Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Foundations for Projecting Away the Irrelevant in ASP Programs

Zeynep G. Saribatur , Stefan Woltran
Institute of Logic and Computation, TU Wien
{zeynep.saribatur, stefan.woltran}@tuwien.ac.at

Abstract

Simplification of logic programs under the answer set semantics has been studied from the very beginning of the field. One natural simplification is the removal of atoms that are deemed irrelevant. While equivalence-preserving rewritings are well understood and incorporated in state-of-the-art systems, more careful rewritings in the realm of strong or uniform equivalence have received considerably less attention. This might be due to the fact that these equivalence notions rely on comparisons with respect to context programs that remain the same for both the original and the simplified program. In this work, we pursue the idea that the atoms considered irrelevant are disregarded accordingly in the context programs of the simplification, and propose novel equivalence notions for this purpose. We provide necessary and sufficient conditions for these kinds of simplifiability of programs, and show that such simplifications, if possible, can actually be achieved by just projecting the atoms from the programs themselves. We furthermore provide complexity results for the problems of deciding simplifiability and equivalence testing.

1 Introduction

The desire to get rid of irrelevant details/rules that overcomplicate an ASP program and its evaluation motivated many works over the years on simplification of ASP programs while preserving their semantics, such as equivalence-based rewriting (Gebser et al. 2008; Pearce 2004), partial evaluation (Brass and Dix 1997; Janhunen et al. 2006) and forgetting (see (Leite 2017) for a recent survey). In all of these works (apart from forgetting), even though the program gets simplified, the overall signature would remain the same, allowing for different forms of equivalence of the original program and the simplified program to be investigated.

The equivalence of logic programs is determined by means of the answer set semantics (Gelfond and Lifschitz 1991): a program P is equivalent to a program Q if $AS(P) = AS(Q)$, where $AS(\cdot)$ denotes the collection of answer sets of a program. *Strong equivalence* (Lifschitz, Pearce, and Valverde 2001) is a much stricter condition over the two programs: P and Q are strongly equivalent if, for any set R of rules, the programs $P \cup R$ and $Q \cup R$ are equivalent, i.e., if

$$AS(P \cup R) = AS(Q \cup R). \quad (1)$$

This notion makes it possible to simplify a part of a logic program without looking at the rest of it: if a subprogram P of Π is strongly equivalent to a simpler program Q, then P can be replaced by Q without changing the answer sets of Π. The works (Osorio, Navarro, and Arrazola 2002; Turner 2003; Eiter et al. 2004; Pearce 2004) show ways of transforming programs by ensuring that the property holds. There are also more liberal notions of equivalence, such as *uniform equivalence* (Maher 1986; Sagiv 1987; Eiter and Fink 2003) where R is restricted to a set of facts, and *relativised* versions of strong and uniform equivalence (Woltran 2004) that are defined for the case of having the newly added rules or facts, R, in a specific language. In addition, (Eiter, Tompits, and Woltran 2005) also addresses equivalence under projection of answer sets. Note that all these equivalence notions consider the same R on both sides of Equation (1) thus leaving the signature of the potentially simplified program Q untouched.[1] However truly capturing the irrelevance in programs requires to remove those details even from the signature, so that they are no longer taken into account, while ensuring that the semantics of the original program P is preserved w.r.t. the modified signature.

The grounders for ASP solvers already conduct such simplifications by, for example, recursively eliminating facts. If the programs get extended, the simplification is reapplied. One can in fact project away all the facts from these simplified programs and the overall vocabulary, while still ensuring that the original semantics is preserved under projection. Though the theory behind this is not known. Furthermore, not all possible such simplifications can be detected by the grounders, especially when there is a guess involved among the atoms that could in fact be projected away. Moving towards applications, we are motivated by the following example, which currently cannot be captured by the representations existing in the literature of equivalence. Consider the Blocksworld planning problem where the goal is to pile up the blocks from a given initial layout. Now also consider the case that the blocks are colored and they can be of different colors in different initial states, I_i. However, if colors are not of relevance to the computation of the plan, then one

[1]Only (Pearce and Valverde 2004) considers a setting where the programs are formulated in different languages, but each language needs to be bijectively interpretable into the other.

would expect to remove the details about the colors from the planning domain description and still obtain the same plans. Since I_i's are described through sets of facts, the desired relation hints at uniform equivalence. However, the signature also gets modified, due to removing the details regarding colors, also from the initial states I_i.

The research on forgetting (Gonçalves, Knorr, and Leite 2016a; Delgrande 2017) is actually closer to our interest, since there for a program P the aim is to construct a program $f(P, V)$ by applying an operator f on P to forget the atoms in V from the vocabulary, so that the resulting program is over $\overline{V}$ (i.e., the universe of atoms without V). Among the many properties that have been investigated in the forgetting literature, the notion of Strong Persistence **(SP)** requires the correspondence between answer sets of the result of forgetting and those of the original program be preserved in the presence of any additional set of rules not containing the atoms to be forgotten, shown as $AS(f(P, V) \cup R) = AS(P \cup R)_{|\overline{V}}$, where the vocabulary of R is restricted to $\overline{V}$. Uniform Persistence **(UP)** accordingly considers R to be a set of facts. Such a restriction over the vocabulary of the added set of rules/facts prevents these notions from truly capturing the forgetting of atoms, since it avoids possible interferences between the rules in R and the rules in P containing the atoms to be forgotten.

While all the notions discussed above thus have in common that the context does not respect the considered simplification, we are interested here in equivalence-like notions that compare a program P over $\mathcal{U}$ with a potential simplification Q over $\mathcal{U} \setminus A$, in short $\overline{A}$, in the sense that

$$AS(P \cup R)|_{\overline{A}} = AS(Q \cup R^*|_{\overline{A}}) \tag{2}$$

holds for all R. In particular, we introduce the notions of

- uniform A simplification where R is a set of facts and $R^*|_{\overline{A}}$ amounts to $R \cap \overline{A}$; and
- strong A simplification where R is a set of rules and $R^*|_{\overline{A}}$ amounts to the rules from $\overline{A}$.[2]

The intuition behind this notion is to not lose the information on interactions between P and R when simplifying. It is not surprising that for given P and A the existence of a strong or uniform A simplification is not guaranteed. We thus establish criteria for simplifiability, and in case they are fulfilled, characterizations for deciding (2) in a way that is similar to what SE- and UE-models do for the respective equivalence notions. Such a Q can be seen as an abstraction of P, relating to the works (Saribatur and Eiter 2021; Saribatur, Eiter, and Schüller 2021), where the aim is to construct an abstract program Q (with a reduced signature according to some mapping) from P so that the answer sets of P are *over-approximated* in Q, i.e., any answer set in P can be mapped to some answer set in Q. Q is considered to be a *faithful* abstraction if it does not have any spurious answer sets. However, we are interested in those Q's which could replace P while fully preserving its semantics w.r.t. the mapping, here considered as projection, especially when there are newly added rules or facts which also get abstracted.

