INSTITUT FÜR LOGIC AND COMPUTATION

FACHBEREICH WISSENSBASIERTE SYSTEME

WITNESSES FOR ANSWER SETS OF LOGIC PROGRAMS

YISONG WANG THOMAS EITER
YUANLIN ZHANG FANGZHEN LIN

LOGCOMP RESEARCH REPORT 22-01

OCTOBER 2022

LOGCOMP RESEARCH REPORT
LOGCOMP RESEARCH REPORT 22-01, OCTOBER 2022

WITNESSES FOR ANSWER SETS OF LOGIC PROGRAMS

Yisong Wang[1] Thomas Eiter[2] Yuanlin Zhang[3] Fangzhen Lin[4]

Abstract. In this paper, we consider Answer Set Programming (ASP). It is a declarative problem solving paradigm that can be used to encode a problem as a logic program whose answer sets correspond to the solutions of the problem. It has been widely applied in various domains in AI and beyond. Given that answer sets are supposed to yield solutions to the original problem, the question of "why a set of atoms is an answer set" becomes important for both semantics understanding and program debugging. It has been well investigated for normal logic programs. However, for the class of disjunctive logic programs, which is a substantial extension of that of normal logic programs, this question has not been addressed much. In this paper, we propose a notion of reduct for disjunctive logic programs and show how it can provide answers to the aforementioned question. First of all, we show that for each answer set, its reduct provides a resolution proof for each atom in it. We then further consider minimal sets of rules that will be sufficient to provide resolution proofs for sets of atoms. Such sets of rules will be called witnesses and are the focus of this paper. We study complexity issues of computing various witnesses and provide algorithms for computing them. In particular, we show that the problem is tractable for normal and headcycle-free disjunctive logic programs, but intractable for general disjunctive logic programs. We also conducted some experiments and found that for many well-known ASP and SAT benchmarks, computing a minimal witness for an atom of an answer set is often feasible.

[1]Department of Computer Science and Technology, Guizhou University; Institute for Artificial Intelligence, Guizhou University, China; email: yswang@gzu.edu.cn
[2]Institute of Logic and Computation, TU Wien, Austria; email: eiter@kr.tuwien.ac.at.
[3]Texas Tech University, Lubbock, Texas, USA; email: y.zhang@ttu.edu
[4]Hong Kong University of Science and Technology, Hong Kong; email: flin@cs.ust.hk.

This report is a preprint version of an article to appear in *ACM Transactions on Computational Logic*.

Contents

1 Introduction

Interpretability/explainability and explanation convey safety and trust in the 'how' and 'why' of decision-making by a system to users to increase transparency of the system Hempel and Oppenheim [1948]; Miller [2019]; Srinivasan and Chander [2020]; Confalonieri *et al.* [2021]. They play an important role in medicine, justice, and artificial intelligence (AI) in particular Almada [2019]; Mittelstadt *et al.* [2019]; Costabello *et al.* [2019]; Dazeley *et al.* [2021]; Bochman [2021]. Such explanation tasks in AI include justifying autonomous agent behaviour, debugging of machine learning models, explaining medical decision-making, and explaining predictions of classifiers Burkart and Huber [2021]. From the perspective of knowledge representation, an explanation for a solution may be equally or more important than the solution itself Pollock [1974]; Lin and Shoham [1992]; Sosa [2019]. For logic programming under answer sets semantics (ASP), which is a declarative problem solving paradigm Brewka *et al.* [2011], an explanation for an answer set amounts to answer "why a set of atoms is an answer set of a logic program" in a well-justified manner, instead of just by definition. In other words, every atom in an answer set of a logic program must be founded or justified by a given derivation. It provides deep insights into ASP semantics that are useful for debugging and various crucial scenarios in which ASP is applied, including health and life sciences Erdem *et al.* [2011] and nuclear engineering Hanna *et al.* [2020]. These justifications are usually not provided in the semantics itself (note exceptions such as causal stable models Cabalar and Fandinno [2016]).

This challenging task has been widely explored Fandinno and Schulz [2019]; Arias *et al.* [2020] in terms of off-line justification for normal logic programs which uses an explanation graph to describe the "reason" for the truth value of an atom *w.r.t.* a given answer set Pontelli *et al.* [2009], attack trees in argumentation theory for extended normal logic programs Schulz and Toni [2016], self-justified $\triangleright$-computation for logic programs with abstract constraint atoms Liu *et al.* [2010], causal graphs and causal stable models Cabalar *et al.* [2014]; Cabalar and Fandinno [2016]; Cabalar *et al.* [2020] for (disjunctive) logic programs. Explanations for why a logic program has no answer set were also recently explored from the perspective of inconsistency proofs Alviano *et al.* [2019] and unsatisfiable cores Alviano and Dodaro [2020]. The notion of justification was also investigated in terms of abstract logical or algebraic frameworks for normal logic programs Denecker *et al.* [2000, 2015].

For normal logic programs, explanations of answer sets are in our view rather easy and intuitive to obtain. The reason is that every atom p in an answer set M of a logic program Π has a step-by-step construction from the GL-reduct Π^M Gelfond and Lifschitz [1988] in terms of the *immediate consequence operator* T_{Π^M}, i.e., the canonical one-step derivation operator for the logic program Π^M van Emden and Kowalski [1976]. This step-by-step construction properly plays a role of "explanation". For instance, considering the answer set $M = \{a, c\}$ of the logic program $\Pi = \{a \leftarrow not\ b,\quad b \leftarrow not\ a,\quad c \leftarrow a,\quad c \leftarrow b\}$, the step-by-step construction for $c \in M$ is $T_{\Pi^M}(\emptyset) = \{a\}$ and $T_{\Pi^M}(\{a\}) = \{a, c\}$, from which the explanation for $c \in M$ can be extracted:

(1) The atom a is first justified by the rule a (called a *fact*) from Π^M;

(2) The atom c is then justified by the rule $c \leftarrow a$ from Π^M and the justified atom a.

For disjunctive logic programs, the self-justified $\triangleright$-computation and attack trees do not work while causal stable models are not constructive. To wit, consider the logic program $\Pi = \{r_1 : a \vee b,\quad r_2 : a \leftarrow b,\quad r_3 : b \leftarrow a\}$, which has the unique answer set $M = \{a, b\}$. According to Cabalar and Fandinno [2016], the logic program Π has two causal stable models I and I' that informally assign each atom that is **true** a derivation in terms of a sequence of rule applications:

$$I(a) = r_1^a, I(b) = r_1^a \cdot r_3; \quad I'(a) = r_1^b \cdot r_2, I'(b) = r_1^b.$$

According to I, the atom a is a non-deterministic effect of the disjunctive rule r_1 by the notation r_1^a, and b is derived from a by r_1^a through r_3. Analogously, I' makes b **true** because of the non-deterministic choice for b from r_1 (in the notation r_1^b) and then obtains a from the chosen b through r_2.

Note that the unique answer set of the Π is $\{a,b\}$. Both a and b are indeed logical consequences of the logic program Π when interpreting '$\leftarrow$' as material implication. In this sense, a and b must be in every model of the logic program, and an explanation for a can be any logical proof for a from Π, which is different from "a non-deterministic effect of disjunction". In fact, a resolution proof for a is immediate by r_1 and r_2, and similarly one for b by r_1 and r_3. These explanations in terms of resolution proof are constructive.

For normal logic programs, such explanations by resolution proofs are readily available as we have illustrated: an answer set M of a normal logic program Π is the least model of the GL-reduct Π^M, which is a Horn logic program; this least model can be efficiently constructed by unit resolution, implementing the *immediate consequence operator*. Thus, unit resolution proofs from Π^M for the atoms in M play a proper role of an explanation for M. This unit resolution proof is also at hand for answer sets of headcycle-free disjunctive logic programs Ben-Eliyahu and Dechter [1994].

Fortunately, we will show that such explanations in terms of resolution proofs exist for disjunctive logic programs as well if a *new reduct* is applied. Resolution proofs can be found using theorem provers, but this is intractable in various settings Haken [1985]; Chvátal and Szemerédi [1988]; Fellows *et al.* [2006]. According to the Occam's razor principle, it is usually desirable that a derivation step is involved in an explanation only if it is necessary for the explanation, and the overall proof involves no redundant clauses to alleviate the burden of interpretation for the user. For this purpose, we are interested in which rules (called *witnesses*) are (minimally) sufficient to build up an explanation.

The main contributions of this paper are briefly summarized as follows.

1. We propose a new notion of *reduct* which replaces every atom that is assumed to be **false** by **false** in addition to the GL-reduct Gelfond and Lifschitz [1988]. This reduct is slightly different from the Ferraris-reduct Ferraris [2005] , which replaces every formula evaluated to be **false** by **false**. It provides a new characterization for answer sets of logic programs, *i.e.*, a set M of atoms is an answer set of a logic program Π whenever M is the least model of this reduct of Π *w.r.t.* M, and achieves explanations for answer sets via resolution proofs. As a result, we expand the minimal model decomposition theorem of Ben-Eliyahu-Zohary *et al.* in Ben-Eliyahu-Zohary *et al.* [2016]. In particular, we show that their sound decomposition algorithm $\mathsf{CheckMin}(\Sigma, M)$ for checking whether a model M of a clause theory Σ is a minimal model of Σ is complete for clause theories in the reduced form; as the latter is easily computed and preserves minimal models of Σ that are subsets of M, the algorithm can thus with a minor modification be used for modular minimal model checking for arbitrary clause theories. Along the same line, this result extends to answer set checking for disjunctive logic programs.

2. We introduce the notions of α-witness and β-witness and variants thereof, viz. *minimal, compact, full-split*, $\alpha^\star$- and $\beta^\star$-witnesses respectively, from which explanations for answer sets can be built up in a stepwise manner. While α-witnesses are for sets of atoms, β-witnesses are more fine-grained and target a single atom at a time. Roughly speaking, an α-witness is a sequence $[(S_1,\Pi_1),(S_2,\Pi_2),\ldots,(S_n,\Pi_n)]$ of pairs (S_i,Π_i) of sets S_i of atoms and rules Π_i from a logic program Π where the S_i forms a partitioning of an answer set M such that each S_i can be derived from the reduct of Π_i *w.r.t.* M and all its ancestors S_j's. A β-witness is a sequence $[(p_1,\Pi_1),(p_2,\Pi_2),\ldots,(p_n,\Pi_n)]$ of pairs (p_i,Π_i) of an atom p_i and a set Π_i of rules from Π, where $M = \{p_1,\ldots,p_n\}$ is an answer set of Π, such that

$p_1, \dots, p_{i-1}$ plus the reduct of Π_i *w.r.t.* M allows to derive p_i. Minimality refers to the subset-minimality of each Π_i, and compactness to the additional property that all Π_i are pairwise disjoint. The * variants impose further restrictions to reflect the dependency among its components (S_i, Π_i)s in terms of a directed graph, while for β-witnesses, a partial ordering of the atoms p_i similar as for α-witnesses is requested. While computing a compact α-witness is tractable, the β-witness allows to focus on some S_i in the α-witness. Thus, α- and β-witnesses can be combined to form a flexible explanation: a low level (atomic) explanation for the atoms in S_i and a high level (collection of atoms) explanation for S_j $(j \neq i)$.

3. We present methods and algorithms for computing α- and β-witnesses and their variants, and we study complexity issues in this context. In particular, we show that every answer set of a logic program has a proper justification in terms of each of the above witness notions, with the exception of compact β- and $\beta^\star$-witnesses; deciding whether such a witness exists turns out to be Σ_2^p-complete. We further prove that the recognition problem for all these witnesses is intractable in general, and is in fact for most of them D_1^p-complete, and thus only mildly harder than NP and co-NP. Furthermore, we consider also computing some witness for selected notions, and relate this problem to computing some minimal unsatisfiable set (MUS) of a given (unsatisfiable) clause theory, which has been studied in many works, cf. Silva [2010]; Janota and Marques-Silva [2016]. In particular, computing some full-split $\alpha^\star$-witness turns out to be equivalent to computing some MUS under log-space reductions. Furthermore, computing some MUS can be reduced in log-space to computing some minimal β-witness resp. minimal $\beta^\star$-witness; the converse is not known. Since computing some MUS is $\mathrm{FP}_{\|}^{\mathrm{NP}}$-hard Chen and Toda [1995]; Janota and Marques-Silva [2016], all of these problems are thus $\mathrm{FP}_{\|}^{\mathrm{NP}}$-hard as well. Notably, deciding whether a minimal β-witness may start with a particular atom p (i.e., $p_1 = p$) is Σ_2^p-complete and thus even harder than computing some minimal β-witness. On the other hand, by known results for MUS and our algorithms, we obtain that computing some full-split $\alpha^\star$-witness and some minimal $\beta^\star$-witness is feasible in polynomial time with an NP oracle. Furthermore, the complexity is tractable for all problems when the logic program Π at hand is normal, *i.e.*, has no disjunction in rule heads, or headcycle-free Ben-Eliyahu and Dechter [1994].

4. We profile the algorithms with experiments on well-known ASP and SAT benchmarks. In particular, we tested two ASP benchmarks, namely minimal diagnosis and strategic companies and five industrial SAT competition benchmarks of moderate size, in addition to random and handcrafted CNFs and random disjunctive logic programs. The ASP benchmarks are beyond NP complexity of deciding the existence of an answer set and involve disjunction, but without any aggregations or optimization. In these experiments, finding minimal β-witnesses is very often feasible under resource constraints, and an explanation (resolution proof) is quite clear from a minimal β-witness in most cases since only one rule is involved. Thus, any such witness provides an easy-to-read proof for the stepwise construction of an answer set of the logic programs that we encountered.

Organization

The rest of the paper is organized as follows. In Section 2, we briefly review necessary concepts and notions of answer set programming, as well as the notion of minimal model decomposition for clause theories Ben-Eliyahu-Zohary *et al.* [2016]; Ben-Eliyahu-Zohary *et al.* [2017]. After that we introduce in Section 3 the notion of reduct and investigate some of its properties. We then present the notions of α- and β-witnesses and variants for answer sets of logic programs in Section 4, where we then also discuss methods and algorithms for computation. This is followed by addressing complexity issues for the various notions of witnesses in Section 5. We report about and analyze the experimental evaluation of the proposed algorithms in Section 6. In Section 7, we discuss related work where we make a more detailed comparison with causal stable models and off-line justification. The final Section 8 concludes the paper with a brief summary and remarks on extensions of the results as well as future work. In order not to distract from the flow of reading, all detailed technical proofs have been moved to the electronic Appendix.

2 Preliminaries

We assume an underlying propositional language $\mathscr{L}$ over a finite set $\mathscr{A}$ of *atoms* together with two additional propositional constants $\bot$ (falsity) and $\top$ (tautology). The notions of literal, clause, formula, (clause) theory, interpretation, satisfiability, model, logical consequence ($\models$) and equivalence ($\equiv$) of $\mathscr{L}$ are defined as usual. For $S \subseteq \mathscr{A}$, we let $\neg S = \{\neg p \mid p \in S\}$, $not\ S = \{not\ p \mid p \in S\}$ and $not\ not\ S = \{not\ not\ p \mid p \in S\}$; moreover, for a finite set L of literals, we use the notation $\bigvee L = \bigvee_{\ell \in L} \ell$ and $\bigwedge L = \bigwedge_{\ell \in L} \ell$. A clause α is usually represented as a set of literals, and we let $\alpha^+ = \alpha \cap \mathscr{A}$ and $\alpha^- = \{p \mid \neg p \in \alpha\}$. By $\Sigma \models_{min} \alpha$ we denote that $\Sigma \models \alpha$ and no $\Sigma' \subset \Sigma$ satisfies $\Sigma' \models \alpha$. By $\mathrm{var}(e)$ we mean the set of atoms occurring in the expression e. For simplicity, a singleton set $\{e\}$ is usually abbreviated as e when it is clear from its context.

2.1 General logic programs

A *(general) logic program* (or an *answer set program*) Π is a finite set of *(general) rules* r Gelfond and Lifschitz [1991]; Lifschitz *et al.* [1999] of the form

$$p_1 \vee \cdots \vee p_k \leftarrow p_{k+1}, \cdots, p_m, not\ p_{m+1}, \cdots, not\ p_n, not\ not\ p_{n+1}, \cdots, not\ not\ p_l \tag{1}$$

where $p_i \in \mathscr{A}$, $1 \leq i \leq l$. We let $r^+ = \{p_1, \ldots, p_k\}$, $r^- = \{p_{k+1}, \ldots, p_m\}$ and $r^{not} = \{p_{m+1}, \ldots, p_n\}$, $r^{2not} = \{p_{n+1}, \ldots, p_l\}$, $hd(r) = p_1 \vee \cdots \vee p_k$ and $bd(r) = r^- \cup not\ r^{not} \cup not\ not\ r^{2not}$. We also write r as

$$hd(r) \leftarrow bd(r) \quad \text{respectively} \quad r^+ \leftarrow r^- \cup not\ r^{not} \cup not\ not\ r^{2not}. \tag{2}$$

The notation $hd(r)$ (resp. $bd(r)$) refers to the *head* (resp. *body*) of r. Intuitively, r^+ (resp. r^-) means the set of atoms that occur in the rule r positively (resp. negatively) when viewing the rule containing no *n*ot as a clause, while r^{not} (resp. r^{2not}) stands for these occurrences of atoms that are in the scope of single (resp. double) negation. The rule r is *nesting-free* or *disjunctive* if $n = l$. The nesting-free rule r is *positive* (resp. *normal*, a *constraint*) if $n = m$ (resp. $k \leq 1$, $k = 0$). A positive rule r is *Horn* (resp. *definite*), if $|r^+| \leq 1$ (resp. $|r^+| = 1$). A logic program Π is a *positive* (resp., *nesting-free* or *disjunctive*, *normal, Horn, definite) (logic) program* if every rule in Π is so.

Example 2.1. *Let Π be the logic program consisting of the following three rules: $r_1 : a \vee b \leftarrow not\ not\ c$; $r_2 : c \leftarrow not\ a$; $r_3 : b \leftarrow a$. Note that $r_1^+ = \{a,b\}$, $r_1^- = r_1^{not} = \emptyset$, $r_1^{2not} = \{c\}$, $r_2^+ = \{c\}$, $r_2^{not} = \{a\}$, $r_3^+ = \{b\}$ and $r_3^- = \{a\}$. The rule r_1 is not nesting-free since it involves 'not not'. The rule r_2 is normal, while r_3 is Horn. Thus, Π is not nesting-free.*

An *interpretation* is a set $I \subseteq \mathscr{A}$ of atoms. The *satisfaction* (*model*) relation $I \models \alpha$ where α is an atom p, a rule r or a logic program Π, is defined as follows:

- $I \models p$ if $p \in I$; $I \models not\ p$ if $I \not\models p$; $I \models not\ not\ p$ if $I \not\models not\ p$, *i.e.*, $p \in I$;

- $I \models r$ if $I \models hd(r)$ or $I \not\models \gamma$ for some $\gamma \in bd(r)$;
- $I \models \Pi$ if $I \models r$ for every $r \in \Pi$.

A model I of Π is *minimal*, if Π has no model I' satisfying $I' \subset I$; moreover I is the *least* (under set inclusion) model Π, if I is the single minimal model of Π, i.e., $I \subseteq I'$ for every model I' of Π. We recall that a set $M \subseteq \mathscr{A}$ is a *stable model* (or *answer set*) of Π, if M is a minimal model of the *GL-reduct* Π^M of Π *w.r.t.* M Gelfond and Lifschitz [1991]; Lifschitz *et al.* [1999], where

$$\Pi^M = \{hd(r) \leftarrow r^- \mid r \in \Pi \text{ such that } M \models \gamma \text{ for each } \gamma \in bd(r) \setminus r^-\}. \tag{3}$$

Example 2.2 (Example 2.1 cont'd)**.** *Let $M_1 = \{b,c\}$. We have $\Pi^{M_1} = \{a \vee b, \quad c, \quad b \leftarrow a\}$, whose minimal model is $\{b,c\}$. Thus, M_1 is an answer set of Π. For $M_2 = \{c\}$, we have $\Pi^{M_2} = \Pi^{M_1}$. Since M_2 is not a minimal model of Π^{M_2}, M_2 is not an answer set of Π.*

We note that a positive rule r of the form (1) has the same models as the clause $p_1 \vee \cdots \vee p_k \vee \neg p_{k+1} \vee \cdots \vee \neg p_m$, or represented as the set $r^+ \cup \neg r^-$ of literals. Furthermore, the GL-reduct Π^I of a positive program Π *w.r.t.* any interpretation I always equals Π, *i.e.*, $\Pi^I = \Pi$, which implies that the answer sets of a positive program are its minimal models. For this reason, we shall identify a positive rule (resp. a positive program) with its corresponding clause (resp. clause theory), if clear from context, and vice versa identify a clause (resp. clause theory) with the corresponding positive rule (resp. positive program) .

It is well-known that every definite logic program Π has the least (under set inclusion) model $LM(\Pi)$, which is computable in polynomial time by the least fixpoint iteration as $LM(\Pi) = T_\Pi^\infty = \lim_{i\to\infty} T_\Pi^i$, where $T_\Pi^0 = \emptyset$ and $T_\Pi^{i+1} = T_\Pi(T_\Pi^i)$, for $i \geq 0$, with the immediate consequence operator van Emden and Kowalski [1976] $T_\Pi : 2^{\mathscr{A}} \to 2^{\mathscr{A}}$ defined by

$$T_\Pi(S) = \{p \mid p \in r^+, r \in \Pi, r^- \subseteq S\}.$$

As well-known, computing T_Π^∞ Dowling and Gallier [1984] is feasible in linear time.

The *(positive) dependency graph* of a logic program Π is the directed graph $G_\Pi = (V,E)$ whose vertices in V are the atoms occurring in Π and with edges $(p,q) \in E$ if $q \in r^+$ and $p \in r^-$ for some $r \in \Pi$. A nonempty set $L \subseteq \mathscr{A}$ of atoms is a *loop* of Π, if G_Π has a cyclic path $v_0,\ldots,v_{n+1}$, i.e., $(v_i,v_{i+1}) \in E$, $0 \leq i \leq n$, and $v_{n+1} = v_0$, such that $L = \{v_0,\ldots,v_n\}$. A logic program Π is *headcycle-free*, if $|L \cap r^+| \leq 1$ for every loop L of Π and every rule $r \in \Pi$.

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., such that a path $v_0,\ldots,v_{n+1}$ with $v_0 = u$ and $v_{n+1} = v$ exists in G; furthermore, for any set $V' \subseteq V$ of nodes, we let $D_G(V') = \bigcup_{v\in V'} D_G(v')$. A vertex v is a *source* of G, if $D_G(v) = \emptyset$.

Given a directed graph $G = (V,E)$, the *collapsed dependency graph* $\mathbb{S}_G = (\mathbb{V},\mathbb{E})$ of G Leone *et al.* [1997] is the directed acyclic graph (DAG) 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 G' by V' when it is clear from its context, and
- $\mathbb{E}$ consists of all edges (C,C') such that $C \neq C'$ and $C \cap D_G(C') \neq \emptyset$.

For simplicity, we write $\mathbb{S}_\Pi$ instead of $\mathbb{S}_{G_\Pi}$ for a logic program Π.

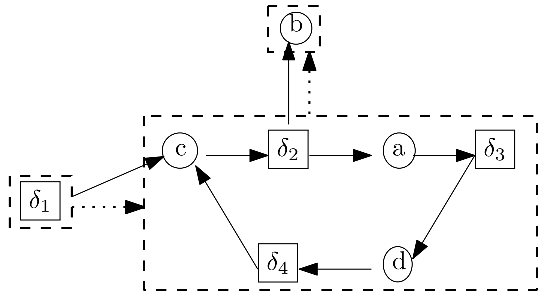


Figure 1: The atom-clause dependency graph $\mathbb{G}_\Sigma$ and the super-dependency graph $\mathbb{SG}_\Sigma$ (dashed lines) of Σ in Example 2.3

2.2 Minimal model decomposition

As we have seen above, the definition of answer set requires a minimality check for models of a positive program. For this task, Ben-Eliyahu-Zohary *et al.* [2016,2017] proposed an algorithm $\mathsf{CheckMin}(\Sigma,M)$ for checking whether a model M of a clause theory Σ is a minimal model of Σ, based on minimal model decomposition using a super-dependency graph with a more fine-grained dependency representation for positive programs respectively clause theories.

The *atom-clause dependency graph* of a clause theory Σ is the directed graph $\mathbb{G}_\Sigma = (V,E)$, where

- the vertices V are the atoms and clauses occurring in Σ;
- for each clause $\delta \in \Sigma$, E contains edges $(p,\delta) \in E$ and $(\delta,q) \in E$ for all $p \in \delta^-$ and $q \in \delta^+$, respectively.

Intuitively, if both (p,δ) and (δ,q) are in E, then p depends (backwards) on q via the clause δ. The *super-dependency graph* $\mathbb{SG}_\Sigma$ of Σ is then the collapsed dependency graph of $\mathbb{G}_\Sigma$, i.e., $\mathbb{SG}_\Sigma = \mathbb{S}_{\mathbb{G}_\Sigma}$.

Example 2.3. *Consider the clause theory (written as positive program) $\Sigma = \{\delta_1 : c, \quad \delta_2 : a \vee b \leftarrow c, \quad \delta_3 : d \leftarrow a, \quad \delta_4 : c \leftarrow d\}$. Then $\mathbb{G}_\Sigma = (V,E)$ has nodes $V = \{a,b,c,d\} \cup \{\delta_1,\delta_2,\delta_3,\delta_4\}$ and edges $E = \{(c,\delta_2)$, (a,δ_3), (d,δ_4), (δ_1,c), (δ_2,a), (δ_2,b), (δ_3,d), $(\delta_4,c)\}$. Furthermore, $\mathbb{SG}_\Sigma$ has the nodes $v_1 = \{\delta_1\}$, $v_2 = \{a,c,d,\delta_2,\delta_3,\delta_4\}$, and $v_3 = \{b\}$, and the edges (v_1,v_2) and (v_2,v_3). Fig. 1 graphically illustrates $\mathbb{G}_\Sigma$ and $\mathbb{SG}_\Sigma$. It is not difficult to check that Σ has two minimal models $\{a,c,d\}$ and $\{b,c\}$.*

We call a source S of $\mathbb{SG}_\Sigma$ *empty*, if $S \cap \mathscr{A} = \emptyset$, and denote by Σ_S the set of clauses in Σ that mention only atoms from S. For a clause theory Σ and disjoint sets X and Y of atoms, *Reduce*(Σ,X,Y) is obtained from Σ by setting all atoms in X to **true** and all atoms in Y to **false**. Ben-Eliyahu-Zohary *et al.* [2016] showed the following result:

Theorem 1 (Minimal model decomposition theorem by Ben-Eliyahu-Zohary *et al.* 2016)**.** *Let M be a minimal model of a clause theory Σ and let S be a source of $\mathbb{SG}_\Sigma$ s.t. $X = S \cap M$ is a minimal model of Σ_S. Then $M - X$ is a minimal model of $\Sigma' = Reduce(\Sigma, X, S \cap \mathscr{A} - X)$.*

Based on this result, Ben-Eliyahu-Zohary *et al.* proposed an algorithm $\mathsf{CheckMin}(\Sigma,M)$, shown as Algorithm 1, and proved that a model M of Σ is minimal if $\mathsf{CheckMin}(\Sigma,M)$ returns **true**. However, while the algorithm is sound for minimal model checking (the model M is indeed minimal if $\mathsf{CheckMin}(\Sigma,M)$ returns **true**), it is not complete (it is possible that $\mathsf{CheckMin}(\Sigma,M)$ return **false** even if M is a minimal model of

```
Algorithm 1: CheckMin $(\Sigma, M)$
   Input: A clause theory $\Sigma$ and a model $M$ of $\Sigma$
   Output: true (only if $M$ is minimal) or false
1  $G \leftarrow \mathbb{SG}_\Sigma$;
2  Recursively delete from $G$ all empty sources;
3  while $G$ has a source $S$ which is a minimal model of $\Sigma_S$ do
4      $X \leftarrow M \cap S$;
5      $M \leftarrow M - X$;
6      $\Sigma \leftarrow Reduce(\Sigma, X, S \cap \mathscr{A} - X)$;
7      $G \leftarrow \mathbb{SG}_\Sigma$;
8      Recursively delete from $G$ all empty sources;
9  end
10 if $M = \emptyset$ then return true else return false
```

Σ). Ben-Eliyahu-Zohary *et al.* gave the following example showing that CheckMin (Σ, M) may return **false** although M is a minimal model of Σ.

Example 2.4 (Example 2.3, cont'd)**.** *It is not difficult to check in Fig. 1 that the only source* $\{\delta_1\}$ *of the super-dependency graph* $\mathbb{SG}_\Sigma$ *is empty. As* $\mathbb{SG}_\Sigma$ *has the single source* $\{\delta_1\}$ *(which is empty), after deleting* $\{\delta_1\}$ *from* $\mathbb{SG}_\Sigma$*, the only source is* $S = \{a, c, d, \delta_2, \delta_3, \delta_4\}$ *and* $\Sigma_S = \{\delta_1, \delta_3, \delta_4\}$*. Consider now* $M = \{a, c, d\}$*. Then CheckMin*(Σ, M) *returns* ***false****, as* $M \cap S = \{a, c, d\}$ *is not a minimal model of* Σ_S*, whose unique minimal model is* $\{c\}$*. However, M is a minimal model of* Σ*.*

To address this issue, a sufficient condition called *modular property* was proposed. Formally, a minimal model M of a clause theory Σ has the *modular property w.r.t.* Σ, if either

1. $\mathbb{SG}_\Sigma$ has only one strongly connected component, or

2. $\mathbb{SG}_\Sigma$ has a source S *s.t.* $M \cap S$ is a minimal model of Σ_S, and $M - S$ is a minimal model of $\Sigma' = Reduce(\Sigma, M \cap S, S \cap \mathscr{A} - M)$ that enjoys the modular property *w.r.t.* Σ'.

Ben-Eliyahu-Zohary *et al.* proved that if a minimal model M of Σ has the modular property *w.r.t.* Σ, then CheckMin (Σ, M) always return **true**, i.e., the algorithm is complete for minimal model checking under this assertion [Ben-Eliyahu-Zohary *et al.*, 2016, Theorem 6.4].

3 Reduct

In this section, we present the notion of reduct[1] and show how the algorithm CheckMin can take advantage of it. More specifically, we show that for clause theories which are invariant under the reduct, the algorithm CheckMin from above is complete. As the reduct preserves the minimal models that contain at most the considered reducing atoms of each clause theory and can be computed efficiently, we obtain that CheckMin can be easily adapted to a sound and complete minimality check for arbitrary clause theories with little overhead.

[1]It is similar to but different from the *disjunctive reduct* for positive logic programs Zhang and Zhang [2017], which does not change rule bodies.

Definition 3.1 (reduct)**.** *Let* Π *be a logic program and let* $S \subseteq \mathscr{A}$*. The* reduct *of* Π *w.r.t.* S*, denoted by* $\mathrm{MR}(\Pi, S)$*, is the positive program or clause theory:*

$$\{r^+ \cap S \leftarrow r^- \mid r \in \Pi \text{ such that } S \models \gamma \text{ for each } \gamma \in bd(r)\}.$$

Intuitively, performing the reduct transforms a logic program according to the closed world assumption: the atoms not in S are assumed to be **false**. Under this assumption, building the reduct substitutes each atom that is assumed to be **false** with **false** and similarly substitutes every atom in the scope of "*not*" by its truth value.

In particular, when Π is a clause theory, then

$$\mathrm{MR}(\Pi, S) = \{\, (\alpha^+ \cap S) \cup \neg\alpha^- \mid \alpha \in \Pi,\ \alpha^- \subseteq S\}. \tag{4}$$

In what follows, we view the reduct $\mathrm{MR}(\Pi, M)$ also as a clause theory even if Π is a logic program, unless explicitly stated. In this sense $\mathrm{MR}(\Pi, M) \models \varphi$ means that the formula φ is a logical consequence of $\mathrm{MR}(\Pi, M)$.

Example 3.1. *For the logic program* $\Pi = \{a \vee b, \quad a \leftarrow b, \quad b \leftarrow a\}$ *in the Introduction, we have:*

- $\mathrm{MR}(\Pi, \emptyset) = \{\bot\}$ *as* $a \vee b$ *is turned into the empty disjunction,* $\mathrm{MR}(\Pi, \{a, b\}) = \Pi$ *(no rule is altered) , and* $\mathrm{MR}(\Pi, \{a\}) = \{a, \quad \leftarrow a\}$ *is logically equivalent to* $\mathrm{MR}(\Pi, \{b\}) = \{b, \quad \leftarrow b\} \equiv \bot$*. Thus, the reduct does not preserve satisfiability.*

For Σ *and* M *in Example 2.4, we have* $\Sigma' = \mathrm{MR}(\Sigma, M) = \{\delta_1, \delta_3, \delta_4\} \cup \{a \leftarrow c\}$*, which yields a definite program.*

The reduct is similar to the GL-reduct for logic programs Gelfond and Lifschitz [1991] and the reduct for logic programs with nested expressions Lifschitz *et al.* [1999]. However, the latter two handle only the atoms in the scope of "*not*". Furthermore, the reduct is also different from the reduct in Ferraris [2005] for propositional formulas;[2] *e.g.*, the Ferraris-reduct of $\{\neg a\}$ *w.r.t.* $\{a\}$ is $\{\bot\}$, while $\mathrm{MR}(\{\neg a\}, \{a\}) = \{\neg a\}$.

The next proposition shows that the reduct $\mathrm{MR}(\Pi, S)$ is logically equivalent to the Ferraris-reduct $\tau(\Pi)^S$ when S is a model of Π, where $\tau(\Pi) = \bigwedge\{\tau(r) \mid r \in \Pi\}$ and for each rule r,

$$\tau(r) \;=\; \left[\bigwedge r^- \wedge \bigwedge_{p \in r^{not}} (p \supset \bot) \wedge \bigwedge_{q \in r^{2not}} (q \supset \bot) \supset \bot)\right] \supset \bigvee r^+. \tag{5}$$

Proposition 3.1. *Let* Π *be a general logic program and* S *a model of* Π*. Then it holds that* $\mathrm{MR}(\Pi, S) \equiv \tau(\Pi)^S$.

The next proposition shows that the reduct "preserves" minimal models for clause theories in the following sense.

Proposition 3.2. *A set* $S \subseteq \mathscr{A}$ *is a minimal model of a clause theory* Σ *iff* S *is a minimal model of* $\mathrm{MR}(\Sigma, S)$.

Example 3.2 (Example 3.1 cont'd)**.** *Recall that for the model* $M = \{a, c, d\}$ *of the clause theory* Σ *in Example 2.4, we have* $\mathrm{MR}(\Sigma, M) = \{\delta_1, \delta_3, \delta_4\} \cup \{\delta_2' : a \leftarrow c\} = \{c, \quad d \leftarrow a, \quad c \leftarrow d, \quad a \leftarrow c\}$*. It is readily checked that* M *is a minimal model of* $\mathrm{MR}(\Sigma, M)$.

[2]The connectives are $\bot, \wedge, \vee$ and $\supset$, while $\neg\psi$ stands for $\psi \supset \bot$.

The following theorem establishes a constructive characterization of the minimal models of a clause theory via logical entailment.

Theorem 2 (Minimal model characterization)**.** *For every clause theory Σ and $S \subseteq \mathscr{A}$, the items (i)–(iii) are equivalent:*

(i) S is a minimal model of Σ.

(ii) S is the least model (under set inclusion) of $\mathrm{MR}(\Sigma, S)$.

(iii) $S = \{q \in \mathscr{A} \cup \{\bot\} \mid \mathrm{MR}(\Sigma, S) \models q\}$.

