r-justification *for a w.r.t. P and I if the following conditions hold:*

(a) If $a \in I$, then $\circ = +$, $Q^I \models_J a$ and there is no $R \subset Q$ such that $R^I \models_J a$.

(b) If $a \notin I$, then $\circ = -$, $Q = \{r \in P \mid r$ is a failed support of a w.r.t. $I\}$, and for every $r \in Q$, $J \models \overline{A}$ for some $A \in bd(r)$.

The r-justification is *concise* if J is $\leq$-minimal.

Definition 10 (r-justification chain)**.** *A sequence $\mathcal{J} = (a_1^\circ, Q_1, J_1), \cdots, (a_n^\circ, Q_n, J_n)$ of r-justifications w.r.t. P and I is an* r-justification chain *for a_1, if for every (a_i°, Q_i, J_i) $(1 \leq i \leq n)$, it holds that*

(a) for each $a \in J_i^+$ some (a_j^+, Q_j, J_j) exists such that $a_j = a$ and either $j > i$ or $a_j = a_1$ and there is no subsequence $(a_{k_1}^+, Q_{k1}, J_{k_1}), \cdots, (a_{k_\ell}^+, Q_{k_\ell}, J_{k_\ell})$ of $\mathcal{J}$ where $k_1 = 1$, $k_\ell = i$, and $a_{k_{h+1}} \in J_{k_h}^+$, for all $1 < h < \ell$;

(b) for each $a \in J_i^-$ some (a_j^-, Q_j, J_j) exists s.t. $a_j = a$;

(c) if $i > 1$, then some (a_j°, Q_j, J_j) exists such that $a_i \in J_j^\circ$ and $j < i$; and

(d) $a_i = a_j$ for $1 \leq j \leq n$ implies $i = j$.

It was shown that, for every logic program P and every FLP-answer I of P, every atom in I has an r-justification chain *w.r.t.* P and I, cf. Theorem 4 of [Eiter and Geibinger, 2023].

The next example shows some subtle difference between r-justification and witness.

Example 7. *Let us consider the basic logic program P consisting of the following rules, where $A = (\{b, a\}, \{\emptyset, \{b, a\}\})$:*[5]

$$r_1 : a \leftarrow b, \quad r_2 : b \leftarrow a, \quad r_3 : a \leftarrow A, \quad r_4 : a \vee b.$$

It is not difficult to verify that $I = \{a, b\}$ is the unique SPT-answer set of P. Note that $A_c^* = \{\emptyset \uplus \emptyset, \{a, b\} \uplus \emptyset\}$ and MR(P, I) consists of

$$\alpha_1 : a \leftarrow b, \quad \alpha_2 : b \leftarrow a,$$
$$\alpha_3 : a \leftarrow (\{b, a\}, \{\{b, a\}\}), \quad \alpha_4 : a \vee b.$$

Clearly $\{\alpha_1, \alpha_4\} \vdash_{CA} a$ and $\{\alpha_2, a\} \vdash_{CA} b$. Thus, we have the following compact β-witness W for I w.r.t. P:

$$W = [(a, \{r_1, r_4\}), (b, \{r_2\})]. \tag{6}$$

From this β-witness and the fact that the total models of $\{r_1, r_4\}$ that extend the partial interpretation $\langle \emptyset, \emptyset \rangle$ are $\langle \{a\}, \{b\} \rangle$ and $\langle \{a, b\}, \emptyset \rangle$, we obtain the following r-justification chain $\mathcal{J}$ for b:

$$\mathcal{J} = (b^+, \{r_2\}, \langle \{a\}, \emptyset \rangle), (a^+, \{r_1, r_4\}, \langle \emptyset, \emptyset \rangle).$$