[2]As we shall discuss later, we will restrict R here to programs where each rule is either given over $\mathcal{U} \setminus A$ or A.

Our main contributions are thus as follows

- We define notions of simplifications of programs over projected vocabularies satisfying the strong and uniform equivalence in such a setting.
- We provide necessary and sufficient conditions for testing whether a program can have strong (resp. uniform) A-simplifications, for a set A of atoms. We also give characterizations of strong (resp. uniform) A-simplifications, which do not require an explicit enumeration of the context programs R.
- We show that for strong (resp. uniform) A-simplifiable programs, simply removing the atoms in A from the rules achieves the desired simplifications.
- We provide complexity results for the problems of deciding simplifiability and equivalence testing; some of them turn out to be hard for the third level of the polynomial hierarchy.

2 Background

We use the traditional representation of a rule r as the form

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

where A_i ($1 \leq i \leq n$, $0 \leq l \leq m \leq n$) are atoms from a first-order language, and not is default negation. We also write r as $H(r) \leftarrow B(r)$ or $H(r) \leftarrow B^+(r)$, $not\ B^-(r)$. We call $H(r) = \{A_1, \ldots, A_l\}$ the *head* of r, $B^+(r) = \{A_{l+1}, \ldots, A_m\}$ the *positive body* of r and $B^-(r) = \{A_{m+1}, \ldots, A_n\}$ the *negative body* of r. If $H(r) = \emptyset$, then r is a *constraint*. r is a (non-disjunctive) *fact* if $B(r) = \emptyset$ and $card(H(r)) \leq 1$; for $H(r) = \emptyset$, we occasionally write $\bot$. A rule r is *normal*, if $card(H(r)) \leq 1$, and *positive* if $B^-(r) = \emptyset$. A *disjunctive logic program* (DLP) is a finite set of rules. In the rest of the paper, we focus on propositional programs over a set of atoms from the universe $\mathcal{U}$. Programs with variables reduce to their ground versions as usual. Unless stated otherwise the term *program* refers to a (propositional) DLP.

Let $I \subseteq \mathcal{U}$ be an intepretation. The *GL-reduct* of a program P w.r.t. I is given by $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap I = \emptyset\}$. An interpretation I is a *model* of a program P (in symbols $I \models P$) if, for each $r \in P$, $(H(r) \cup B^-(r)) \cap I \neq \emptyset$ or $B^+(r) \not\subseteq I$; I is an *answer set*, if it is a minimal model of P^I. We denote the set of all answer sets by $AS(P)$. Two programs P_1, P_2 are *equivalent* if $AS(P_1) = AS(P_2)$, *strongly equivalent (SE)*, denoted by $P_1 \equiv P_2$, if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for every R over $\mathcal{U}$, and *uniformly equivalent (UE)*, denoted by $P_1 \equiv_u P_2$ if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for any set of facts R over $\mathcal{U}$.

An *SE-interpretation* is a pair $\langle X, Y \rangle$ such that $X \subseteq Y \subseteq \mathcal{U}$; it is *total* if $X = Y$ and *non-total* otherwise. An SE-interpretation $\langle X, Y \rangle$ is an *SE-model* of a program P if $Y \models P$ and $X \models P^Y$. An SE-model $\langle X, Y \rangle$ of P is called *UE-model* of P if $X = Y$ or there is no SE-model $\langle X', Y \rangle$ with $X \subset X' \subset Y$. We denote by $SE^V(P)$ (resp. UE^V) the set of all SE-models (UE-models) of P over the set of atoms $V \subseteq \mathcal{U}$. If $V = \mathcal{U}$, we drop the superscript for simplicity. Note that a set Y of atoms is an answer set of P if $\langle Y, Y \rangle \in SE(P)$ and no non-total $\langle X, Y \rangle \in SE(P)$

exists. Two programs P_1 and P_2 are strongly equivalent iff $SE(P_1) = SE(P_2)$ (Turner 2001); they are uniformly equivalent iff $UE(P_1) = UE(P_2)$ (Eiter and Fink 2003).

For a set $S \subseteq \mathcal{U}$ of atoms, $S_{|A}$ denotes the projection to the atoms in A and $\overline{S}$ is a shorthand for $\mathcal{U} \setminus S$. We also use the notion on pairs, i.e. $\langle X, Y\rangle_{|A} = \langle X_{|A}, Y_{|A}\rangle$ and on sets of objects, i.e. $\mathcal{S}_{|A} = \{S_{|A} \mid S \in \mathcal{S}\}$.

Finally, the following property on the DLPs will be useful for our results.

Proposition 1 ((Eiter et al. 2013))**.** *For each DLP P, it holds that for all $\langle X, Y\rangle, \langle Z, Z\rangle \in SE(P)$ such that $X \subseteq Y \subseteq Z$, $\langle X, Z\rangle \in SE(P)$.*

3 Simplifications and Simplifiability

In this section, we introduce the notions for the simplification of programs by removing atoms from their vocabulary, while preserving the desired UE and SE-like semantics.[3] We show the necessary and sufficient conditions for simplifiability and the characterizations for the simplifications.

3.1 Uniform Simplification

We begin with introducing a notion to capture the motivating example by lifting the existing uniform equivalence notion to consider a Q with a smaller vocabulary. Due to the programs P and Q being of different vocabularies "equivalence" can no longer hold between them, since the symmetry condition of equivalence cannot be satisfied. For this, we define Q to be a simplification of P w.r.t. removal of a set of atoms, while preserving the semantics of P in the spirit of uniform equivalence.