We note that item (iii) does not hold whenever $\mathrm{MR}(\Sigma, M)$ is inconsistent since $S \subseteq \mathscr{A}$ and $\mathscr{A}$ does not contain $\bot$. As regards item (iii) of Theorem 2, one can use resolution to compute the atoms that are logical consequences of $\mathrm{MR}(\Sigma, M)$. Such a resolution proof for an atom can be taken as a "justification" (or an explanation) for why the atom belongs to the minimal model M of $\mathrm{MR}(\Sigma, M)$. Many efficient theorem provers such as Z3[3] and iprover[4] can be utilized for this task.

The next proposition shows that this reduct can be employed to simplify clause theories for computing minimal models that are contained in a given model.

Proposition 3.3. *Suppose M is a model of a clause theory Σ and let $M' \subset M$. Then M' is a minimal model of Σ if and only if M' is a minimal model of $\mathrm{MR}(\Sigma, M)$.*

Consequently, we can replace for the minimality check of a nonempty model M of the clause theory Σ with the reduct $\mathrm{MR}(\Sigma, M)$ (for the empty model the check is trivial). For clause theories in the reduced form, i.e., that satisfies $\Sigma = \mathrm{MR}(\Sigma, M)$, we can complement the Minimal Model Decomposition Theorem with the following result.

Proposition 3.4. *Let M be a nonempty minimal model of a clause theory Σ and $\mathrm{MR}(\Sigma, M) = \Sigma$. Then*

(i) $\mathscr{A} \cap S = \emptyset$ for every source S of $\mathbb{SG}_\Sigma$.

(ii) Let $\mathbb{S}$ be resulted from $\mathbb{SG}_\Sigma$ by removing all empty sources. Then for every source S of $\mathbb{S}$, $S \cap \mathscr{A}$ is a minimal model of Σ_S, and $M - S \cap \mathscr{A}$ is a minimal model of $Reduce(\Sigma, S \cap \mathscr{A}, \emptyset)$.

(iii) Let S be a node of $\mathbb{SG}_\Sigma$ and $S \cap \mathscr{A} \neq \emptyset$, $X_S = \{S' \cap \mathscr{A} \mid S' \in D_{\mathbb{SG}_\Sigma}(S)\}$ and $\Gamma = Reduce(\Sigma, X_S, \emptyset)$. Then S is a minimal model of Γ_S.

This proposition suggests a revised minimal model checking algorithm CheckMinMR, shown in Algorithm 2; the main difference is that it uses the reduct, *i.e.*, it computes the reduct of Σ *w.r.t.* M at first (Line 1). The other minor revisions include simplifying the remaining process of CheckMin according to Proposition 3.4, *e.g.*, it returns **false** if there is a nonempty source S and $S \cap \mathscr{A}$ is not a minimal model of Σ_S (Line 4); the "Reduce" (Line 6) is also simplified due to the fact that $S \cap \mathscr{A} = S \cap M$. Modulo these minor differences, CheckMinMR(Σ, M)=CheckMin$(\mathrm{MR}(\Sigma, M), M)$. Combined with Theorem 6.4 of Ben-Eliyahu-Zohary *et al.* [2016], we obtain the following result.

Theorem 3. *Let M be a model of a clause theory Σ. Then it holds that*

(i) M has the modular property w.r.t. Σ if M is a minimal model of Σ;

[3]`https://github.com/Z3Prover/z3`

[4]`http://www.cs.man.ac.uk/~korovink/iprover/`

```
Algorithm 2: CheckMinMR($\Sigma, M$)
  Input: A clause theory $\Sigma$ and a model $M$ of $\Sigma$
  Output: true if $M$ is a minimal model of $\Sigma$, else false
1 $\Sigma \leftarrow \mathrm{MR}(\Sigma, M)$;   $G \leftarrow \mathbb{SG}_\Sigma$;
2 Recursively delete from $G$ all empty sources;
3 while $G$ is nonempty, i.e., has some source $S$ do
4     if $S \cap \mathscr{A}$ is not a minimal model of $\Sigma_S$ then break $M \leftarrow M - S$;
5     $\Sigma \leftarrow Reduce(\Sigma, S \cap \mathscr{A}, \emptyset)$;
6     $G \leftarrow \mathbb{SG}_\Sigma$;
7     Recursively delete from $G$ all empty sources;
8 end
9 if $M = \emptyset$ then return true else return false
```

```
Algorithm 3: MinModel($\Sigma$)
  Input: A clause theory $\Sigma$
  Output: A minimal model of $\Sigma$ if it is satisfiable, and UNSAT otherwise
1 if $\Sigma$ is unsatisfiable then return UNSAT while $\Sigma$ is satisfiable do
2     Let $M$ be a model of $\Sigma$;
3     if CheckMinMR($\Sigma, M$) = true then break $\Sigma \leftarrow \mathrm{MR}(\Sigma, M) \cup \{\bigvee \neg M\} \cup \{\neg p \mid p \in \mathrm{var}(\Sigma) - M\}$;
4 end
5 return $M$;
```

(ii) M is a minimal model of Σ iff CheckMinMR(Σ, M) *returns* **true**.

That is, CheckMinMR(Σ, M) is an algorithm for modular minimal model checking that is sound and complete, in contrast to the original algorithm CheckMin(Σ, M).

Example 3.3 (Example 3.2 cont'd)**.** *Recall that for Σ and M in Example 2.4, $M = \{a, c, d\}$ is a minimal model of $\Sigma' = \mathrm{MR}(\Sigma, M) = \{\delta_1 : c, \quad \delta_3 : d \leftarrow a, \quad \delta_4 : c \leftarrow d, \quad \delta_2' : a \leftarrow c\}$. Clearly, M is the least model of Σ' and* CheckMinMR(Σ, M) *returns* **true***: G is the supergraph $\mathbb{SG}_{\Sigma'} = (\{v_1, v_2\}, \{(v_1, v_2)\})$ resulted from the supergraph $\mathbb{SG}_\Sigma$ in Example 2.4 (cf. Fig. 1) by removing the node $v_3 = \{b\}$ and the associated edge (v_2, v_3) and replacing δ_2 by δ_2'. The single source $v_1 = \{\delta_1\}$ of $\mathbb{SG}_{\Sigma'}$ is empty, and thus is deleted in line 2; the remaining node $v_2 = \{a, c, d, \delta_2', \delta_3, \delta_4\}$ is then a nonempty source S. It fulfills that $S \cap \mathscr{A} = \{a, c, d\} = M$ is a minimal model of $\Sigma'_S = \Sigma'$, and thus M is updated in line 5 to $M - S = \emptyset$. Furthermore, Σ' is updated in line 6 to $Reduce(\Sigma', S \cap \mathscr{A}, \emptyset) = Reduce(\Sigma', M, \emptyset) = \emptyset$, which means the G is updated in line 7 to the empty graph. Consequently, the loop quits and in line 10, and* **true** *is returned.*

Based on algorithm CheckMinMR, we also obtain a new algorithm Algorithm MinModel(Σ) for computing minimal models of clause theories, shown as Algorithm 3, whose correctness is easy to establish from Theorem 3. Intuitively, it generates a model M of Σ (if possible) and then checks whether M is a minimal model of Σ by CheckMinMR. In case M is not a minimal model of Σ, it computes a model $M' \subset M$ of Σ at Line 5. Here, $\{\neg p \mid p \in \mathrm{var}(\Sigma) - M\}$ requires $M' \subseteq M$, while $\bigvee(\neg M)$ asserts that some atoms in M must be **false**.

The next theorem shows that the characterization via the reduct extends naturally to all logic programs that we consider. In this way, we obtain a unified answer set definition for both normal and disjunctive logic programs in terms of a least model, although $\mathrm{MR}(\Pi,M)$ may have no least model and may be intractable to be computed if Π is disjunctive, *e.g.* $\mathrm{MR}(\{a\vee b\},\{a,b\})=\{a\vee b\}$, which has two minimal models $\{a\}$ and $\{b\}$, but no least model.

Theorem 4 (Answer set characterization)**.** *For every logic program* Π *and* $M\subseteq\mathscr{A}$*, the items (i)–(iii) are equivalent:*

(i) *M is an answer set of* Π.

(ii) *M is the least model of* $\mathrm{MR}(\Pi,M)$.

(iii) $M=\{q\in\mathscr{A}\cup\{\bot\}\mid \mathrm{MR}(\Pi,M)\models q\}$.

It is evident that the item (iii) in the above theorem can be $M=\{q\in\mathscr{A}\mid \mathrm{MR}(\Pi,M)\models q\}$ if M is a model of Π.

We stress that by Theorems 3 and 4, $\mathsf{CheckMinMR}(\mathrm{MR}(\Pi,M),M)$ can be used for checking whether a model M of a logic program Π is an answer set; furthermore, since $\mathrm{MR}(\mathrm{MR}(\Pi,M),M)=\mathrm{MR}(\Pi,M)$, by replacing Σ in $\mathsf{CheckMinMR}$ with a logic program Π, we can use $\mathsf{CheckMinMR}(\Pi,M)$ right away for answer set checking. We remark that algorithm $\mathsf{MinModel}$ could be similarly adapted for computing answer sets of general logic programs. However, as not every model M of a logic program Π contains some answer set, completeness would have to be ensured by looping over models.

4 Witnesses for Answer Sets

Recall that as a consequence of Theorem 4, every atom in the least model of $\mathrm{MR}(\Pi,M)$ (provided that it exists) has a resolution proof from $\mathrm{MR}(\Pi,M)$. Such a proof provides a justification for "why an atom belongs to the answer set M". Notice further that the algorithm $\mathsf{CheckMinMR}$ decomposes a minimal model checking task into a series of smaller-scale subtasks. Thus, it provides some evidence for "why some atoms are in an answer set" on a high level since each atom in an answer set has a resolution proof and each component in the decomposition of an answer set has a justification based on the previously established components.[5] These considerations motivate the notion of witness for answer sets below, from which a structural explanation for an answer set can be built up.

To this end, we introduce in this section first a generic notion of witness and minimal witness for justifying the presence of a set B of atoms in an answer set M by resorting to Theorem 4 from above, in terms of rules from a logic program Π at hand, relative to some context S of atoms that are asserted to be **true**. We then present the notions of α- and β-witnesses , which use syntactic dependency information in order to account for the structure of explanations of answer sets. Roughly speaking, α-witnesses are modularly composed witnesses $W_1,\dots,W_n$ of an answer set $M=S_1\cup\cdots\cup S_n$ partitioned into components S_i, $i=1,\dots,n$ such that for S_i, the parts S_j on which S_i depends provide the context S, and W_i explains S_i and no other part S_j. The notion of β-witness is more refined as the sets S_i have to be singletons. For both notions, in the spirit of Occam's razor also non-redundant versions and further restrictions are considered.

4.1 Witnesses

We start with a formal definition of the notions of witness and minimal witness.

[5] Sources need no resolution proof.

Definition 4.1 (witness and minimal witness)**.** *Let $M \subseteq \mathscr{A}$, and B, S be disjoint subsets of M. A logic program Π is a* witness of B under S *w.r.t. M, if* $\mathrm{MR}(\Pi, M) \cup S \models B$*; moreover Π is* minimal*, if no $\Pi' \subset \Pi$ exists s.t.* $\mathrm{MR}(\Pi', M) \cup S \models B$*. For every logic program Π, we denote by* $\mathrm{MW}(B, \Pi, S, M)$ *the set of all minimal witnesses $\Pi' \subseteq \Pi$ of B under S w.r.t. M.*

Intuitively, the set M in the above definition plays the role of closed-world assumption for the witness Π, while S stands for a set of already justified atoms. In this sense, a witness provides some evidence as a set of rules from which the atoms in B can be properly justified in terms of logical reasoning. Recall that we identify clause theories as positive logic programs. The notion of witness and its variants are all applicable for clause theories as well.

Example 4.1. *For the logic program $\Pi = \{r_1\colon a \vee b, \quad r_2\colon a \leftarrow b, \quad r_3\colon b \leftarrow a\}$ and $M = \{a, b\}$ in the Introduction, $\Pi' = \{r_1, r_2\}$ is a minimal witness of $B = \{a\}$ under $S = \emptyset$ w.r.t. M. In fact, we have that it is the only such witness, i.e.,* $\mathrm{MW}(B, \Pi, S, M) = \{\Pi'\}$*. Similarly, $\Pi'' = \{r_1, r_3\}$ is the only minimal witness of $B = \{b\}$ under $S = \emptyset$ w.r.t. M.*

As can be seen, minimal witnesses are closely related to minimal unsatisfiable subformulas (MUS) and minimal unsatisfiable cores Marques-Silva [2010]; Liffiton *et al.* [2016]. Formally, an MUS of a unsatisfiable clause theory Σ is a subset Σ' of Σ such that Σ' is unsatisfiable and every proper subset Σ'' of Σ' is satisfiable. We then have the following relationships.

Proposition 4.1. *Let $M \subseteq \mathscr{A}$, let $B, S \subseteq M$ be disjoint subsets of M, and let Π be a logic program.*

(i) If $\Pi' \in \mathrm{MW}(B, \Pi, S, M)$ then some $S' \subseteq S$ exists such that $\mathrm{MR}(\Pi', M) \cup S' \cup \{\bigvee \neg B\}$ is an MUS.

(ii) If $\mathrm{MR}(\Pi', M) \cup S' \cup \{\bigvee \neg B\}$ is an MUS of $\mathrm{MR}(\Pi, M) \cup S \cup \{\bigvee \neg B\}$ such that $S' \subseteq S$, then some $\Pi'' \in \mathrm{MW}(B, \Pi, S, M)$ exists such that $\Pi'' \subseteq \Pi'$.

Exploiting this proposition, we can compute a minimal witness Π^* for B under S *w.r.t.* M by the algorithm MinWitness, which is shown in Algorithm 4. It uses an MUS solver as an optional parameter; several such solvers are available, e.g., picomus[6] and SAT4j[7], which can be leveraged for computing minimal witnesses. We note that the **ForEach** loop in lines 5-8 of this algorithm serves to compute some $\Pi^* \subseteq \Pi$ such that $\mathrm{MR}(\Pi^*, M) = \Pi'$, which is effected by line 3. The soundness and completeness of Algorithm 4 when a complete MUS solver is used is ensured by the items (ii) and (i) of Proposition 4.1, respectively; in the solverless case, it follows from the monotonicity of classical inference.

Proposition 4.2. *Algorithm 4 is sound and moreover complete if the involved MUS solver mus is complete.*

The following example illustrates how Algorithm 4 can be employed to compute a minimal witness.

Example 4.2 (Example 4.1 cont'd)**.** *Recall that $\Pi' = \{r_1, r_2\}$ is a minimal witness of $B = \{a\}$ under $S = \emptyset$ w.r.t. $M = \{a, b\}$. It is not difficult to check that for $S' = S = \emptyset$, we have that $\Pi' \cup S' \cup \{\bigvee \neg B\} = \Pi' \cup \{\neg a\}$ is an MUS of $\Pi \cup S \cup \{\bigvee \neg B\} = \{r_1, r_2, r_3, \neg a\}$, in line with item (i) of Proposition 4.1. Thus Π' may be computed by an MUS solver mus in lines 2 and 3 of Algorithm 4; then $\Pi^* = \{r_1, r_2\}$ is constructed in lines 4-8 and output. In the solverless case, only the rule r_3 can be removed from the initial rule set $\Pi^* = \Pi = \{r_1, r_2, r_3\}$; the resulting rule set $\Pi^* = \{r_1, r_2\}$ is then likewise found as a minimal witness and output. We note that $\Pi' = \{r_1, r_2\}$ is the unique set that satisfies the MUS condition in item (ii) of Proposition 4.1, and thus a unique minimal witness exists, as observed in Example 4.1.*

[6]`https://www.mankier.com/1/picomus`

[7]`http://sat4j.org/`

Algorithm 4: MinWitness(B,Π,M,S)

```
Input: A logic program $\Pi$; $M \subseteq \mathscr{A}$ and $B,S \subseteq M$ s.t. $B \cap S = \emptyset$ and $M$ is a minimal model of
       $\mathrm{MR}(\Pi,M) \cup S$
Parameter: an MUS solver mus (nil = no MUS solver)
Output: A minimal witness $\Pi^* \subseteq \Pi$ of $B$ under $S$ w.r.t. $M$
if $mus \neq nil$ then
    compute an MUS $\Pi'$ of $\mathrm{MR}(\Pi,M) \cup S \cup \{\bigvee \neg B\}$ by calling the MUS solver mus;
    remove all elements in $S \cup \{\bigvee \neg B\}$ from $\Pi'$;
    $\Pi^* \leftarrow \emptyset$;
    foreach $r' \in \Pi'$ do
        select some rule $r \in \Pi$ with $\mathrm{MR}(\{r\},M) = \{r'\}$;
        $\Pi^* \leftarrow \Pi^* \cup \{r\}$;
    end
else
    $\Pi^* \leftarrow \Pi$;
    foreach $r \in \Pi^*$ do
        if $\mathrm{MR}(\Pi^* - \{r\},M) \cup S \models B$ then $\Pi^* \leftarrow \Pi^* - \{r\}$
    end
end
return $\Pi^*$;
```

We note that for $M = \emptyset$, the single minimal witness for B under S *w.r.t.* M is the empty set of clauses, as $B = S = \emptyset$ must hold and no rules are needed to explain an empty B. In particular, if $M = \emptyset$ is an answer set of a logic program Π, then $\Pi' = \emptyset$ is trivially the single minimal witness of M under S from Π. In the rest of this paper, we thus concentrate on witnesses for nonempty answer sets of logic programs.

4.2 $\alpha^\star$- and α-witnesses

We are now in a position to present various witnesses for answer sets of logic programs starting from the most general notion of $\alpha^\star$-witness.

Definition 4.2 ($\alpha^\star$- and α -witnesses)**.** *Let* Π *be a logic program and* $M \neq \emptyset$ *be an answer set of* Π. *Furthermore, let B and S be disjoint subsets of M. Then, an* $\alpha^\star$-witness *of B under* Π *and S w.r.t. M is a DAG* $G = (\{(S_i,\Pi_i) \mid 1 \leq i \leq n\},E)$ *where* $\{S_i \mid 1 \leq i \leq n\}$ *is a partitioning of B (i.e.* $\bigcup_{i=1}^n S_i = B$ *and* $S_i \neq \emptyset$, $1 \leq i \leq n$*) and, for every* $i, 1 \leq i \leq n$,

(i) $\Pi_i \subseteq \Pi$ *is a witness of* S_i *under* $S \cup X_i$ *w.r.t. M, and*

(ii) Π_i *is not a witness of* S_j *under* $S \cup X_i$ *w.r.t. M, for every* $1 \leq j \neq i \leq n$*, where*

$$X_k = \bigcup \{S' \mid (S',\Pi') \in D_G((S_k,\Pi_k))\}, \qquad 1 \leq k \leq n. \tag{6}$$

If G induces a total order $(S_1,\Pi_1) < (S_2,\Pi_2) < \cdots < (S_n,\Pi_n)$*, i.e.* $E = \{((S_i,\Pi_i),(S_{i+1},\Pi_{i+1})) \mid 1 \leq i < n\}$*, we call it an* α-witness *of B under* Π *and S w.r.t. M and write it as*

$$W = [(S_1,\Pi_1),\ldots,(S_n,\Pi_n)]. \tag{7}$$

We call G minimal, *if every Π_i is minimal and G is* compact, *if in addition to minimality $\Pi_i \cap \Pi_j = \emptyset$ for all $1 \leq i < j \leq n$.*

If $B = M$ and $S = \emptyset$, we call G an (minimal, compact) $\alpha^\star$-witness resp. α-witness of M w.r.t. Π.

Intuitively, in Definition 4.2 the atoms in S are assumed to be justified resp. explained and need no further explanation. They can be used for justifying resp. explaining more atoms in an answer set, which are given by the set B that has to be disjoint from S. Refining the notion of witness from the previous section, $\alpha^\star$-witnesses and their specializations take a modular perspective where parts $S_1,\ldots,S_n$ of B must be justified locally, using sets $\Pi_1,\ldots,\Pi_n$ of rules. To this end, for deriving the atoms in S_i the atoms in $X_i = \bigcup\{S' \mid (S',\Pi') \in D_G((S_i,\Pi_i))\}$, which are derived by rules of the program Π that in the graph G intuitively feed into the rules Π_i, are taken as already justified; this yields condition (i). In particular, note that $X_i = \emptyset$ if $D_G((S_i,\Pi_i)) = \emptyset$. Thus, once the atoms in each S_i $(1 \leq i \leq n)$ are justified, the atoms in B are structurally justified. In particular, if $B = M$ and $S = \emptyset$ then the answer set M can be structurally justified resp. explained by resolution proofs using the rules in a witness under CWA *w.r.t.* M. Condition (ii) in Definition 4.2 imposes that Π_i with input $S_i \cup X_i$ covers S_i but no other part S_j of B; this ensures that the explanatory power of local witnesses is exploited and that components are not unnecessarily introduced. Notably, if S_j precedes S_i *w.r.t.* G, *i.e.* $(S_j,\Pi_j) \in D_G((S_i,\Pi_i))$, the condition in (ii) is always fulfilled according to Definition 4.1: as $S_j \subseteq X_i$, S_j and $X_i \cup S_i$ are not disjoint, and thus Π_i is not a witness of S_j under $X_i \cup S_i$ *w.r.t.* M. For any $S_j \neq S_i$ not preceding S_i, a violation of condition (ii) would mean that S_j could be merged into S_i and witnessed by Π_i. Condition (ii) also excludes some "undesired" orders among the witnesses to capture derivation steps of answer sets.

Example 4.3. *Let us consider the program $\Pi = \{r_1 : a_1, \quad r_2 : a_2 \leftarrow a_1, \quad r_3 : a_3 \leftarrow a_2\}$, and the sequences $W = [(\{a_2,a_1\},\{r_1,r_2\}),(\{a_3\},\{r_3\})]$ and $W' = [(\{a_3\},\Pi),(\{a_2,a_1\},\{r_1,r_2\})]$. Then W is an α-witness of the answer set $M = \{a_1,a_2,a_3\}$ of Π and captures a proper derivation for M from Π: it derives first a_1 and a_2 from $\{r_1,r_2\}$, then a_3 is derived from $\{r_3\}$ together with the derived a_1 and a_2. The sequence W' is not an α-witness: the reason is that a_2 and a_1 have been derived when a_3 is derived at the beginning. In this case, the witness Π of $\{a_3\}$ can replace the witness $\{r_1,r_2\}$ of $\{a_1,a_2\}$ in W. This reflects the principle of Occam's razor: a derivation in an explanation is involved when it is necessary.*

Let us next revisit our Example 4.1

Example 4.4 (Example 4.1 cont'd)**.** *The answer set M of Π has multiple $\alpha^\star$-witnesses, including*

- *$G_1 = (V_1,\emptyset)$ with $V_1 = \{(\{a,b\},\Pi)\}$;*
- *$G_2 = (\{v_1,v_2\},\{(v_1,v_2)\})$ with $v_1 = (\{a\},\{r_1,r_2\})$ and $v_2 = (\{b\},\{r_3\})$;*
- *$G_3 = (\{v_1',v_2'\},\{(v_1',v_2')\})$ with $v_1' = (\{b\},\{r_1,r_3\})$ and $v_2' = (\{a\},\{r_2\})$;*
- *$G_4 = (\{v_1,v_1'\},\emptyset)$.*

As for G_1, clearly $\Pi \models a \wedge b$; as for G_2, $r_1 \wedge r_2 \models a$ and $r_3 \wedge a \models b$, while $r_1 \wedge r_2 \not\models b$. As for G_3, similarly $r_1 \wedge r_3 \models b$ and $r_2 \wedge b \models a$, while $r_1 \wedge r_3 \not\models a$. For G_4, we then also obtain that the conditions of an $\alpha^\star$-witness are fulfilled; notice that $D_{G_4}(v_1) = D_{G_4}(v_1') = \emptyset$. If we remove in G_4 the rules r_1 from v_1, i.e. replace v_1 by v_2', then the resulting sequence G_4' is not an $\alpha^\star$-witness, as a cannot be derived from r_2 alone and $D_{G_4}(v_2') = \emptyset$. Notice that G_1, G_2 and G_3 are α-witnesses, and can thus be written as $W_1 = [(\{a,b\},\Pi)]$, $W_2 = [(\{a\},\{r_1,r_2\}),(\{b\},\{r_3\})]$, and $W_3 = [(\{b\},\{r_1,r_3\}),(\{a\},\{r_2\})]$, respectively, The minimality property of an $\alpha, \alpha^\star$-witness ensures that there are no redundant rules in the witness components Π_i and the compactness property that moreover no rule can occur twice *e.g.* in Example 4.3, W is minimal and compact. Compactness is in particular relevant for disjunctive rules, which may be used for deriving (combined with other rules) different atoms in their heads.

Example 4.5 (Example 4.4 cont'd)**.** *The $\alpha^\star$-witnesses G_1, G_2 and G_3 of M w.r.t. Π are all compact, which are also compact α-witnesses W_1, W_2 and W_3 as in Example 4.4. Moreover, G_2 and G_3 are also compact β-witnesses, that will be defined later.*

4.2.1 Constructing $\alpha^\star$-witnesses and specialisations

A natural question is whether each answer set M of a logic program has some $\alpha^\star$-witness, and in particular whether always some compact $\alpha^\star$-witness exists. The answer is positive[8] and the next proposition shows that a compact $\alpha^\star$-witness can be obtained from the collapsed dependency graph of the reduct $\mathrm{MR}(\Sigma, M)$ based on the following lemma:

Lemma 4.1. *Let Σ be a clause theory and $M \neq \emptyset$ a minimal model of Σ with $\Sigma = \mathrm{MR}(\Sigma, M)$. Then $G = (\{(S_i, \Sigma_i) \mid 1 \leq i \leq n\}, E')$ obtained from $\mathbb{S}_\Sigma = (\{S_1, \ldots, S_n\}, E)$ s.t.*

- *$\Sigma_i \in \mathrm{MW}(S_i, \Sigma, S_{\triangleright i}, M)$ $(1 \leq i \leq n)$, where $S_{\triangleright i} = \bigcup D_{\mathbb{S}_\Sigma}(S_i)$, and*
- *$((S_i, \Sigma_i), (S_j, \Sigma_j)) \in E'$ whenever $(S_i, S_j) \in E$*

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

The next proposition shows that every answer set of a logic program Π has some syntax-guided compact $\alpha^\star$-witness *w.r.t.* Π.

Proposition 4.3. *Let $M \neq \emptyset$ be an answer set of a logic program Π and $\Sigma = \mathrm{MR}(\Pi, M)$. Then the DAG $G = (\{(S_i, \Pi_i) \mid 1 \leq i \leq n\}, E')$ obtained from $\mathbb{S}_\Sigma = (\{S_1, \ldots, S_n\}, E)$ s.t.*

- *$\Pi_i \in \mathrm{MW}(S_i, \Pi, S_{\triangleright i}, M)$ $(1 \leq i \leq n)$, where $S_{\triangleright i} = \bigcup D_{\mathbb{S}_\Sigma}(S_i)$, and*
- *$((S_i, \Pi_i), (S_j, \Pi_j)) \in E'$ whenever $(S_i, S_j) \in E$*

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

We call a compact $\alpha^\star$-witness as in Proposition 4.3 a *full-split $\alpha^\star$-witness*, denoted as $\alpha^\star_{fs}$*-witness*, of M *w.r.t.* Π.

Example 4.6 (Example 4.4 cont'd)**.** *For the logic program $\Pi = \{r_1 : a \vee b, \quad r_2 : a \leftarrow b, \quad r_3 : b \leftarrow a\}$ and $M = \{a, b\}$, we have that $\Sigma = \mathrm{MR}(\Pi, M) = \Pi$ and thus $\mathbb{S}_\Sigma = (\{\{a, b\}\}, \emptyset)$. Consequently, G_1 is an $\alpha^\star_{fs}$-witness of M w.r.t. Π. In fact, G_1 is the only such $\alpha^\star_{fs}$-witness.*

We note that while in the above example, the full-split $\alpha^\star_{fs}$-witness of M *w.r.t.* Π is unique, in general multiple distinct full-split $\alpha^\star_{fs}$-witnesses are possible.

[8]It is indeed trivial since $G = (\{(M, \Pi')\}, \emptyset)$ with Π' is a minimal subset of Π such that $\mathrm{MR}(\Pi', M) \models M$ is a compact $\alpha^\star$-witness of M *w.r.t.* Π. However, deciding whether some $\alpha^\star$-witness with more than one node exists is more involving; this is clearly in Σ^p_2, while a matching lower bound is open.

Example 4.7. *Consider the logic program* Π *consisting of the rules:*

$$r_1 : c, \quad r_2 : a \leftarrow c, \quad r_3 : b \leftarrow c, \quad r_4 : b \leftarrow a, \quad r_5 : a \leftarrow b, \quad r_6 : c \leftarrow b$$

and its answer set $M = \{a,b,c\}$. *Then* $\Sigma = \mathrm{MR}(\Pi, M) = \Pi$ *and* $\mathbb{S}_\Sigma = (\{\{a,b,c\}\}, \emptyset)$, *as all atoms in* Π *mutually depend on each other. It is not hard to check that* $G = (\{(\{a,b,c\}, \{r_1, r_2, r_4,\})\}, \emptyset)$ *and* $G' = (\{(\{a,b,c\}, \{r_1, r_3, r_5,\})\}, \emptyset)$ *are both* $\alpha^\star_{fs}$*-witnesses of* M *w.r.t.* Π. *Note that* r_1 *in* Π *may be replaced by a rule* $c \vee d$ *or* $c \leftarrow not\ d$*; the presence of facts in* Π *is thus not important for having multiple* $\alpha^\star_{fs}$*-witnesses.*

The $\alpha^\star_{fs}$ witnesses are relevant from an evaluation perspective, since the SCCs of a logic program Π are in practice often the basis for modular evaluation, where the answer sets of Π can be built, starting from the sources of the collapsed dependency graph $\mathbb{S}_{G_\Pi}$, incrementally along its edges. Proposition 4.3 ensures that we can find a minimal witness for the SCC S_i at hand without reusing any of the rules that have been used for witness formation in the SCCs S_j on which S_i depends. The same generalizes if we merge SCCs appropriately.

Given an $\alpha^\star$-witness $G = (V, E)$ of an answer set M *w.r.t.* a logic program Π, we call a graph $G' = (V', E')$ an *edge-contraction* of G, if some edge $(v_1, v_2) \in E$ exists, where $v_i = (S_i, \Pi_i)$, $i = 1, 2$, such that $V' = V \setminus \{v_1, v_2\} \cup \{v\}$ where $v = (v_1 \cup v_2, \Pi_1 \cup \Pi_2)$ and

$$\begin{aligned} E' = E \cap (V \setminus \{v_1, v_2\})^2 &\cup \{(v, v') \mid (v'', v') \in E, v'' \in \{v_1, v_2\}, v' \notin \{v_1, v_2\}\} \\ &\cup \{(v', v) \mid (v', v'') \in E, v'' \in \{v_1, v_2\}, v' \notin \{v_1, v_2\}\}, \end{aligned}$$

i.e., the neighboured nodes v_1 and v_2 are merged into v_1 and the edges are adjusted. Then we obtain:

Proposition 4.4. *Let* G *be an* $\alpha^\star_{fs}$*-witness of an answer set* M *w.r.t. a logic program* Π *and* $G = G_0, G_1, G_2, \ldots, G_k$ *be graphs such that each* G_i, $i = 1, \ldots, k$ *is an edge-contraction of* G_{i-1}. *Then* G_k *is a compact* $\alpha^\star$*-witness of* M *w.r.t.* Π, *where the edge-contraction of* $((S, \Pi), (S', \Pi'))$ *is the node* $(S \cup S', \Pi \cup \Pi')$.

As a consequence, for modular program evaluation[9] where the evaluation units comprise as customary single or multiple SCCs of a logic program Π, compact $\alpha^\star$-witnesses for an answer set $M = M_1 \cup \cdots \cup M_k$ can be simply composed from the compact $\alpha^\star_{fs}$-witnesses of M_i $(1 \le i \le k)$, where M_i is an answer set of a module Π_i of Π. Notably, the SCCs for modular evaluation of Π take in addition to the positive dependencies also the negative dependencies into account, i.e., the (full) dependency graph has edges (p, q) if $p \in r^-$ or $p \in r^{not}$, and $q \in r^+$ for some rule $r \in \Pi$ as well; thus each SCC S'_i of the full dependency graph of Π contains each SCC S_j of the positive dependency graph of Π *w.r.t.* M such that $S_j \cap S'_i \neq \emptyset$. Consequently, by merging all the respective nodes (S_j, Π_j) of G into one node $(S'_i \cap M, \Pi'_i)$ and adjusting the edges, a compact $\alpha^\star$-witness G' of M *w.r.t.* Π is obtained such that Π'_i witnesses $S'_i \cap M$.

4.3 $\beta^\star$- and β-witnesses

From Example 4.4, we can see that G_2 and G_3 are more fine-grained than G_1 in the sense that the witnesses in G_2 and G_3 have more refined dependencies than the one of G_1. In particular, the $\alpha^\star_{fs}$-witness is not appropriate for normal or headcycle-free logic programs since the justification for answer sets in this case has a more natural formalism in terms of the resolution proof when the new reduct is applied. For instance, $M = \{a, b\}$ is the unique answer set of $\Pi = \{r_1 : a \leftarrow b, \quad r_2 : b \leftarrow a, \quad r_3 : a \leftarrow not\ c\}$. Note that $G = (V, \emptyset)$ with $V = \{(\{a,b\}, \{r_2, r_3\})\}$ is the unique $\alpha^\star_{fs}$-witness of M *w.r.t.* Π since $\mathbb{S}_{\mathrm{MR}(\Pi,M)} = (\{\{a,b\}\}, \emptyset)$. It means that a

[9]Each answer set of a modular program Π can be composed from the answer sets of the modules of Π.

and b are collectively justified by $\{r_2,r_3\}$. In fact, the ideal witness for M should be $[(\{a\},\{r_3\}),(\{b\},\{r_2\})]$. It means that a and b are separately and sequentially justified. This motivates the notion of $\beta^{\star}$-witness and specializations.

Definition 4.3 ((minimal, compact) $\beta^{\star}$- and β-witness)**.** *Let Π be a logic program, $M \neq \emptyset$ an answer set of Π, and B,S be disjoint subsets of M. Then every $\alpha^{\star}$-witness $G = (\{(S_i,\Pi_i) \mid 1 \leq i \leq n\},E)$ of B under Π and S w.r.t. M such that $S_i = \{p_i\}$ for some atom p_i, for all $(1 \leq i \leq n)$, is a $\beta^{\star}$-witness of B under Π and S w.r.t. M, also written as $G = (\{(p_i,\Pi_i) \mid 1 \leq i \leq n\},E)$. The notions of β-witness and minimal / compact $\beta^{\star}$- resp. β-witness of B under Π and S w.r.t. M are defined analogously from α-witness and minimal / compact $\alpha^{\star}$- resp. α-witness of B under Π and S w.r.t. M.*

Thus, the notion of $\beta^{\star}$-witness captures the important case where the atoms p_i in B are justified one by one, while the notion of α- (resp. β-) witness is the restriction that the parts S_i resp. atoms p_i of B can be explained in a sequential order.