Note that the FLP-reduct P^I of P w.r.t. I is P. The total models of $\{r_1, r_3\}$ are $\langle \{a\}, \{b\} \rangle$ and $\langle \{a, b\}, \emptyset \rangle$, i.e., the classical interpretations $\{a\}$ an $\{a, b\}$. It is tedious but not difficult to check that

$$(a^+, \{r_1, r_3\}, \langle \emptyset, \emptyset \rangle) \tag{7}$$

is an r-justification for a w.r.t. P and I since $\{r_1, r_3\}$ is (subset) minimal s.t. $\{r_1, r_3\}^I = \{r_1, r_3\}$ and $\{r_1, r_3\}^I \models_{\langle \emptyset, \emptyset \rangle} a$ (every total model of $\{r_1, r_3\}$ that extends the partial interpretation $\langle \emptyset, \emptyset \rangle$ satisfies a). The r-justification chain (7) is indeed a chain. It explains why a is in the FLP-answer set by the argument that every total model of $\{r_1, r_3\}$ that extends $\langle \emptyset, \emptyset \rangle$ satisfies a. Due to the fact that MR$(\{r_1, r_3\}, I) = \{\alpha_1, \alpha_3\}$, we have MR$(\{r_1, r_3\}, I) \not\vdash_{CA} a$; hence, this r-justification for a cannot be mapped to a witness for a. For the same reason, the following r-justification chain

$$\mathcal{J}' = (b^+, \{r_2\}, \langle \{a\}, \emptyset \rangle), (a^+, \{r_1, r_3\}, \langle \emptyset, \emptyset \rangle)$$

for b cannot be mapped to a witness for b yet. Since $\{r_1, r_4\}$ can derive a by classical resolution, while $\{r_1, r_3\}$ cannot, the justification chain $\mathcal{J}$ is more intuitive than $\mathcal{J}'$.

The next proposition shows that every minimal witness can capture an r-justification. Its converse does not hold, as shown by Example 7.

Proposition 6. *Let P be a basic logic program, I be an SPT-answer set of P, $S \subseteq I$, and $p \in I \setminus S$. If $Q \in$ MW$(\{p\}, P, S, I)$, then $(p^+, Q, \langle S, \overline{I} \rangle)$ is an r-justification for p w.r.t. P and I.*

Hence, when we consider r-justifications under SPT-answer set semantics, the minimal witness provides a more selective and intuitive way to chose rules as justifications as seen the $\mathcal{J}$ and $\mathcal{J}'$ in Example 7.

7 Conclusion

For logic programs with abstract constraint atoms, we have proposed the notions of minimal reduct, the resolution proof, and witness for SPT-answer sets. We showed that this minimal reduct provides a new characterization for SPT-answer sets of basic logic programs. It also facilitates an extended resolution for atoms in SPT-answer sets. Thus, an explanation in terms of "proof" can be obtained for the SPT-answer sets.

It is well-known that the problem of finding shorter resolution proofs in propositional logic is intractable [Haken, 1985]. To construct short CA-resolution-based explanations for SPT-answer sets is a challenge, but it deserves further investigation. In addition, we will analyze the computational complexity of (compact) α-witnesses and β-witnesses.

Acknowledgments

This research has been partially supported by the National Natural Science Foundation of P.R. China under grants 62376066 and 61976065 and was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12.

[5] This logic program is similar to the one $P_7 \cup P_8$ in Example 13 of [Eiter and Geibinger, 2023]

References

[Alviano *et al.*, 2023] Mario Alviano, Wolfgang Faber, and Martin Gebser. Aggregate semantics for propositional answer set programs. *Theory and Practice of Logic Programming*, 23(1):157–194, January 2023.

[Alviano *et al.*, 2024] Mario Alviano, Ly Ly T. Trieu, Tran Cao Son, and Marcello Balduccini. The XAI system for answer set programming xasp2. *J. Log. Comput.*, 34(8):1500–1525, 2024.

[Angiulli *et al.*, 2022] Fabrizio Angiulli, Rachel Ben-Eliyahu-Zohary, Fabio Fassetti, and Luigi Palopoli. Graph-based construction of minimal models. *Artificial Intelligence*, 313:103754, 2022.