Definition 1. *Given $A \subseteq \mathcal{U}$ and a program P (over $\mathcal{U}$), program Q (over $\overline{A} = \mathcal{U} \setminus A$) is a* uniform A-simplification of P *if for any set F of facts over $\mathcal{U}$, we have*

$$AS(P \cup F)_{|\overline{A}} = AS(Q \cup F_{|\overline{A}}). \tag{3}$$

We say that P is uniform A-simplifiable if there exists a program Q such that (3) holds.

Note that in the left-hand side of the equation we project on the level of answer sets, while on the right-hand side any $Q \cup F_{|\overline{A}}$ is already given over $\overline{A}$ and thus has all answer sets being subsets of $\overline{A}$.

Example 1. *Let program P consist of rules*

$$a \leftarrow c, not\ b. \qquad b.$$

Since the body of the first rule can never be satisfied due to the existence of the fact b, the answer sets of $P \cup F$ for any F relies only on the fact b. Due to this, the use of c to obtain a in an answer set cannot occur, which makes it an irrelevant detail. Thus, a program Q over signature $\{a, b\}$ consisting only of the fact $\{b.\}$ would be a $\{c\}$-simplification of P.

Not every program might have a uniform A-simplification. We can see this from investigating the undesired case that prevents a program P from being uniform A-simplifiable, which comes from the following.

[3]"Simplification" does not necessarily mean smaller programs. Though as we shall show later, smaller programs can be obtained.

Proposition 2. *If there is a uniform A-simplification of P, then P satisfies the condition*

$$\begin{aligned}\forall Y \forall Z, Y \in AS(P \cup Z), \forall Z', Z'_{|\overline{A}} = Z_{|\overline{A}}\\ \exists Y', Y'_{|\overline{A}} = Y_{|\overline{A}}, Y' \in AS(P \cup Z')\end{aligned} \tag{4}$$

The intuition comes from the fact that for any added facts Z we need to ensure that there is a correspondence among the answer sets of $P \cup Z$ (projected onto remaining atoms) and the answer sets obtained from the uniform A-simplification combined with $Z_{|\overline{A}}$. Such a correspondence cannot occur if P has a mismatch of answer sets for different Z's that agree on the projected atoms. So for any Y and Z where $Y \in AS(P \cup Z)$ it cannot be the case that there exists some Z' (that agrees on the remaining atoms with Z), where there is no answer set Y' of $P \cup Z'$, which agrees on the remaining atoms with Y.

Example 2. *Let program P consist of rules*

$$a \leftarrow not\ b. \qquad b \leftarrow not\ a.$$

Say we consider removing the atom b from the vocabulary. However observe that $AS(P \cup \emptyset) = \{\{a\}, \{b\}\}$ while $\{a\}, \{a, b\} \notin AS(P \cup \{b\})$.

Using this knowledge, one can obtain necessary conditions for uniform A-simplifiability.

Proposition 3. *If there is a uniform A-simplification of P, then P satisfies the following:*

Δ_{u_1}: *For each $\langle X, Y\rangle \in SE(P)$ with $X \subset Y$ and $X_{|\overline{A}} = Y_{|\overline{A}}$, there exists $Y' \supseteq X$, $Y'_{|\overline{A}} = Y_{|\overline{A}}$, such that $\langle Y', Y'\rangle \in SE(P)$ and for each M with $X \subseteq M \subset Y'$, $\langle M, Y'\rangle \notin SE(P)$.*

Δ_{u_2}: *For each X, Y with $X \subset Y$, $\langle Y, Y\rangle \in SE(P)$, such that for each M with $X \subseteq M \subset Y$, $\langle M, Y\rangle \notin SE(P)$, and for each X' with $X_{|\overline{A}} = X'_{|\overline{A}}$, there exists $\langle Y', Y'\rangle \in SE(P)$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$ such that $X' \subseteq Y'$ and for each M' with $X' \subseteq M' \subset Y'$, $\langle M', Y'\rangle \notin SE(P)$.*

Δ_{u_3}: *$\langle Y, Y\rangle \in SE(P)$ implies $\langle Y \cup A, Y \cup A\rangle \in SE(P)$.*

Proof. Let Q be a uniform A-simplification of P.

Δ_{u_1}: Assume that for some $\langle X, Y\rangle \in SE(P)$ no such Y' exists. We have $Y \in AS(P \cup Y)$ and thus $Y_{|\overline{A}} \in AS(Q \cup Y_{|\overline{A}})$. Now consider $P \cup X$. Then we have $Y_{|\overline{A}} \in AS(Q \cup X_{|\overline{A}})$ (recall $X_{|\overline{A}} = Y_{|\overline{A}}$). However by our assumption we cannot find a $Y' \in AS(P \cup X)$ such that $Y'_{|\overline{A}} = Y$ which contradicts Proposition 2.

Δ_{u_2}: Assume that for some X, Y, X' as given in the condition, no such Y' exists. So for all $\langle Y', Y'\rangle \in SE(P)$ either (1) $X' \not\subseteq Y'$ or (2) for some M' with $X' \subseteq M' \subset Y'$, $\langle M', Y\rangle \in SE(P)$. We have $\langle Y, Y\rangle \in AS(P \cup X)$, and thus $Y_{|\overline{A}} \in AS(Q \cup X_{|\overline{A}})$. However by our assumption we cannot find a $Y' \in AS(P \cup X')$ such that $Y'_{|\overline{A}} = Y_{|\overline{A}}$ and $X_{|\overline{A}} = X'_{|\overline{A}}$ which contradicts Proposition 2.

Δ_{u_3}: Assume for $\langle Y, Y\rangle \in SE(P)$ this is not the case. Then $AS(P \cup (Y \cup A)) = \emptyset$ while $Y \in AS(P \cup Y)$. $\square$

Example 3 (Ex. 2 ctd)**.** *The SE-models of P are $\langle a, a\rangle$, $\langle b, b\rangle, \langle\emptyset, ab\rangle, \langle a, ab\rangle, \langle b, ab\rangle, \langle ab, ab\rangle$.*[4] *Now observe that for $\langle a, a\rangle \in SE(P)$ and $X = \emptyset$, for $X' = \{b\}$ ($X'_{|\overline{A}} = X_{|\overline{A}}$) we look at $\langle ab, ab\rangle \in SE(P)$. But $\langle b, ab\rangle \in SE(P)$, thus Δ_{u_2} is not satisfied.*

For program P in Example 2, we can see that b is actually not an irrelevant atom that could be removed, since the truth value of a depends on the truth value of b.