Example 4.8 (Example 4.4, cont'd)**.** *In Example 4.4, $G_2 - G_4$ are $\beta^{\star}$-witnesses, and thus can be written as*

- $\bar{G}_2 = (\{\bar{v}_1,\bar{v}_2\},\{(\bar{v}_1,\bar{v}_2)\})$ *with* $\bar{v}_1 = (a,\{r_1,r_2\})$ *and* $\bar{v}_2 = (b,\{r_3\})$*;*
- $\bar{G}_3 = (\{\bar{v}'_1,\bar{v}'_2\},\{(\bar{v}'_1,\bar{v}'_2)\})$ *with* $\bar{v}'_1 = (b,\{r_1,r_3\})$ *and* $\bar{v}'_2 = (a,\{r_2\})$*;*
- $\bar{G}_4 = (\{\bar{v}_1,\bar{v}'_1\},\emptyset)$

by replacing $\{a\}$ and $\{b\}$ with a and b, respectively, in v_1, v_2, v'_1, and v'_2. As $\bar{G}_2$ and $\bar{G}_3$ are also β-witnesses, they can be written as $W_2 = [(a,\{r_1,r_2\}),(b,\{r_3\})]$ and $W_3 = [(b,\{r_1,r_3\}),(a,\{r_2\})]$. Each of $\bar{G}_2$, $\bar{G}_3$ and $\bar{G}_4$ is minimal, but only $\bar{G}_2$ and $\bar{G}_3$ are compact, as r_1 occurs in both local witnesses $\{r_1,r_3\}$ and $\{r_1,r_2\}$ of $\bar{G}_4$. Furthermore, if the nodes v_1 and $\bar{v}'_1$ of $\bar{G}_4$ would be ordered, e.g. to $\bar{v}_1 < \bar{v}'_1$, then the resulting β-witness would not be minimal.

Recall that $T^{\infty}_{\Pi^M} = M$ where M is an answer set of a normal logic programs Π. One can build up a compact β-witness of M *w.r.t.* Π by setting $W = [(p_{1,1},\emptyset),\dots(p_{1,k_1},\emptyset)] + \dots + [(p_{n,1},\Pi_{n,1}),\dots,(p_{n,k_n},\Pi_{n,k_n})]$, where "$+$" denotes concatenation of sequences and n is the least number satisfying $T^{n}_{\Pi^M} = M$ and, for $1 \leq i \leq n$,

- $T^{i}_{\Pi^M} - T^{i-1}_{\Pi^M} = \{p_{i,1},\dots,p_{1,k_i}\}$,
- $\{r_{i,j}\} = \Pi_{i,j} \subseteq \Pi$, $M \models r_{i,j}$ and $r^{-}_{i,j} \subseteq T^{i-1}_{\Pi^M}$ for $i > 1$.

Note further that if $[(p_1,\Pi_1),\dots,(p_n,\Pi_n)]$ is a minimal β-witness of $M = \{p_1,\dots,p_n\}$ *w.r.t.* Π, then $|\Pi_i| = 1$ $(1 \leq i \leq n)$. Otherwise Π_i is a witness of p_j $(j > i)$ under $\{p_1,\dots,p_{j-1}\}$ for some $r_j \in \Pi_i$ with $r^{+}_j = \{p_j\}$, since $p_j \notin \{p_1 \dots, p_{i-1}\}$. We obtain the following corollary.

Corollary 4.1. *Let $M \neq \emptyset$ be an answer set of a normal logic program Π. Then*

(i) There is a compact β-witness of M w.r.t. Π.

(ii) For every minimal β-witness $[(p_1,\Pi_1),\dots,(p_n,\Pi_n)]$ of M w.r.t. Π, it holds that $|\Pi_i| = 1$ $(1 \leq i \leq n)$.

Unlike α-witnesses, a compact β-witness may not always exist for some answer sets of disjunctive logic programs, as illustrated by the following example.

Algorithm 5: MinBetaBSWitness(Π, B, S, M)

```
Input: A logic program $\Pi$, an answer set $M \neq \emptyset$ of $\Pi$, and two disjoint subsets $B, S$ of $M$
Output: A minimal $\beta$-witness of $B$ under $\Pi$ 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
       /* unit propagation */
3      while $\exists \alpha \in \Sigma$ s.t. $\alpha^- \subseteq T$, $\alpha^+ = \{p\}$, $p \in M - T$ do
4          $T \leftarrow T \cup \alpha^+$;
5          $\Pi'.append((p, rm(\{\alpha\}, \Pi, M)))$;
           /* $rm(\Sigma, \Pi, M)$ returns a minimal subset $\Pi'$ of $\Pi$ s.t. $\mathrm{MR}(\Pi', M) = \Sigma$ */
6      end
7      if $M - T = \emptyset$ then break Let $u \in B - T$;
8      Let $\Sigma_u \leftarrow$ MinWitness$(\{u\}, \Sigma, T, M)$;
9      while $\Sigma_u \cup T \models v$ for some atom $v \notin \{u\} \cup T$ do
10         Let $\Sigma_v \leftarrow$ MinWitness$(\{v\}, \Sigma_u, T, M)$;
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 $\Pi'$;
```

Example 4.9. *Let $M = \{p, q, r\}$ and Π consist of*

$$\delta_1 : p \vee q \vee r, \quad \delta_2 : p \leftarrow q, \quad \delta_3 : q \leftarrow r, \quad \delta_4 : r \leftarrow p.$$

Then M is an answer set of Π and that $\Sigma = \mathrm{MR}(\Pi, M) = \Pi$. The sequence $[(r, \{\delta_1, \delta_2, \delta_4\}), (q, \{\delta_3\}), (p, \{\delta_2\})]$ is a minimal β-witness of M w.r.t. Σ. For the atom r, the unique $\Sigma' \subseteq \Sigma$ s.t. $\Sigma' \models_{min} p$ is $\Sigma' = \{\delta_1, \delta_2, \delta_4\}$. The clause δ_2 must be reused to derive p. All other minimal β-witnesses require a similar reuse. Thus, M has no compact β-witness w.r.t. Π. One can similarly verify that M has no compact $\beta^\star$-witness w.r.t. Π yet.

4.3.1 Constructing $\beta^\star$-witnesses and specialisations

The algorithm MinBetaBSWitness, shown as Algorithm 5, serves to compute a minimal β-witness of B under Π relative to S asserted as facts *w.r.t.* M. In the case $B = M$ and $S = \emptyset$ it computes a minimal β-witness of M *w.r.t.* Π.

Intuitively, it loops in the outer **while** until each atom p in M gets a minimal witness (i.e., is in T). The first inner **while-loop** (3-6) identifies such p by unit propagation, while the second inner **while** (10-14) searches for an atom u and some minimal witness Σ_u of it under T such that no atom v outside $T \cup \{u\}$ is entailed by $\Sigma_u \cup T$; in line (15) then $rm(\Sigma_u, \Pi, M)$ yields a minimal subset Π' of Π such that $\mathrm{MR}(\Pi', M) = \Sigma_u$. The following example demonstrates a run of the algorithm.

Example 4.10. *Let $M = \{p, q, r, s\}$ and consider the positive logic program*

$$\Pi = \{\ \alpha_1 : p \vee r,\ \ \alpha_2 : p \vee q \leftarrow r,\ \ \alpha_3 : r \vee s \leftarrow p,\ \ \alpha_4 : p \leftarrow q,\ \ \alpha_5 : q \leftarrow p,\ \ \alpha_6 : r \leftarrow s,\ \ \alpha_7 : s \leftarrow r\}.$$

One can check that M is an answer set of Π, hence a minimal model of $\Sigma = \mathrm{MR}(\Pi,M) = \Pi$. As all atoms in Σ do mutually depend on each other, the collapsed dependency graph of Σ is $\mathbb{S}_\Sigma = (\{\{p,q,r,s\}\},\emptyset)$. Furthermore, no proper subset $\Sigma' \subset \Sigma$ can derive all atoms in M. Hence an $\alpha^\star$-witness of M w.r.t. Σ is $G = (\{(\{p,q,r,s\},\Sigma)\},\emptyset)$, which is evidently compact. An execution of MinBetaBSWitness$(\Sigma,M,\emptyset,M)$ *is illustrated in Table 1. Please note here that $B = M$. It is not difficult but tedious to check that* MinBetaBSWitness$(\Sigma,M,\emptyset,M)$ *may compute the compact β-witness*

$$W = [(p,\{\alpha_1,\alpha_2,\alpha_4\}),(q,\{\alpha_5\}),(s,\{\alpha_3,\alpha_7\}),(r,\{\alpha_6\})] \tag{8}$$

of M w.r.t. Σ in the following steps according to Table 1:

(1) In the first iteration of the outer **while-loop** *(lines 2-17), the first inner* **while-loop** *(lines 3-6) makes no contribution for T; then the atom $u = s$ is chosen from $B - T$ and a minimal witness Σ_u is built up; finally, the second inner* **while-loop** *(lines 10-14) finds the atom $v = p$ with $\Sigma_u \cup T \models v$ and builds up a minimal witness $\Sigma_u = \{\alpha_1,\alpha_2,\alpha_4\}$.*

(2) In the second iteration of the outer **while-loop**, *unit propagation finds the rule $\alpha = \alpha_5$ and builds up the minimal witness for q; then the atom $u = s$ is chosen again from $B - \{p,q\}$ and a minimal witness $\Sigma_u = \{\alpha_3,\alpha_7\}$ is built up; the second inner* **while-loop** *finds no atom v with $\Sigma_u \cup T \models v$ with $v \notin T \cup \{u\}$; then a minimal witness Σ_u is established.*

(3) In the third iteration of the outer **while-loop**, *unit propagation finds the rule α_6 and builds the minimal witness for the atom r; the outer* **while-loop** *terminates at line 7 due to $B - T = \emptyset$.*

Regarding the termination of algorithm MinBetaBSWitness, the outer **while**-loop and the first inner **while**-loop (3-6) clearly terminate, as gradually atoms p respectively u from $B \setminus T$ are added to T; for the second inner **while**-loop (10-14), we must be sure that the search for a minimal witness Σ_v in line (11) makes progress, i.e., that $\Sigma_v \neq \Sigma_u$ (and thus $\Sigma_v \subset \Sigma_u$) holds. The next lemma asserts that this is indeed the case.

Lemma 4.2. *Let l be a literal and let Σ be a satisfiable clause theory s.t. $\Sigma \models l$ and no $\Sigma' \subset \Sigma$ satisfies $\Sigma' \models l$. Then:*

(i) the opposite literal $\bar{l}$ does not occur in Σ, and

(ii) if $\Sigma \models l'$ for some literal $l' \neq l$, then some proper subset $\Sigma' \subset \Sigma$ exists such that $\Sigma' \models l'$.

Armed with this lemma, we then prove the correctness of Algorithm 5, which is stated in the following Proposition.

Table 1: A possible run of MinBetaBSWitness$(\Sigma,M,\emptyset)$

while(2-17)	**while**(3-6)		u	Σ_u	$\mathrm{CnA}(\Sigma_u \cup T) =$	**while**(10-14)		T
#i = i-th iteration	α	T			$\{p \in \mathscr{A} \mid \Sigma_u \cup T \models p\}$	u	Σ_u	
#1		$\emptyset$	s	$\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_7\}$	$\{p,s\}$	p	$\{\alpha_1,\alpha_2,\alpha_4\}$	$\{p\}$
#2	α_5	$\{p,q\}$	s	$\{\alpha_3,\alpha_7\}$	$\{p,q,s\}$	s	$\{\alpha_3,\alpha_7\}$	$\{p,q,s\}$
#3	α_6	$\{p,q,s,r\}$						

Algorithm 6: MinBetaWitness(Π, M)

```
Input: A logic program $\Pi$, an answer set $M \neq \emptyset$ of $\Pi$
Output: A minimal $\beta$-witness of $M$ w.r.t. $\Pi$
1 $\Pi' \leftarrow [\,]$;   $\Sigma \leftarrow \mathrm{MR}(\Pi, M)$;   $G \leftarrow \mathbb{SG}_\Sigma$;
2 Recursively delete from $G$ all empty sources;
3 while $G$ has a source $S$ do
4     $U \leftarrow \mathscr{A} \cap (\bigcup D_{\mathbb{SG}_\Sigma}(S))$;
5     $\Pi'$.append(MinBetaBSWitness$(\Pi, S \cap \mathscr{A}, U, M)$);
6     Delete $S$ from $G$;
7     Recursively delete from $G$ all empty sources;
8 end
9 return $\Pi'$;
```

Proposition 4.5. *Given a logic program Π, an answer set $M \neq \emptyset$ of Π and two disjoint subsets B, S of M, the call of MinBetaBSWitness(Π, B, S, M) returns some minimal β-witness W of B under Π and S w.r.t. M.*

For computing some a β-witness of an answer set of a logic program, we present the algorithm MinBetaWitness(Π, M), shown as Algorithm 6. It makes use of minimal model decomposition property with reduct (Theorem 3). Its correctness is guaranteed by Proposition 4.5 and the minimal model decomposition theorem (Proposition 3.4).

Proposition 4.6. *Given a logic program Π and an answer set $M \neq \emptyset$ of Π, the call of MinBetaWitness(Π, M) returns some minimal β-witness W of M w.r.t. Π.*

Example 4.11. *Let Π be the logic program consisting of*

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

and $M = \{a, b, c\}$. It is evident that M is the unique answer set of Π and $\Sigma = \mathrm{MR}(\Pi, M)$. Note that $G = \mathbb{SG}_\Sigma = (V, E)$ with

- *$V = \{v_1, v_2, v_3\}$ with $v_1 = \{r_1, r_2, r_3, a, b\}$, $v_2 = \{r_4\}$, $v_3 = \{c\}$;*
- *$E = \{(v_1, v_2), (v_2, v_3)\}$.*

The unique source of $\mathbb{SG}_\Sigma$ is v_1. In the first iteration of the loop (lines 3-8), MinBetaBSWitness computes $U = \emptyset$ and a minimal β-witness of $v_1 \cap \{a, b, c\} = \{a, b\}$ under Π and $\emptyset$ w.r.t. S by calling MinBetaBSWitness$(\Pi, \{a, b\}, \emptyset, M)$; the latter may return $[(a, \{r_1, r_2\}), (b, \{r_3\})]$ or $[(b, \{r_1, r_3\}), (a, \{r_2\})]$. Suppose $\Pi' = [(a, \{r_1, r_2), (b, \{r_3\})]$. MinBetaBSWitness then removes the source v_1 from G (line 6). Now v_2 is an empty source of G and will be deleted (line 7). The unique remaining nonempty source of G is v_3. In the second iteration of the loop (lines 3-8), MinBetaBSWitness computes $U = \{a, b\}$ and a minimal β-witness of $\{c\}$ under Π and $\{a, b\}$ w.r.t. S by calling MinBetaBSWitness$(\Pi, \{c\}, \{a, b\}, M)$, which definitely returns $[(c, \{r_4\})]$. After removing v_3 from G, there is no source. Thus, the final minimal β-witness Π' is

$$[(a, \{r_1, r_2\}), (b, \{r_3\}), (c, \{r_4\})] \tag{9}$$

For $\Pi' = [(b, \{r_1, r_3\}), (a, \{r_2\})]$, we similarly obtain the final minimal β-witness

$$[(b, \{r_1, r_3\}), (a, \{r_2\}), (c, \{r_4\})]. \tag{10}$$

Algorithm 7: MinBetaStarWitness(Π, M)

```
    Input: A logic program $\Pi$ and an answer set $M$ of $\Pi$
    Output: A minimal $\beta^\star$-witness of $M$ w.r.t. $\Pi$
 1  Let $[(p_i,\Sigma_i) \mid 1 \leq n]$ be the output of MinBetaWitness$(\Pi, M)$;
 2  $V \leftarrow \{(p_i,\Sigma_i) \mid 1 \leq i \leq n\}$;   $E \leftarrow \emptyset$;   $G \leftarrow (V,E)$;
 3  for $i = 2$ to $n$ do
 4      $\Delta \leftarrow \bigcup\{\alpha^- \mid \alpha \in \Sigma_i\}$;
 5      while $\exists q, q' \in \Delta \cap \{p_1,\ldots,p_{i-1}\}$ such that $q \neq q'$ do
 6          if $G$ has a path from $(q,\Sigma_q)$ to $(q',\Sigma_{q'})$ then $\Delta \leftarrow \Delta - \{q\}$
 7      end
 8      foreach $q \in \Delta \cap \{p_1,\ldots,p_{i-1}\}$ do $E \leftarrow E \cup \{((q,\Sigma_q),(p_i,\Sigma_i))\}$
 9  end
10  return $G$;
```

We note that the β-witness form $W = [(p_1,S_i),\ldots,(p_n,S_n)]$ intuitively means that p_i "depends on" p_j $(j < i)$. According to (9), b depends on a in the first β-witness, while a depends on b in the second β-witness; c depends on a and b in both β-witnesses. However, an atom $p_i \in B$ may not necessarily depend on $p_j \in B$ for some $j < i$. For instance, if the rule r_4 in the above example is replaced by "$c \leftarrow b$" then one may still have the minimal β-witnesses (9) and (10). Nevertheless, c "directly" depends on b only, while it "indirectly" depends on a via b. The notion of $\beta^\star$-witness allows to faithfully capture this dependency relation.

Example 4.12 (Example 4.11 cont'd)**.** *We obtain a minimal $\beta^\star$-witness from Π' in (9) by defining the graph $G = (V,E)$, where $V = \{v_1,v_2,v_3\}$ with $v_1 = (a,\{r_1,r_2\})$, $v_2 = (b,\{r_3\})$, $v_3 = (c,\{r_4\})$ and $E = \{(v_1,v_2),(v_2,v_3)\}$. Note that $X_1 = \emptyset$, $X_2 = \{a\}$, and $X_3 = \{b\}$. From Π' in (10), we obtain similarly a minimal $\beta^\star$-witness, by defining $G = (V,E)$ with $V = \{v_1,v_2,v_3\}$ with $v_1 = (b,\{r_1,r_3\})$, $v_2 = (a,\{r_2\})$, $v_3 = (c,\{r_4\})$ and $E = \{(v_1,v_2),(v_1,v_3)\}$; here $X_1 = \emptyset$, and $X_2 = X_3 = \{b\}$.*

To compute a minimal $\beta^\star$-witness of an answer set M for a logic program Π (i.e., under Π and $B = M$, hence $S = \emptyset$), we can compute a minimal β-witness of M and then distill its true dependencies by Algorithm 7. The next proposition shows that this algorithm is correct and the returned $\beta^\star$-witness has no redundant dependency.

Proposition 4.7. *Let $M \neq \emptyset$ be an answer set of a logic program Π and let furthermore be G the output of MinBetaStarWitness(Π, M).*

(i) G is a minimal $\beta^\star$-witness of M w.r.t. Π.

(ii) G has no redundant edges, i.e., if removing an edge $((p_i,\Sigma_i),(p_j,\Sigma_j))$ from G then Σ_j is not a witness of p_j under X w.r.t. M, where $X = \{p \mid (p,\Sigma) \in D_G((p_j,\Sigma_j))$.

The following example illustrates how a minimal $\beta^\star$-witness can be gradually evaluated by Algorithm 7.

Example 4.13. *Consider the logic program Π consisting of the rules*

$$\delta_1 : a \leftarrow not\ b, \quad \delta_2 : b \leftarrow not\ a, \quad \delta_3 : r \leftarrow p, a, \quad \delta_4 : p \vee q, \quad \delta_5 : p \leftarrow q, \quad \delta_6 : q \leftarrow p.$$

It holds that $M = \{p,q,a,r\}$ *is an answer set of* Π *and* $\Pi^M = \{\delta_3,\ldots,\delta_6\} \cup \{\delta_1' : a\}$. *The* $\alpha^\star_{fs}$*-witness of* M *w.r.t.* Π $(\{v_1, v_2^*, v_4\}, \{(v_1,v_4),(v_2^*,v_4)\})$ *is illustrated in Fig. 2, in which the compact* β*-witness for each vertex is attached. Two compact* β*-witnesses of* $\{p,q\}$ *under* Π *and* $\emptyset$ *w.r.t.* M *are* $[v_2 : (p,\{\delta_4,\delta_5\}), v_3 : (q,\{\delta_6\})]$ *and* $[u_2 : (q,\{\delta_4,\delta_6\}), u_3 : (p,\{\delta_5\})]$. *Suppose* $\mathsf{MinBetaWitness}(\Pi,M)$ *outputs the following minimal* β*-witness of* M *w.r.t.* Π*:*

$$[v_1 = (a,\{\delta_1\}), v_2 = (p,\{\delta_4,\delta_5\}), v_3 = (q,\{\delta_6\}), v_4 = (r,\{\delta_3\})].$$

There are three iterations of the **for** *loop (lines 3-9) in MinBetaStarWitness:*

(1) $\Sigma_2 = \{\delta_4,\delta_5\}$ *and* $\Delta = \{q\}$. *As* $\Delta \cap \{p_1\} = \{q\} \cap \{a\} = \emptyset$, *the condition of the* **while** *loop (line 5) is not satisfied, and in line 8 no edge is added to E;*

(2) $\Sigma_3 = \{\delta_6\}$ *and* $\Delta = \{p\}$. *As* $\Delta \cap \{p_1,p_2\} = \{p\} \cap \{a,p\} = \{p\}$, *the condition of the* **while** *loop (line 5) is not satisfied; in line 8, the edge* (v_2,v_3) *is added to E by the* **foreach** *loop.*

(3) $\Sigma_4 = \{\delta_3\}$ *and* $\Delta = \{p,a\}$. *As* $\Delta \cap \{p_1,p_2,p_3\} = \{p,a\} \cap \{a,p,q\} = \{p,a\}$, *the condition of the* **while** *loop (line 5) is again not satisfied; in line 8, the edges* (v_1,v_4) *and* (v_2,v_4) *are added to E by the* **foreach** *loop.*

Consequently, the corresponding minimal $\beta^\star$*-witness* $G = (V,E)$ *of* M *w.r.t.* Π *is built:*

- $V = \{v_1,v_2,v_3,v_4\}$, *and* $E = \{(v_2,v_3),(v_2,v_4),(v_1,v_4)\}$.

Similarly, when $\mathsf{MinBetaWitness}(\Pi,M)$ *outputs the following minimal* β*-witness of* M *w.r.t.* Π*:*

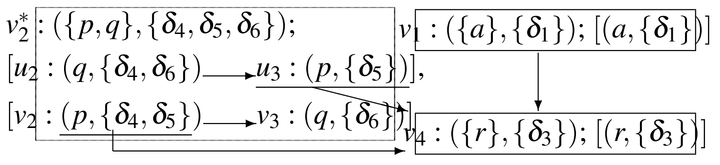
$$[v_1 = (a,\{\delta_1\}), u_2 = (q,\{\delta_4,\delta_6\}), u_3 = (p,\{\delta_5\}), v_4 = (r,\{\delta_3\})],$$

MinBetaStarWitness builds up the minimal $\beta^\star$*-witness* $G' = (V',E')$ *with* $V' = \{v_1,u_2,u_3,v_4\}$ *and* $E' = \{(u_2,u_3),\ (u_3,v_4),\ (v_1,v_4)\}$. *In fact, the two minimal* $\beta^\star$*-witnesses are compact.*

4.4 Relationship among witness notions

We have so far presented various notions of witnesses. The α- and $\alpha^\star$- witnesses, and the $\alpha^\star_{fs}$-witnesses in particular, provide a very coarse-grained justification for answer sets of logic programs. The β- and $\beta^\star$- witnesses justify answer sets in the finest-grained way. When an answer set of a logic program can be decomposed into components in terms of Theorem 1, one can flexibly choose between α-witness and β-witness on demand.

The different notions of witnesses from above are related among each other as follows. For convenience, we denote for each (minimal resp. compact) $w \in \{\alpha,\beta,\alpha^\star,\beta^\star,\alpha^\star_{fs}\}$ with (min- resp. comp-) $w(\Pi,M)$ the collection of (minimal resp. compact) w-witnesses of M *w.r.t.* Π, respectively. Then we have



v_2^* : $(\{p,q\},\{\delta_4,\delta_5,\delta_6\})$;
$[u_2 : (q,\{\delta_4,\delta_6\}) \longrightarrow u_3 : (p,\{\delta_5\})]$,
$[v_2 : (p,\{\delta_4,\delta_5\}) \longrightarrow v_3 : (q,\{\delta_6\})]$
$v_1 : (\{a\},\{\delta_1\});\ [(a,\{\delta_1\})]$
$v_4 : (\{r\},\{\delta_3\});\ [(r,\{\delta_3\})]$

Figure 2: The $\alpha^\star/\beta^\star$-witness of M *w.r.t.* Π in Example 4.13

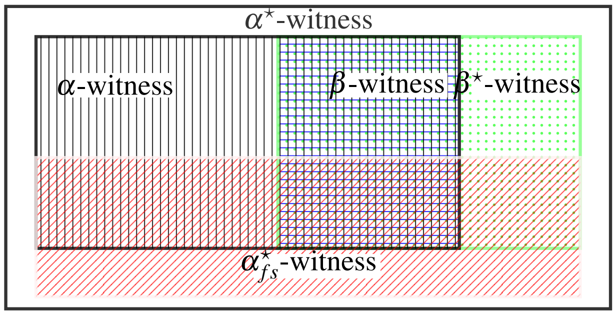


Figure 3: The relationships among α-, $\alpha^\star$-, β-, $\beta^\star$- and $\alpha^\star_{fs}$-witnesses
Legend: $\alpha^\star_{fs}$-witness is the area filled with north west red lines; α-witness is the area filled with vertical black lines; $\beta^\star$-witness is the area filled with green dots; β-witness is the area filled with horizontal blue lines;

Proposition 4.8 (Relationships among witnesses)**.** *Let $M \neq \emptyset$ be an answer set of a logic program Π. Then*

(i) $\alpha(\Pi,M) \cap \beta^\star(\Pi,M) = \beta(\Pi,M)$*;*

(ii) $\text{min-}\beta(\Pi,M) = \text{min-}\beta^\star(\Pi,M) \cap \text{min-}\alpha(\Pi,M) = \beta^\star(\Pi,M) \cap \text{min-}\alpha(\Pi,M) = \text{min-}\beta^\star(\Pi,M) \cap \alpha(\Pi,M)$*;*

(iii) $\text{comp-}\beta(\Pi,M) = \text{comp-}\beta^\star(\Pi,M) \cap \text{comp-}\alpha(\Pi,M) = \beta^\star(\Pi,M) \cap \text{comp-}\alpha(\Pi,M) = \text{comp-}\beta^\star(\Pi,M) \cap \alpha(\Pi,M)$.

Accordingly, the overall relationships among these α-, β-, $\alpha^\star$-, $\beta^\star$- and $\alpha^\star_{fs}$- witnesses can be illustrated in a Venn diagram shown in Fig 3. If an $\alpha^\star_{fs}$-witness has a graph $G = (V,E)$ that is a chain and S is a singleton for every (S,Π') in V, then the $\alpha^\star_{fs}$-witness can be regarded as a β-witness as well. What the relationships shown can be extended for the minimal and compact variants. For instance, if a minimal $\beta^\star$-witness is a minimal α-witness then it is also a minimal β-witness. Here, when we say that an $\alpha^\star$- (resp. $\beta^\star$-) witness is an α- (resp. β-) witness, we mean that the directed graph of $\alpha^\star$- (resp. $\beta^\star$-) witness is a chain. An interesting question is whether an $\alpha^\star$- (resp. $\beta^\star$-) witness W with an arbitrary directed graph $G = (V,E)$ amounts to some α- (resp. β-) witness, i.e., that some $\alpha^\star$- (resp. $\beta^\star$-) witness W' exists with an associated graph $G' = (V,E')$ that is a chain. If this is the case, we call W *serializable*. However, $\alpha^\star$- and $\beta^\star$-witnesses are not always serializable.

Example 4.14. *Let us consider the logic program Π in Example 4.9, which consists of*

$$\delta_1 : p \vee q \vee r, \quad \delta_2 : p \leftarrow q, \quad \delta_3 : q \leftarrow r, \quad \delta_4 : r \leftarrow p.$$

Then $M = \{p,q,r\}$ is an answer set of Π and $\mathrm{MR}(\Pi,M) = \Pi$. The DAG $G = (\{v_1,v_2,v_3\},\emptyset)$ with $v_1 = (p,\{\delta_1,\delta_2,\delta_3\}), v_2 = (q,\{\delta_1,\delta_3,\delta_4\}$ and $v_3 = (r,\{\delta_1,\delta_2,\delta_4\})$ is a minimal (but not compact) $\beta^\star$-witness of M w.r.t. Π. However, no serizalization of v_1,v_2,v_3 is a β-witness of M w.r.t. Π. For instance, let us consider $W = [v_1,v_2,v_3]$. The witness $\{\delta_1,\delta_3,\delta_4\}$ of q under $\emptyset$ is a witness of r under $\{p,q\}$. It means that the witness of q can replace the witness of r in the sequence. Thus, this sequence cannot be a β-witness of M w.r.t. Π, hence also not a minimal or compact β-witness of M w.r.t. Π.

As illustrated by the following example, it is also possible that a compact $\beta^\star$-witness is not serializable. And since a (minimal resp. compact) $\beta^\star$-witness can be a special case of a (minimal resp. compact) $\alpha^\star$-witness, (minimal resp. compact) $\alpha^\star$-witnesses cannot always be serializable either.

Example 4.15. *Let Π be the logic program consisting of*

$$r_1 : a \leftarrow d,\ \ r_2 : a \vee d \vee c,\ \ r_3 : a \leftarrow c,\ \ r_4 : b \leftarrow a,\ \ r_5 : c \leftarrow a,\ \ r_6 : a \vee b \vee c,\ \ r_7 : a \leftarrow b,\ \ r_8 : d \leftarrow c.$$

It is not difficult to check that $M = \{a,b,c,d\}$ is the unique answer set of Π and $\mathrm{MR}(\Pi,M) = \Pi$. The directed graph $G = (\{v_1,v_2,v_3,v_4\},\{(v_1,v_2),(v_3,v_4)\})$ with

$$v_1 = (a,\{r_1,r_2,r_3\}), v_2 = (b,\{r_4\}), v_3 = (c,\{r_5,r_6,r_7\}), v_4 = (d,\{r_8\})$$

is a compact $\beta^\star$-witness of M w.r.t. Π. However, there is not an order for v_1,v_2,v_3,v_4 which forms a minimal β-witness of M w.r.t. Π. The reason is that $\{r_1,r_3\}$ is a witness of $\{a\}$ under $\{c\}$ w.r.t. Π and $\{r_5\}$ is a witness of $\{c\}$ under $\{a\}$ w.r.t. Π. Thus, the minimality of minimal β-witness will be violated by any order $[v_1,\ldots,v_3,\ldots]$ or $[v_3,\ldots,v_1,\ldots]$. A minimal β-witness of M w.r.t. Π can be $W = [v_1,v_2,v_3',v_4]$ with $v_3' = (c,\{r_5\})$.

The next proposition shows that every $\alpha^\star_{fs}$-witness is seralizable.

Proposition 4.9. *Let Π be a logic program, M an answer set of Π and the DAG $G = (V,E)$ be an $\alpha^\star_{fs}$-witness of M w.r.t. Π. Then there is a compact α-witness $W = [w_1,\ldots,w_n]$ of M w.r.t. Π such that w_i $(1 \leq i \leq n)$ occurs in V.*

Note that the proof of this proposition shows in fact that every topological sorting of an $\alpha^\star_{fs}$-witness W leads to a corresponding compact α-witness. Informally, repeated edge-contractions of W fix successor positions in a topological sorting of W; it thus can be seen that each compact $\alpha^\star$-witness obtained by repeated edge-contractions is also serializable.

5 Complexity

In this section, we address complexity issues of the notions that we have introduced in the previous sections.

5.1 Recognizing witnesses

Let us first consider the problem of recognizing witnesses. Clearly, this problem is intractable for all kinds of witnesses that we have considered in the previous section, and all notions have the same complexity excepting the basic notion which has slightly lower complexity. Furthermore, in all cases the worst-case complexity holds if we start from an answer set M of a positive (negation-free) program Π that we want to explain. Specifically, we obtain the following results.

Proposition 5.1. *Deciding given logic programs Π,Π', sets S,B and an interpretation M, whether Π' is a witness of B under S w.r.t. M is* co-NP*-complete in general. The* co-NP*-hardness holds even if Π is positive, $S = \emptyset$, $B = M$ and M is an answer set of Π.*

The notion of minimal witness involves a further satisfiability test, viz. that by omitting any rule r in Π', B is no longer entailed, i.e., $\mathrm{MR}(\Pi' \setminus \{r\},M) \cup S \cup \{\neg p \mid p \in B\}$ is satisfiable. This raises the complexity of the recognition problem to D_1^p, which intuitively contains the problems that are expressible as the "conjunction" of two problems in NP and in co-NP, respectively, that are independent. The D_1^p-hardness for recognizing a minimal witness is immediate from the proof of Proposition 5.1, if we assume that Σ is a CRITICAL SAT (CSAT) instance, which is to decide whether a given clause theory Σ is unsatisfiable but always satisfiable if a single clause $c \in \Sigma$ is removed.

The notions of α- and β-witnesses with their variants all involve similarly unsatisfiability and satisfiability tests; as it turns out, they are all D_1^p-complete.

Theorem 5. *Let Π be a logic program Π. The following problems are D_1^p-complete:*

(i) deciding whether $\Pi' \in \mathrm{MW}(B,\Pi,S,M)$, with D_1^p-hardness if M is an answer set of Π;

(ii) deciding whether $W = [(S_1,\Pi_1),\ldots,(S_n,\Pi_n)]$ resp. $G = (\{(S_i,\Pi_i) \mid 1 \leq i \leq n\},E)$ is a (minimal, compact) α- resp. $\alpha^\star$-witness or $\alpha^\star_{fs}$-witness of an answer set M of Π;

(iii) deciding whether $W = [(p_1,\Pi_1),\ldots,(p_n,\Pi_n)]$ resp. $G = (\{(p_i,\Pi_i) \mid 1 \leq i \leq n\},E)$ is a (minimal, compact) β- resp. $\beta^\star$-witness of an answer set M of Π.

Furthermore, the D_1^p-hardness holds in all cases if Π is a positive (negation-free) program.

On the other hand, for normal logic programs Π, the witness test is feasible in polynomial time for all notions of witnesses that we considered, since the reduct $\mathrm{MR}(\Pi,M)$ of Π *w.r.t.* any interpretation M is Horn, and thus deciding the entailment problem $\mathrm{MR}(\Pi,M) \cup S \models B$ is feasible in polynomial time. This extends to headcycle-free programs, due to the following property. We note that, according to Theorem 2.3 of Ben-Eliyahu and Dechter [1994], for an answer set M of a headcycle-free program Π, each atom $p \in M$ has a *proof w.r.t.* M and Π, which is a sequence $r_1,\ldots,r_k$ of rules in Π such that

- $M \models bd(r_i)$ and $|r_i^+ \cap M| = 1$, for every i $(1 \leq i \leq k)$,
- $bd(r_1) = \emptyset$ and $r_i^- \subseteq r_1^+ \cup \cdots \cup r_{i-1}^+$, and
- $r_k^+ = \{p\}$.

Proposition 5.2. *Suppose that Π is headcycle-free and M an answer set of Π. Then $\{r \in \mathrm{MR}(\Pi,M) \mid |r^+| = 1\} \models M$.*

That is, we can confine to the definite rules in the reduct $\mathrm{MR}(\Pi,M)$, and thus the entailment of B is decidable in polynomial time.

5.2 Computing witnesses

We now turn to computing witnesses, where we have differentiated picture. For comparing the relationship of problems, we use an adjusted notion of reduction. Namely, a problem A is reducible to problem B in polynomial time (resp., in log-space), if from every instance I_A of A, some instance I_B of B can be computed in polynomial time resp. in log-space, such that (1) I_B has some solution if I_A has some solution, and (2) given any solution S_B of I_B, we can compute some solution S_A of I_A from I_A and S_B in polynomial time respectively in log-space.