[Arias *et al.*, 2020] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. Justifications for goal-directed constraint answer set programming. In Francesco Ricca, Alessandra Russo, Sergio Greco, Nicola Leone, Alexander Artikis, Gerhard Friedrich, Paul Fodor, Angelika Kimmig, Francesca A. Lisi, Marco Maratea, Alessandra Mileo, and Fabrizio Riguzzi, editors, *Proceedings 36th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2020, (Technical Communications) UNICAL, Rende (CS), Italy, 18-24th September 2020*, volume 325 of *EPTCS*, pages 59–72, 2020.

[Arioua *et al.*, 2015] Abdallah Arioua, Nouredine Tamani, and Madalina Croitoru. Query answering explanation in inconsistent datalog$$+/-$$knowledge bases. In Qiming Chen, Abdelkader Hameurlain, Farouk Toumani, Roland Wagner, and Hendrik Decker, editors, *Database and Expert Systems Applications*, pages 203–219, Cham, 2015. Springer International Publishing.

[Baral, 2003] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, New York, NY, 2003.

[Ben-Eliyahu-Zohary *et al.*, 2017] Rachel Ben-Eliyahu-Zohary, Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Modular construction of minimal models. In *Logic Programming and Nonmonotonic Reasoning - 14th International Conference*, pages 43–48, Espoo, Finland, July 2017. Springer.

[Bienvenu *et al.*, 2019] Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Computing and explaining query answers over inconsistent dl-lite knowledge bases. *Journal of Artificial Intelligence Research*, 64:563–644, 2019.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Brewka *et al.*, 2016] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming: An introduction to the special issue. *AI Mag.*, 37(3):5–6, 2016.

[Burkart and Huber, 2021] Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[Cabalar and Fandinno, 2016] Pedro Cabalar and Jorge Fandinno. Justifications for programs with disjunctive and causal-choice rules. *Theory and Practice of Logic Programming*, 16(5-6):587–603, 2016.

[Cabalar *et al.*, 2020] Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. A system for explainable answer set programming. In *Proceedings 36th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2020, (Technical Communications) UNICAL, Rende (CS), Italy, 18-24th September 2020*, pages 124–136, 2020.

[Cheney *et al.*, 2009] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474, 2009.

[Dazeley *et al.*, 2021] Richard Dazeley, Peter Vamplew, Cameron Foale, Charlotte Young, Sunil Aryal, and Francisco Cruz. Levels of explainable artificial intelligence for human-aligned conversational explanations. *Artificial Intelligence*, 299:103525, 2021.

[Dell'Armi *et al.*, 2003] Tina Dell'Armi, Wolfgang Faber, Giuseppe Ielpa, Nicola Leone, and Gerald Pfeifer. Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in DLV. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 847–852. Morgan Kaufmann, 2003.

[Eiter and Geibinger, 2023] Thomas Eiter and Tobias Geibinger. Explaining answer-set programs with abstract constraint atoms. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3193–3202. ijcai.org, 2023.

[Faber *et al.*, 2004] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In José Júlio Alferes and João Alexandre Leite, editors, *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings*, volume 3229 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2004.

[Fandinno and Schulz, 2019] Jorge Fandinno and Claudia Schulz. Answering the "why" in answer set programming - A survey of explanation approaches. *Theory and Practice of Logic Programming*, 19(2):114–203, 2019.

[Ferraris, 2011] Paolo Ferraris. Logic programs with propositional connectives and aggregates. *ACM Trans. Comput. Log.*, 12(4):25:1–25:40, 2011.

[Gebser *et al.*, 2012] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen,

editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, pages 1070–1080, Seattle, Washington, USA, 1988. MIT Press.

[Gelfond and Zhang, 2019] Michael Gelfond and Yuanlin Zhang. Vicious circle principle, aggregates, and formation of sets in ASP based languages. *Artificial Intelligence*, 275:28–77, 2019.