Example 4 (Ex. 1 (ctd))**.** *We have $SE(P) = \{\langle b, b\rangle, \langle b, ab\rangle, \langle ab, ab\rangle, \langle b, bc\rangle, \langle bc, bc\rangle, \langle b, abc\rangle, \langle ab, abc\rangle, \langle bc, abc\rangle, \langle abc, abc\rangle\}$. For Δ_{u_1}, the only relevant non-total model is $\langle b, bc\rangle \in SE(P)$, and with $\langle b, b\rangle \in SE(P)$ the condition is satisfied. For Δ_{u_2}, let's look at $\langle b, b\rangle$ and $X = \emptyset$, since $\langle\emptyset, b\rangle \notin SE(P)$. For $X' = c$, $\langle c, bc\rangle \notin SE(P)$. Now let's look at $\langle ab, ab\rangle$ and $X = a$, since $\langle a, ab\rangle \notin SE(P)$. For $X' = ac$, $\langle ac, abc\rangle \notin SE(P)$. Thus Δ_{u_2} is satisfied. It is easy to see that Δ_{u_3} is also satisfied.*

Let us have a final, slightly more involved, example to illustrate the functioning of Δ_{u_1} and Δ_{u_2}. Later in Section 5, we use similar structures for a complexity result regarding uniform simplifiability.

Example 5. *Consider a program P with the SE-models*

$$\begin{array}{lll} \langle abc, abc\rangle & \langle abd, abd\rangle & \langle abcd, abcd\rangle \\ \langle abc, abcd\rangle & \langle abd, abcd\rangle & \langle\emptyset, abcd\rangle \\ \langle a, abc\rangle & \langle a, abcd\rangle & \end{array}$$

and the set $A = \{a, b, c, d\}$; therefore for all $\langle X, Y\rangle \in SE(P)$, $X_{|\overline{A}} = Y_{|\overline{A}}$. Moreover, Δ_{u_3} is satisfied, due to the presence of $\langle abcd, abcd\rangle$.

We first argue that P satisfies Δ_{u_1} and Δ_{u_2}. We start with Δ_{u_1}. To this end, we need for each $\langle X, Y\rangle \in SE(P)$ with $X \subset Y$ an Y', such that $\langle Y', Y'\rangle \in SE(P)$ and for each M of the form $X \subseteq M \subset Y'$, $\langle M, Y'\rangle \notin SE(P)$. For $X = abc$ we can use $Y' = abc$ (likewise for $X = abd$, use $Y' = abd$). For $X = a$ and $X = \emptyset$, we can use $Y' = abd$ (but not $Y' = abc$). Let us turn to Δ_{u_2}. Since we have, e.g. $\langle bcd, abcd\rangle \notin SE(P)$ we have to check the particular condition of Δ_{u_2} for each $X' \subseteq abcd$. For the total SE-models $\langle X', X'\rangle \in SE(P)$ this clearly holds. For all $X' \subset abd$, we have $\langle X', abd\rangle \notin SE(P)$, and $\langle abd, abd\rangle \in SE(P)$. Likewise, for all X' with $cd \subseteq X' \subset abcd$, we have $\langle X', abcd\rangle \notin SE(P)$ and $\langle abcd, abcd\rangle \in SE(P)$. Finally, for all X' with $c \subseteq X' \subset abc$, we have $\langle X', abc\rangle \notin SE(P)$ and $\langle abc, abc\rangle \in SE(P)$. All cases are thus covered.

Let us now change P to P' such that the following two additional SE-models are present:

$$\langle b, abd\rangle \qquad \langle b, abcd\rangle$$

Δ_{u_1} and Δ_{u_2} are now violated and it can be checked that the pair $\langle\emptyset, Y\rangle$ becomes the problematic SE-model. In fact, for Δ_{u_1} the "path" to $\langle abd, abd\rangle$ becomes blocked, while for $X' = b$ we still would have that for all M with $X' \subseteq M \subset abc$, $\langle M, Y'\rangle \notin SE(P)$. For Δ_{u_2}, the same reasoning applies.

By using the property on DLPs from Proposition 1, we can obtain a further condition of uniform A-simplifiability.

[4]We occasionally use strings to denote sets for SE-models.

Proposition 4. *Let P satisfy Δ_{u_2}. If $\langle X, Y\rangle \in UE(P)$ with $X_{|\overline{A}} \subset Y_{|\overline{A}}$ and $X \cap A = Y \cap A$, then for all $\langle Y', Y'\rangle \in SE(P)$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$, $\langle X_{|\overline{A}} \cup (Y' \cap A), Y'\rangle \in UE(P)$.*

Proof. Assume for $\langle X, Y\rangle \in UE(P)$ there exists some $\langle Y', Y'\rangle \in SE(P)$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$ such that $\langle X_{|\overline{A}} \cup (Y' \cap A), Y'\rangle \notin UE(P)$. So either (1) $\langle X_{|\overline{A}} \cup (Y' \cap A), Y'\rangle \notin SE(P)$ or (2) there exists some $X' \supset X_{|\overline{A}} \cup (Y' \cap A)$ such that $\langle X', Y'\rangle \in UE(P)$. For case (1), if there is some $X' \supset X_{|\overline{A}} \cup (Y' \cap A)$ such that $\langle X', Y'\rangle \in UE(P)$ we apply step (2). So consider $\langle X', Y'\rangle \notin SE(P)$ for all $X' \supset X_{|\overline{A}} \cup (Y' \cap A)$. By Δ_{u_2}, we need to ensure that for all X''s with $X''_{|\overline{A}} = X_{|\overline{A}}$ there is some $\langle Y'', Y''\rangle$ such that for all M, $X'' \subseteq M \subset Y$, $\langle M, Y''\rangle \notin SE(P)$. So also for X, we need to find such a $Y'' \neq Y$ while $X \subseteq Y''$. Since $Y''_{|\overline{A}} = Y_{|\overline{A}}$, Y'' and Y can only differ on atoms from A. But then since $X \cap A = Y \cap A$, $Y'' \cap A \supset Y \cap A$ should hold, thus $Y'' \supset Y$, which brings us to a contradiction with Proposition 1. For case (2), if there is some $\langle X', Y'\rangle \in UE(P)$ with $X' \supset X_{|\overline{A}} \cup (Y' \cap A)$ then it also satisfies $X'_{|\overline{A}} \subset Y'_{|\overline{A}}$ and $X' \cap A = Y' \cap A$. Then we can apply the same reasoning for X' and consider the two cases. By doing this recursively, eventually case (2) will not be applicable. Thus a contradiction through case (1) will be achieved. ☐

We now define A-UE models, that project those UE-models of importance for uniform A-simplification.