5.2.1 Minimal witnesses

As regards the computation of minimal witnesses, algorithm MinWitness implements a polynomial-time reduction of computing some minimal witness $\Pi' \in \mathrm{MW}(B,\Pi,S,M)$ of B where M is an answer set of Π with S asserted and $B \subseteq M \setminus S$, to computing some MUS of a clause theory; note that the steps in lines 2 and 4-8, respectively, amount to constructing from an instance I of minimal witness computation an instance I' of MUS computation, and from a solution S' of I' a solution S of I, respectively, in polynomial time. In fact, the computations can be done in logarithmic space.

Conversely, we can reduce MUS computation easily to minimal witness computation as follows. Given a clause theory Σ, let $\Pi = \{r^+ \cup \{p\} \leftarrow r^- \mid r \in \Sigma\}$, where p is a fresh atom, consist of all clauses in Σ

with a positive literal p added, written as rules. Then the minimal witnesses $\Pi' \subseteq \Pi$ of $B = \{p\}$ *w.r.t.* $M = \mathscr{A}(\Pi) \cup \{p\}$, where $\mathscr{A}(\Pi)$ is a set of the atoms occurring in Π, correspond to the MUS $\Sigma' \subseteq \Sigma$, such that $\Sigma' = \{r^+ \cup \neg r^- \mid r^+ \cup \{p\} \leftarrow r^- \in \Pi'\}$. Again, the reduction is computable in logarithmic space. We thus obtain:

Corollary 5.1. *Computing some minimal witness $\Pi' \in \mathrm{MW}(B,\Pi,M,S)$ as in the algorithm MinWitness and computing some MUS of a clause theory Σ are equivalent under logspace-reductions.*

Computing an MUS of a clause theory Σ is feasible in polynomial time with an NP oracle, and is thus in the class $\mathrm{FP}^{\mathrm{NP}}$. Furthermore, computing some MUS is hard for the class $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$ Chen and Toda [1995]; Janota and Marques-Silva [2016], which are the functions computable by a Turing machine in polynomial time with parallel NP oracle access. However, the precise complexity of MUS computation is not known Janota and Marques-Silva [2016]; in particular, it is also unknown whether computing some MUS is hard for or in the class $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$, which contains the multi-valued functions f for which some possible function value $y \in f(x)$ is computable in polynomial time with logarithmically many calls to a *witness oracle* for NP [Buss *et al.*, 1993; Janota and Marques-Silva, 2016], i.e., an oracle that returns some satisfying assignment for the oracle SAT instance if it is satisfiable. When only few (a constant number of) MUSs exist, then the problem is in $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ Janota and Marques-Silva [2016].

For normal logic programs, $\mathrm{MR}(\Pi,M) \cup S$ is Horn and thus algorithm MinWitness can be implemented to run in polynomial time, as the entailment test on line 12 can be done in polynomial time. This extends to headcycle-free programs, due to Proposition 5.2.

5.2.2 α-witnesses

Regarding α-witnesses, the main interest is in computing some compact $\alpha^\star$-witness, and in particular to compute some $\alpha^\star_{fs}$-witness of an answer set, since such a witness is in line with the decomposition of the logic program $\mathrm{MR}(\Pi,M)$ into its strongly connected components, which often form the basis for modular evaluation; furthermore, it provides greatest flexibility for serialization, as minimality and compactness are retained regardless of the concrete serialization, and even if SCCs are merged to larger units for evaluation (cf. Proposition 4.4). This problem reduces in view of Proposition 4.3 in log-space to solving independent instances $I_1,\dots,I_m$ of minimal witness computation (which is thus possible in parallel). On the other hand, solving such instances is reducible in log-space to a single minimal witness computation. Indeed, the instances I_j can be reduced to MUS instances Σ_j, $j = 1,\dots,m$ on disjoint sets of variables; the latter can be reduced to a single MUS computation for a clause theory $\Sigma = \{c \vee \neg a_i \mid 1 \le i \le m, c \in \Sigma_i\} \cup \{a_1 \vee \cdots \vee a_m\}$, where the a_is are fresh variables; indeed, the MUS of Σ is given by all unions $X_1 \cup \cdots \cup X_m \cup \{a_1 \vee \cdots \vee a_m\}$ where X_i is an MUS of Σ_i, $1 \le m$. In turn, computing some MUS of an unsatisfiable theory Σ is reducible to computing some $\alpha^\star_{fs}$-witness of an answer set of a logic program. This can be accomplished by extending the reduction from MUS computation to minimal witness computation in the previous section. To this end, we add the clause $p' \vee \neg p$ to Σ for each $p' \in \mathscr{A}$. Then $M = \mathscr{A} \cup \{p\}$ is an answer set of the resulting clause theory, and assuming without loss of generality that $r^- \neq \emptyset$ for some $r \in \Sigma$, the dependency graph G_Σ of Σ is strongly connected. Thus, every $\alpha^\star_{fs}$-witness of M is of the form (S_1,Π_1) where $S_1 = M$ and $\{r \in \Sigma \mid r^+ \cup \{p\} \leftarrow r^- \in \Pi_1\}$ is an MUS of Σ. We thus obtain:

Corollary 5.2. *Computing some $\alpha^\star_{fs}$-witness of an answer set $M \neq \emptyset$ of a logic program and computing some MUS of a clause theory are equivalent under logspace-reductions.*

5.2.3 β-witnesses

A minimal β-witness for an answer set M of a logic program Π can be computed using the algorithm MinBetaWitness modulo the test "$\Sigma_u \cup T \models v$ for some atom $v \notin \{u\} \cup T$" in line 10 and the calls to MinWitness in polynomial time. As both the test and each call can be reduced in polynomial time to MUS computation,[10] a minimal β-witness is computable in polynomial time with a procedure for MUS computation.

Since an MUS of a clause theory is computable in polynomial time with an NP oracle, or also directly from the algorithms MinBetaWitness and MinBetaBSWitness, we obtain:

Corollary 5.3. *Given an answer set M of a disjunctive logic program Π, some minimal β- resp. $\beta^\star$-witness of M w.r.t. Π is computable in polynomial time with an* NP *oracle.*

Conversely, we can reduce MUS computation of a given unsatisfiable clause theory Σ to computing a minimal β-witness for an answer set M of a positive logic program Π.

Let Σ be an unsatisfiable clause theory and Σ' be a copy of Σ on the alphabet $\mathscr{A}' = \{p' \mid p \in \mathscr{A}\}$, and define

$$\Sigma_1 = \{c \vee x \mid c \in \Sigma\},\ \ \Sigma_2 = \{c' \vee \neg x \vee x' \mid c' \in \Sigma'\},\ \ \Sigma_3 = \{\neg x \vee \neg x' \vee p \mid p \in \mathscr{A} \cup \mathscr{A}'\}, \tag{11}$$

where x, x' are fresh atoms, and let $\Pi = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$. Then $M = \mathscr{A} \cup \mathscr{A}' \cup \{x, x'\}$ is the single answer set of Π. Intuitively, in a minimal β-witness $[(p_1, \Pi_1), \ldots, (p_m, \Pi_m)]$ for M, either the rules in $\Pi_1 \cap \Sigma_1$ or those in $\Pi_1 \cap \Sigma_2$ must derive x resp. $\neg x \vee x'$; otherwise, it is not possible that Π_1 is a witness of p_1, and as necessary, does not derive any other atom. Formally,

Proposition 5.3. *Let Σ be an unsatisfiable clause theory, and let $\Pi = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ and $M = \mathscr{A} \cup \mathscr{A}' \cup \{x, x'\}$ be as (11). Then,*

(i) for every minimal β-witness $[(p_1, \Pi_1), \ldots, (p_m, \Pi_m)]$ of M w.r.t. Π, if $p_1 = x$ the set $\Gamma = \{r \in \Sigma \mid (r^+ \cup \{x\}) \leftarrow r^- \in \Pi_1\}$ and otherwise the set $\Gamma = \{r \in \Sigma \mid (r^+ \cup \{x'\}) \leftarrow r^- \cup \{x\} \in \Pi_1\}$ is an MUS of Σ;

(ii) from every MUS of Σ, some minimal β-witness of M w.r.t. Π can be constructed.

Since computing an MUS of a clause theory is known to be $\mathrm{FP}^{\mathrm{NP}}_{\|}$-hard Chen and Toda [1995]; Janota and Marques-Silva [2016], from Proposition 5.3 we thus obtain

Proposition 5.4. *Given an answer set M of a logic program Π, computing some minimal β-witness (resp. minimal $\beta^\star$-witness) for M w.r.t. Π is $\mathrm{FP}^{\mathrm{NP}}_{\|}$-hard.*

Notably, it is not clear whether the gap between $\mathrm{FP}^{\mathrm{NP}}_{\|}$-hardness and membership in $\mathrm{FP}^{\mathrm{NP}}$ can be improved, e.g., hardness for membership in the class $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$. Intuitively, a minimal β-witness seems to be more difficult to compute than an MUS, since we have a sequence of dependent MUS problems to solve, according to Definition 4.3 of minimal β-witness. However, while for each answer set $M = \{p_1, \ldots, p_n\}$ of a logic program Π always some minimal β-witness of $B = \{p_1\}$ under Π and $S = \emptyset$ *w.r.t.* M exists, no such witness $[(p_1, \Pi)]$ may exist that can be extended to some minimal β-witness $[(p_1, \Pi), \ldots, (p_n, \Pi_n)]$ of M under Π. As it turns out, deciding this problem is actually at the second level of the Polynomial Hierarchy.

[10]For the test in line 10, we can e.g., exploit a reduction from SAT to CRITICAL SAT in Papadimitriou and Wolfe [1988].

Theorem 6. *Let M be an answer set of a logic program Π, and let $p \in M$. Deciding whether there exists some minimal β-witness $W = [(p_1,\Pi_1),\dots,(p_n,\Pi_n)]$ for $M = \{p_1,\dots,p_n\}$ w.r.t. Π such that $p = p_1$ is Σ_2^p-complete.*

On the other hand, for normal and headcycle-free programs, computing some minimal β-witness is tractable; in fact, we can even compute some compact β-witnesses in polynomial time. The reason is that $\mathrm{MR}(\Pi,M)$ is a Horn theory. Thus, every atom in M has a witness with exact one rule. Thus, the inner **while** loop (lines 3-6) of MinBetaBSWitness is sufficient to find such a witness. For headcycle-free disjunctive logic programs, this is likewise, i.e., the algorithm MinBetaBSWitness will return the Π' constructed after this loop; this is due to the fact that headcycle-freeness ensures that minimal models can be incrementally be built up from facts by applying the rules.

Corollary 5.4. *For an answer set M of a normal or headcycle-free logic program Π, computing a compact β-witness for M w.r.t. Π is feasible in polynomial time.*

However, answer sets of general disjunctive logic programs may lack a compact β-witness and a compact $\beta^\star$-witness as shown by Example 4.9. The problem of deciding whether an answer set has a compact β- resp. $\beta^\star$-witness is intractable in general and in fact located at the second level of the Polynomial Hierarchy.

Theorem 7. *Deciding whether an answer set M of a disjunctive logic program Π has a compact β-witness (resp. compact $\beta^\star$-witness) is Σ_2^p-complete, and the Σ_2^p-hardness holds even if* $\mathrm{MR}(\Pi,M) = \Pi$.

Indeed, we can guess and check a compact β-witness in polynomial time with an NP oracle; the conditions in Definition 4.2 involve minimal witness tests which are feasible in polynomial time with an NP oracle. The Σ_2^p-hardness is by a $\exists\forall$-QBF encoding, where the rule selection for a compact β-witness mimicks the $\exists$-part of the formula.

In constructing a potential compact β-witness (if possible), it is usually preferable to construct the witness for an atom q before an atom p whenever any witness of p may be a witness of q. In this case the atom q is called *necessary* for p. Formally, let M be an answer set of a logic program Π and $S \subseteq M$. An atom q is *necessary* for an atom p *w.r.t.* M and S under Π, if $\mathrm{MR}(\Pi',M) \cup S \models q$ holds for every witness $\Pi' \subseteq \Pi$ of p under S *w.r.t.* M. Intuitively, it means that every derivation for p from $\mathrm{MR}(\Pi,M) \cup S$ makes use of q, which has been derived from $\mathrm{MR}(\Pi,M) \cup S$ before p. In this case p cannot be prior to q in a compact β-witness of M *w.r.t.*Π. This condition is hard to check in terms of the next theorem.

Theorem 8. *Let M be an answer set of a logic program Π, $S \subseteq M$ and $p,q \in \mathscr{A}$ s.t.* $\mathrm{MR}(\Pi,M) \cup S \models p \wedge q$. *The problem of deciding whether q is necessary for p w.r.t. M and S under Π is Π_2^p-complete, and the Π_2^p-hardness holds even if* $\mathrm{MR}(\Pi,M) = \Pi$ *and* $S = \emptyset$.

6 Experimental Evaluation

To experimentally evaluate the approach of computing minimal β-witnesses, we have implemented a prototype system in Python 3.7 and in addition also the revised minimal model checking algorithm. Since the above two algorithms involve logical reasoning, we make use of Glucose 4.0[11] in the python-sat package Ignatiev *et al.* [2018]. The package provides actually many other SAT solvers, including *CaDiCaL, Lingeling, Minisat* etc. According to Proposition 4.1, we can utilize MUS solvers for computing β-witnesses; we used the MUS solver *picomus* for this purpose.

[11] `https://www.labri.fr/perso/lsimon/glucose/`

Recall that the notion of witnesses developed in this paper serve the offline verification of given answer sets, produced by any answer set solver, and are not geared to online construction of answer sets by a particular solver, taking possible solver peculiarities into account.

The experiments were aimed to shed light on the following aspects:

A1. To see whether β-witnesses can be computed for logic programs in practice. As a side issue, this involved computing minimal models of positive programs respectively clause theories as well as minimality checking to have instances for the problem.

A2. How often do actually disjunctive rules occur in such witnesses, more precisely in the reduct *w.r.t.* to the given answer set; this links to the question how much disjunction with headcycles is needed to produce an answer set.

A3. The impact of using MUS solvers compared to using proprietary minimal β-witness computation.

We shall turn back to these aspects in Subsection 6.3 when experimental results are reported. Since the notions and constructions hinge much on positive programs and minimal models, positive programs are in the focus of the experiments.

The remainder of this section is organized as follows. We first give an overview of the benchmarks that we considered, which is followed by a description of the experimental platform and the results. We then conclude the section with a general discussion of the results. All experimental data (benchmarks and results) are online available.[12]

6.1 Benchmark problems

For our evaluation, we considered both synthetic and use case benchmark problem from ASP and SAT solving. The former include random k-CNF theories, random disjunctive logic programs and hand-coded CNF theories, while the latter include *minimal diagnosis* and *strategic companies* problems, which are two typical beyond-NP problems in ASP competitions, and some small size problems from SAT competitions (*Collatz, Johnson, Giraldez, crypto, grieu*) as benchmarks.

6.1.1 Benchmark generation

For the evaluation on CNF theories, one has to firstly compute a minimal model for each satisfiable CNF theory. For this purpose, one can translate a CNF theory into a logic program whose answer sets are exactly the minimal models of the CNF theory. In this case, efficient solvers for logic programs (under answer set semantics) can be employed, such as *DLV*[13], *clingo*[14], etc. Another possibility is to make use of SAT solvers in terms of Algorithm 3; note that line 4 in this algorithm can be safely removed for minimal model computation. We have implemented this algorithm in the prototype system *mr_minimal*[15] based on MiniSat source code. [16] Preliminary experimental results showed that this algorithm achieves better performance than the ASP solver *clingo-5.4* on both random clause theories with more than 200 variables and some industrial benchmarks from SAT competitions Li *et al.* [2021]. We also considered a trivial procedure Algorithm 8 using a SAT solver to compute a minimal model of a clause theory, in order to exploit the latest SAT solvers.

[12] `https://github.com/yswang168/witness`
[13] `http://www.dlvsystem.com/dlv/`
[14] `https://potassco.org/clingo/`
[15] `https://github.com/zhangli-hub123/minimal-model`
[16] `http://minisat.se/downloads/minisat-2.2.0.tar.gz`

Algorithm 8: MinModelSAT (Σ)

```
   Input: A clause theory $\Sigma$
   Parameter: SAT solver $sat$
   Output: A minimal model of $\Sigma$ or $UNSAT$ if no model of $\Sigma$ exists
1  $M \leftarrow sat(\Sigma)$ ;                // Compute a model $M$ of $\Sigma$ invoking $sat$
2  if $M$ is $UNSAT$ then return $UNSAT$ while $True$ do
3      $M' \leftarrow M$;
4      $\Sigma \leftarrow \Sigma \cup \{\neg q \mid q \in \text{var}(\Sigma) - M\} \cup \bigvee\{\neg p \mid p \in M\}$;
5      $M \leftarrow \alpha(\Sigma)$;
6      if $M$ is $UNSAT$ then return $M'$
7  end
```

6.1.2 Random k-CNFs

These satisfiable k-CNFs are randomly generated with the number n of variables ranging from 50 to 200 with interval 10, *i.e.*, $n \in \{50, 60, \ldots, 200\}$, while the number of clauses ranges from $3 \times n$ to $5 \times n$ with interval $0.1 \times n$. For each case we used 20 trials and report the average CPU time. The SAT solver *cadical*[17] is employed to test whether a k-CNFs is satisfiable and to compute a minimal model of a satisfiable k-CNFs instance according to Algorithm 3; notably, *cadical* won the first place in the SAT track of the SAT Race 2019 and second overall place.

6.1.3 Random disjunctive logic programs

Chen and Interian Chen and Interian [2005] proposed a model $t\text{-}Q(a, e; A, E; m)$ for generating random QBFs of the form $\forall X \exists Y F$, where F is a CNF formula, t is the number of components (QBFs), $|X| = A$, $|Y| = E$, m is the number of clauses in F, each clause contains a literals with variables in X and e literals with variables in Y. Combining the Chen-Interian model with the mapping between true QBFs of the form $\exists X \forall Y F$ and consistent disjunctive logic programs in Eiter and Gottlob [1995], Amendola *et al.* Amendola *et al.* [2018] have recently implemented a parametric generator for random disjunctive logic programs $t\text{-}D_{dlp}(e, a; E, A; m)$ that is the dual counterpart of the Chen-Interian model. We considered the parameter settings $t\text{-}D_{dlp}(1, 3; A, 60; m)$ with $t \in \{1, 2\}$, $A = 20, 30, \ldots, 140$ and $m = 50, 70, \ldots, 490$. In each case 20 randomly generated consistent disjunctive programs were considered. The answer set solver *clingo* was employed to compute an answer set of ech of these $2 \times 13 \times 23 \times 20 = 11960$ instances.

6.1.4 Handcrafted CNFs

To test for a harder case, we first considered the following clause theory Σ_n with n atoms $p_1, \ldots, p_n$, defined as

$$\Sigma_n = \{p_1 \vee \neg p_i, \neg p_1 \vee p_i \mid 2 \le i \le n\} \cup \{p_2 \vee \cdots \vee p_n\}. \tag{12}$$

The unique minimal model of Σ_n is $M_n = \{p_i \mid 1 \le i \le n\}$. It is evident that $\text{MR}(\Sigma, M) = \Sigma$ and the super-dependency graph $\mathbb{SG}_{\Sigma_n}$ has only one vertex. We tested the above theory with n ranging from 50 to 400 with interval 10 *i.e.*, $n \in \{50, 60, \ldots, 400\}$.

[17] http://fmv.jku.at/cadical/

We then considered also a cascading version of the above theory, where we have m "stages":

- $\Sigma_{n,1} = \{c^{(1)} \mid c \in \Sigma_n\}$,
- $\Sigma_{n,m} = \Sigma_{n,m-1} \cup \{p_1^{(m-1)} \wedge \cdots \wedge p_n^{(m-1)} \to c^{(m)} \mid c \in \Sigma_n\}, \quad$ for $m > 1$,

where $c^{(i)}$ means that each propositional symbol p in c is replaced by $p^{(i)}$. It is not difficult to see that $\Sigma_{n,m}$ has a unique minimal model $M_{n,m}$ containing every atom occurring in the theory $\Sigma_{n,m}$. We tested the cascading theories with $m = 5$ and n ranging from 50 to 400 with interval 10, with a CPU time limit of 30 minutes for each instance.

We also considered the following "cycle" version of the cascading case as follows: let x, y be two fresh atoms and

- $\Pi_{n,1} = \{\neg x \vee c^{(1)}, \neg y \vee c^{(1)} \mid c \in \Sigma_n\}$,
- $\Pi_{n,m} = \Sigma_{n,m-1} \cup \{p_1^{(m-1)} \wedge \cdots \wedge p_n^{(m-1)} \to c^{(m)} \mid c \in \Sigma_n\}$, for $m > 1$, and
- $\Sigma^c_{n,m} = \Pi_{n,m} \cup \{x \vee y, \quad p_1^{(m)} \wedge \cdots \wedge p_n^{(m)} \to x, \quad p_1^{(m)} \wedge \cdots \wedge p_n^{(m)} \to y\}$.

Intuitively, one of x and y is sufficient to derive all atoms $p_j^{(m)}$, which in turn derive both x and y. More formally, it is evident that $\Pi_{n,1} \cup \{x \vee y\}$ can derive $\{p_1^{(1)}, \ldots, p_n^{(1)}\}$ and more generally that $\Pi_{n,m} \cup \{x \vee y\}$ can derive $\{p_1^{(j)}, \ldots, p_n^{(j)}\}$, for all $j = 1, \ldots, m$. Hence $\Sigma^c_{n.m}$, which contains $x \vee y$, derives all $p_i^{(j)}$ and moreover x, y, which are derived using $p_1^{(m)} \wedge \cdots \wedge p_n^{(m)}$. Thus the unique minimal model of $\Sigma^c_{n,m}$ is $M^c_{n,m} = \{p_i^{(j)} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{x, y\}$. We tested the "cycle" theories $\Sigma^c_{n,m}$ with $m = 5$ and n ranging from 50 to 400 with interval 10, and again with a CPU time limit of 30 minutes for each instance.

6.1.5 ASP Competition benchmarks

We consider the beyond-NP search benchmarks *minimal diagnosis*, *strategic company* from the 3rd[18] and the 4th[19] ASP competitions. The ones from the 4th ASP competition were used in the 5th, 6th and 7th ASP competitions as well. The *strategic company* benchmarks from the 4th ASP competition are for answering a query, *e.g.*, "non_strategic_pair(c1,c2)?" is a ground query at the end of an instance file. Such ground queries are removed for our evaluation. One more beyond-NP search benchmark from the 4th ASP competition is *Complex Optimization of Answer Sets*. It is excluded since it involves non-disjunctive (choice) rules, for which the tool LP2NORMAL Bomanson [2017] still cannot translate such programs possibly due to disjunction. The other beyond-NP benchmarks in the two ASP competitions are for optimization and involving weak constraints. For each instance, its grounding was obtained by *gringo*-3.0.5 with the option "-t", while an answer set was obtained by *clingo*-4.4.0 within 2 hours time limitation. Though there are many other benchmark classes for disjunctive logic programs Gebser *et al.* [2013], including *ConformantPlanning*, *MaximalSatisfiableSet*, *2QBF* and *Repair*, these benchmarks do not fit for evaluation since they are in *lparse*[20] internal format and involve *choice, maximize, aggregate* or other non-disjunctive rule types or constructors. The current implementation cannot deal with these extensions.

[18] `https://www.mat.unical.it/aspcomp2011/FrontPage`
[19] `https://www.mat.unical.it/aspcomp2013/FrontPage`
[20] `http://www.tcs.hut.fi/Software/smodels/lparse.ps`

Table 2: The numbers of atoms(n) and clauses(nc) of random benchmarks for SAT 2007 competition

class \ k		3	5	7
LargeSize	n	$m \times 1000 \mid m = 4, 7, 10, 13, 16, 19$	$m \times 100 \mid m = 6, 7, 8, 9, 10, 11$	140,160,180,200,220,240
	nc	$n \times 4.2$	$n \times 20$	$n \times 85$
OnThreshold	n	360,400,450,500,550,600,650	70,80,90,100,110,120	45,50,55,60,65,70,75
	nc	$n \times 4.26$	$n \times 21.3$	$n \times 89$

6.1.6 SAT Competition benchmarks

A large number of benchmarks for SAT competitions and races is available.[21] As we need to compute a minimal model first, which is ususally time consuming, relatively small size CNF instances were chosen: *Collatz* and *Johnson* from the SAT-2019 competition, the application benchmark *Giraldez* from the SAT-2016 competition, and the industrial benchmarks *crypto* and *grieu* from the SAT-2007 competition.

As the above random k-CNFs are relatively small in size, we choose the random k-CNF benchmarks from the SAT 2007 competition as well. It includes two classes 'LargeSize' and 'OnTreshold' for $k \in \{3, 5, 7\}$ with different numbers of atoms. The number n of atoms and the number nc of clauses for different class and different k are shown in Table 2; for each value pair of k and n, 10 instances were considered.

For each of these instances, we employed the SAT solver *cadical* to compute a minimal model M within 2 hours, and then we computed a minimal β-witness for each M obtained *w.r.t.* the corresponding clause theory (timeout 2 hours).

6.2 Experimental results

All experiments were conducted on a server running Linux 3.10.0 with 48 cores Intel(R) Xeon(R) Silver 4214 CPU at 2.20GHz and 132G memory, without using multi-processes or multi-threads.

6.2.1 Random k-CNFs

We first report in Table 3 the average CPU time for computing minimal models of randomly generated 3-CNFs by four solvers: *DLV, clingo*, Algorithm 8 with *cadical*, and *mr_minimal*. Each cell of the table includes 21×20 satisfiable instances with $n \in \{50, 100, 150, 200\}$ variables and $n \times \gamma$ clauses, where γ ranges over 3.0, 3.1, 3.2, ..., 5.0, and all clauses were randomly generated in a uniform manner; the number of randomly generated unsatisfiable instances is also reported. As can be seen from the table, using the latest SAT solver *cadical* yields a slightly better performance in terms of CPU time for the largest number of variables. Thus, we considered Algorithm 8 as a minimal model solver in the subsequent experiments. A notable observation is that it was more difficult to randomly generate satisfiable 3-CNF theories when the number of atoms was increasing. In particular, the minimal model solver has for each random k-CNF instance a limit of 30 minutes to compute a minimal model. It took for $n = 150$ and $n = 200$ almost 0.5 and 10 hours, respectively, to randomly generate $16 \times 21 \times 20 = 6720$ satisfiable 3-CNF theories and to compute some minimal models of them; for $n = 300$, this task could not be completed within 5 days.

The overall CPU time (in seconds) for computing some minimal β-witnesses of given random k-CNFs is shown in Table 4, where $k \in \{3, 4, 5, 6, 10, 20\}$. Each cell of the table includes the data for $16 \times 21 \times 20 = 6720$ satisfiable k-CNF instances with $n \in \{50, 60, \ldots, 200\}$ atoms and $n \times \gamma$ clauses where $\gamma \in \{3.0, 3.1, \ldots, 5.0\}$ for each n. The total number of atoms p_i whose witness Π_i contains more than one clause, denoted by $\#|\Pi_i|\geq 2$, is also reported in the table. In case $|\Pi_i| \geq 2$, Π_i must contain at least one clause whose reduct

[21] `http://satcompetition.org/`

Table 3: The average CPU time (in seconds) for computing a minimal model of 3-CNFs

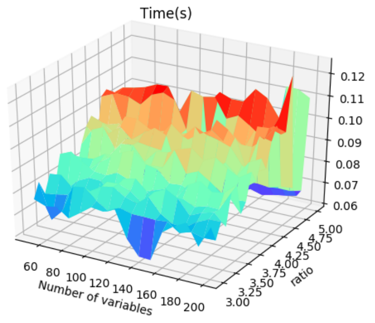
solver \ n	50	100	150	200
dlv	0.0269	0.0338	0.0638	0.3201
clingo	0.0686	0.1691	0.3916	0.9696
mr_minimal	0.0254	0.0424	0.1071	0.5826
Algorithm 8 (cadical)	0.0883	0.1204	0.1570	0.2753
#UNSAT	668	4252	65669	528767

Table 4: The result of computing minimal β-witness of random k-CNFs

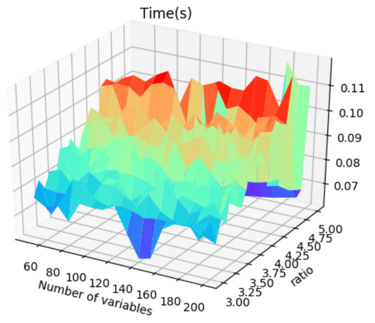
k	3	4	5	6	10	20
without MUS solver						
Time (secs)	674.73	633.93	638.68	637.80	620.01	671.05
$\#\|\Pi_i\|\geq 2$	13	9	0	0	0	1
$\#\|\Pi_i\|^c \geq 2$	13	10	0	0	0	1
$\#\|\Pi_i\|^w \geq 2$	28	21	0	0	0	2
non-compactness	2	1	0	0	0	0
with the MUS solver picomus						
Time (secs)	676.77	634.25	639.67	637.27	623.09	672.82
$\#\|\Pi_i\|\geq 2$	13	9	0	0	0	1
$\#\|\Pi_i\|^c \geq 2$	13	10	0	0	0	1
$\#\|\Pi_i\|^w \geq 2$	30	21	0	0	0	2
non-compactness	3	1	0	0	0	0

has more than one positive literal. The overall number of such clauses in all respective Π_i is denoted by $\#|\Pi_i|^c \geq 2$, and the overall number of clauses in all respective Π_i is denoted by $\#|\Pi_i|^w \geq 2$. As can be seen from Table 4, computing some minimal β-witness can in general be accomplished quite efficiently, and the computation time is not much dependent on the clause size. There are very few atoms (in a minimal model) whose minimal β-witness involves multiple clauses. It means that almost every atom has just one clause as its witness, from which an explanation (resolution proof) for the atom is trivial. In other words, minimal β-witnesses for minimal models of random k-CNFs are almost definite when applying the reduct. In this case, the unit propagation in Algorithm 5, *i.e.* the **while-loop** (lines 3-6), is sufficient to compute such a minimal β-witness. Furthermore, even if atoms p_i with $\#|\Pi_i| \geq 2$ were in minimal β-witnesses obtained, only 3 (resp. 4) of these minimal β-witnesses were not compact β-witnesses when not MUS solver was used (resp. the MUS solver *picomus* was used).

We further illustrate in Fig. 4 the average CPU time for computing minimal β-witnesses of 20 3-CNF instances with the same n and ratio γ. In this figure, (a) uses no MUS solver, while (b) uses the MUS solver *picomus*. An expected behaviour is that an increasing number of clauses requires more CPU time in both cases. According to both Table 4 and Fig. 4, for the tested random CNFs there is not much difference between using the MUS solver or not. The other cases for $k \in \{4, 5, 6, 10, 20\}$ are all similar to that of $k = 3$.



(a) without MUS solver



(b) with MUS solver picomus

Figure 4: The average CPU time for computing minimal β-witnesses of random 3-CNFs

Table 5: The result of computing minimal β-witness of random disjunctive logic program

t	# instances	With MUS solver *picomus*?	$\#\lvert\Pi_i\rvert \geq 2$	$\#\lvert\Pi_i\rvert^c \geq 2$	$\#\lvert\Pi_i\rvert^w \geq 2$	compact	CPU(s)
1	5980	No	5980	5980	18410	5510	0.29
1	5980	Yes	5980	5980	18729	5191	0.16
2	5980	No	12044	12044	36712	5214	0.51
2	5980	Yes	12314	12314	38455	4595	0.18

6.2.2 Random disjunctive logic programs

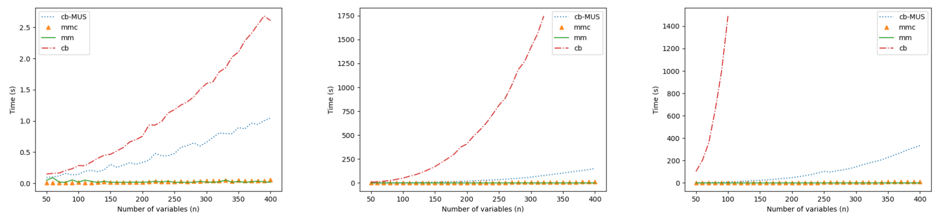
Firstly, we report the overall metric values of $\#|\Pi_i| \geq 2$,$\#|\Pi_i|^c \geq 2$, $\#|\Pi_i|^w \geq 2$, number of compact witnesses and the average CPU time in seconds per instance in Table 5.

As can be seen, the use of the MUS solver *picomus* reduces the overall CPU time of computing some minimal β-witness up to $0.51/0.18 \approx 2.83$ times; on the other hand, the number of compact β-witnesses computed also decreases for $t = 1$ from 5510 to 5191 and for $t = 2$ from 5214 to 4595. In addition, when $t = 1$ each computed β-witness has exactly one atom whose local witness involves more than one rule, among them exactly one is non-definite. When $t = 2$, the metric value of $\#|\Pi_i| \geq 2$ (resp. $\#|\Pi_i|^c \geq 2$, $\#|\Pi_i|^w \geq 2$) is about 2 times larger than that of $t = 1$. Notably, computing a minimal β-witness was for each of these random disjunctive logic programs feasible within one second and for each atom the local witness involved quite few rules.

6.2.3 Handcrafted CNFs

For the handcrafted clause theories $\Sigma_{n,m}$, the CPU time of computing a minimal β-witness without an MUS solver (cb) and with the MUS solver *picomus* (cb-MUS) is shown in Fig. 5. For comparing the efficiency between minimal model checking and computing minimal β-witnesses, the algorithm CheckMinMR for minimal model checking was implemented and run on the clause theories; the CPU time is reported in Fig. 5 as 'mmc', in addition to the CPU time of computing the minimal model by Algorithm 3, reported as 'mm'.

It can be seen that our algorithm MinBetaWitness (Algorithm 6) is still efficient for these theories. As expected, the time increases with the number n of atoms in the basic clause theory and also with the number m of stages, but for the latter in a non-proportional manner (for $n = 100, 200, 300$ it increases from roughly 0.25,0.75,1.5 secs for $m = 1$ to 60,400,1500 secs for $m = 5$, respectively); the cyclic version further increases



(a) $m = 1$ (b) $m = 5$ (c) $m = 5$ with cycle

Figure 5: CPU time for computing minimal β-witness and minimal model checking of $\Sigma_{n,m}$

the computation time. Notably, the MUS solver *picomus* helps a lot in computing some β-witness in terms of the computing cost. According to Fig. 5.b, MinBetaWitness times out under a limit of 30 mins for $n \geq 320$ when no MUS solver was used. With the help of *picomus*, the computation time of the cycle version is slightly higher than that of the cascading version ("cb-MUS" in (b) vs. "cb-MUS" in (c)). However, the cycle version is much harder to be solved without the help of the MUS solver ("cb" in (b) vs. "cb" in (c)). A notable comparison shows that the minimal model computation and minimal model checking are far more efficient than computing some minimal β-witness, and have nearly the same (small) cost for these instances.