[Haken, 1985] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[Leone *et al.*, 1997] Nicola Leone, Pasquale Rullo, and Francesco Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. *Information and Computation*, 135(2):69–112, 1997.

[Li *et al.*, 2021] Fang Li, Huaduo Wang, Kinjal Basu, Elmer Salazar, and Gopal Gupta. Discasp: A graph-based ASP system for finding relevant consistent concepts with applications to conversational socialbots. In Andrea Formisano, Yanhong Annie Liu, Bart Bogaerts, Alex Brik, Verónica Dahl, Carmine Dodaro, Paul Fodor, Gian Luca Pozzato, Joost Vennekens, and Neng-Fa Zhou, editors, *Proceedings 37th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2021, Porto (virtual event), 20-27th September 2021*, volume 345 of *EPTCS*, pages 205–218, 2021.

[Lifschitz, 2019] Vladimir Lifschitz. *Answer Set Programming*. Springer, 2019.

[Lin and You, 2002] Fangzhen Lin and Jia-Huai You. Abduction in logic programming: A new definition and an abductive procedure based on rewriting. *Artificial Intelligence*, 140(1/2):175–205, 2002.

[Liu *et al.*, 2010] Lengning Liu, Enrico Pontelli, Tran Cao Son, and Miroslaw Truszczyński. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence*, 174:295–315, 2010.

[Marek and Truszczynski, 2004] Victor W. Marek and Miroslaw Truszczynski. Logic programs with abstract constraint atoms. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 86–91, California, USA, September 2004. AAAI Press / The MIT Press.

[Marques-Silva and Huang, 2024] João Marques-Silva and Xuanxiang Huang. Explainability is *Not* a game. *Commun. ACM*, 67(7):66–75, 2024.

[Peñaloza, 2020] Rafael Peñaloza. Axiom pinpointing. In Giuseppe Cota, Marilena Daquino, and Gian Luca Pozzato, editors, *Applications and Practices in Ontology Design, Extraction, and Reasoning*, volume 49 of *Studies on the Semantic Web*, pages 162–177. IOS Press, 2020.

[Pontelli *et al.*, 2009] Enrico Pontelli, Tran Cao Son, and Omar El-Khatib. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming*, 9(1):1–56, 2009.

[Shen *et al.*, 2009] Yi-Dong Shen, Jia-Huai You, and Li-Yan Yuan. Characterizations of stable model semantics for logic programs with arbitrary constraint atoms. *Theory and Practice of Logic Programming*, 9(4):529–564, 2009.

[Shen *et al.*, 2014] Yi-Dong Shen, Kewen Wang, Thomas Eiter, Michael Fink, Christoph Redl, Thomas Krennwallner, and Jun Deng. FLP answer set semantics without circular justifications for general logic programs. *Artificial Intelligence*, 213:1–41, 2014.

[Simons *et al.*, 2002] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.

[Son *et al.*, 2007] Tran Cao Son, Enrico Pontelli, and Phan Huy Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research*, 29:353–389, 2007.

[Sosa, 2019] Ernest Sosa. *Knowledge and Justification*, chapter 15, pages 220–228. John Wiley & Sons, Ltd, 2019.

[Wang *et al.*, 2023] Yisong Wang, Thomas Eiter, Yuanlin Zhang, and Fangzhen Lin. Witnesses for answer sets of logic programs. *ACM Transactions on Computational Logic*, 24(2):15:1–15:46, 2023.

[Weber *et al.*, 2024] Rosina O. Weber, Adam J. Johs, Prateek Goel, and João Marques-Silva. XAI is in trouble. *AI Mag.*, 45(3):300–316, 2024.

[İsmail İlkan Ceylan *et al.*, 2025] İsmail İlkan Ceylan, Thomas Lukasiewicz, Enrico Malizia, and Andrius Vaicenavičius. Explanations for query answers under existential rules. *Artificial Intelligence*, 341:104294, 2025.