Definition 2. *Given a program P over $\mathcal{U}$ and $A \subseteq \mathcal{U}$, the A-UE-models of P are given by*

$$UE_A(P) = \{\langle X_{|\overline{A}}, Y_{|\overline{A}}\rangle \mid (X, Y) \in UE(P) \text{ and } X \cap A = Y \cap A\}$$

The aim with A-UE models is to collect the projection of all the total models and those non-total UE-models which differ on the projection but agree on the removed atoms, i.e., $X_{|\overline{A}} \subset Y_{|\overline{A}}$ and $X \cap A = Y \cap A$. We observe that UE models of any uniform A-simplification, if exists, need to adhere with the A-UE models of P.

Proposition 5. *If Q is a uniform A-simplification for P, then it satisfies*

$$UE_A(P) = UE^{\overline{A}}(Q) \tag{5}$$

Proof. By Proposition 3, P satisfies Δ_{u_2}. First assume $\langle X, Y\rangle \in UE_A(P)$ but $\langle X, Y\rangle \notin UE^{\overline{A}}(Q)$. If $X = Y$, this means there is $\langle Y \cup A', Y \cup A'\rangle \in SE(P)$ for some $A' \subseteq A$. Thus $(Y \cup A') \in AS(P \cup (Y \cup A'))$, while $Y \notin AS(Q \cup Y)$, contradicting the assumption of Q being a uniform A-simplification of P. If $X \subset Y$, this means there is $\langle X \cup A', Y \cup A'\rangle \in UE(P)$, but either (1) there exists $\langle M, Y\rangle \in UE^{\overline{A}}(Q)$ such that $M \supset X$, or (2) $\langle X, Y\rangle \notin SE^{\overline{A}}(Q)$. For case (1), consider $M \cup A'$; $Y \cup A' \in AS(P \cup (M \cup A'))$ while $Y \notin AS(Q \cup M)$ which is a contradiction. For case (2), if some $\langle M, Y\rangle \in UE^{\overline{A}}(Q)$ for $M \supset X$ we apply case (1). Assume there is no $\langle M, Y\rangle \in$

$SE^{\overline{A}}(Q)$ such that $M \supset X$. Then $Y \in AS(Q \cup X)$. By Proposition 4 we know that for any $\langle Y', Y' \rangle \in SE(P)$ such that $Y'_{|\overline{A}} = Y$ we have $\langle X \cup (Y' \cap A), Y' \rangle \in UE(P)$. Thus there is no $Y' \in AS(P \cup (X \cup A'))$ for any $A' \subseteq A$ with $Y'_{|\overline{A}} = Y$ which is again a contradiction.

Now assume $\langle X, Y \rangle \in UE^{\overline{A}}(Q)$ but $\langle X, Y \rangle \notin UE_A(P)$. If $X = Y$, we have $Y \in AS(Q \cup Y)$. However $\langle Y, Y \rangle \notin UE_A(P)$ means for any $A' \subseteq A$, $\langle Y \cup A', Y \cup A' \rangle \notin SE(P)$. Thus there is no $Y \cup A' \in AS(P \cup (Y \cup A'))$ for any $A' \subseteq A$ which contradicts the assumption of Q being a uniform A-simplification of P. If $X \subset Y$, this means for all $A' \subseteq A$ with $\langle Y \cup A', Y \cup A' \rangle \in SE(P)$, $\langle X \cup A', Y \cup A' \rangle \notin UE(P)$. So either (1) there exists $\langle X', Y \cup A' \rangle \in UE(P)$ such that $X' \supset X \cup A'$, or (2) $\langle X \cup A', Y \cup A' \rangle \notin SE(P)$. For case (1), for $X'_{|\overline{A}}$, $Y \in AS(Q \cup X'_{|\overline{A}})$. Since $X' \cap A = Y \cap A$ (due to $X' \supset X \cup A'$), Proposition 4 applies, thus for any $\langle Y', Y' \rangle \in SE(P)$ such that $Y'_{|\overline{A}} = Y$, $\langle X'_{|\overline{A}} \cup (Y' \cap A), Y' \rangle \in UE(P)$. So there is no $Y' \in AS(P \cup (X'_{|\overline{A}} \cup A')$ for any $A' \subseteq A$ with $Y'_{|\overline{A}} = Y$ which is a contradiction. For case (2), if some $\langle X', Y \cup A' \rangle \in UE(P)$ for $X' \supset X \cup A'$ we apply case (1). Assume there is no $\langle X', Y \cup A' \rangle \in SE(P)$ such that $X' \supset X \cup A'$. Then $Y \cup A' \in AS(P \cup (X \cup A'))$ but $Y \notin AS(Q \cup X)$ which is again a contradiction. ☐

Example 6 (Ex. 1 (ctd))**.** *Over signature $\{a, b\}$, Q has SE-models $\langle b, b \rangle, \langle b, ab \rangle$, and $\langle ab, ab \rangle$, thus satisfying $UE_A(P) = \{\langle b, b \rangle, \langle ab, ab \rangle\} \cup \{\langle b, ab \rangle\} = UE^{\overline{A}}(Q)$.*

We now show that the necessary conditions for uniform A-simplifiability are also sufficient.