6.2.4 ASP and SAT competition benchmarks

The CPU time and other metrics of computing a minimal β-witness for the ASP and SAT benchmarks are shown in Tables 6 and 7, respectively. Specifically, the following values are shown:

- total, select: the number of overall (resp. the selected in lexicographic order) instances;
- sat, unsat, solved: the number of satisfiable (resp. unsatisfiable, solved) instances within a time limit of 2h;
- compact: the number of solved instances with a compact β-witness as result;
- $|\Sigma|$(MB): the average solved instances size (in Megabytes);
- $|M|$(KB): the average size of answer sets (minimal model) (in Kilobytes) for solved instances;
- CPU: the average CPU time (secs) to compute a minimal β-witness for solved instances within a time limit of 2h;
- Mem: the average memory usage in Megabytes for solved instances;
- $\#|\Pi_i| \geq 2$: the overall number of atoms with more than one clause in their local witness over the solved instances;
- $\#|\Pi_i|^c \geq 2$: the overall number of non-definite clauses (i.e., with more than one positive literal) in witnesses;
- $\#|\Pi_i|^w \geq 2$: the overall size of local witnesses with more than one clause;
- +: maximum recursion depth 1000 exceeded;

Table 6: Computing minimal β-witnesses for ASP benchmarks – … timeout; further legend see text

	minimal diagnosis		strategic companies	
metrics	3rd	4th	3rd	4th
total / select	551/100	250/20	51/20	37/5
sat	68	13	15	3
unsat	32	7	0	0
without / with MUS-solver *picomus*				
solved	68	5	15	0
$\lvert\Sigma\rvert$ (MB)	3.72	25.80	1.23	–
$\lvert M \rvert$ (KB)	237.30	1547.62	410.22	–
CPU	161.30 / 157.32	3140.85 / 3199.37	65.17 / 65.12	–
Mem	411.18 / 406.62	2500.33 / 2499.52	179.00 / 178.99	–
compact	54 / 55	4 / 4	15 / 15	–
$\#\lvert\Pi_i\rvert \geq 2$	15 / 14	2 / 2	0 / 0	–
$\#\lvert\Pi_i\rvert^c \geq 2$	21 / 20	9 / 9	0 / 0	–
$\#\lvert\Pi_i\rvert^w \geq 2$	116 / 109	47 / 47	0 / 0	–

Table 7: Computing minimal β-witnesses for SAT benchmarks – … timeout; further legenda see text

metrics	Collatz	Johnson	Giraldez	crypto	grieu
total / select	19/19	19/19	29/29	10/10	10/10
sat	16	4	18	10	6
unsat	3	0	9	0	0
without / with MUS-solver *picomus*					
solved	9 / 10	4	18	0	6
$\lvert\Sigma\rvert$ (MB)	6.61 / 8.59	2.66	0.15	+	1.25
$\lvert M \rvert$ (KB)	8.40 / 8.02	11.32	5.44	+	0.13
CPU	29.50 / 13.61	14.91 / 14.43	2.19 / 1.27	+	1.82 / 1.83
mem	491.82 / 563.97	192.64 / 192.67	36.09 / 36.01	+	119.87 / 119.87
compact	6 / 6	4 / 4	18 / 18	+	6 / 6
$\#\lvert\Pi_i\rvert \geq 2$	9 / 31	0 / 0	3 / 3	+	0 / 0
$\#\lvert\Pi_i\rvert^c \geq 2$	21 / 53	0 / 0	3 / 3	+	0 / 0
$\#\lvert\Pi_i\rvert^w \geq 2$	415 / 502	0 / 0	6 / 6	+	0 / 0

- – : timeout after 2 hours.

As can be seen from Tables 6 and 7, MinBetaWitness(Π, M) computed 68+5+15+0=88 minimal β-witnesses for the 68+13+15+3=99 of the consistent ASP instances regardless of whether no MUS solver or *picomus* was used, and 37 resp. 38 minimal β-witnesses for 54 consistent SAT instances without resp. with the MUS solver *picomus*.

However, no *strategic companies* instance from the 4th ASP competition could be solved due to their overlarge size of around 140 MB, and similarly no *crypto* instance whose model size is more than 100 KB was solved due to exceeding the maximum recursion depth 1000.

In few cases, an atom p_i in an answer set resp. minimal model had a local witness Π_i that contained more than one rule resp. clause; furthermore, few rules (resp., clauses) with at least 2 head atoms (resp., positive literals) occurred in the minimal β-witnesses computed, and for non-singleton local witnesses Π_i the number of rules resp. clauses in it was small, with the exception of *minimal diagnosis (3rd edition)* and *Collatz*. In

Table 8: Computing minimal β-witnesses for 'LargeSize' class benchmarks

k-n / metrics	3-4000	5-600	5-700	5-800	7-180
sat	2	9	8	6	1
unsat	0	0	0	0	0
without / with MUS-solver *picomus*					
CPU	4.93 / 4.78	0.68 / 0.68	0.87 / 0.91	1.09 / 1.14	11.21 / 0.60
compact	2 / 2	9 / 9	8 / 8	6 / 6	0 / 0
$\#\lvert\Pi_i\rvert \geq 2$	0 / 0	0 / 0	0 / 0	0 / 0	2 / 2
$\#\lvert\Pi_i\rvert^c \geq 2$	0 / 0	0 / 0	0 / 0	0 / 0	66 / 12
$\#\lvert\Pi_i\rvert^w \geq 2$	0 / 0	0 / 0	0 / 0	0 / 0	134 / 32

particular, when using the MUS solver *picomus*, one more instance, C3-2-31.cnf from *Collatz* was solved in about 30 seconds and in the returned β-witness there are 23 atoms involving overall 75 clauses in their witnesses, among them 36 clauses have at least two positive literals. Overall, the MUS solver was helpful for computing some β-witness in terms of computing cost; in particular, the computation was about two times faster than without the MUS solver for the benchmark *Collatz*, at the expense of more memory and some more atoms that have multiple-clause witnesses, while the other benchmarks showed no such trade-off. In the other experiments there was a small but not significant run time gain or loss. Furthermore, many of the minimal β-witnesses computed, viz. 73 (resp. 74) out of 88 for the ASP benchmarks and 34 out of 37 (resp. 38) for the SAT benchmarks with no MUS solver (resp. with the MUS solver *picomus*), were in fact compact witnesses.

For the random benchmarks of the SAT 2007 competition, we report the cases that a minimal model is computed in the 'LargeSize' class in Table 8. The satisfiability of the other instances is unknown due to timeout. It can be seen that our algorithm for computing some minimal β-witness is efficient for all of these minimal models. The worst situation (11.21 seconds) happens for the random 7-CNF with 180 atoms and $180 \times 85 = 15300$ clauses when it uses no MUS solver. For this instance, using the MUS solver *picomus* yielded a speed up of more than 18 times. The minimal β-witness computed in this case was the only non-compact one among all the minimal β-witnesses that were computed.

For the 'OnThreshold' class, we computed a minimal model for five of the random instances with $k = 3$ and each $n \in \{360, 400, \ldots, 650\}$ in 2 hours for each instance. The satisfiability of the others are unknown due to timeout, with the exception of $n = 360$ where the other instances are unsatisfiable. For each of the 35 minimal models obtained for these 35 random 3-CNF instances, our algorithm computed a minimal β-witnesses in less than 0.3 seconds. These witnesses are all compact and no β-witness of an atom needs more than 2 clauses. For the random 5-CNF benchmarks, a minimal model is returned for 5 instances for each $n \in \{90, 100, 110, 130\}$. Our implementation computes a minimal β-witness for each of these 20 minimal models in 0.4 seconds on average. All the minimal β-witnesses are compact and no β-witness of an atom needs more than 2 clauses. For the other random 5-CNF instances with $n \in \{70, 80, 120\}$ and all the random 7-CNF instances, the experimental results of computing minimal β-witnesses are reported in Table 9. They further confirm that the MUS solver does improve the computation for minimal β-witnesses, *e.g.*, it yielded a speed up of more than $121.83/1.57 \approx 77$ times for the instances of 7-CNF and n=75. For the 32 minimal models of the random 7-CNF instances, only $10/27 \approx 37\%$ of the minimal β-witnesses computed with the help of the MUS solver were compact. Furthermore, many atoms in answer sets need more than 2 clauses in their β-witnesses and many clauses in these witnesses are non-definite. With the help of the MUS solver, our implementation produced less than one-third of non-definite clauses. This is very different from all the other

Table 9: Computing minimal β-witnesses for 'OnThreshold' class benchmarks (without/with MUS solver)

k-n	sat	unsat	compact	cpu(s)	$\#\lvert\Pi_i\rvert \geq 2$	$\#\lvert\Pi_i\rvert^c \geq 2$	$\#\lvert\Pi_i\rvert^w \geq 2$
5-70	5	5	4 / 4	0.40 / 0.16	4 / 1	57 / 6	124 / 14
5-80	5	4	4 / 3	0.23 / 0.14	2 / 3	3 / 4	11 / 17
5-120	5	0	4 / 4	0.39 / 0.15	2 / 2	9 / 5	24 / 18
7-45	5	5	1 / 1	3.79 / 0.29	24 / 25	610 / 201	959 / 376
7-50	5	4	0 / 1	5.22 / 0.30	23 / 23	524 / 133	871 / 306
7-55	5	0	2 / 2	4.86 / 0.30	19 / 22	596 / 172	951 / 348
7-60	5	0	1 / 1	5.82 / 0.37	22 / 23	914 / 176	1429 / 375
7-65	4	0	2 / 2	1.39 / 0.21	6 / 5	175 / 39	264 / 69
7-70	5	0	3 / 3	3.11 / 0.29	8 / 10	367 / 50	634 / 136
7-75	3	0	0 / 0	121.83 / 1.57	24 / 25	3061 / 837	4195 / 1363

tested random $3,5$-CNF, industrial SAT and ASP competition use cases.

6.3 Summary and Discussion

The above experimental results lead us to the following insights for the aspects (A1)-(A3) from above.

First, regarding (A1), they show that computing some minimal β-witness is usually feasible for both randomly generated clause theories and for the handcrafted clause theories. For the ASP and SAT encodings from the considered use cases, the picture is different, as with the time limit of 1h for ASP and 45 secs for SAT, for about (68+4+15)/(68+13+15+3)=87/99≈87% resp. (8+4+18+6)/(16+4+18+10+6)=36/54≈64% of the instances some minimal β-witness was computed[22]. However, the percentiles for the individual benchmarks vary a lot and suggest that there is a range of "easy" to "hard" problems, depending on the problem structure, for both ASP and SAT. This is further nurtured by that observation that in case a minimal β-witness was found, the computation terminated rather quickly compared to the time limit, with the exception of *minimal diagnosis (4th edition)*.

Regarding (A2), except for the 7-CNFs in the 'OnThreshold' class, we found that very few atoms p_i needed more than one rule resp. clause in their local witness Π_i in a minimal β-witness as shown by $\#|\Pi_i|^c \geq 2$, for all the tested cases above, and that for five out of benchmarks such a local witness Π_i had few clauses on average (less than two). Thus, the justification of an atom (why this atom belongs to a minimal model) in terms of a resolution proof from the reduct $\mathrm{MR}(\Pi, M)$ of the program Π *w.r.t.* the answer set M at hand (which incorporates the closed world assumption of M into the program Π), is quite obvious in these cases. For the other two benchmarks, constructing a resolution proof was narrowed down to local sets of manageable size of about 10 rules resp. 50 clauses. Furthermore, as definite rules r are not repeated in minimal β-witnesses (they serve to derive the atom in r^+), the computed minimal β-witness were also "nearly" compact, as only for few distinct atoms p_i and p_j the associated witnesses Π_i and Π_j, respectively, may overlap. In fact, about 84% resp. 91% of them were for the ASP resp. SAT benchmarks indeed compact; since algorithm MinBetaWitness is not geared towards compact witnesses, this rate may possibly be even increased by tuning it.

As for (A3), the use of an MUS solver is beneficial for instances that bear structure, random disjunctive logic programs and random SAT benchmarks, in particular for the handcrafted instances this led to a significant speedup and a more graceful increase of computation time, while it had almost no effect for the

[22]Only one solved ASP instance of minimal_diagnosis from the 4th ASP competition took more than 1 hour, and one solved SAT instance from Collatz took more than 45 seconds in the case of without MUS solver.

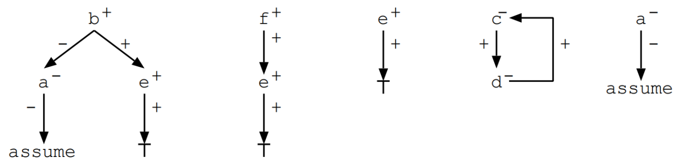


Figure 6: The off-line justifications for atoms in M_1 *w.r.t.* M_1 and $\{a\}$ (left to right)

randomly generated k-CNF instances (resulting in marginally higher runtime). For the ASP and non-random SAT benchmarks, the runtime did in most cases not significantly improve, and the quality of the minimal β-witnesses produced was very similar to the setting without MUS solver, with one exception. For obtaining a clear picture on the benefits/drawbacks of using an MUS solver in practice, further research will be necessary.

7 Related Work

Justifications for logic programs have been extensively investigated from the perspective of off-line justifications Pontelli *et al.* [2009], why-not provenance Damásio *et al.* [2015] and argumentative explanations Schulz and Toni [2016] for non-disjunctive logic programs, semantical error finding and debugging Gebser *et al.* [2008]; Oetsch *et al.* [2018]; Dodaro *et al.* [2019], causal graphs Cabalar and Fandinno [2016] and inconsistency proofs Alviano *et al.* [2019] for logic programs; we refer to Fandinno and Schulz [2019] for a comprehensive survey. In the section, we discuss off-line justifications, causal graphs, inconsistency proofs and answer-set program debugging more in detail as they are closely related to the notion of β-witness in the paper, while the purpose of semantical error finding is to answer why a set of atoms is not an answer set. We briefly address also the relationship of our notions to inconsistency proofs, which aim to answer why a logic program has no answer set.

7.1 Off-line justification

Pontelli *et al.* Pontelli *et al.* [2009] proposed off-line justification for normal logic programs, which has some similarity to the concept of β-witness from above; the formal definition takes positive and negative dependencies among atoms into account and is rather involved; for this reason, we refrain from introducing it and illustrate here the difference by the following example.

Example 7.1 (Example 10 in Pontelli *et al.* [2009])**.** *Let* Π *be the logic program consisting of*

$$\delta_1 : a \leftarrow f, not\ b, \quad \delta_2 : c \leftarrow d, f, \quad \delta_3 : e, \quad \delta_4 : b \leftarrow e, not\ a, \quad \delta_5 : d \leftarrow c, e, \quad \delta_6 : f \leftarrow e.$$

This program has two answer sets, viz. $M_1 = \{f, e, b\}$ *and* $M_2 = \{f, e, a\}$*. The off-line justifications for atoms in* M_1 *w.r.t.* M_1 *and* $\{a\}$ *are shown in Fig. 6. Intuitively, the reason that b belongs to the answer set* M_1 *is:*

(a) a is assumed (to be **false***), and*

(b) e belongs to M_1 *which can be justified by* δ_3 *and the rule* δ_4*.*

The reason why a is not in the answer set M_1 *is due to assumption.*

In terms of the reduct $\mathrm{MR}(\Pi, M)$ that we introduced in the paper, we have

$$\mathrm{MR}(\Pi, M_1) = \{\delta_3 : e,\ \ \delta_4' : b,\ \ \delta_6 : f \leftarrow e\} \ \text{ and } \ \mathrm{MR}(\Pi, M_2) = \{\delta_1' : a \leftarrow f,\ \ \delta_3 : e,\ \ \delta_6 : f \leftarrow e\}.$$

One can easily see that M_1 and M_2 have *w.r.t.* Π the compact β-witnesses $W_1 = [(e,\delta_3),(f,\delta_6),(b,\delta_4)]$ and $W_2 = [(e,\delta_3),(f,\delta_6),(a,\delta_1)]$, respectively. As for M_1, W_1 answers why $\{b,e,f\}$ is a minimal model of $\mathrm{MR}(\Pi, M_1)$ and why every atom in M_1 must occur in M_1. Comparing with the justification in Fig. 6, the minimal β-witness W_1 does not answer why the atom a is not in M_1, while this is answered according to the off-line justification, namely by assumption. An important difference between off-line justifications and our minimal β-witnesses is that for the latter we are asserted that a given interpretation M is an answer set of the program Π at hand. The reduct $\mathrm{MR}(\Pi, M)$ incorporates the CWA assumption of M, such that a justification of atoms that are not in M, i.e., are false in M, is not needed; in particular, why a,c,d are false in M_1 needs no further justification. In other words, the knowledge about M being an answer set allows us to give more lightweight justifications.

7.2 Causal stable models

Cabalar and Fandinno Cabalar and Fandinno [2016] investigated justifications for (labelled) disjunctive logic programs,[23] where each atom that is true in a model is associated with an algebraic expression (in terms of rule labels) that represents its justification. For instance, let us consider the following logic program Π consisting of

$$r_1 : \mathit{dead} \leftarrow \mathit{shoot}, \quad r_2 : \mathit{shoot} \leftarrow \mathit{tails}, \quad r_3 : \mathit{head} \vee \mathit{tails} \leftarrow \mathit{harvey}, \quad r_4 : \mathit{harvey},$$

which is an adaption of the logic program P_2 in Cabalar and Fandinno [2016], where r_1, r_2, r_3, r_4 are called *labels*. The rule r_3 means that Harvey throws a coin and only shoots when it ends up tails. Note that $M = \{\mathit{dead}, \mathit{shoot}, \mathit{tails}, \mathit{harvey}\}$ is an answer set of Π. Accordingly, the explanation for the atom $\mathit{dead} \in M$ can be

$$\mathit{harvey} \cdot r_3^{\mathit{tails}} \cdot r_2 \cdot r_1. \tag{13}$$

It means that the cause of *dead* in the stable model M of Π is the sequential application of rules: *harvey* first, r_3 second, and then r_2, finally r_1. In particular, when applying r_3, *tails* is chosen from its rule head, written r_3^{tails}. This explanation actually is associated with a causal stable model of Π, which exactly corresponds to the stable model M of Π.

Note that the “causal stable model” semantics of (labelled) logic programs is declarative, where an interpretation I is a mapping from the set of atoms to a set of terms of the form (13). Cabalar and Fandinno showed that, for each stable model of a logic program, there is a corresponding causal stable model of its corresponding labelled logic program, and vice versa, cf. [Cabalar and Fandinno, 2016, Theorem 4]. Actually, the following interpretation I is a causal stable model of Π with:

$$\begin{aligned}&I(\mathit{harvey}) = \mathit{harvey}, \quad I(\mathit{head}) = 0, \quad I(\mathit{tails}) = \mathit{harvey} \cdot r_3 \cdot \mathit{tails},\\ &I(\mathit{shoot}) = \mathit{harvey} \cdot r_1 \cdot \mathit{tails} \cdot r_2 \cdot \mathit{shoot}, \quad I(\mathit{dead}) = \mathit{harvey} \cdot r_1 \cdot \mathit{tails} \cdot r_2 \cdot \mathit{shoot} \cdot r_3 \cdot \mathit{dead}.\end{aligned}$$

The causal stable model I exactly captures the above stable model M of the logic program Π. Though one can read out the justification for each element of M from the corresponding causal stable model I, the new

[23] Each rule is associated with a label.

issue "why I is a causal stable model of Π" arises, viz. what are the justification for causal stable models. In addition, it is not clear how to efficiently compute a causal stable model and how difficult this task is, though a trivial guess and check algorithm exists.

Note that $\mathrm{MR}(\Pi,M)=\{r_1,r_2\}\cup\{harvey,\quad tails\leftarrow harvey\}$. It can be easily verified that a compact β-witness of M *w.r.t.* $\mathrm{MR}(\Pi,M)$ is $W=[(harvey,\{r_4\}),\ \underline{(tails,\{tails\leftarrow harvey\})},\ (shoot,\{r_2\}),\ (dead,\{r_1\})]$. It actually explains why M is a minimal model of $\mathrm{MR}(\Pi,M)$. To answer the question how M can be a minimal model of Π, one may replace the above underlined part by $(tails,\{r_3\})$, which intuitively means that *tails* is chosen at that step, while *head* is not. In fact, we get the following result when running our prototype system for this situation:

```
harvey:
  harvey.
tails:
  tails | head :- harvey.
shoot:
  shoot :- tails.
dead:
  dead :- shoot.
```

It properly explains that why M is an answer set of the logic program Π in terms of resolution refutation and reduct in the following manner:

(1) The element *harvey* of M is due to the fact rule r_4;

(2) One can derive *tails* from the reduct of the rule of r_3 *w.r.t.* M and the derived *harvey* in step (1);

(3) One can further derive *shoot* from r_2 and the derived *tails* in step (2);

(4) Finally, we have *dead* from r_1 and the derived *shoot* in step (3).

In this way, once a proposition has been properly justified, it can play the ground truth role in the explanation of other propositions. Thus, the notion of witness in this paper provides a structure explanation. This is not so for causal stable models, *e.g.*, the above causal stable model I yields no relationship among $I(harvey), I(tails), I(shoot)$ and $I(dead)$.

A limitation of the causal stable models approach has already been addressed in the Introduction, where we discussed the logic program $\Pi=\{r_1: a\vee b,\quad r_2: a\leftarrow b,\quad r_3: b\leftarrow a\}$, which has the unique answer set $M=\{a,b\}$. The causal stable model I where $I(a)=r_1^a, I(b)=r_1^a\cdot r_3$ is a declarative construction of M, but the choice of a from the rule r_1 expressed by r_1^a is in a sense not founded.

The minimal β-witness $W=[(a,\{r_1,r_2\}),(b,\{r_3\})]$ of M *w.r.t.* Π intuitively corresponds to I, but gives a more founded justification for M being an answer set, as a is a logical consequence of r_1,r_2 (and thus of Π) by resolution. Furthermore, W is compact and thus the disjunctive rule r_1 is used only once for derivation. The second causal stable model I' of Π, where $I'(a)=r_1^b\cdot r_2$ and $I'(b)=r_1^b$, corresponds to $W'=[(b,\{r_1,r_3\}),(a,\{r_2\})]$ in a symmetric fashion.

7.3 Inconsistency proofs

Besides answering why a given set of atoms is an answer set of a logic program, it is of equal interest to certify that a logic program has no answer set (referred to as being *inconsistent*). The latter is in particular useful for expressing properties like validity of a Boolean formula, that a given graph is not 3-colorable, that a plan for solving a problem is conformant, etc.

That an ASP solver merely reports that no answer set was found for a given program Π is not compelling. A proof of inconsistency may be desired, and finding an informative such proof can be far from trivial. Furthermore, since deciding inconsistency of Π is Π_2^p-complete in general and co-NP-complete already for normal programs Π, cf. Dantsin *et al.* [2001], no polynomial size proof for inconsistency that can be verified in polynomial time is feasible in general. In particular, this applies to resolution proofs, which by Haken's [1985] celebrated result may have super-polynomial size.

Notwithstanding these aspects, recently the ASP-DRUPE format for inconsistency proofs of disjunctive logic programs has been proposed Alviano *et al.* [2019]. It was inspired by work on inconsistency proofs for SAT and hinges on clausal proof variants Gelder [2008]; Goldberg and Novikov [2003] that have certain properties, in particular the RUP (reverse unit propagation) and RAT (Resolution Asymmetric Tautology) properties. These proof formats share verifiability in polynomial time in the size of the proof and the input formula, and they can be tightly coupled with modern SAT solving techniques Alviano *et al.* [2019].

ASP-DRUPE is based on RUP and works as follows. It converts the program Π into SAT format, using Clarke completion and loop nogoods, and aims at constructing a proof during solving; notably, an ASP-DRUPE proof is verifiable in polynomial time in its length and the size of the completion nogoods. When the evaluation of Π ends with no answer set, then the constructed proof is validated in order to check whether the assessment of the solver is in accordance with the proof in the output.

The main differences between our notions of witnesses and ASP-DRUPE proofs are as follows. First and foremost, as already said our witness notions provide an explanation why atoms occur in an answer set, while ASP-DRUPE proofs aim to elucidate why no answer set is possible. It is non-obvious how the former can be efficiently reduced to the latter (and it may not be possible). Second, ASP-DRUPE constructs a proof at solving time, while our witness notions aim at ex post forensics, in which an already asserted answer set is analyzed. In principle, we could thanks to Theorem 4 generalize our algorithms to input interpretations M that are not necessarily answer sets: $\mathrm{MR}(\Pi, M)$ must not contain any constraint and each atom $p \in M$ must occur in the witness, in a set S_i of some pair (S_i, Π_i) resp. as atom p_i in some pair (p_i, Π_i). Third, ASP-DRUPE is geared towards the working of a modern ASP solver that uses unit propagation and clause-driven nogood learning (CDNL), in which clause transformations play an important role. One of the aims the proofs constructed is validating whether the solver in fact works correctly. Our witness notions instead take user perspective and aim to provide an explanation of an answer set in terms of the original clauses resp. rules in the program.

In conclusion, ASP-DRUPE proofs and our witness notions have similar yet different aims and features. They can in some respects be viewed as complementary; combining them in a single framework is of interest for future research.

7.4 ASP debugging

A general aim of logic program debugging is to explain for a given logic program which of its components cause a semantically unexpected effect that is usually called a *fault* or an *error*. Debugging logic programs amounts to answer why the semantics of a given program differs from expectations Brain and Vos [2005].

The existing debugging approaches for answer-set programs can be roughly classified into three realms: algorithmic Brain and Vos [2005]; Syrjänen [2006], meta-programming Gebser *et al.* [2008]; Oetsch *et al.* [2010]; Polleres *et al.* [2013]; Damásio *et al.* [2015]; Shchekotykhin [2015]; Dodaro *et al.* [2019] and stepping Pontelli *et al.* [2009]; Oetsch *et al.* [2018]. The meta-programming method uses logic program itself to debug a faulty logic program. It converts the input logic program and a possible candidate answer set into a logic program over a meta-language and then executes it together with a debugging logic program, which finds causes of a fault, where each cause is encoded by specific atoms in an answer set of the debugging logic program. It aims at explaining why an expected answer set is actually not an answer set of a given logic program, or why some atoms are not contained in any answer set of the logic program, or why no answer set exists. Our work aims to explain why some atoms belong to a given answer set of the program, which is opposite to that of the meta-programming approach. In this sense, our method can be regarded complementary.

The stepping method constructs interpretations stepwise by considering rules of an answer-set program at hand in a successive manner. Thus, users can observe the effects that rule applications have in the computation and decide which rule to consider active in the debugging. It guarantees that either an answer set will be reached, or some error will occur that provides hints why the semantics of the logic program differs from the user's expectations. A core concept of the stepping method is the one of computation, which a sequence $C = S_0, \ldots, S_n$ of states S_i such that S_0 is an initial state and S_{i+1} is a successor of S_i for all i $(0 \leq i < n)$. Informally, a state is a tuple (P, I, I^-, Υ) where

- every atom occurring in P belongs to $I \cup I^-$,
- I and I^- are disjoint sets of atoms considered to be **true** and **false**, respectively,
- P is a set of rules that are active (rule bodies are satisfied) and satisfied by I, and
- Υ is the collection of subsets of I considered to be unfounded in P *w.r.t.* I, *i.e.*, there is no rule in P which is an external support for X *w.r.t.* I. Recall that a rule r is an *external support* for a set X of atoms *w.r.t.* an interpretation I if $I \models bd(r)$, $I - X \models bd(r)$, $hd(r) \cap I \cap X \neq \emptyset$, and $p \in I$ implies $p \in X$ for each $p \in hd(r)$.

The state is *stable* if I is an answer set of P, or alternatively $\Upsilon = \{\emptyset\}$. It is *complete* for a logic program Π if $P = \{r \in \Pi \mid I \models bd(r)\}$. A state $S' = (P', I', I^{-'}, \Upsilon')$ is a *successor* of a state $S = (P, I, I^-, \Upsilon)$ if there are $r \in P' - P$ and two sets Δ, Δ^- of atoms occurring in r such that

- $P' = P \cup \{r\}, I \models bd(r)$,
- $I' = I \cup \Delta, I^{-'} = I^- \cup \Delta^-, (I \cup I^-) \cap (\Delta \cup \Delta^-) = \emptyset$,
- $X' \in \Upsilon'$ iff $X' = X \cup \Delta'$ with $X \in \Upsilon$ and $\Delta' \subseteq \Delta$ and r is not an external support for X' *w.r.t.* I'.

Succeeded computations are ones that end with a complete and stable state. Answer sets can be characterized in terms of succeeded computations.

In each computation step that is the extension of a computation by a further state, exactly one more rule is considered. This is somewhat dual to β-witnesses, where exactly one atom is derived in each step (p_i, Π_i), possibly using multiple rules in Π_i. A further difference is that users may decide (select) which atoms (in Δ) from the head of the chosen (disjunctive) rule r are considered to be **true**; the latter has no counterpart for β-witnesses.

Example 7.2 (example 4.9, cont'd)**.** *A succeeded computation for* Π *may be* $C = S_0, \ldots, S_4$ *where*

$$S_0 = (\emptyset, \emptyset, \emptyset, \{\emptyset\}), \qquad S_1 = (\{\delta_1\}, \{p, q, r\}, \emptyset, \{\emptyset\}), \qquad S_2 = (\{\delta_1, \delta_2\}, \{p, q, r\}, \emptyset, \{\emptyset\}),$$
$$S_3 = (\{\delta_1, \delta_2, \delta_3\}, \{p, q, r\}, \emptyset, \{\emptyset\}), \qquad S_4 = (\{\delta_1, \delta_2, \delta_3, \delta_4\}, \{p, q, r\}, \emptyset, \{\emptyset\}).$$

It is evident that any complete computation for Π involving (P,I,I^-,Υ) with $I^- \neq \emptyset$ will not lead to a stable state, otherwise the set of atoms that are considered to be **true** *in its final state is a proper subset of $\{p,q,r\}$. To justify or explain "why $M = \{p,q,r\}$ is an answer set of Π" according to the above succeeded computation C, one has to rely on "$\{p,q,r\}$ is an answer set of Π" at the state S_4, in addition to selecting p,q,r to be* **true** *at the state S_1.*

8 Conclusion

In this paper, we have considered the issue of giving a justification for an answer set of a logic program in terms of sets of rules that support building a modular proof for the atoms that are true in it. This enables one to provide a trustable answer to the question why some atoms occur in an answer set. To this end, we have introduced the new reduct $\mathrm{MR}(\Pi,M)$ of a logic program Π *w.r.t.* an interpretation M, which allows for a new characterization of the answer sets of a logic program with disjunctive rule heads. As a first application, we showed how this characterization can be fruitfully used to establish completeness of the modular minimal model decomposition result by Ben-Eliyahu-Zohary *et al.* [2016] for positive logic programs. Based on the reduct, we then introduced the notions of α- and β-witness of an answer set M of a logic program Π, which describe how to construct an explanation sequentially in a stepwise manner. We have furthermore developed variations and combinations of these notions that take into account the structural restrictions (compact witnesses, which in essence exclude the repeated use of disjunctive rules in a modular proof) and more fine-grained dependency conditions ($\alpha^\star$- and $\beta^\star$-witnesses, which allow for partially ordered proofs). We have studied the complexity of and provided algorithms for finding various witnesses. We also revealed connections between witness computation and MUS. Finally, our experimental results on synthetic, handcrafted, and legacy ASP and SAT benchmarks showed that computing a minimal β-witness is often feasible. And from the resulted β-witness, an explanation is quite evident in most cases.

The work in this paper complements previous work on explanations of answer sets of a logic program, cf. Pontelli *et al.* [2009]; Liu *et al.* [2010]; Cabalar *et al.* [2014]; Cabalar and Fandinno [2016]; Schulz and Toni [2016]; Fandinno and Schulz [2019]; Cabalar *et al.* [2020], and can be extended in several directions. On such direction is language extensions. For example, one can see that the notion of reduct can be extended to programs with strong ("classical") negation and rules of the form

$$l_1 \vee \cdots \vee l_k \leftarrow l_{k+1},\ldots,l_m, not\ l_{m+1},\ldots,not\ l_n, not\ not\ l_{n+1},\ldots,not\ not\ l_s\ (k \leq m \leq n \leq s)$$

where each l_i $(1 \leq i \leq s)$ is a literal Gelfond and Lifschitz [1991]; Lifschitz [1996]; see the electronic Appendix for more details. Notably, the Ferraris reduct Ferraris [2005] is not (directly) applicable here. Other examples would include choice rules, optimal answer sets, and aggregates, cf. Calimeri *et al.* [2020], where the latter are among the most diverse and controversial linguistic extensions of answer set programming Alviano and Faber [2018], as well as external atoms Eiter *et al.* [2005]; Gebser *et al.* [2014, 2016]; Dodaro *et al.* [2016]; Dodaro and Ricca [2020]. On the other hand, the approach that we presented may be explored for variations of the standard answer set semantics for logic programs, e.g., for paracoherent semantics Sakama and Inoue [1995]; Amendola *et al.* [2016] and for determining inference semantics Shen and Eiter [2019], as well as for other nonmonotonic formalisms, e.g., in circumscription. Using our new reduct, we can explain why an atom is in an answer set in terms of a witness. It would be worthwhile to investigate whether this reduct and the notions of witnesses in the paper can be used to explain why an atom is not in an answer set and why a logic program has no answer set, respectively.

The complexity study can be expanded and refined under the lens of parametric complexity and restrictions on the problem instances. To characterize the complexity of computing some minimal β-witness precisely is interesting in its own right and may give us more insight in the power of computations with limited use of NP and NP witness oracles.

Finally, another direction of research is to see how the results and algorithms in this work can be exploited in practice. For this, the development of dedicated and optimized implementations, e.g., for minimal model checking and witness computation, may be conceived for stand-alone usage or for integration into existing tools and software. It would be interesting to study how resolution proof explanations built from minimal $\alpha^{\star}_{fs}$ or $\beta^{\star}$-witness can be applied to improve ASP debugging Gebser *et al.* [2008]; Oetsch *et al.* [2010, 2018]; Dodaro *et al.* [2019].

Acknowledgments

Wang's work is partially supported by NSFC under grants 61976065 and U1836205, and Guizhou Science Support Project (2022-259). Eiter's research is supported by the European Humane-AI Net project (grant ICT-48-2020-RIA / 952026). Zhang's work is partially supported by National Science Foundation (NSF) grant DRL-1901704.

Proofs for Section 3

Proposition 3.1. *Let* Π *be a general logic program and* S *a model of* Π*. Then it holds that* $\mathrm{MR}(\Pi,S) \equiv \tau(\Pi)^S$.

Proof. Note that, according to Ferraris [2005],

- $\psi^X = \bot$ if $X \not\models \psi$;
- $p^X = p$ if $p \in X$ and $p \in \mathscr{A}$;
- $(\phi \otimes \psi)^X = \phi^X \otimes \psi^X$ for $\otimes \in \{\wedge, \vee, \supset\}$.

It is evident that $S \models r$ whenever $S \models \tau(r)$. It suffices to show that, for any $M \subseteq \mathscr{A}$, $M \models \mathrm{MR}(r,S)$ if and only if $M \models \tau(r)^S$ by $S \models \Pi$. Let $r \in \Pi$ and $\{\alpha\} = \mathrm{MR}(r,S)$. It is clear $\alpha^+ = r^+ \cap S$, $\alpha^- = r^-$, $r^{not} \cap S = \emptyset$ and $(r^- \cup r^{2not}) \subseteq S$. We have
$M \not\models \mathrm{MR}(r,S)$
iff $M \not\models \alpha$
iff $\alpha^- \subseteq M$ and $\alpha^+ \cap M = \emptyset$
iff $r^- \subseteq M$ and $r^+ \cap S \cap M = \emptyset$ due to $\alpha^- = r^-$ and $\alpha^+ = r^+ \cap S$
iff $M \models (\bigwedge r^-)^S$ and $M \not\models (\bigvee r^+)^S$
iff $M \models [(\bigwedge r^-) \wedge (\bigwedge_{p \in r^{not}} p \supset \bot) \wedge (\bigwedge_{q \in r^{2not}} (q \supset \bot) \supset \bot)]^S$ and $M \not\models (\bigvee r^+)^S$ since $(\bigwedge_{p \in r^{not}} \bot \supset p)^S \equiv \top$ by $S \cap r^{not} = \emptyset$, and $[\bigwedge_{q \in r^{2not}} (q \supset \bot) \supset \bot]^S \equiv \top$ by $r^{2not} \subseteq S$
iff $M \not\models \tau(r)^S$. □

Lemma 3.1. *Let* Π *be a logic program and* $M \subseteq \mathscr{A}$*. Then* $\mathrm{MR}(\Pi,M) = \mathrm{MR}(\Pi^M,M)$.