Theorem 6. *If P satisfies $\Delta_{u_1}, \Delta_{u_2}$ and Δ_{u_3}, then it has a uniform A-simplification.*

Proof. We will prove that a program Q that satisfies (5) is a uniform A-simplification of P. Assume it is not the case. Suppose that there exist sets of atoms F and Z such that $Z \in AS(P \cup F)$ and $Z_{|\overline{A}} \notin AS(Q \cup F_{|\overline{A}})$. Since $Z \in AS(P \cup F)$ we have that $F \subseteq Z$ and moreover $Z \models P$. Consequently, $\langle Z, Z \rangle$ is an SE-model of P. Since $Z_{|\overline{A}} \notin AS(Q \cup F_{|\overline{A}})$, let us first assume $Z_{|\overline{A}} \not\models (Q \cup F_{|\overline{A}})^{Z_{|\overline{A}}}$. Then, since $(Q \cup F_{|\overline{A}})^{Z_{|\overline{A}}} = Q^{Z_{|\overline{A}}} \cup F_{|\overline{A}}$ and $F_{|\overline{A}} \subseteq Z_{|\overline{A}}$, it follows that $Z_{|\overline{A}} \not\models Q^{Z_{|\overline{A}}}$. This implies $Z_{|\overline{A}} \not\models Q$. Thus $\langle Z, Z \rangle \in SE(P)$ while $\langle Z_{|\overline{A}}, Z_{|\overline{A}} \rangle \notin SE^{\overline{A}}(Q)$, which contradicts (5). It follows that $Z_{|\overline{A}} \models (Q \cup F_{|\overline{A}})^{Z_{|\overline{A}}}$ holds and that there exists $Z' \subset Z_{|\overline{A}}$ such that $Z' \models (Q \cup F_{|\overline{A}})^{Z_{|\overline{A}}} = Q^{Z_{|\overline{A}}} \cup F_{|\overline{A}}$. We conclude $Z_{|\overline{A}} \models Q$ and that $\langle Z', Z_{|\overline{A}} \rangle \in UE^{\overline{A}}(Q)$. Now consider $Z_1 = Z' \cup (Z \cap A)$. Since $Z' \subset Z_{|\overline{A}}$ we can infer that $Z_1 \subset Z$ should hold. Then since $Z_{1|\overline{A}} = Z' \models F_{|\overline{A}}$ and $Z \models F$, we immediately get that $Z_1 \models F$ should hold. However since $Z \in AS(P \cup F)$, $\langle Z_1, Z \rangle$ cannot be an SE-model of P (thus also cannot be a UE-model of P). By Proposition 4 we get that then there must be no Z_2 with $\langle Z_2, Z_2 \rangle \in SE(P)$ and $Z_{2|\overline{A}} = Z_{|\overline{A}}$, $\langle Z' \cup (Z_2 \cap A), Z_2 \rangle \in UE(P)$, which contradicts (5).

Suppose now that there exist a set F of atoms such that for some $Z \in AS(Q \cup F_{|\overline{A}})$, we have for all $Z \cup A'$ for $A' \subseteq A$, $Z \cup A' \notin AS(P \cup F)$. Since $Z \in AS(Q \cup F_{|\overline{A}})$ we have that $F_{|\overline{A}} \subseteq Z$ and moreover $Z \models Q$. Consequently, $\langle Z, Z \rangle \in SE^{\overline{A}}(Q)$. So $\langle Z, Z \rangle \in UE_A(P)$ which means for some A', $\langle Z \cup A', Z \cup A' \rangle \in SE(P)$. Due to Δ_{u_3}, $\langle Z \cup A, Z \cup A \rangle \in SE(P)$, thus $Z \cup A \models P \cup F$. In order to have $Z \cup A' \notin AS(P \cup F)$ for any $A' \subseteq A$, it should hold that for those A' with $Z \cup A' \models P \cup F$ there exists $Z_1 \subset Z \cup A'$ such that $Z_1 \models (P \cup F)^{Z \cup A'}$. We conclude $Z \cup A' \models P$ and that $\langle Z_1, Z \cup A' \rangle \in UE(P)$. Note that $Z_{1|\overline{A}} = Z$ is not possible. Otherwise, since P satisfies Δ_{u_1}, some other Z_1' exists such that $Z_1' \in AS(P \cup F)$, which is against our assumption that $Z \cup A' \notin AS(P \cup F)$ for all A'. For $Z_{1|\overline{A}} \subset Z$, if $Z_1 \cap A \subset A'$, then since $\langle Z_1 \cup A', Z \cup A' \rangle \notin SE(P)$, actually $Z \cup A' \in AS(P \cup (Z_1 \cup A'))$ which is against our assumption. So $Z_1 \cap A = A'$ and $\langle Z_{1|\overline{A}}, Z \rangle \in UE_A(P)$. However, since $Z \in AS(Q \cup F_{|\overline{A}})$, $\langle Z_{1|\overline{A}}, Z \rangle$ cannot be in $UE^{\overline{A}}(Q)$ which contradicts (5).

Since it is possible to construct a DLP from a given set of SE-models (or UE-models), cf. Section 3.1 in (Eiter et al. 2013), some Q satisfying (5) can always be found. ☐

We can infer from Proposition 5 and Theorem 6 that (5) is a characterization for uniform A-simplification.

Before concluding the section, note that the simplification in Example 1 can also be achieved by grounders, since the existence of the fact b guides the grounder to delete the first rule. However grounders cannot detect such possible simplifications when they are involved in a guess or disjunction.

Example 7. *Consider the program P_1 consisting of rules*

$$x \leftarrow not\ y. \qquad y \leftarrow not\ x. \qquad a \leftarrow c, not\ x, not\ y.$$

Similar to Example 1, the body of the third rule can never be satisfied no matter what truth values x or y have. As for the program P_2 consisting of rules

$$x \vee y. \qquad a \leftarrow x, c. \qquad a \leftarrow y, c.$$

since one of x or y will be true, deriving a actually only depends on whether or not c is true. In both of these programs, the grounders do not apply simplifications, while actually x and y can be seen as irrelevant; indeed these programs are uniform $\{x, y\}$-simplifiable.

3.2 Strong Simplification

In the previous section we introduced a simplification notion in the spirit of uniform equivalence. This brings the question of how the notion could be generalized to capture the spirit of strong equivalence. However an immediate generalization is not easily possible, since the atoms A would also have to be removed from the context program R. In order to avoid having too many assumptions on how some operator removing a set A of atoms should act on R, instead we take a view on R similar to how the facts F behave in the uniform simplification. There the atoms to be removed are rule-wise separated from the atoms that remain, since they are all facts. Thus projecting away A from F does not interfere with rules that contain atoms that remain. In order to have a similar behavior in the strong simplification setting, we fix R to have two separate parts, one only consists of the atoms in A and the other only consists of $\mathcal{U} \setminus A$.

Definition 3. *A program R over $\mathcal{U}$ is called* A-separated*, if $R = R_1 \cup R_2$ for programs R_1 and R_2 that are defined over $\mathcal{U} \setminus A$ and A, respectively.*

Due to the A-separation, any *reasonable* operator that operates on R to remove the atoms in A would remove part R_2 and keep R_1 as it is. Thus, we can simply denote the resulting program for removing atoms in A from R to be $R_{|\overline{A}}$. Furthermore, as we will discuss below, we require each rule in R to contain at least one atom in the head, thus disallowing constraints of form $\bot$ in the rule head.[5]

We now define a simplification notion for removing a set of atoms in the spirit of strong equivalence.

Definition 4. *Given $A \subseteq \mathcal{U}$ and a program P (over $\mathcal{U}$), a program Q (over $\mathcal{U} \setminus A$) is* a strong A-simplification of P *if for any program R over $\mathcal{U}$ that is A-separated, we have*

$$AS(P \cup R)_{|\overline{A}} = AS(Q \cup R_{|\overline{A}}) \quad (6)$$

We say that P is strong A-simplifiable if there is a program Q such that (6) holds.

Example 8. *Let program P consist of rules*

$$p \leftarrow q, r. \qquad q. \qquad r.$$

The atom p will appear in the answer sets of $P \cup R$ for any q, r-separated R. Thus, the program Q consisting only of the fact $\{p\}$ is a strong $\{q, r\}$-simplification of P.

In the above example, we can observe the need for disallowing the use of $\bot$ in R; for $R = \{\bot \leftarrow q.\}$, $P \cup R$ would have no answer sets, while $R_{|\overline{qr}} = \emptyset$ gets rid of this triggering constraint, which makes it impossible to find a strong simplification satisfying (6) even though by intuition the facts q, r are expected to be removable.

Proposition 2 can be lifted to the strong case, where Z and Z' in (4) are A-separated programs. Thus we see that not every program might be strong A-simplifiable.

Example 9. *Let program P consist of rules*

$$a \leftarrow p. \quad b \leftarrow q. \quad p \leftarrow not\ q. \quad q \leftarrow not\ p.$$

Consider $A = \{p, q, a\}$. We have $AS(P \cup \emptyset) = \{\{p, a\}, \{q, b\}\}$ but for $R = \{p.; q \leftarrow p.\}$ which is A-separated, we get $AS(P \cup R) = \{p, q, a, b\}$. Even though $R_{\overline{A}} = \emptyset$, for $\{p, a\} \in AS(P \cup \emptyset)$, none of Y's with $Y_{\overline{A}} = \emptyset = \{p, a\}_{|\overline{A}}$ is an answer set of $P \cup R$, thus P does not satisfy (4) *for when Z and Z' are A-separated programs.*

We can then obtain the necessary conditions for strong simplifiabiliy. Moreover, the use of UE_A for uniform simplification, which projects the UE-models of P importance, already hints on how to achieve the strong simplification, which is via projecting the SE-models of P. Thus we show below the necessary and sufficient conditions for a program to be strong A-simplifiable.[6]

[5]Constraints can still be described using $f \leftarrow B, not f$.

[6]In the rest of the paper, for full proofs see https://www.dbai.tuwien.ac.at/user/saribat/pub/kr23_supp.pdf.

Theorem 7. *There exists a strong A-simplification of P iff P satisfies the following*

Δ_{s_1}: *$\langle Y, Y \rangle \in SE(P)$ implies $A \subseteq Y$.*

Δ_{s_2}: *For any $\langle X, Y \rangle \in SE(P)$, $X_{|\overline{A}} = Y_{|\overline{A}}$ implies $X = Y$.*

Δ_{s_3}: *$\langle X, Y \rangle \in SE(P)$ implies $\langle X \cup (Y \cap A), Y \rangle \in SE(P)$.*

Proof (Sketch). ($\Rightarrow$) Let Q be a strong A-simplification of P, but some $\langle Y, Y \rangle$ violates Δ_{s_1}. Then $\langle Y \cup A, Y \cup A \rangle \in SE(P)$ needs to hold. Since otherwise we would have $AS(P \cup (Y \cup A)) = \emptyset$ while $Y \in AS(P \cup Y)$. Then by Proposition 1, $\langle Y, Y \cup A \rangle \in SE(P)$ holds. Now consider $R = (Y \cup A) \cup \{f \leftarrow y, not\ f \mid y \notin Y \cup A\}$ and $R' = Y \cup \{f \leftarrow y, not\ f \mid y \notin Y \cup A\} \cup \{y \leftarrow not\ y \mid y \in (Y \cup A) \setminus Y\}$, where f is an auxiliary atom in $\overline{A}$. We have $\langle Y \cup A, Y \cup A \rangle \in AS(P \cup R)$, thus $\langle Y_{|\overline{A}}, Y_{|\overline{A}} \rangle \in AS(Q \cup R_{|\overline{A}})$. Both R and R' are A-separated with $R_{|\overline{A}} = R'_{|\overline{A}}$, but $AS(P \cup R') = \emptyset$, contradicting (6). A similar proof can be obtained for Δ_{s_2} and Δ_{s_3}, where through R we target the answer set related with the violating SE-model, while another R', that agrees on the projection with R, obtains no answer set.

($\Leftarrow$) We show that a program Q satisfying $SE(P)_{|\overline{A}} = SE^{\overline{A}}(Q)$ is a strong A-simplification of P. □

Example 10 (Ex. 9 (ctd))**.** *For $A = \{p, q, a\}$, P does neither satisfy Δ_{s_1} ($\langle pa, pa \rangle \in SE(P)$) nor Δ_{s_2} ($\langle pab, paqb \rangle \in SE(P)$ with $\{pab\}_{|\overline{A}} = \{paqb\}_{|\overline{A}} = \{b\}$).*

Example 11 (Ex. 8 (ctd))**.** *Δ_{s_1}, Δ_{s_2} and Δ_{s_3} are satisfied by the only SE-model $\langle pqr, pqr \rangle$ of P and $SE^{\overline{A}}(Q) = \langle p, p \rangle = SE(P)_{|\overline{qr}}$*

The below result together with Theorem 7 then gives the characterization for strong simplification.