Proof. Let $r \in \Pi$. We have that
$r^+ \cap M \leftarrow r^-$ belongs to $\mathrm{MR}(\Pi,M)$
iff $r^{not} \cap M = \emptyset$ and $r^- \subseteq M$
iff $r^+ \leftarrow r^-$ belongs to Π^M and $r^- \subseteq M$
iff $r^+ \cap M \leftarrow r^-$ belongs to $\mathrm{MR}(\Pi^M,M)$. □

Lemma 3.2. *Let α be a clause and S be an interpretation. Then*

(i) $S \models \alpha$ *iff* $S \models \mathrm{MR}(\{\alpha\}, S)$*;*

(ii) $\neg\overline{S} \cup \{\alpha\} \equiv \neg\overline{S} \cup \mathrm{MR}(\{\alpha\}, S)$.

Proof. (i) In the case $\alpha^- - S \neq \emptyset$, $\mathrm{MR}(\{\alpha\}, S) = \emptyset$ holds. It is trivial that $S \models \alpha$ and $S \models \mathrm{MR}(\{\alpha\}, S)$.

In the case $\alpha^- - S = \emptyset$, *i.e.*, $\alpha^- \subseteq S$, let $\{\beta\} = \mathrm{MR}(\{\alpha\}, S)$. It is evident that $\alpha^- = \beta^-$ and $\beta^+ = \alpha^+ \cap S$. We have
$S \models \alpha$
iff $\alpha^- \subseteq S$ implies $S \cap \alpha^+ \neq \emptyset$
iff $S \cap \alpha^+ \neq \emptyset$ since $\alpha^- \subseteq S$
iff $S \cap (\alpha^+ \cap S) \neq \emptyset$
iff $S \cap \beta^+ \neq \emptyset$ by $\beta^+ = \alpha^+ \cap S$
iff $S \models \beta$ and $\beta^- = \alpha^- \subseteq S$.

(ii) Similar to the case (i), it trivially holds when $\alpha^- - S \neq \emptyset$. Suppose $\alpha^- - S = \emptyset$ and $\{\beta\} = \mathrm{MR}(\{\alpha\}, S)$. For any interpretation I, we have
$I \models \neg\overline{S} \cup \{\alpha\}$
iff $I \models \neg\overline{S}$ and $I \models \alpha$
iff $I \models \neg\overline{S}$ and, $\alpha^- \subseteq I$ implies $I \cap \alpha^+ \neq \emptyset$
iff $I \models \neg\overline{S}$ and, $\beta^- \subseteq I$ implies $I \cap \alpha^+ \neq \emptyset$ by $\alpha^- = \beta^-$
iff $I \models \neg\overline{S}$ and, $\beta^- \subseteq I$ implies $I \cap (\alpha^+ \cap S) \neq \emptyset$ by $I \subseteq S$
iff $I \models \neg\overline{S}$ and, $\beta^- \subseteq I$ implies $I \cap \beta^+ \neq \emptyset$ by $\beta^+ = \alpha^+ \cap S$
iff $I \models \neg\overline{S} \cup \mathrm{MR}(\{\alpha\}, S)$. □

Proposition 3.2. *A set $S \subseteq \mathscr{A}$ is a minimal model of a clause theory Σ iff S is a minimal model of* $\mathrm{MR}(\Sigma, S)$.

Proof. Let $\beta \in \mathrm{MR}(\Sigma, S)$ which is obtained from the clause $\alpha \in \Sigma$ by the reduction, *i.e.*,

$$\beta^+ = \alpha^+ \cap S, \qquad \alpha^- = \beta^- \text{ and } \qquad \alpha^- \subseteq S. \tag{14}$$

By (i) of Lemma 3.2, S is a model of Σ if and only if S is a model of $\mathrm{MR}(\Sigma, S)$.

The model S of Σ is minimal
iff $\Sigma \cup \neg\overline{S} \cup \{\bigvee \neg S\}$ is unsatisfiable
iff $\Sigma \cup \neg\overline{S} \models p$, for each $p \in S$
iff $\mathrm{MR}(\Sigma, S) \cup \neg\overline{S} \models p$ for each $p \in S$ by (ii) of Lemma 3.2
iff $\mathrm{MR}(\Sigma, S) \cup \neg\overline{S} \cup \{\bigvee \neg S\}$ is unsatisfiable
iff The model S of $\mathrm{MR}(\Sigma, S)$ is minimal. □

Theorem 1 (Minimal model characterization)**.** *For every clause theory Σ and $S \subseteq \mathscr{A}$, the items (i)–(iii) are equivalent:*

(i) *S is a minimal model of Σ.*

(ii) *S is the least model (under set inclusion) of* $\mathrm{MR}(\Sigma, S)$.

(iii) $S = \{q \in \mathscr{A} \cup \{\bot\} \mid \mathrm{MR}(\Sigma, S) \models q\}$.

Proof. $(i) \Rightarrow (ii)$. S is a minimal model of Σ
$\Rightarrow S$ is a minimal model of $\mathrm{MR}(\Sigma, S)$ by Proposition 3.2
$\Rightarrow S$ is a minimal model of $\mathrm{MR}(\Sigma, S)$ and $S \subseteq \mathrm{var}(\mathrm{MR}(\Sigma, S))$
$\Rightarrow S$ is a minimal model of $\mathrm{MR}(\Sigma, S)$ and $S = \mathrm{var}(\mathrm{MR}(\Sigma, S))$ by $\mathrm{var}(\mathrm{MR}(\Sigma, S)) \subseteq S$
$\Rightarrow S$ is the unique minimal model of $\mathrm{MR}(\Sigma, S)$
$\Rightarrow S$ is the least model of $\mathrm{MR}(\Sigma, S)$.

$(ii) \Rightarrow (iii)$. Let p be an atom.

$(\subseteq)$ $p \in S$
$\Rightarrow p$ belongs to the least model S of $\mathrm{MR}(\Sigma, S)$ by (ii)
$\Rightarrow p \in M$ for each $M \models \mathrm{MR}(\Sigma, S)$
$\Rightarrow p \in \{q \text{ is an atom} \mid \mathrm{MR}(\Sigma, S) \models q\}$.

$(\supseteq)$ $p \in \{q \text{ is an atoms} \mid \mathrm{MR}(\Sigma, S) \models q\}$
$\Rightarrow p \in \{q \text{ is an atom} \mid q \in M \text{ for any } M \models \mathrm{MR}(\Sigma, S)\}$
$\Rightarrow p \in M$ for any $M \models \mathrm{MR}(\Sigma, S)$
$\Rightarrow p \in S$ by (ii).

$(iii) \Rightarrow (i)$. Note that if $\mathrm{MR}(\Sigma, S)$ is unsatisfiable, then $\mathrm{MR}(\Sigma, M) \models \bot$ and $S \subset \{q \in \mathscr{A} \cup \{\bot\} \mid \mathrm{MR}(\Sigma, M) \models q\}$, which contradicts with the precondition. Thus, $\mathrm{MR}(\Sigma, S)$ is satisfiable.

$S = \{q \in \mathscr{A} \cup \{\bot\} \mid \mathrm{MR}(\Sigma, S) \models q\}$
$\Rightarrow \mathrm{MR}(\Sigma, S) \models \bigwedge S$ and $\mathrm{MR}(\Sigma, S)$ is satisfiable
$\Rightarrow S \subseteq S'$ for each $S' \models \mathrm{MR}(\Sigma, S)$
$\Rightarrow S$ is the least model of $\mathrm{MR}(\Sigma, S)$ by $\mathrm{var}(\mathrm{MR}(\Sigma, S)) \subseteq S$
$\Rightarrow S$ is a minimal model of $\mathrm{MR}(\Sigma, S)$
$\Rightarrow S$ is a minimal model of Σ by Proposition 3.2. □

Lemma 3.3. *Let Σ be a clause theory and $M' \subseteq M$. Then $M' \models \mathrm{MR}(\Sigma, M)$ if and only if $M' \models \Sigma$.*

Proof. $(\Rightarrow)$ Let $\alpha \in \Sigma$ and suppose $\alpha^- \subseteq M'$. We have
$\alpha^- \subseteq M'$
$\Rightarrow \alpha^- \subseteq M$ by $M' \subseteq M$
$\Rightarrow \neg\alpha^- \cup \alpha^+ \cap M$ belongs to $\mathrm{MR}(\{\alpha\}, M)$
$\Rightarrow \neg\alpha^- \cup \alpha^+ \cap M$ belongs to $\mathrm{MR}(\Sigma, M)$
$\Rightarrow \alpha^+ \cap M' \neq \emptyset$ by $M' \models \mathrm{MR}(\Sigma, M)$.

It implies $M' \models \alpha$, thus $M' \models \Sigma$.

$(\Leftarrow)$ Let $\alpha \in \Sigma$ with $\alpha^- \subseteq M'$ and $\neg\alpha^- \cup \alpha^+ \cap M \in \mathrm{MR}(\Sigma, M)$. We have
$\alpha^- \subseteq M'$
$\Rightarrow \alpha^+ \cap M' \neq \emptyset$ by $\alpha \in \Sigma$ and $M' \models \Sigma$
$\Rightarrow \alpha^+ \cap M' \cap M \neq \emptyset$ by $M' \subseteq M$.

It follows $M' \models \mathrm{MR}(\{\alpha\}, M)$, thus $M' \models \mathrm{MR}(\Sigma, M)$. □

Proposition 3.3. *Suppose M is a model of a clause theory Σ and let $M' \subset M$. Then M' is a minimal model of Σ if and only if M' is a minimal model of* $\mathrm{MR}(\Sigma, M)$.

Proof. $(\Rightarrow)$ We show that (1) $M' \models \mathrm{MR}(\Sigma, M)$ and (2) $M^* \not\models \mathrm{MR}(\Sigma, M)$ for any $M^* \subset M'$.

(1) Let $\alpha \in \Sigma$ such that $\alpha^- \subseteq M'$ and $\mathrm{MR}(\{\alpha\}, M) \neq \emptyset$. We have
$\alpha^- \subseteq M'$
$\Rightarrow \alpha^- \subseteq M$ by $M' \subset M$
$\Rightarrow \alpha^+ \cap M' \neq \emptyset$ since $M' \models \Sigma$ and $\alpha \in \Sigma$.

Thus, $M' \models \alpha$, which implies $M' \models \mathrm{MR}(\Sigma, M)$.

(2) If there is $M^* \subset M'$ such that $M^* \models \mathrm{MR}(\Sigma, M)$, then $M^* \models \Sigma$ by Lemma 3.3. It is a paradox since M' is a minimal model of Σ.

($\Leftarrow$) We show that (1) $M' \models \Sigma$ and $M^* \not\models \Sigma$ for any $M^* \subset M'$.

(1) Let $\alpha \in \Sigma$ and $\alpha^- \subseteq M'$. We have

$\alpha^- \subseteq M'$

$\Rightarrow \alpha^- \subseteq M$ since $M' \subseteq M$

$\Rightarrow \neg\alpha^- \cup \alpha^+ \cap M \in \mathrm{MR}(\{\alpha\}, M)$

$\Rightarrow \neg\alpha^- \cup \alpha^+ \cap M \in \mathrm{MR}(\Sigma, M)$

$\Rightarrow \alpha^+ \cap M \cap M' \neq \emptyset$ by $M' \models \mathrm{MR}(\Sigma, M)$

$\Rightarrow \alpha^+ \cap M' \neq \emptyset$ by $M' \subset M$.

It follows $M' \models \alpha$, which implies $M' \models \Sigma$.

(2) Suppose that $M^* \models \Sigma$ for some $M^* \subset M'$. It implies $M^* \models \mathrm{MR}(\Sigma, M)$ by $M^* \subset M$ and Lemma 3.3. It is a paradox since M' is a minimal model of $\mathrm{MR}(\Sigma, M)$. □

The following lemma establishes the 1-1 correspondence between $\mathbb{S}_\Sigma$ and $\mathbb{SG}_\Sigma$ for a clause theory Σ. The following lemma is evident.

Lemma 3.4. *Let Σ be a clause theory and $p, q \in \mathscr{A}$.*

(i) $\mathbb{G}_\Sigma$ has a path (p, δ, q) if and only if (p, q) is an arc of G_Σ where $\delta \in \Sigma$.

(ii) If S is a vertex of $\mathbb{SG}_\Sigma$ with $S \cap \mathscr{A} \neq \emptyset$ then $S \cap \mathscr{A}$ is a vertex of $\mathbb{S}_\Sigma$.

(iii) If S is a vertex of $\mathbb{S}_\Sigma$ then $\mathbb{SG}_\Sigma$ has a vertex S' with $S = S' \cap \mathscr{A}$.

Proof. (i) It is evident.

(ii) S is a vertex of $\mathbb{SG}_G$ with $S \cap \mathscr{A} \neq \emptyset$

$\Rightarrow S \cap \mathscr{A}$ is maximal *s.t.* $\mathbb{SG}_G$ has a path from p to q for any p and q in $S \cap \mathscr{A}$

$\Rightarrow S \cap \mathscr{A}$ is maximal *s.t.* $\mathbb{S}_\Sigma$ has a path from p to q for any p and q in $S \cap \mathscr{A}$ by (i)

$\Rightarrow S \cap \mathscr{A}$ is a vertex of $\mathbb{S}_\Sigma$.

(iii) S is a vertex of $\mathbb{S}_\Sigma$

$\Rightarrow S$ is maximal *s.t.* $\mathbb{S}_\Sigma$ has a path from p to q for any p and q in S

$\Rightarrow$ there is S' with $S = S' \cap \mathscr{A}$ *s.t.* S' is maximal and $\mathbb{SG}_G$ has a path from p to q for any p and q in S' by (i)

$\Rightarrow S'$ is a vertex of $\mathbb{SG}_G$. □

Proposition 3.4. *Let M be a nonempty minimal model of a clause theory Σ and $\mathrm{MR}(\Sigma, M) = \Sigma$. Then:*

(i) $\mathscr{A} \cap S = \emptyset$ for every source S of $\mathbb{SG}_\Sigma$.

(ii) Let $\mathbb{S}$ be resulted from $\mathbb{SG}_\Sigma$ by removing all empty sources. Then for every source S of $\mathbb{S}$, $S \cap \mathscr{A}$ is a minimal model of Σ_S, and $M - S \cap \mathscr{A}$ is a minimal model of $Reduce(\Sigma, S \cap \mathscr{A}, \emptyset)$.

(iii) Let S be a node of $\mathbb{SG}_\Sigma$ and $S \cap \mathscr{A} \neq \emptyset$, $X_S = \{S' \cap \mathscr{A} \mid S' \in D_{\mathbb{SG}_\Sigma}(S)\}$ and $\Gamma = Reduce(\Sigma, X_S, \emptyset)$. Then S is a minimal model of Γ_S.

Proof. (i) S is a nonempty source of $\mathbb{SG}_\Sigma$

$\Rightarrow S$ must be a singleton $\{p\}$

$\Rightarrow p$ occurs negatively in every clauses of Σ

$\Rightarrow \Sigma \not\models p$.
It is a contradiction since $p \in M$, $\Sigma = \mathrm{MR}(\Sigma, M)$ and $\mathrm{MR}(\Sigma, M) \models p$ by Theorem 2.

(ii) Firstly, we have
$S \not\models \Sigma_S$
$\Rightarrow S \not\models \alpha$ for some $\alpha \in \Sigma_S$
$\Rightarrow S \not\models \alpha$ for some $\alpha \in \Sigma$ by $\Sigma_S \subseteq \Sigma$
$\Rightarrow S \not\models \alpha$ for some $\alpha \in \mathrm{MR}(\Sigma, M)$ by $\Sigma = \mathrm{MR}(\Sigma, M)$
$\Rightarrow \alpha^+ \cap S = \emptyset$ and $\alpha^- \subseteq S$
$\Rightarrow \alpha^+ = \emptyset$ and $\alpha^- \subseteq S$ by $\alpha \in \Sigma_S$
$\Rightarrow \alpha^+ \cap M = \emptyset$ and $\alpha^- \subseteq M$ by $S \subseteq M$
$\Rightarrow M \not\models \alpha$
$\Rightarrow M \not\models \Sigma$ by $\alpha \in \Sigma$.
It contradicts with M is a minimal model of Σ. Thus, $S \models \Sigma_S$.

Suppose $\exists S' \subset S$ and $S' \models \Sigma_S$. We will show that $M' = M - (S - S')$ is a model of Σ. Let $\alpha \in \Sigma$ and $\alpha^- \subseteq M'$. We consider the following two cases:

(a) $\alpha \in \Sigma_S$. We have that
$\alpha^- \subseteq M - (S - S')$
$\Rightarrow \alpha^- \subseteq S \cap (M - (S - S'))$ since $\alpha^- \subseteq S$ by $\alpha \in \Sigma_S$
$\Rightarrow \alpha^- \subseteq S'$ by $S' = S \cap (M - (S - S'))$ and $S' \subset S$
$\Rightarrow S' \cap \alpha^+ \neq \emptyset$ by $S' \models \Sigma_S$ and $\alpha \in \Sigma_S$
$\Rightarrow \alpha^+ \cap (M - (S - S')) \neq \emptyset$ by $S' \subseteq M - (S - S')$.
It implies $M' \models \alpha$.

(b) $\alpha \in \Sigma - \Sigma_S$. Recall that S is a source of Σ. We consider the two cases:

- $\alpha^- - S = \emptyset$
  $\Rightarrow \alpha^- \subseteq S$
  $\Rightarrow \alpha^+ - S \neq \emptyset$ by $\alpha \in \Sigma - \Sigma_S$
  $\Rightarrow (M - (S - S')) \cap \alpha^+ \neq \emptyset$ by $\alpha^+ \subseteq M$
  $\Rightarrow M' \models \alpha$.
- $\alpha^- - S \neq \emptyset$
  $\Rightarrow \alpha^+ \cap S = \emptyset$ since S is a source of Σ
  $\Rightarrow (M - (S - S')) \cap \alpha^+ \neq \emptyset$ by $\alpha^+ \subseteq M$ and $\alpha^+ \neq \emptyset$
  $\Rightarrow M' \cap \alpha^+ \neq \emptyset$
  $\Rightarrow M' \models \alpha$.

This implies $M' \models \Sigma$. A contradiction follows since $M' \subset M$ and M is a minimal model of Σ. Thus, S is a minimal model of Σ_S.

Secondly, let $S' = M - S \cap \mathscr{A}$ and $\Gamma = Reduce(\Sigma, S \cap \mathscr{A}, \emptyset)$. we show by contradiction that (a) $S' \models \Gamma$ and (b) for any $S'' \subset S'$, $S'' \not\models \Gamma$.

(a) $S' \not\models \Gamma$
$\Rightarrow \exists r \in \Sigma\ s.t.\ \{r'\} = Reduce(\{r\}, S \cap \mathscr{A}, \emptyset) \subseteq Reduce(\Sigma, S \cap \mathscr{A}, \emptyset)$ and $S' \not\models r'$
$\Rightarrow S' \cup S \cap \mathscr{A} \not\models r'$
$\Rightarrow M = S' \cup S \cap \mathscr{A} \not\models r$
$\Rightarrow M \not\models \Sigma$, a contradiction.

(b) There is $S'' \subset S'$ *s.t.* $S'' \models \Gamma$. Let $\alpha \in \Sigma$ and let us consider the following two cases:

(b.1) $Reduce(\{r\}, S \cap \mathscr{A}, \emptyset) \equiv \{\top\}$
$\Rightarrow r^+ \cap S \neq \emptyset$
$\Rightarrow r^+ \cap (S \cup S'') \neq \emptyset$
$\Rightarrow S \cup S'' \models r$.

(b.2) $Reduce(\{r\}, S \cap \mathscr{A}, \emptyset) = \{r'\}$ and $r' \not\equiv \top$
$\Rightarrow S' \cap r^+ \neq \emptyset$ since $Var(\Gamma) = S'$ and $S' \models \Gamma$
$\Rightarrow (S' \cup S'') \cap r^+ \neq \emptyset$
$\Rightarrow S \cup S'' \models r$.

The above two cases show that the proper subset $S \cup S''$ of M is a model of Σ, a contradiction.

(iii) It follows from (ii) by an iterative application. □

Theorem 2. *Let M be a model of a clause theory Σ. Then it holds that*

(i) M has the modular property w.r.t. Σ if M is minimal and $\mathrm{MR}(\Sigma, M) = \Sigma$*;*

(ii) M is a minimal model of Σ iff CheckMinMR(Σ, M) returns **true**.

Proof. (i) It follows from (ii) of Proposition 3.4.

(ii) It follows from (i) and Theorem 6.4 of Ben-Eliyahu-Zohary *et al.* [2016]. □

Theorem 3 (Answer set characterization)**.** *For every logic program Π and $M \subseteq \mathscr{A}$, the items (i)–(iii) are equivalent:*

(i) M is an answer set of Π.

(ii) M is the least model of $\mathrm{MR}(\Pi, M)$.

(iii) $M = \{q \in \mathscr{A} \cup \{\bot\} \mid \mathrm{MR}(\Pi, M) \models q\}$.

Proof. $(i) \Leftrightarrow (ii)$ M is an answer set of Π
iff M is a minimal model of Π^M
iff M is the least model of $\mathrm{MR}(\Pi^M, M)$ by Theorem 2
iff M is the least model of $\mathrm{MR}(\Pi, M)$ by Lemma 3.1.

$(ii) \Rightarrow (iii)$ and $(iii) \Rightarrow (i)$ can be similarly proved as that of Theorem 2. □

Proofs in Section 4

Proposition 4.1. *Let $M \subseteq \mathscr{A}$, let $B, S \subseteq M$ be disjoint subsets of M, and let Π be a logic program.*

(i) If $\Pi' \in \mathrm{MW}(B, \Pi, S, M)$ then some $S' \subseteq S$ exists such that $\mathrm{MR}(\Pi', M) \cup S' \cup \{\bigvee \neg B\}$ is an MUS.

(ii) If $\mathrm{MR}(\Pi', M) \cup S' \cup \{\bigvee \neg B\}$ is an MUS of $\mathrm{MR}(\Pi, M) \cup S \cup \{\bigvee \neg B\}$ such that $S' \subseteq S$, then some $\Pi'' \in \mathrm{MW}(B, \Pi, S, M)$ exists such that $\Pi'' \subseteq \Pi'$.

Proof. Firstly note that $\mathrm{MR}(\Pi, M) \cup S$ is satisfiable since it is positive.

(i) $\Pi' \in \mathrm{MW}(B, \Pi, S, M)$
$\Rightarrow \Pi'$ is a minimal subset of Π such that $\mathrm{MR}(\Pi', M) \cup S \models B$
$\Rightarrow \mathrm{MR}(\Pi', M)$ is a minimal subset of $\mathrm{MR}(\Pi, M)$ such that $\mathrm{MR}(\Pi', M) \cup S \models B$
$\Rightarrow \mathrm{MR}(\Pi', M)$ is a minimal subset of $\mathrm{MR}(\Pi, M)$ such that $\mathrm{MR}(\Pi', M) \cup S \cup \{\bigvee \neg B\}$ is unsatisfiable

$\Rightarrow$ $\mathrm{MR}(\Pi',M)$ is a minimal subset of $\mathrm{MR}(\Pi,M)$ such that $\mathrm{MR}(\Pi',M)\cup S'\cup\{\bigvee\neg B\}$ is minimally unsatisfiable for some $S'\subseteq S$ since $\mathrm{MR}(\Pi',M)\cup S$ is satisfiable
$\Rightarrow$ $\mathrm{MR}(\Pi',M)\cup S'\cup\{\bigvee\neg B\}$ is an MUS for some $S'\subseteq S$.

(ii) Note that $\mathrm{MR}(\Pi',M)\cup S'\cup\{\bigvee\neg B\}$ is unsatisfiable. It implies $\mathrm{MR}(\Pi',M)\cup S'\models B$. We have $\mathrm{MR}(\Pi',M)\cup S'\cup\{\bigvee\neg B\}$ is an MUS
$\Rightarrow$ $\mathrm{MR}(\Pi',M)\cup S'\cup\{\bigvee\neg B\}$ is unsatisfiable
$\Rightarrow$ $\mathrm{MR}(\Pi',M)\cup S\cup\{\bigvee\neg B\}$ is unsatisfiable
$\Rightarrow$ There is a minimal subset Π'' of Π' such that $\mathrm{MR}(\Pi'',M)\cup S\cup\{\bigvee\neg B\}$ is unsatisfiable
$\Rightarrow$ There is a minimal subset Π'' of Π' such that $\mathrm{MR}(\Pi'',M)\cup S\models B$
$\Rightarrow$ There is a minimal subset Π'' of Π such that $\mathrm{MR}(\Pi'',M)\cup S\models B$
$\Rightarrow$ $\Pi''\in\mathrm{MW}(B,\Pi,S,M)$ for some $\Pi''\subseteq\Pi'$. □

Proposition 4.2. *Algorithm 4 is sound and moreover complete if the involved MUS solver mus is complete.*

Proof. (1) In the case $mus = nil$, it is evident that the returned Π^* is a witness of B under S *w.r.t.* M in line 12. The minimality of Π^* follows from the fact that no proper subset of Π^* is a witness of B under S *w.r.t.* M. Thus, it is sound.

In addition, for each minimal witness $\Pi^o\subseteq\Pi$ of B under S *w.r.t.* M, Π^o can be obtained by setting the order on the rules in Π: every rule in Π^o is ordered after all the rules in $\Pi-\Pi^o$. The **ForEach** loop lines 11-13 will remove exactly all the rules in $\Pi-\Pi^o$. This implies the completeness of the algorithm.

(2) In the case $mus\neq nil$, the resulting Π' in line 3 is a minimal witness of B under S *w.r.t.* M by (ii) of Proposition 4.1. Note further that the **ForEach** loop lines 5 and 6 computes $\Pi^*\subseteq\Pi$ such that $|\Pi^*|=\Pi'$ and $\mathrm{MR}(\Pi^*)=\Pi'$. Thus, the returned Π^* is a minimal witness of B under S *w.r.t.* M, which implies the soundness of this algorithm.

Furthermore, for each minimal witness $\Pi^*\subseteq\Pi$ of B under S *w.r.t.* M, any complete MUS solver *mus* will compute an MUS Π' of $\mathrm{MR}(\Pi,M)\cup S\cup\{\bigvee\neg B\}$ by (i) of Proposition 4.1. Then Π^* can be computed by the **ForEach** loop on lines 5-8 by setting an order on the rules of Π: the rules in $\Pi^*\subseteq\Pi$ are before the the rules in $\Pi-\Pi^*$. Hence, the completeness of Algorithm 4 follows. □

Lemma 4.1. *Let Σ be a clause theory and M a minimal model of Σ with $\Sigma=\mathrm{MR}(\Sigma,M)$. Then $G=(\{(S_i,\Sigma_i)\mid 1\leq i\leq n\},E')$ obtained from $\mathbb{S}_\Sigma=(\{S_1,\ldots,S_n\},E)$ s.t.*

- *$\Sigma_i\in\mathrm{MW}(S_i,\Sigma,S_{\triangleright i},M)$ $(1\leq i\leq n)$, where $S_{\triangleright i}=\bigcup D_{\mathbb{S}_\Sigma}(S_i)$, and*
- *$((S_i,\Sigma_i),(S_j,\Sigma_j))\in E'$ whenever $(S_i,S_j)\in E$*

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

Proof. (a) It is clear that G is directed and acyclic since $\mathbb{S}_\Sigma$ is so.

(b) $\{S_1,\ldots,S_n\}$ is clearly a partition of M.

(c) Recall that $\mathrm{MR}(\Sigma,M)\models M$ by Theorem 2. It implies that $\mathrm{MW}(S_i,\Sigma,S_{\triangleright i},M)$ $(1\leq i\leq n)$ is nonempty and Σ_i is a witness of S_i under X_i *w.r.t.* M. By (ii) of Proposition 3.4, S_i is a minimal model of $Reduce(\Sigma_i,X_i,\emptyset)$. It implies Σ_i is not a witness of S_j $(j\neq i)$ under X_i *w.r.t.* M, where X_i is defined in Definition 4.2.What remains is to show that $\Sigma_i\cap\Sigma_j=\emptyset$ for any $1\leq i<j\leq n$. Suppose that there is $r\in\Sigma_i\cap\Sigma_j$ for some $1\leq i\neq j\leq n$. Let $r^t=Reduce(r,S_{\triangleright t},\emptyset)$ and $\Sigma^t=Reduce(\Sigma_t,S_{\triangleright t},\emptyset)$ for $t\in\{i,j\}$. We have
$\Sigma_i\in\mathrm{MW}(S_i,\Sigma,S_{\triangleright i},M)$
$\Rightarrow$ $\Sigma_i\subseteq\Sigma$ is minimal *s.t.* $\mathrm{MR}(\Sigma_i,M)\cup S_{\triangleright i}\models S_i$ by Definition 4.1
$\Rightarrow$ Σ_i is minimal *s.t.* $\Sigma_i\cup S_{\triangleright i}\models S_i$ by $\Sigma_i\subseteq\Sigma$ and $\mathrm{MR}(\Sigma_i,M)=\Sigma_i$

$\Rightarrow \Sigma^i$ is minimal *s.t.* $\Sigma^i \models S_i$ by $S_{\triangleright i} \models \Sigma^i \leftrightarrow \Sigma_i$
$\Rightarrow S_i$ is the unique minimal model of $\Sigma^i_{S_i}$ by (iii) of Lemma 3.4 and (ii) of Proposition 3.4
$\Rightarrow r^i \in \Sigma^i_{S_i}$ (otherwise, $\Sigma^i = \Sigma^i - \{r^i\} \models S_i$, *i.e.*, $\Sigma_i - \{r\} \cup S_{\triangleright i} \models S_i$)
$\Rightarrow atoms(r^i) \subseteq S_i$ and $atoms(r^i) \neq \emptyset$ (otherwise r^i is an empty clause, which implies that Σ is unsatisfiable), where $atoms(e)$ is the set of atoms occurring in e
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in S_i and those from $atoms(r^i)$ in particular
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in S_i and some atoms from $atoms(r)$
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in S_j and some atoms from $atoms(r)$ for the same above reason
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in $S_i \cup S_j$, a contradiction. □

Proposition 4.3. *Let M be an answer set of a logic program Π and $\Sigma = \mathrm{MR}(\Pi, M)$. Then the DAG $G = (\{(S_i, \Pi_i) \mid 1 \leq i \leq n\}, E')$ obtained from $\mathbb{S}_\Sigma = (\{S_1, \dots, S_n\}, E)$ s.t.*

- *$\Pi_i \in \mathrm{MW}(S_i, \Pi, S_{\triangleright i}, M)$ $(1 \leq i \leq n)$, where $S_{\triangleright i} = \bigcup D_{\mathbb{S}_\Sigma}(S_i)$, and*
- *$((S_i, \Pi_i), (S_j, \Pi_j)) \in E'$ whenever $(S_i, S_j) \in E$*

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

Proof. (a) It is clear that G is directed and acyclic since $\mathbb{S}_\Sigma$ is so.

(b) $\{S_1, \dots, S_n\}$ is clearly a partition of M.

(c) Recall that $\mathrm{MR}(\Sigma, M) \models M$ by Theorem 4. It implies that $\mathrm{MW}(S_i, \Pi, S_{\triangleright i}, M)$ $(1 \leq i \leq n)$ is nonempty and Π_i is a witness of S_i under X_i *w.r.t.* M. Since S_i is an answer set of $Reduce(\mathrm{MR}(\Pi_i), X_i, \emptyset)$, it implies that Π_i is not a witness of S_j $(j \neq i)$ under X_i *w.r.t.* M, where X_i is defined in Definition 4.2. What remains is to show that $\Pi_i \cap \Pi_j = \emptyset$ for any $1 \leq i < j \leq n$. Suppose that there is $r \in \Pi_i \cap \Pi_j$ for some $1 \leq i \neq j \leq n$. Let $r^t = Reduce(\mathrm{MR}(r, M), S_{\triangleright t}, \emptyset)$, $\Sigma_t = \mathrm{MR}(\Pi_t, M)$ and $\Sigma^t = Reduce(\Sigma_t, S_{\triangleright t}, \emptyset)$ for $t \in \{i, j\}$. Note that
$\Pi_i \in \mathrm{MW}(S_i, \Pi, S_{\triangleright i}, M)$
$\Rightarrow \Pi_i \subseteq \Pi$ is minimal *s.t.* $\mathrm{MR}(\Pi_i, M) \cup S_{\triangleright i} \models S_i$ by Definition 4.1
$\Rightarrow \Sigma_i \subseteq \Sigma$ is minimal *s.t.* $\Sigma_i \cup S_{\triangleright i} \models S_i$ by $\mathrm{MR}(\Pi_i, M) = \Sigma_i$
$\Rightarrow \Sigma_i \subseteq \Sigma$ is minimal *s.t.* $\mathrm{MR}(\Sigma_i, M) \cup S_{\triangleright i} \models S_i$ by $\mathrm{MR}(\Sigma_i, M) = \Sigma_i$
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in S_i and some atoms from $atoms(r)$ by Lemma 4.1
$\Rightarrow G_\Sigma$ has a cycle which mentions the atoms in $S_i \cup S_j$, a contradiction. □

Proposition 4.4. *Let G be an $\alpha^\star_{fs}$-witness of an answer set M w.r.t. a logic program Π and $G = G_0, G_1, G_2, \dots, G_k$ be graphs such that each G_i, $i = 1, \dots, k$ is an edge-contraction of G_{i-1}. Then G_k is a compact $\alpha^\star$-witness of M w.r.t. Π, where the edge-contraction of $((S, \Pi), (S', \Pi'))$ is the node $(S \cup S', \Pi \cup \Pi')$.*

Proof. We prove the statement by induction on $k \geq 0$. The base case $k = 0$ is trivial since $G_0 = G$. For the induction step, assume the statement holds for k and suppose that G_k is a compact $\alpha^\star$-witness of M *w.r.t.* Π for $k = i$. Let $G_{k+1} = (V_{k+1}, E_{k+1}) = (V - \{v_1, v_2\} \cup \{v\}, E')$ be an edge-contraction of $G_k = (V_k, E_k)$ with $v_i = (S_i, \Pi_i)$ $(i = 1, 2)$ and $v = (S_1 \cup S_2, \Pi_1 \cup \Pi_2)$. Note that, by Proposition 3.4, $Var(\mathrm{MR}(\Pi_j, M)) = S_j$ for any $v_j = (S_j, \Pi_j)$ in V_{k+1}. It is evident $\Pi_1 \cup \Pi_2$ is a minimal witness of $S_1 \cup S_2$ under $X = \bigcup_{(S', \Pi') \in D_{G_{k+1}}(v)} S'$, and $(\Pi_1 \cup \Pi_2) \cap \Pi' = \emptyset$ for any $(S', \Pi') \in V_k \cap V_{k+1}$. Similarly, for every node $v' = (S', \Pi')$ in V_{k+1} and it is the case that no Π_j can replace Π_i $(j \neq i)$ in the witness of G_{k+1}. The compactness of G_{k+1} is trivial simply due to $\Pi_i \cap \Pi_j = \emptyset$ for any Π_i, Π_j occurring in G_k by the induction assumption. Consequently, G_{k+1} is a compact $\alpha^\star$-witness of M *w.r.t.* Π. □

Lemma 4.2. *Let l be a literal and let Σ be a satisfiable clause theory s.t. $\Sigma \models l$ and no $\Sigma' \subset \Sigma$ satisfies $\Sigma' \models l$. Then:*

(i) the opposite literal $\bar{l}$ does not occur in Σ, and

(ii) if $\Sigma \models l'$ for some literal $l' \neq l$, then some proper subset $\Sigma' \subset \Sigma$ exists such that $\Sigma' \models l'$.

Proof. (i) $\Sigma \models_{min} l$
$\Rightarrow \Sigma \models l$ and $\Sigma - \{\alpha\} \not\models l$ for every $\alpha \in \Sigma$
$\Rightarrow$ for every $\alpha \in \Sigma$, there is an interpretation M *s.t.* $M \models \Sigma - \{\alpha\}$ and $M \not\models l$
$\Rightarrow M \not\models \alpha$ by $\Sigma \models l$ (otherwise $M \models l$, a contradiction), and $M \models \bar{l}$
$\Rightarrow \bar{l}$ does not occur in α for every $\alpha \in \Sigma$
$\Rightarrow \bar{l}$ does not occur in Σ.

(ii) Suppose that there is no proper subset Π of Σ satisfying $\Pi \models l'$, *i.e.* $\Sigma \models_{min} l'$. We have
$\Sigma \models_{min} l'$
$\Rightarrow \Sigma \models l'$
$\Rightarrow l'$ has a linear resolution proof from Σ, in which the last resolvent is l', since l' is a prime implicate of Σ and every prime implicate has a (linear) resolution proof
$\Rightarrow$ the (linear) resolution must involve every clauses in Σ since $\Pi \not\models l'$ for any $\Pi \subset \Sigma$
$\Rightarrow l$ must occur in the last resolvent of the resolution proof by (i)
$\Rightarrow$ it contradicts with the fact that the last resolvent is $l'(\neq l)$. □

Proposition 4.5. *Given a logic program Π, an answer set M of Π and two disjoint subsets B, S of M, the call of MinBetaBSWitness(Π, B, S, M) returns some minimal β-witness W of B under Π and S w.r.t. M.*

Proof. Let $B = [(p_1, \Sigma_1), \ldots, (p_n, \Sigma_n)]$ be returned by MinBetaBSWitness(Σ, B, S, M). It is evident $B = \{p_i \mid 1 \leq i \leq n\}$. By (i) and (ii) of Lemma 4.2, the inner **while-loop** (lines 10-14) will find Σ_{p_k} which always exists and is a minimal witness of $\{p_k\}$ under T $(1 \leq k \leq n)$ *w.r.t.* M. It is easy to see that the Σ_u following the **while-loop** (lines 10-14) cannot derive any other atom under T. Thus the $rm(\Sigma_u, \Pi, M)$ (line-15) is not a witness of any other atom v under T. This algorithm will always terminate since in each iteration of outer **while-loop** (lines 2-17), at least one more atom is added into T, which implies the set $B - T$ will definitely become $\emptyset$. □

Proposition 4.6. *Given a logic program Π and an answer set M of Π, the call of MinBetaWitness(Π, M) returns some minimal β-witness W of M w.r.t. Π.*

Proof. Firstly, the algorithm will terminates since each iteration of the **while-loop** (lines 3-8) will delete at least one (source) node from G and G has only finite number of nodes. Secondly, in each iteration of the loop, all atoms in U ($\mathscr{A} \cap \bigcup D_{\mathbb{SG}_\Sigma}(S)$) have been witnessed by calling MinBetaBSWitness (line 5) whose correctness is guaranteed by Proposition 4.5 and M consists of exactly the atoms in those nodes of G. □

Lemma 4.3. *Let M be an answer set of a logic program Π, let $[(p_i, \Sigma_i) \mid 1 \leq i \leq n]$ be the output of MinBetaWitness(Π, M) and let G be the output of MinBetaStarWitness(Π, M). Then, for every i $(1 \leq i \leq n)$,*

$$\Sigma_i^- \cap \Delta_i = \Sigma_i^- \cap D_G^i, \tag{15}$$

where $\Sigma_i^- = \bigcup\{\alpha^- \mid \alpha \in \Sigma_i\}$, $\Delta_i = \{p_k \mid 1 \leq k \leq i-1\}$, and $D_G^k = \{p \mid (p, \Sigma_p) \in D_G((p_k, \Sigma_k))\}$.

Proof. Note that $\Delta_0 = D_G^0 = \emptyset$ and $\Delta_1 = \{p_1\}$. In the **for** loop (lines 3-9) of MinBetaStarWitness with $i = 2$,
$p_1 \in \Sigma_2^- \cap D_G^2$
iff the edge $((p_1,\Sigma_1),(p_2,\Sigma_2))$ is added into E (line 8)
iff $p_1 \in \Sigma_2^- \cap \Delta_2$. It implies equation (15) holds for $i = 1,2$.

In the **for** loop (lines 3-9) of MinBetaStarWitness with $i > 2$,
$q \in \Sigma_i^- \cap D_G^i$
iff either the edge $((q,\Sigma_q),(p_i,\Sigma_i))$ is added into E (line 8), or $q \in D_G^k$ for the least k $(1 \le k < i)$ such that G has a path from (q,Σ_q) to (p_k,Σ_{p_k}) and $((p_k,\Sigma_{p_k}),(p_i,\Sigma_i))$ is added into E (line 8)
iff either $q \in \Sigma_i^- \cap \Delta_i$ and G has no path from (q,Σ_q) to $(q',\Sigma_{q'})$ for any $q' \in \Sigma_i^- \cap \Delta_i$ by the **while** loop (lines 5-8), or G has a path from (q,Σ_q) to (p_k,Σ_{p_k}) for some $p_k \in \Sigma_i^- \cap \Delta_i$
iff $q \in \Sigma_i^- \cap \Delta_i$. □

Proposition 4.7. *Let $M \neq \emptyset$ be an answer set of a logic program Π and G be the output of MinBetaStarWitness(Π,M).*

(i) G is a minimal $\beta^\star$-witness of M w.r.t. Π.

(ii) G has no redundant edges, i.e., if removing an edge $((p_i,\Sigma_i),(p_j,\Sigma_j))$ from G then Σ_j is not a witness of p_j under X w.r.t. M, where $X = \{p \mid (p,\Sigma) \in D_G((p_j,\Sigma_j))$.

Proof. Let $[(p_i,\Sigma_i) \mid 1 \le i \le n]$ be the output of MinBetaStarWitness(Π,M). Note that Σ_i is not a witness of p_j $(j \neq i)$ under X_i *w.r.t.* M. This still holds for the returned G of MinBetaStarWitness(Π,M). For every $k, 1 \le k \le n$, let $\Delta_k = \{p_i \mid 1 \le i \le k\}$ and $D_G^k = \{p \mid (p,\Sigma_p) \in D_G((p_k,\Sigma_k))\}$, and let $\Delta_0 = D_G^0 = \emptyset$.

(i) First, it is clear that Σ_i $(1 \le i \le n)$ is a witness of p_i under Δ_{k-1} *w.r.t.* M. For every $i, 2 \le i \le n$, let $\Delta = \Sigma_i^-$, where $\Sigma_i^- = \bigcup\{\alpha^- \mid \alpha \in \Sigma_i\}$. For every $q, q' \in \Sigma_i^- \cap \Delta_{i-1}$, we have that after removal of q from Δ in the Algorithm 7 (at line 6), G has a path from q to some atom in $\Sigma_i \cap \Delta_{i-1}$, *i.e.*, $q \in D_G^{i-1}$. It follows that, for every $i, 1 \le i \le n$,

$$\Sigma_i^- \cap \Delta_{i-1} = \Sigma_i^- \cap D_G^i. \tag{16}$$

Thus, Σ_i is a witness of p_i under Δ_{i-1} *w.r.t.* M if and only if Σ_i is a witness of p_i under D_G^i *w.r.t.* M.

Second, suppose $\Sigma_i = \Sigma_j$ for some $1 \le i < j \le n$, *i.e.*, Σ_i is a minimal witness of p_j under D_G^j *w.r.t.* M and let $\Sigma = \mathrm{MR}(\Sigma_i,M)$. By Equation (16), Σ_i is a minimal witness of p_j under Δ_{j-1} *w.r.t.* M. By (i) of Lemma 4.2, p_i occurs positively in Σ since Σ is minimal such that $\Sigma \cup \Delta_{i-1} \models p_i$. Let $\Sigma' = \Sigma - \{\alpha \in \Sigma \mid p_i \in \alpha^+\}$. It is clear that $\Sigma' \subset \Sigma$ and $\Sigma' \cup \{p_i\} \equiv \Sigma$. Note that $\Delta_{i-1} \subset \Delta_{j-1}$ by $i < j$. It follows $\Sigma' \cup \Delta_{j-1} \models p_j$; this contradicts that Σ_i is a minimal witness of p_j under D_G^j.

Thus, G is a $\beta^\star$-witness of M *w.r.t.* Π.

(ii) Let G' be obtained from G by removing the edge $((p_i,\Sigma_i),(p_j,\Sigma_j))$. According to the construction of G, $p_i \in \Sigma_j^-$, *i.e.*, p_i occurs in the body of some rule from Σ_j. Note that Σ_j is a minimal witness of p_j under D_G^j *w.r.t.* M. If $\mathrm{MR}(\Sigma_j,M) \cup D_{G'}^j \models p_j$ then $\mathrm{MR}(\Sigma_j,M) \cup (D_G^j - \{p_i\}) \models p_j$, which implies $\mathrm{MR}(\Sigma_j,M) \cup (D_G^j - \{p_i\}) \models p_i$. This is impossible since $\mathrm{MR}(\Sigma_j,M) \cup \Delta_{j-1} \not\models p'_j$ for any $p'_j \neq p_j$ according to Equation (16) and Algorithm 5. □

Proposition 4.8 (Relationships among witnesses)**.** *Let $M \neq \emptyset$ be an answer set of a logic program Π. Then*

(i) $\alpha(\Pi,M) \cap \beta^\star(\Pi,M) = \beta(\Pi,M)$*;*

(ii) $\min\text{-}\beta(\Pi,M) = \min\text{-}\beta^\star(\Pi,M) \cap \min\text{-}\alpha(\Pi,M) = \beta^\star(\Pi,M) \cap \min\text{-}\alpha(\Pi,M) = \min\text{-}\beta^\star(\Pi,M) \cap \alpha(\Pi,M)$*;*

(iii) comp-$\beta(\Pi,M) =$ comp-$\beta^\star(\Pi,M) \cap$ comp-$\alpha(\Pi,M) = \beta^\star(\Pi,M) \cap$ comp-$\alpha(\Pi,M) =$ comp-$\beta^\star(\Pi,M) \cap \alpha(\Pi,M)$.

Proof. Please note that we identify a singleton set $\{v\}$ with its element v, and identify a linear order $W = [w_1,\dots,w_n]$ with the graph $G = (\{w_1,\dots,w_n\},E)$ where $(w_i,w_j) \in E$ if and only if $i < j$.

(i) $W = [(w_1,\Pi_1),\dots,(w_n,\Pi_n)]$ is a $\beta^\star$- and α-witness of M *w.r.t.* Π
$\Rightarrow$ for each (w_i,Π_i) occurring in W, w_i is a singleton and the order among (w_i,Π_i)s is a linear order
$\Rightarrow W$ is β-witness of M *w.r.t.* Π.

($\Leftarrow$) It follows from the facts $\beta(\Pi,M) \subseteq \beta^\star(\Pi,M)$ and $\beta(\Pi,M) \subseteq \alpha(\Pi,M)$.

(ii) $W = [v_1 = (p_1,\Pi_1),\dots,v_n = (p_n,\Pi_n)]$ is a minimal β-witness of M *w.r.t.* Π
iff $G = (\{v_1,\dots,v_n\},\{(v_i,v_{i+1}) \mid 1 \le i \le n-1\})$ belongs to min-$\beta^\star(\Pi,M)$ and, $W' = [(\{p_1\},\Pi_1),\dots,(\{p_n\},\Pi_n)]$ belongs to min-$\alpha(\Pi,M)$
iff $G \in \beta^\star(\Pi,M)$ and $W' \in$ min-$\alpha(\Pi,M)$
iff $G \in$ min-$\beta^\star(\Pi,M)$ and $W' \in \alpha(\Pi,M)$.

(iii) $W = [v_1 = (p_1,\Pi_1),\dots,v_n = (p_n,\Pi_n)]$ is a compact β-witness of M *w.r.t.* Π
iff $G = (\{v_1,\dots,v_n\},\{(v_i,v_{i+1}) \mid 1 \le i \le n-1\})$ belongs to comp-$\beta^\star(\Pi,M)$ and, $W' = [(\{p_1\},\Pi_1),\dots,(\{p_n\},\Pi_n)]$ belongs to comp-$\alpha(\Pi,M)$
iff $G \in \beta^\star(\Pi,M)$ and $W' \in$ comp-$\alpha(\Pi,M)$
iff $G \in$ comp-$\beta^\star(\Pi,M)$ and $W' \in \alpha(\Pi,M)$.

□

Proposition 4.9. *Let Π be a logic program, M an answer set of Π and the DAG $G = (V,E)$ be an $\alpha^\star_{fs}$-witness of M w.r.t. Π. Then there is a compact α-witness $W = [w_1,\dots,w_n]$ of M w.r.t. Π such that w_i $(1 \le i \le n)$ occurs in V.*

Proof. Given an $\alpha^\star_{fs}$-witness of $M \neq \emptyset$ *w.r.t.* Π, we construct the sequence $W = [w_1,\dots,w_n]$ with $w_i = (S_i,\Pi_i)$ $(1 \le i \le n)$ from G as follows:

(1) $W = []$;

(2) let u be a node of G whose indegree is 0, and append u to W;

(3) remove u from V and its associate edges from E;

(4) if $V \neq \emptyset$ goto (2).

We show that W is a compact α-witness of M *w.r.t.* Π. Let $X_i = \bigcup_{1\le k\le i-1} S_k$ and $Y_i = \bigcup_{(S',\Pi')\in D_G(w_i)} S'$ for $1 \le i \le n$. It is evident that Π_i is not a witness of S_j $(j \neq i)$ under X_i.

(a) It is evident that $\Pi_i \subseteq \Pi$ $(1 \le i \le n)$ and $\Pi_i \cap \Pi_j = \emptyset$ for any $w_i = (S_i,\Pi_i)$ and $w_j = (S_j,\Pi_j)$ in W with $i \neq j$.

(b) Note that, for any $v,v' \in V$, if there is a path from v to v' in G then $v = w_i$ and $v' = w_j$ where $i < j$. It follows that $Y_i \subseteq X_i$ for every $i = 1,\dots,n$. Thus, Π_i is a witness of S_i under X_i.

(c) Suppose that Π_i is not a minimal witness of S_i under X_i for some $i \in \{1,\dots,n\}$. Thus for some $r \in \Pi_i$, $\Pi_i - \{r\}$ is a witness of S_i under X_i *w.r.t.* M, hence $\mathrm{MR}(\Pi_i - \{r\},M) \cup X_i \models S_i$.

Since $\Pi_i \in MW(S_i,\Pi,Y_i,M)$, there exists a model M of $\mathrm{MR}(\Pi_i - \{r\},M) \cup Y_i$ such that $M \not\models S_i$. From (b), we have $Y_i \subseteq X_i$. Since G is an $\alpha^\star_{fs}$-witness of M *w.r.t.* Π, no rule $r \in \Pi_i$ contains some atom $p \in X_i \setminus Y_i$ in the body. Indeed, if this were the case and $p \in S_j$, then $(S',\Pi') \in D_G(w_i)$ with $S' = S_j$ would hold and thus $S_j \subseteq Y_i$, which contradicts $p \in X_i \setminus Y_i$. Consequently, if we set in M all atoms in $X_i \setminus Y_i$ to true, the resulting interpretation M' satisfies $\mathrm{MR}(\Pi_i - \{r\},M) \cup X_i$ but $M' \not\models S_i$. It follows that Π_i is not a witness of M *w.r.t.* Π, which is a contradiction. □

Proofs in Section 5

Proposition 5.1. *Deciding given programs Π, Π', sets S, B and an interpretation M, whether Π' is a witness of B under S w.r.t. M is* co-NP*-complete in general. The* co-NP*-hardness holds even if Π is positive, $S = \emptyset$, $B = M$ and M is an answer set of Π.*

Proof. The problem is in co-NP since for any $\Pi' \subseteq \Pi$, $\mathrm{MR}(\Pi, M)$ is computable in polynomial time (in fact, with logarithmic workspace); thus a guess for an interpretation I that satisfies $\mathrm{MR}(\Pi', M) \cup S$ but violates B can be verified in polynomial time, from which co-NP-membership follows.

The co-NP-hardness is by a simple reduction from SAT. Given a clause theory Σ, let

$$\Pi = \{c \vee x \mid c \in \Sigma\} \cup \{x,\ p,\ \neg x \vee p \mid p \in \mathscr{A}\}$$

where x is a fresh atom. Note that $M = \mathscr{A} \cup \{x\}$ is an answer set of Π; furthermore, $\Pi' = \{c \vee x \mid c \in \Sigma\} \cup \{\neg x \vee p \mid p \in \mathscr{A}\}$ is a witness of $B = M$ under $S = \emptyset$ *w.r.t.M* iff Σ unsatisfiable. □

Theorem 1. *Let Π be a logic program Π. The following problems are D_1^p-complete:*

(i) deciding whether $\Pi' \in \mathrm{MW}(B, \Pi, S, M)$, with D_1^p-hardness if M is an answer set of Π;

(ii) deciding whether $W = [(S_1, \Pi_1), \ldots, (S_n, \Pi_n)]$ resp. $G = (\{(S_i, \Pi_i) \mid 1 \leq i \leq n\}, E)$ is a (minimal, compact) α- resp. $\alpha^\star$-witness or $\alpha^\star_{fs}$-witness of an answer set M of Π;

(iii) deciding whether $W = [(p_1, \Pi_1), \ldots, (p_n, \Pi_n)]$ resp. $G = (\{(p_i, \Pi_i) \mid 1 \leq i \leq n\}, E)$ is a (minimal, compact) β- resp. $\beta^\star$-witness of an answer set M of Π.

Furthermore, the D_1^p-hardness holds in all cases if Π is a positive (negation-free) program.

Proof. The D_1^p membership holds in all cases: modulo the computation of auxiliary objects like X_i, $S_{\triangleright i}$ etc. and checking syntactic conditions on them and the witness at hand (like $\Pi_i \cap \Pi_j = \emptyset$), the recognition test reduces to polynomially many witness and non-witness tests, which can be transformed in polynomial time into a single SAT and UNSAT instance, respectively. Furthermore, the auxiliary computations and syntactic checks can be done in polynomial time. As D_1^p is closed under polynomial time transformations, it follows that all problems are in D_1^p. The D_1^p-hardness parts are shown as follows.
(i) Assume that in the co-NP-hardness proof of Proposition 5.1, the given clause theory Σ is a CSAT instance; then Π' is a minimal witness of M *w.r.t.* Π iff Σ is unsatisfiable and $\Sigma \setminus \{c\}$ is satisfiable for each clause $c \in \Sigma$, i.e., Σ is a yes-instance of CSAT.
(ii) For all (minimal, compact) α- resp. $\alpha^\star$-witnesses, the D_1^p-hardness follows from the reduction in (i), when we set $W = [(S_1, \Pi_1)]$ resp. $G = \{(S_1, \Pi_1)\}$ with $S_1 = M$ and $\Pi_1 = \Pi'$ and in case of $\alpha^\star_{fs}$-witnesses add to Π all clauses $\neg x \vee \neg p' \vee p$, where $p, p' \in \mathscr{A}$; this ensures that the dependency graph of Π is strongly connected.

For $\alpha^\star$-witnesses, we reduce deciding whether given clause theories Σ_1 and Σ_2 are unsatisfiable and satisfiable, respectively, to α-witness testing as follows. Suppose that Σ_1 and Σ_2 are over disjoints sets $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively, and let y_1 and y_2 be fresh atoms. Define

$$\Pi = \{c \vee y_1 \mid c \in \Sigma_1\} \cup \{c \vee y_2 \mid c \in \Sigma_2\} \cup \mathscr{A}_1 \cup \mathscr{A}_2 \cup \{y_1,\ y_2\}. \tag{17}$$

Then, $M = \mathscr{A}_1 \cup \mathscr{A}_2 \cup \{y_1,\ y_2\}$ is clearly a minimal model and thus an answer set of Π. We set $G = (\{(S_1, \Pi_1), (S_2, \Pi_2), (S_3, \Pi_3)\}, E)$ where

- $S_1 = \{y_1\}$ and $\Pi_1 = \{c \vee y_1 \mid c \in \Sigma_1\} \cup \{c \vee y_2 \mid c \in \Sigma_2\}$,
- $S_2 = \{y_2\}$ and $\Pi_2 = \{y_2\}$, and
- $S_3 = \Pi_3 = \mathscr{A}_1 \cup \mathscr{A}_2$,

and E has the edge $(S_1, \Pi_1) \to (S_2, \Pi_2)$; thus $X_1 = X_3 = \emptyset$, and $X_2 = \{y_1\}$.

We verify that G is an $\alpha^\star$-witness of M *w.r.t.* Π iff Σ_1 is unsatisfiable and Σ_2 is satisfiable. As for the first condition of $\alpha^\star$-witness, clearly, Π_1 is a witness of S_1 under X_1 *w.r.t.M* iff Σ_1 is unsatisfiable; for $i = 2,3$, Π_i is always a witness of S_i under X_i. Regarding the second condition of $\alpha^\star$-witness, Π_1 is not a witness of $S_2 = \{y_2\}$ under $X_1 = \emptyset$, iff Σ_2 is satisfiable; in all other cases, Π_i is trivially not a witness of S_j $(j \neq i)$, under X_i.

For α-witness, we set $W = [(S_1, \Pi_1), (S_2, \Pi_2), (S_3, \Pi_3)]$ and one can similarly verify that W is an α-witness of M *w.r.t.* Π iff Σ_1 is unsatisfialbe and Σ_2 is satisfiable. This proves the claim.

(*iii*) For β-witnesses, the proof is similar to the one of α-witness. We take the same program Π and model M, and define $W = [(p_1, \Pi_1), \dots, (p_n, \Pi_n)]$ where $n = m+2$ and $\mathscr{A}_1 = \{q_1, \dots, q_m\}$: $p_1 = y_1$ and Π_1 is as above, $p_2 = y_2$ and Π_2 is as above, and for $i = 1, \dots, m$, we have $p_{i+2} = q_i$ and $\Pi_{i+2} = \{q_i\}$; that is, S_3 is split into atomic components. This can be readily extended to $\beta^\star$-witnesses by defining for the graph G the single edge $(p_1, \Pi_1) \to (p_2, \Pi_2)$.

For all other notions of β- and $\beta^\star$- witnesses, we adapt the proof for minimal α-witness in (*ii*), by defining $p_1 = x$ and $p_{i+1} = q_i$, where $\mathscr{A} = \{q_1, \dots, q_m\}$ and $n = m+1$, and splitting Π' into $\Pi_1 = \{c \vee x \mid c \in \Sigma\}$, and $\Pi_{i+1} = \{\neg x \vee q_i\}$, $1 \leq i \leq m$. Furthermore, in case of $\beta^\star$-witnesses, we define for the graph G edges $(p_1, \Pi_1) \to (p_{1+i}, \Pi_{i+1})$, for $i = 1, \dots n$. Then, W resp. G is a minimal/compact β resp. $\beta^\star$-witness for M *w.r.t.* Π iff the original clause theory Σ is a yes-instance of CSAT. $\square$

Proposition 5.2. *Suppose that* Π *is headcycle-free and* M *an answer set of* Π. *Then* $\{r \in \mathrm{MR}(\Pi, M) \mid |r^+| = 1\} \models M$.

Proof. For any $p \in M$, there is a proof $r_1, \dots, r_k$ *w.r.t.* M and Π by Theorem 2.3 of Ben-Eliyahu and Dechter [1994]
$\Rightarrow \mathrm{MR}(\{r_i \mid 1 \leq i \leq k\}, M) \models p$
$\Rightarrow \{r \in \mathrm{MR}(\Pi, M) \mid |r^+| = 1\} \models p$ since $\mathrm{MR}(\{r_i \mid 1 \leq i \leq k\}, M) \subseteq \{r \in \mathrm{MR}(\Pi, M) \mid |r^+| = 1\}$.
Thus $\{r \in \mathrm{MR}(\Pi, M) \mid |r^+| = 1\} \models M$. $\square$

Proposition 5.3. *Let* Σ *be an unsatisfiable clause theory, and let* $\Pi = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ *and* $M = \mathscr{A} \cup \mathscr{A}' \cup \{x, x'\}$ *be as (11). Then,*

(i) for every minimal β*-witness* $[(p_1, \Pi_1), \dots, (p_m, \Pi_m)]$ *of* M *w.r.t.* Π, *if* $p_1 = x$ *the set* $\Gamma = \{r \in \Sigma \mid (r^+ \cup \{x\}) \leftarrow r^- \in \Pi_1\}$ *and otherwise the set* $\Gamma = \{r \in \Sigma \mid (r^+ \cup \{x'\}) \leftarrow r^- \cup \{x\} \in \Pi_1\}$ *is an MUS of* Σ*;*

(ii) from every MUS of Σ *a minimal* β*-witness w.r.t.* Π *can be constructed.*

Proof. (i) We analyse the cases for p_1. If $p_1 = x$, then by minimality $\Pi_1 \subseteq \Sigma_1$ must hold (indeed, from $\Pi_1 \cup \{\neg x\}$, every clause in $\Sigma_2 \cup \Sigma_3$ could be removed without establishing satisfiability). Hence the set Γ constructed for this case must be minimally unsatisfiable.

If $p_1 \neq x$, consider first that $p_1 \in \mathscr{A}' \cup \{x'\}$. As $\Pi_1 \not\models x$ must hold, Π_1 has a model M in which x is false. But then $M' = M \setminus (\mathscr{A}' \cup \{x'\})$ is also a model of Π_1; this contradicts that $\Pi_1 \models p_1$, however. Thus it follows

that $p_1 \in \mathscr{A}$. We conclude that we have $\Pi_1 \cap \Sigma_2 \models \neg x \vee x'$: if not, $\Pi_1 \cap \Sigma_2 \cup \{x, \neg x'\}$ has some model M, which is a model of Π_1; then also $M \setminus \{p_1\}$ is a model of Π_1, which contradicts $\Pi_1 \models p_1$. As each clause in Σ_2 is of the form $c \vee \neg x \vee x'$ where x, x' do not occur in c, it follows that the set Γ described for Π_1 and p_1 must fulfill $\Gamma \models \bot$, i.e., Γ is unsatisfiable; by the minimality of Π_1, moreover Γ is minimal. This shows the claim.
(ii) For each MUS Γ of Σ, we can build a minimal β-witness $W = [(p_1,\Pi_1),\dots,(p_m,\Pi_m)]$ of $\{p_1,\dots,p_m\}$ *w.r.t.* Π by setting

- $p_1 = x$, $\Pi_1 = \{c \vee x \mid c \in \Gamma\}$,
- $p_2 = x'$, $\Pi_2 = \{c' \vee \neg x \vee x' \mid c \in \Gamma\}$, and
- $p_{i+2} = q_i$, $\Pi_{i+2} = \{\neg x \vee \neg x' \vee q_i\}$ for each i $(1 \leq i \leq m-2)$, where $\mathscr{A} \cup \mathscr{A}' = \{q_1,\dots,q_{m-2}\}$.

Clearly $\Pi_i \cup \{p_1,\dots,p_{i-1}\} \models p_i$ and $\Pi_i \cup \{p_1,\dots,p_{i-1}\} \not\models p_j$ $(1 \leq i \neq j \leq m)$, and no rules can be omitted from any Π_i without losing this property. ☐

Proposition 5.4. *Given an answer set M of a logic program Π, computing some minimal β-witness (resp. minimal $\beta^\star$ witness) for M w.r.t. Π is $\mathrm{FP}_{\|}^{\mathrm{NP}}$-hard.*

Proof. In Janota and Marques-Silva [2016], the $\mathrm{FP}_{\|}^{\mathrm{N}P}$-hardness of computing some MUS (assuming that the input formula F is unsatisfiable) is attributed to Chen and Toda [1995]. This result is not stated explicitly there, but can be shown using Lemma 4.7 in that paper formulating MUS computation as a maximization problem as considered there for suitable UNSAT instances. The result follows then from Proposition 5.3. For self-containedness, we instead reduce the canonical $\mathrm{FP}_{\|}^{\mathrm{N}P}$-complete problem of computing the answers to given SAT instances $F_1,\dots,F_n$ to minimal β-witness computation.

By a reduction in Papadimitriou and Wolfe [1988], any SAT instance F_i can be reduced in polynomial time into a CNF G_i *s.t.* (i) G_i is unsatisfiable and (ii) F_i is satisfiable iff every subformula G_i' that results from G_i by omitting some clause is satisfiable. Assuming that all G_i are over disjoint atoms, we construct for each G_i the program Π^i and the set M^i of atoms as in Proposition 5.3 such that all Π^i are on disjoint atoms. Then $M = \bigcup_{i=1}^n M_i$ is the single answer set of $\Pi = \bigcup_{i=1}^n \Pi^i$, and by the disjointness of the Π^i, the minimal β- resp. $\beta^\star$-witnesses W of M *w.r.t.* Π are composed of minimal β- resp. $\beta^\star$-witnesses W_i of the answer sets M_i *w.r.t.* Π_i, for $i = 1,\dots,n$, where W_i can be obtained by projecting W to the components over the alphabet of Π^i. Since every such W_i corresponds by item (ii) of Proposition 5.3 to an MUS Γ_i of G_i, and since $\Gamma_i = G_i$ must hold iff F_i is unsatisfiable, we can compute the answers of the SAT instances $F_1,\dots,F_n$ easily from W. As the logic program Π and M are easily constructed from $G_1,\dots,G_n$, this proves $\mathrm{FP}_{\|}^{\mathrm{N}P}$-hardness. ☐

Theorem 2. *Let M be an answer set of a logic program Π, and let $p \in M$. Deciding whether there exists some minimal β-witness $W = [(p_1,\Pi_1),\dots,(p_n,\Pi_n)]$ for $M = \{p_1,\dots,p_n\}$ w.r.t. Π such that $p = p_1$ is Σ_2^p-complete.*

Proof. The problem is in Σ_2^p, as we can guess a minimal β-witness W of M such that $p = p_1$ and check with an NP oracle in polynomial time whether W is indeed a minimal β-witness of M (cf. Theorem 5).

The Σ_2^p-hardness is shown by a reduction from deciding whether for a given unsatisfiable clause theory Σ, a particular clause $c \in \Sigma$ belongs to some MUS Σ' of Σ. This problem is known to be Σ_2^p-complete, even if the clause c is a single literal $\neg p$, as follows from the results on minimal equivalent subformulas in Liberatore [2005].

For the reduction, we consider the clause theory Π constructed from Σ and the answer set M as in the proof of Proposition 5.3. We claim that some minimal β-witness $W = [(p_1,\Pi_1),\ldots,(p_n,\Pi_n)]$ of $M = \{p_1,\ldots,p_n\}$ *w.r.t.* Π exists such that $p = p_1$ iff some MUS $\Sigma' \subseteq \Sigma$ exists such that $\neg p \in \Sigma'$.

($\Leftarrow$) Assume that Σ' is an MUS such that $\neg p \in \Sigma'$. Then $\Pi' = \Sigma' \setminus \{\neg p\}$ is a minimal set such that $\Pi' \models p$. Consequently, if we define

$$\begin{aligned}
&\Pi_1 = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \ \textit{with}\ \Sigma_1 = \{c \cup \{x\} \mid c \in \Pi'\}, \Sigma_2 = \{c \cup \{\neg x, x'\} \mid c \in \Sigma'\}, \Sigma_3 = \{\neg x \vee \neg x' \vee p\},\\
&\Pi_2 = \{\neg p \vee x\},\\
&\Pi_3 = \{c \cup \{\neg x, x'\} \mid c \in \Sigma'\},\\
&\Pi_i = \{\neg x \vee \neg x' \vee p_i\} \text{ for } i = 4,\ldots,n,
\end{aligned}$$

then $W = [(p_1,\Pi_1),\ldots,(p_n,\Pi_n)]$ for $M = \{p_1,\ldots,p_n\} = \mathscr{A} \cup \mathscr{A}' \cup \{x,x'\}$ where $p_1 = p$, $p_2 = x$, $p_3 = x'$, is a minimal β-witness of M. Indeed, by $\Pi' \models p$, $\Pi_1 \models p \vee x$ holds and $\Pi_1 \models p \vee \neg x \vee x'$, thus $\Pi_1 \models p \vee x'$; since Π_1 contains $p \vee \neg x \vee \neg x'$, it follows that $\Pi_1 \models p$. Furthermore, $\Pi_1 \not\models x$ and $\Pi_1 \not\models x$: Π' is satisfiable, and so Π_1 has a model in which p is true and x, x' are false. Finally, $\Pi_1 \not\models q$ for any other atom q since Π_1 has a model in which p, x, x' are true and q is false. As for Π_2, note that $\neg p \in \Sigma$, and thus $\neg p \vee x \in \Sigma_1$; hence $\Pi_2 \cup \{p\} \models x$. Next, by construction $\Pi_3 \models \neg x \vee x'$; thus $\Pi_3 \cup \{p,x\} \models x'$. Finally, for each Π_i, $i = 4,\ldots,n$, we have that $\Pi_i \cup \{x,x'\} \models p_i$ and Π_i is not a witness of p_j $(j \neq i)$ under $\{p_1,\ldots,p_{i-1}\}$. The minimality of Π_1 follows from the minimality of Σ'.

($\Rightarrow$) Suppose that $W = [(p_1,\Pi_1),\ldots,(p_n,\Pi_n)]$ is a minimal β-witness of M such that $p = p_1$. Then we obtain that $\Pi_1 \cap \Sigma_1 \models p \vee x$: If not, then $\Pi_1 \cap \Sigma_1$ has a model in which both p and x are false; but then Π_1 has a model in which p, x are false and x' is true. Thus the set $\Gamma = \{r \in \Sigma \mid r^+ \cup \{x\} \leftarrow r^- \in \Pi_1 \cap \Sigma_1\}$ is such that $\Gamma \cup \{\neg p\}$ is unsatisfiable. Furthermore, by the minimality of W, no $\Pi_1' \subset \Pi_1 \cap \Sigma_1$ exists such that $\Pi_1' \models p \vee x$. Otherwise, $\Pi_1' \cup (\Pi_1 \cap (\Sigma_2 \cup \Sigma_3))$ would logically imply p but no other atoms, and thus contradict that W is a minimal β-witness. Consequently $\Gamma \cup \{\neg p\}$ is an MUS that contains $\neg p$. This proves the result. □

Theorem 3. *Deciding whether an answer set M of a disjunctive logic program Π has a compact β-witness (resp. compact $\beta^\star$-witness) is Σ_2^p-complete, and the Σ_2^p-hardness holds even if* $\mathrm{MR}(\Pi,M) = \Pi$.

Proof. (Membership): a guess for a compact β-witness $W = [(p_i,\Pi_i)]_{i=1}^n$ (resp., compact $\beta^\star$-witness $W = (\{(p_i,\Pi_i) \mid 1 \leq i \leq n\}, E))$ of $M = \{p_1,\ldots,p_n\}$ *w.r.t.* Π has polynomial size and can be checked easily using an NP-oracle in polynomial time by verifying the conditions for Π^M.

(Hardness): We first consider compact β-witnesses, and then extend the argument to compact $\beta^\star$-witnesses. Let Φ be a QBF of the form

$$\Phi = \exists X \forall Y E(X,Y)$$

where $X = \{x_1,\ldots,x_n\}$, $Y = \{y_1,\ldots,y_m\}$ and $E(X,Y) = \bigvee_{j=1}^k D_j$ is a 3DNF over the variables $X \cup Y$ with $D_j = l_{j,1} \wedge l_{j,2} \wedge l_{j,3}$.

We construct a clause theory $\Sigma = \bigcup_{i=1}^3 \Sigma_i$ that consists of the following subsets:

- Σ_1 consists of the following rules,

$$p_1 \vee p_2 \vee p_3, \quad p_1 \leftarrow p_2, \quad p_2 \leftarrow p_3, \quad p_3 \leftarrow p_1.$$

  Note that Σ_1 has the unique minimal model $M_1 = \{p_1,p_2,p_3\}$ which has no compact β-witness *w.r.t.* Σ_1.

- $\Sigma_2 = \{\tau(D_j) \mid 1 \le j \le k\}$ where
$$\tau(D_j) = \{p_1\} \cup D_j - \leftarrow D_j +,$$
and D^+ (resp. D^-) is the set of atoms that occur in D positively (resp. negatively).

- Σ_3 consists of the following rules, for $1 \le i \le n$ and $1 \le j \le m$:
$$p_1 \leftarrow z_i, \quad z_i \leftarrow x_i, \quad x_i \leftarrow r_i, \tag{18}$$
$$p_1 \vee z_i', \quad x_i \leftarrow z_i', \quad r_i \leftarrow x_i, \tag{19}$$
$$r_i \vee c_i, \quad c_i \leftarrow z_i, \tag{20}$$
$$z_i' \vee c_i', \quad c_i' \leftarrow r_i, \tag{21}$$
$$c_i \leftarrow d_i', \quad d_i' \leftarrow c_i', \quad c_i' \leftarrow d_i, \quad d_i \leftarrow c_i, \tag{22}$$
$$z_i \leftarrow M, \quad z_i' \leftarrow M, \quad r_i \leftarrow M, \quad x_i \leftarrow M, \tag{23}$$
$$y_j \leftarrow M \tag{24}$$

where $z_i, z_i', c_i, c_i', d_i, d_i', r_i$ $(1 \le i \le n)$ are pairwise different fresh atoms and $M = M_1 \cup M_2$ with $M_2 = \{c_i, c_i', d_i, d_i' \mid 1 \le i \le n\}$.

Let $N = M \cup N_1 \cup N_2$ where $N_1 = \{x_i, z_i, z_i', r_i \mid 1 \le i \le n\}$ and $N_2 = \{y_j \mid 1 \le j \le m\}$.

Note that the first two rules from (18) and (19) respectively can derive p_1. While the last two rules from (18) and the rules from (20) can derive c_i, the last two rules from (19) and the rules from (21) can derive c_i'. Thus, $\Sigma_1 \cup \Sigma_3$ can derive every atom in N. It is evident that N is a minimal model of Σ.

We will show N has a compact β-witness *w.r.t.* Σ if and only if Φ evaluates to true.

($\Leftarrow$) There is an assignment σ for X *s.t.* $\forall Y E(\sigma(X), Y)$ evaluates to true. We construct the following β-witness of N *w.r.t.* Σ:
$$W = [(p_1, \Pi), (q_{i,s}, \delta_{i,s})\ (1 \le s \le 4, 1 \le i \le n)] + \Delta \tag{25}$$
where

- $\Pi = \Pi_1 \cup \Pi_2 \cup \Pi_3$ is minimal *s.t.* $\Pi \models p_1$ with
$$\Pi_1 \subseteq \Sigma_2,$$
$$\Pi_2 = \{p_1 \leftarrow z_i, \quad z_i \leftarrow x_i \mid \sigma(x_i) = 1, 1 \le i \le n\},$$
$$\Pi_3 = \{p_1 \vee z_i', \quad x_i \leftarrow z_i' \mid \sigma(x_i) = 0, 1 \le i \le n\}.$$

- If $\sigma(x_i) = 0$ then
  - $q_{i,1} = c_i$ and $\delta_{i,1}$ consists of the last two rules from (18) and the rules from (20);
  - $q_{i,2} = d_i$ and $\delta_{i,2} = \{d_i \leftarrow c_i\}$;
  - $q_{i,3} = c_i'$ and $\delta_{i,3} = \{c_i' \leftarrow d_i\}$;
  - $q_{i,4} = d_i'$ and $\delta_{i,4} = \{d_i' \leftarrow c_i'\}$;

  otherwise
  - $q_{i,1} = c_i'$ and $\delta_{i,1}$ consists of the last two rules from (19) and the rules from (21);

  - $q_{i,2} = d_i'$ and $\delta_{i,2} = \{d_i' \leftarrow c_i'\}$;
  - $q_{i,3} = c_i$ and $\delta_{i,3} = \{c_i \leftarrow d_i'\}$;
  - $q_{i,4} = d_i$ and $\delta_{i,4} = \{d_i \leftarrow c_i\}$.

- Δ is a compact β-witness of $N_1 \cup N_2$ under $\Gamma \cup M$ and M *w.r.t.* N, where Γ consists of the rules from (23) and (24).

It is tedious but not difficult to check that this β-witness W of N *w.r.t.* Σ is compact.

($\Rightarrow$) In the case $E(X,Y)$ is a tautology, Φ is trivially valid. Suppose that $E(X,Y)$ is not a tautology in what follows. Thus, Σ_2 cannot derive p_1. Let W be a compact β-witness of N *w.r.t.* Σ, which we assume to be of the form

$$W = \Delta_1 + \Delta_2 + \Delta_3, \tag{26}$$

where the subparts Δ_i, $(1 \leq i \leq 3)$ will be clarified in the sequel. Firstly, note that, for any atom q,

$$\Sigma_3 \models q \text{ iff } q \in \{p_1\} \cup M_2.$$

If the first two rules from (18) and (19) respectively are used to derive p_1 in the above compact β-witness (26), then it is impossible to construct a compact β-witness of $\{c_i, c_i', d_i, d_i'\}$ *w.r.t.* the set of rules in (20)-(22). For instance, if the rules $r_i \vee c_i, c_i' \leftarrow r_i, c_i \leftarrow d_i', d_i' \leftarrow c_i'$ are used to derive c_i, then one cannot derive d_i' since the rule $d_i' \leftarrow c_i'$ is not allowed anymore for the compactness.

It implies that the compact β-witness (26) contains either

- $(c_i, \delta_1 = \{z_i \leftarrow x_i, \quad x_i \leftarrow r_i, \quad r_i \vee c_i, \quad c_i \leftarrow z_i\})$ or
- $(c_i', \delta_2 = \{x_i \leftarrow z_i', \quad r_i \leftarrow x_i, \quad z_i' \vee c_i', \quad c_i' \leftarrow r_i\})$,

but not both. Otherwise, without the rules in $\delta_1 \cup \delta_2$, one can not have a compact β-witness for $N - \{c_i, c_i'\}$. Without loss of generality, we assume that Δ_1 of (26) is a compact β-witness of M_2 under Σ and $\emptyset$ *w.r.t.* N.

Secondly, to derive any atom in $N_1 \cup N_2$ from Σ, it must depend on all atoms in M. Thus, we may assume that Δ_3 of (26) is a compact β-witness of $N_1 \cup N_2$ under Σ and M *w.r.t.* N.

Thirdly, Δ_2 of (26) must be a compact β-witness of M_1 under Σ and M_2 *w.r.t.* N, which does not contain any rules occurring in $\Delta_1 \cup \Delta_3$. Thus, to construct a compact β-witness for M_1 under Σ and M_2 *w.r.t.* N, one has to derive first p_1 by Σ. In this case, the only possible rules to derive p_1 are from Σ_2 and from, for $1 \leq i \leq n$,

(a) $\{p_1 \leftarrow z_i,\ z_i \leftarrow x_i\}$ or

(b) $\{p_1 \vee z_i',\ x_i \leftarrow z_i'\}$, but not both.

Thus, Σ_2 together with, for i $(1 \leq i \leq n)$, $\{p_1 \leftarrow x_i\}$ or $\{p_1 \vee x_i\}$ can derive p_1. This corresponds to a selection of x_i or $\neg x_i$ for i $(1 \leq i \leq n)$. It implies that $\neg E(X,Y)$ is unsatisfiable under this selection of the variables in X. This selection exactly corresponds to an assignment σ for X, *i.e.*, $\sigma(x_i) = 0$ if $p_1 \leftarrow x_i$ is selected, and $\sigma(x_i) = 1$ if $p_1 \vee x_i$ is selected. It follows that $E(\sigma(X),Y)$ is valid, which implies $\exists X \forall Y E(X,Y)$ evaluates to true. This establishes Σ_2^p-hardness for compact β-witnesses under the given restrictions.

For $\beta^\star$-witnesses, the same reduction works. As each compact β-witness is a compact $\beta^\star$-witness, it remains to consider the only-if direction ($\Rightarrow$). Notably, Σ_1 has *w.r.t.* M_1 also no compact $\beta^\star$-witness, and thus p_1 must be derived from rules in $\Sigma_2 \cup \Sigma_3$. With a similar line of argumentation, where $\beta^\star$ replaces β, it can be seen that from a compact $\beta^\star$-witness W, an assignment σ for X is obtainable such that $E(\sigma(X),Y)$ is valid, i.e., $\exists X \forall Y E(X,Y)$ evaluates to true. Here W is without loss of generality of similar form as in (26), where inside Δ_i components may not be totally ordered. $\square$

Theorem 4. *Let M be an answer set of a logic program Π, $S \subseteq M$ and $p,q \in \mathscr{A}$ s.t. $\mathrm{MR}(\Pi,M) \cup S \models p \wedge q$. The problem of deciding whether q is necessary for p w.r.t. M and S under Π is Π_2^p-complete, and the Π_2^p-hardness holds even if $\mathrm{MR}(\Pi,M) = \Pi$ and $S = \emptyset$.*

Proof. Membership: If q is not necessary for p *w.r.t.* M and S under Π then there exists some $\Pi' \subseteq \Pi$ *s.t.* (a) $\mathrm{MR}(\Pi',M) \cup S \models p$ and (b) $\mathrm{MR}(\Pi',M) \cup S \not\models q$. It is clear that one can guess such Π' and verify the conditions (a) and (b) in polynomial time using an NP-oracle. Hence the complement of the problem in Σ_2^P, which implies that the problem is in Π_2^p.

Hardness: Let $\Phi = \exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m \phi_1 \vee \cdots \vee \phi_k$ $(m,n \geq 1)$ be a 2QBF formula over variables $X \cup Y$, where $X = \{x_1,\ldots,x_n\}$, $Y = \{y_1,\ldots,y_m\}$ and $\phi_j = l_{j_1} \wedge l_{j_2} \wedge l_{j_3}$ is a conjunction of literals. The problem of deciding whether Φ is valid is Σ_2^P-complete.

Let $S = \emptyset$ and Π be the clause theory consisting of the following clauses built from Φ:

$$\begin{aligned}
&r_0 : \neg p' \vee p,\\
&r_{1i} : x_i \vee p', \qquad r_{2i} : \neg x_i \vee p',\ (1 \leq i \leq n),\\
&r_{3i} : \sigma(l_{i_1}) \vee \sigma(l_{i_2}) \vee \sigma(l_{i_3}) \vee p' \vee p,\ (1 \leq i \leq k),\\
&r_{x_i} : \neg p' \vee \neg p \vee x_i,\ (1 \leq i \leq n), \qquad r_{y_j} : \neg p' \vee \neg p \vee y_j,\ (1 \leq j \leq m),
\end{aligned}$$

where p and p' are two fresh atoms, $\sigma(\neg\alpha) = \alpha$ and $\sigma(\alpha) = \neg\alpha$ for each $\alpha \in X \cup Y$. It is evident that $\Pi \models p \wedge p'$ in terms of the clauses in $\{r_{1i}, r_{2i} | 1 \leq i \leq n\} \cup \{r_0\}$, and hence that $M = X \cup Y \cup \{p,p'\}$ is by the clauses r_{x_i} and r_{y_j} the single model and answer set of Π. Furthermore, $\mathrm{MR}(\Pi,M) = \Pi$.

In the following, we show that Φ evaluates to true if and only if $q = p'$ is not necessary for p *w.r.t.* M and $S = \emptyset$ under Π.

($\Rightarrow$) Let v be an assignment for X *s.t.* $\forall y_1 \cdots \forall y_m v(\phi_1 \vee \cdots \vee \phi_k) = 1$. It implies that the following entailment holds:

$$\{x \in X \mid v(x) = 1\} \cup \{\neg x \mid x \in X \ \&\ v(x) = 0\} \models \phi_1 \vee \cdots \vee \phi_k. \tag{27}$$

Let Σ consisting of the clauses in Π:

$$\begin{aligned}
&r_0, \qquad r_{3i}, && && (28)\\
&r_{1i},\ \text{if } v(x_i) = 1, && (1 \leq i \leq n) && (29)\\
&r_{2i},\ \text{if } v(x_i) = 0, && (1 \leq i \leq n). && (30)
\end{aligned}$$

It is evident that $\{x \in X \mid v(x) = 1\} \cup \{p\}$ is a model of $\Sigma \cup \{\neg p'\}$. Thus, $\Sigma \not\models p'$. Suppose that $\Sigma \not\models p$. We have that

$\Sigma \cup \{\neg p\}$ has a model v'

$\Rightarrow \Sigma \cup \{\neg p\} \models \neg p \wedge \neg p'$ by r_0

$\Rightarrow \Sigma \cup \{\neg p\} \models \{x \in X \mid v(x) = 1\} \cup \{\neg x \mid x \in X \ \&\ v(x) = 0\}$ by the clauses in the lines (29) and (30)

$\Rightarrow \Sigma \cup \{\neg p\} \models \phi_1 \vee \cdots \vee \phi_k$ by Eq (27)

$\Rightarrow \Sigma \cup \{\neg p\} \models \bigwedge_{1 \leq i \leq k}(\sigma(l_{i_1}) \vee \sigma(l_{i_2}) \vee \sigma(l_{i_3}))$ by the clauses r_{3i}s

$\Rightarrow \Sigma \cup \{\neg p\} \models \bigwedge_{1\le i\le k} \neg\phi_i$
$\Rightarrow \Sigma \cup \{\neg p\} \models \neg(\phi_1 \vee \cdots \vee \phi_k)$.
This contradicts that $\Sigma \cup \{\neg p\}$ is satisfiable. Thus, $\Sigma \models p$. As $\mathrm{MR}(\Sigma, M) = \Sigma$, it follows that p' is not necessary for p *w.r.t.* M and S under Π.

($\Leftarrow$) Suppose that p' is not necessary for p *w.r.t.* M and S under Π. As $\mathrm{MR}(\Pi', M) = \Pi'$ for each $\Pi' \subseteq \Pi$, it follows that there is a subset Σ of Π *s.t.* $\Sigma \models p$ and $\Sigma \not\models p'$. If $r_0 \notin \Sigma$ then the interpretation $\{p'\}$ satisfies every clause in Π, which implies it is a model of Σ. This contradicts that $\Sigma \models p$. Thus, $r_0 \in \Sigma$. By $\Sigma \not\models p'$, we have, for each i $(1 \le i \le n)$, that at most one of r_{1i} and r_{2i} is in Σ. Let v be an assignment *s.t.*, for each $x \in X$, $v(x) = 1$ if $x \vee p' \in \Sigma$ and $v(x) = 0$ otherwise. To prove that Φ evaluates to true, it suffices to show

$$\{x \in X \mid v(x) = 1\} \cup \{\neg x \mid x \in X \ \&\ v(x) = 0\} \models \phi_1 \vee \cdots \vee \phi_k. \tag{31}$$

Suppose that (31) does not hold. We have that there is an extension v' of v to Y *s.t.* $v'(\phi_1 \vee \cdots \vee \phi_k) = 0$
$\Rightarrow v'(\phi_i) = 0$ for each i $(1 \le i \le k)$
$\Rightarrow v'(\neg\phi_i) = 1$ for each i $(1 \le i \le k)$
$\Rightarrow v'(\sigma(l_{i_1}) \vee \sigma(l_{i_2}) \vee \sigma(l_{i_3})) = 1$ for each i $(1 \le i \le k)$
$\Rightarrow v'(r_{3i}) = 1$ for each $r_{3i} \in \Sigma$
$\Rightarrow v''(\Sigma) = 1$ where v'' is the extension of v' to $\{p, p'\}$ by setting $v''(p) = v''(p') = 0$
$\Rightarrow \Sigma \not\models p$, a contradiction. □

Appendix: reduct for general extended logic programs

Let *Lit* be the set of literals of $\mathscr{L}$. An *general extended logic program* Π is a set of *extended rules* of the form Gelfond and Lifschitz [1991]; Lifschitz *et al.* [1999]

$$l_1 \vee \cdots \vee l_k \leftarrow l_{k+1}, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n, not\ not\ l_{n+1}, \cdots, not\ not\ l_s \tag{32}$$

where each l_i $(1 \le i \le s)$ is a literal.

Let S be a set of literals. The *GL-reduct* of Π *w.r.t.* S, written Π^S, is the logic program

$$\{r^+ \leftarrow r^- \mid r \in \Pi, r^{not} \cap S = \emptyset \text{ and } r^{2not} \subseteq S\}. \tag{33}$$

The set S is an *answer set* of Π if it is minimal such that S is *closed w.r.t.* Π^S, *i.e.*

- for each rule $r \in \Pi^S$, $r^- \subseteq S$ implies $r^+ \cap S \neq \emptyset$, and
- if S contains a pair of complementary literals, then $S = Lit$.

The reduct $\mathrm{MR}(\Pi, S)$ of Π *w.r.t.* S is similarly defined as before, *i.e.*,

$$\mathrm{MR}(\Pi, S) = \{r^+ \cap S \leftarrow r^- \mid r \in \Pi, r^- \subseteq S, r^{not} \cap S = \emptyset, r^{2not} \subseteq S\}. \tag{34}$$

For instance, let $\Pi = \{\neg a \vee b,\quad b \leftarrow \neg a,\quad \neg a \leftarrow b\}$ and $S = \{\neg a, b\}$. It is not difficult to check that S is the unique answer set of Π and $\Pi = \mathrm{MR}(\Pi, S)$.

Lemma 7.1. *Let Π be a positive extended logic program, $S, S' \subseteq Lit$.*

A1. S is closed w.r.t. Π if and only if S is closed w.r.t. $\{r \in \Pi \mid r^- \subseteq S\}$.

A2. *If both S and S' are closed w.r.t.* Π *then* $S \cap S'$ *is also closed w.r.t.* Π.

Proof. (1) S It is trivial when $S = Lit$. Suppose that S is consistent. We have
S is closed *w.r.t.* Π
iff for each $r \in \Pi$, $r^- \subseteq S$ implies $r^+ \cap S \neq \emptyset$
iff for each $r \in \{r \in \Pi \mid r^- \subseteq S\}$, $r^+ \cap S \neq \emptyset$
iff for each $r \in \{r \in \Pi \mid r^- \subseteq S\}$, $r^- \subseteq S$ implies $r^+ \cap S \neq \emptyset$
iff S is closed *w.r.t.* $\{r \in \Pi \mid r^- \subseteq S\}$.

(2) It is evident when either S or S' is Lit. Suppose that both S and S' are consistent. We have
S and S' are closed *w.r.t.* Π
$\Rightarrow$ for each $r \in \Pi$ and $M \in \{S, S'\}$, $r^- \subseteq M$ implies $r^+ \cap M \neq \emptyset$
$\Rightarrow$ for each $r \in \Pi$, $r^- \subseteq S \cap S'$ implies $r^+ \cap (S \cap S') \neq \emptyset$
$\Rightarrow$ $S \cap S'$ is closed *w.r.t.* Π. □

Lemma 7.2. *Let* Π *be an extended logic program,* $S \subseteq Lit$ *and* $r \in \Pi$. *Then*

A1. $r^+ \leftarrow r^- \in \Pi^S$ *and* $r^- \subseteq S$ *if and only if* $r^+ \cap S \leftarrow r^- \in \mathrm{MR}(\Pi, S)$.

A2. *S is closed w.r.t.* Π^S *if and only if S is closed w.r.t.* $\mathrm{MR}(\Pi, S)$.

Proof. (1) $r^+ \leftarrow r^- \in \Pi^S$ and $r^- \subseteq S$
iff $r^- \subseteq S$, $r^{not} \cap S = \emptyset$ and $r^{2not} \subseteq S$
iff $r^+ \cap S \leftarrow r^- \in \mathrm{MR}(\Pi, S)$.

(2) It is evident when $S = Lit$. Let S be a consistent set of literals. We have
S is closed *w.r.t.* Π^S
iff $r^- \subseteq S$ implies $r^+ \cap S \neq \emptyset$ for each $r^+ \leftarrow r^-$ in Π^S
iff $r^+ \cap S \neq \emptyset$ for each $r^+ \leftarrow r^-$ in Π^S and $r^- \subseteq S$
iff $r^+ \cap S \neq \emptyset$ for each $r^+ \cap S \leftarrow r^-$ in $\mathrm{MR}(\Pi, S)$ by (1)
iff S is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$. □

Proposition 7.5. *Let* Π *be an extended logic program and* $S \subseteq Lit$. *Then S is an answer set of* Π *if and only if S is least (under set inclusion) and closed w.r.t.* $\mathrm{MR}(\Pi, S)$.

Proof. ($\Rightarrow$) Note that S is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$ by (2) of Lemma 7.2. Suppose that S is not the least one such that it is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$. It implies there is S' with $S \not\subseteq S'$ which is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$. We consider the following two cases:

A1. $S' \subset S$. We have that S' is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$
$\Rightarrow$ S is closed *w.r.t.* $\{r \in \mathrm{MR}(\Pi, S) \mid r^- \subseteq S'\}$ by (1) of Lemma 7.1
$\Rightarrow$ for each $\alpha \in \{r \in \mathrm{MR}(\Pi, S) \mid r^- \subseteq S'\}$, $\alpha^+ \cap S' \neq \emptyset$
$\Rightarrow$ for each $\alpha \in \{r \in \Pi^S \mid r^- \subseteq S'\}$, $\alpha^+ \cap S \cap S' \neq \emptyset$ by (1) of Lemma 7.2
$\Rightarrow$ for each $\alpha \in \{r \in \Pi^S \mid r^- \subseteq S'\}$, $\alpha^+ \cap S' \neq \emptyset$ by $S' \subseteq S$
$\Rightarrow$ S' is closed *w.r.t.* $\{r \in \Pi^S \mid r^- \subseteq S'\}$
$\Rightarrow$ S' is closed *w.r.t.* Π^S by (1) of Lemma 7.1, which is a contradiction.

A2. $S' \not\subseteq S$. Let $S^* = S \cap S'$. It is evident $S^* \subset S$. By (2) of Lemma 7.1, S^* is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$. Following the proof of the above case (1), we can similarly show that S^* is closed *w.r.t.* Π^S, which is a contradiction.

It follows that S must be the least and closed set *w.r.t.* $\mathrm{MR}(\Pi, S)$.

($\Leftarrow$) Not that S is closed *w.r.t.* Π^S by (2) of Lemma 7.2. Suppose that S is not minimal. It follows that there is $S' \subset S$ which is closed *w.r.t.* Π^S
$\Rightarrow S'$ is closed *w.r.t.* $\{r \in \Pi^S \mid r^- \subseteq S'\}$ by (1) of Lemma 7.1
$\Rightarrow$ for each $r \in \Pi^S$, $r^- \subseteq S'$ implies $r^+ \cap S' \neq \emptyset$
$\Rightarrow$ for each $r \in \Pi^S$ and $r^- \subseteq S' \cap S$, $r^+ \cap S \cap S' \neq \emptyset$ by $S' \subset S$
$\Rightarrow$ for each $r \in \mathrm{MR}(\Pi, S)$ and $r^- \subseteq S'$, $r^+ \cap S' \neq \emptyset$ by (1) of Lemma 7.2
$\Rightarrow S'$ is closed *w.r.t.* $\mathrm{MR}(\Pi, S)$, which is a contradiction.

Thus, S is a minimal and closed set *w.r.t.* Π^S, *i.e.*, S is an answer set of Π. □

References

Marco Almada. Human intervention in automated decision-making: Toward the construction of contestable systems. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, ICAIL '19, pages 2–11, New York, NY, USA, 2019. Association for Computing Machinery.

Mario Alviano and Carmine Dodaro. Unsatisfiable core analysis and aggregates for optimum stable model search. *Fundamenta Informaticae*, 176(3-4):271–297, 2020.

Mario Alviano and Wolfgang Faber. Aggregates in answer set programming. *Künstliche Intelligenz*, 32(2-3):119–124, 2018.

Mario Alviano, Carmine Dodaro, Johannes Klaus Fichte, Markus Hecher, Tobias Philipp, and Jakob Rath. Inconsistency proofs for ASP: the ASP - DRUPE format. *Theory and Practice of Logic Programming*, 19(5-6):891–907, 2019.

Giovanni Amendola, Thomas Eiter, Michael Fink, Nicola Leone, and João Moura. Semi-equilibrium models for paracoherent answer set programs. *Artificial Intelligence*, 234:219–271, 2016.

Giovanni Amendola, Francesco Ricca, and Mirek Truszczynski. A generator of hard 2qbf formulas and ASP programs. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, pages 52–56, Tempe, Arizona, USA, 2018. AAAI Press.

Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. Justifications for goal-directed constraint answer set programming. In Ricca et al. Ricca *et al.* [2020], pages 59–72.

Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):53–87, 1994.

Rachel Ben-Eliyahu-Zohary, Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Decomposing minimal models. In *KnowProS@IJCAI*, pages 1–7, New York City, 2016. CEUR-WS.org.

Rachel Ben-Eliyahu-Zohary, Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Modular construction of minimal models. In Marcello Balduccini and Tomi Janhunen, editors, *LPNMR-2017, Espoo, Finland*, volume 10377 of *Lecture Notes in Computer Science*, pages 43–48, Espoo, Finland, 2017. Springer.

Alexander Bochman. *A Logical Theory of Causality*. The MIT Press, 2021.

Jori Bomanson. lp2normal — a normalization tool for extended logic programs. In Marcello Balduccini and Tomi Janhunen, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 222–228, Cham, 2017. Springer International Publishing.

Martin Brain and Marina De Vos. Debugging logic programs under the answer set semantics. In Marina De Vos and Alessandro Provetti, editors, *Answer Set Programming, Advances in Theory and Implementation, Proceedings of the 3rd Intl. ASP'05 Workshop, Bath, UK, September 27-29, 2005*, volume 142 of *CEUR Workshop Proceedings*, pages 141–152, Bath, UK, 2005. CEUR-WS.org.

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

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

Samuel Buss, Jan Krajìček, and Gaisi Takeuti. On provably total functions in bounded arithmetic theories. In Peter Clote and Jan Krajìček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 116–61. Oxford University Press, 1993.

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.

Pedro Cabalar, Jorge Fandinno, and Michael Fink. Causal graph justifications of logic programs. *Theory and Practice of Logic Programming*, 14(4-5):603–618, 2014.

Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. A system for explainable answer set programming. In Ricca et al. Ricca *et al.* [2020], pages 124–136.

Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca, and Torsten Schaub. Asp-core-2 input language format. *Theory and Practice of Logic Programming*, 20(2):294–309, 2020.

Hubie Chen and Yannet Interian. A model for generating random quantified boolean formulas. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 66–71, Edinburgh, Scotland, UK, 2005. Professional Book Center.

Zhi-Zhong Chen and Seinosuke Toda. The complexity of selecting maximal solutions. *Information and Computation*, 119(2):231–239, 1995.

Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

Roberto Confalonieri, Tillman Weyde, Tarek R. Besold, and FermÃn Moscoso del Prado MartÃn. Using ontologies to enhance human understandability of global post-hoc explanations of black-box models. *Artificial Intelligence*, 296:103471, 2021.

Luca Costabello, Fosca Giannotti, Riccardo Guidotti, Pascal Hitzler, Freddy Lécué, Pasquale Minervini, and Kamruzzaman Sarker. On explainable AI: From theory to motivation, applications and limitations, 2019. AAAI 2019 Tutorial, https://xaitutorial2019.github.io/.

Carlos Viegas Damásio, João Moura Pires, and Anastasia Analyti. Unifying justifications and debugging for answer-set programs. In Marina De Vos, Thomas Eiter, Yuliya Lierler, and Francesca Toni, editors, *ICLP-2015), Cork, Ireland*, volume 1433 of *CEUR Workshop Proceedings*, pages 1–14, Cork, Ireland, 2015. CEUR-WS.org.

Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

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.

Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 127–144. Springer US, Boston, MA, 2000.

Marc Denecker, Gerhard Brewka, and Hannes Strass. A formal theory of justifications. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning: 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, pages 250–264. Springer International Publishing, Cham, 2015.

Carmine Dodaro and Francesco Ricca. The external interface for extending WASP. *Theory and Practice of Logic Programming*, 20(2):225–248, 2020.

Carmine Dodaro, Francesco Ricca, and Peter Schüller. External propagators in WASP: preliminary report. In Stefano Bistarelli, Andrea Formisano, and Marco Maratea, editors, *Proceedings of the 23rd RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion 2016 (RCRA 2016), Genova, Italy, November 28, 2016*, volume 1745 of *CEUR Workshop Proceedings*, pages 1–9, Genova, Italy, 2016. CEUR-WS.org.

Carmine Dodaro, Philip Gasteiger, Kristian Reale, Francesco Ricca, and Konstantin Schekotihin. Debugging non-ground ASP programs: Technique and graphical tools. *Theory and Practice of Logic Programming*, 19(2):290–316, 2019.

William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logical Programming*, 1(3):267–284, 1984.

Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 90–96, Edinburgh, Scotland, UK, 2005. Professional Book Center.

Esra Erdem, Yelda Erdem, Halit Erdogan, and Umut Öztok. Finding answers and generating explanations for complex biomedical queries. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, pages 785–790, San Francisco, California, USA, 2011. AAAI Press.

Francesc Esteva, Joan Gispert, and Felip Manya Barceloà, editors. *40th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2010, Barcelona, Spain, 26-28 May 2010*, 1730 Massachusetts Ave., NW Washington DC, United States, 2010. IEEE Computer Society.

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.

Michael R. Fellows, Stefan Szeider, and Graham Wrightson. On finding short resolution refutations and small unsatisfiable subsets. *Theoretical Computer Science*, 351(3):351–359, 2006.

Paolo Ferraris. On modular translations and strong equivalence. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005, Diamante, Italy, September 5-8, 2005, Proceedings*, volume 3662 of *Lecture Notes in Computer Science*, pages 79–91, Diamante, Italy, 2005. Springer.

Martin Gebser, Jörg Pührer, Torsten Schaub, and Hans Tompits. A meta-programming technique for debugging answer-set programs. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 448–453, Chicago, Illinois, USA, 2008. AAAI Press.

Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Advanced conflict-driven disjunctive answer set solving. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 912–918, Beijing, China, 2013. IJCAI/AAAI.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694:9, 2014.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with Clingo 5. In Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos, editors, *Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, October 16-21, 2016, New York City, USA*, volume 52 of *OASICS*, pages 2:1–2:15, New York City, USA, 2016. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Allen Van Gelder. Verifying RUP proofs of propositional unsatisfiability. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2008, Fort Lauderdale, Florida, USA, January 2-4, 2008*, page 9, Fort Lauderdale, Florida, USA, 2008. online.

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.

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

Evguenii I. Goldberg and Yakov Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10886–10891, Munich, Germany, 2003. IEEE Computer Society.

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

Botros N. Hanna, Ly Ly T. Trieu, Tran Cao Son, and Nam T. Dinh. An application of ASP in nuclear engineering: Explaining the three mile island nuclear accident scenario. *Theory and Practice of Logic Programming*, 20(6):926–941, 2020.

Carl G. Hempel and Paul Oppenheim. Studies in the logic of explanation. *Philosophy of Science*, 15(2):135–175, 1948.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, Oxford, UK, 2018. Springer, Cham.

Mikolás Janota and Joao Marques-Silva. On the query complexity of selecting minimal sets for monotone predicates. *Artificial Intelligence*, 233:73–83, 2016.

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

Zhang Li, Wang Yisong, Xie Zhongtao, and Feng Renyan. Computing propositional minimal models: Minisat-based approaches. *Journal of Computer Research and Development (in Chinese)*, 58:2515–2523, 2021.

Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artificial Intelligence*, 163(2):203–232, 2005.

Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible mus enumeration. *Constraints*, 21(2):223–250, Apr 2016.

Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.

Vladimir Lifschitz. Foundations of logic programming. In *Principles of Knowledge Representation*, pages 69–127. CSLI Publications, 1996.

Fangzhen Lin and Yoav Shoham. A logic of knowledge and justified assumptions. *Artificial Intelligence*, 57(2-3):271–289, 1992.

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.

João Marques-Silva. Minimal unsatisfiability: Models, algorithms and applications (invited paper). In Esteva et al. Esteva *et al.* [2010], pages 9–14.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

Brent D. Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in AI. In danah boyd and Jamie H. Morgenstern, editors, *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, pages 279–288, Atlanta, GA, USA, 2019. ACM.

Johannes Oetsch, Jörg Pührer, and Hans Tompits. Catching the ouroboros: On debugging non-ground answer-set programs. *Theory and Practice of Logic Programming*, 10(4-6):513–529, 2010.

Johannes Oetsch, Jörg Pührer, and Hans Tompits. Stepwise debugging of answer-set programs. *Theory and Practice of Logic Programming*, 18(1):30–80, 2018.

Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

Axel Polleres, Melanie Frühstück, Gottfried Schenner, and Gerhard Friedrich. Debugging non-ground ASP programs with choice rules, cardinality and weight constraints. In Pedro Cabalar and Tran Cao Son, editors, *Proc. 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), Corunna, Spain, September 15-19, 2013*, volume 8148 of *Lecture Notes in Computer Science*, pages 452–464, Corunna, Spain, 2013. Springer.

John L. Pollock. *Knowledge and Justification*. Princeton University Pres, 1974.

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.

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 ICLP, Technical Communications 2020, UNICAL, Rende (CS), Italy, 18-24th September 2020*, volume 325 of *EPTCS*, 2020.

Chiaki Sakama and Katsumi Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5(3):265–285, 1995.

Claudia Schulz and Francesca Toni. Justifying answer sets using argumentation. *Theory and Practice of Logic Programming*, 16(1):59–110, 2016.

Kostyantyn M. Shchekotykhin. Interactive query-based debugging of ASP programs. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 1597–1603, Austin, Texas, USA, 2015. AAAI Press.

Yi-Dong Shen and Thomas Eiter. Determining inference semantics for disjunctive logic programs. *Artificial Intelligence*, 277:115–135, December 2019.

João P. Marques Silva. Minimal unsatisfiability: Models, algorithms and applications (invited paper). In Esteva et al. Esteva *et al.* [2010], pages 9–14.

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

Ramya Srinivasan and Ajay Chander. Explanation perspectives from the cognitive sciences - A survey. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4812–4818, Yokohama, Japan, 2020. ijcai.org. Scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic.

Tommi Syrjänen. Debugging inconsistent answer set programs. In *Proceedings of The 11Th International Workshop on Nonmonotonic Reasoning (NMR'06). Number IFI-06-04 in Technical Report Series, Clausthal University of Technology, Institute for Informatics (2006) 77-83*, pages 77–83, Lake District, England, 2006. Clausthal University of Technology, Institute for Informatics.

Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, 1976.

Yan Zhang and Yuanlin Zhang. Epistemic specifications and conformant planning. In *WS-17-01*, AAAI Workshop - Technical Report, pages 781–787, San Francisco, USA, 2017. AI Access Foundation. Publisher Copyright: © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.; null ; Conference date: 04-02-2017 Through 10-02-2017.