Proposition 8. *If Q is a strong A-simplification for P, then it satisfies $SE(P)_{|\overline{A}} = SE^{\overline{A}}(Q)$.*

Observe that the conditions $\Delta_{s_1}, \Delta_{s_2}$ and Δ_{s_3} are quite restrictive on the SE-models of P. Satisfying all the conditions can occur only when the atoms in A are semantically behaving as facts. Though on the syntactic level this might have various representations, and might not be easily detected, which makes these results still of value.

Example 12. *Programs containing the fact $\{b.\}$ or the rules $\{a \leftarrow not\ b.; b \leftarrow not\ a.; \bot \leftarrow a.\}$ are strong $\{b\}$-simplifiable.*

3.3 Properties

We begin with the familiar observation.

Proposition 9. *If P is strong A-simplifiable, then it is uniform A-simplifiable.*

This can easily be seen through the conditions; Δ_{s_1} and Δ_{s_2} trivially satisfies Δ_{u_3} and Δ_{u_1}, respectively, and by Δ_{s_3}, we can infer that if there is some $\langle X, Y \rangle$ with $\langle Y, Y \rangle \in SE(P)$, where for each M with $X \subseteq M \subset Y$, $\langle M, Y \rangle \notin SE(P)$ (thus $\langle X \cup (Y \cap A), Y \rangle \notin SE(P)$), then for all X' with $X'_{|\overline{A}} = X_{|\overline{A}}$, $\langle Y, Y \rangle$ satisfies the requirement for Δ_{u_2}.

We take a look at the extreme cases of removing atoms from $\mathcal{U}$. If $A = \emptyset$, then all of Δ_{s_1}, Δ_{s_2} and Δ_{s_3} are trivially satisfied. Moreover, any SE-model of a program P satisfies $SE(P)_{|\overline{A}} = SE(P)$, and any non-total UE-model satisfies $X_{|\overline{A}} \subset Y_{|\overline{A}}$ and $X \cap A = Y \cap A$, thus $UE_A(P) = UE(P)$.

Proposition 10. *The following holds.*

- *Any program is strong (resp. uniform) $\emptyset$-simplifiable.*
- *Q is strong (resp. uniform) equivalent to P iff Q is a strong (resp. uniform) $\emptyset$-simplification of P.*

Though not every program has a strong (resp. uniform) $\mathcal{U}$-simplification, e.g., some SE-model $\langle Y, Y\rangle$ might easily have $\mathcal{U} \nsubseteq Y$ violating Δ_{s_1} (resp. might not imply existence of the SE-model $\langle Y \cup \mathcal{U}, Y \cup \mathcal{U}\rangle$ violating Δ_{u_3}).

We now show properties regarding closure under union.

Proposition 11. *If P_1 and P_2 are strong A-simplifiable, then $P_1 \cup P_2$ is strong A-simplifiable.*

By making use of the property $SE(P_1 \cup P_2) = SE(P_1) \cap SE(P_2)$ (Turner 2003), the above result is achieved easily.

Proposition 12. *If P_1 and P_2 are uniform $\{a\}$-simplifiable, then $P_1 \cup P_2$ is uniform $\{a\}$-simplifiable.*

This property holds due to the only relevant total-models being $\langle Y, Y\rangle$ and $\langle Y \cup \{a\}, Y \cup \{a\}\rangle$. It cannot be lifted for a set A of atoms with $|A| > 1$, as P_1 and P_2 might have total models that differ on the subsets of A while satisfying the simplifiability conditions.

Example 13. *Consider the programs P_1 with rules*

$$q \leftarrow a. \quad a \leftarrow b. \quad a \leftarrow not\ b.$$

and P_2 with a and b in P_1 flipped. We have $SE(P_1) = \{\langle qa, qa\rangle, \langle qa, qab\rangle\} \cup S$ and $SE(P_2) = \{\langle qb, qb\rangle, \langle qb, qab\rangle\} \cup S$, where $S = \{\langle \emptyset, qab\rangle, \langle q, qab\rangle, \langle qab, qab\rangle\}$. Both P_1 and P_2 $\{a, b\}$-simplifiable, but $P_1 \cup P_2$ is not, since e.g., $SE(P_1 \cup P_2) = S$ does not satisfy Δ_{u_1} for $\langle q, qab\rangle$.

4 Projecting Away the Irrelevant

In this section we show that by projecting the atoms in A from the given uniform (resp. strong) A-simplifiable program achieves the desired simplification.

Definition 5. *Given a rule $r : H(r) \leftarrow B(r)$, the projection of r onto $\overline{A}$, denoted by $r_{|\overline{A}}$, gives*

$$\begin{cases} \emptyset & \text{if } B^-(r) \cap A \neq \emptyset \text{ or } H(r) \cap A \neq \emptyset \\ H(r) \leftarrow B(r) \setminus A & \text{otherwise.} \end{cases}$$

The resulting program, denoted by $P_{|\overline{A}}$, is then $\bigcup_{r \in P} r_{|\overline{A}}$.

We begin with a simple observation over the SE models of P and their projection on $\overline{A}$.

Lemma 13. $SE^{\overline{A}}(P_{|\overline{A}}) \subseteq SE(P)_{|\overline{A}}$.

Now we move on to obtaining the desired simplification.

Theorem 14. *Let P be a strong A-simplifiable program. Then $P_{|\overline{A}}$ is a strong A-simplification of P.*

Proof (Sketch). As P satisfies the necessary and sufficient conditions, we prove that $P_{|\overline{A}}$ is a strong A-simplification by showing $SE(P)_{|\overline{A}} = SE^{\overline{A}}(P_{|\overline{A}})$. By Lemma 13, it remains to show $SE(P)_{|\overline{A}} \subseteq SE^{\overline{A}}(P_{|\overline{A}})$ by studying the rule $r \in P_{|\overline{A}}$ that might prevent it. Knowing that the corresponding rule $r' \in P$ satisfies $B^-(r') \cap A = \emptyset$ and $H(r') \cap A = \emptyset$, with the help of Δ_{s_1} and Δ_{s_2}, we reach a contradiction. □

In retrospect, in Example 8, we see that Q is $P_{|\overline{qr}}$. As expected, projecting away the facts from a program achieves a simplification over the reduced vocabulary.

Proposition 15. *Let P be a program and a be a fact. Then $P_{|\overline{a}}$ is a strong $\{a\}$-simplification of P.*

The above property can be extended to a set A of facts appearing in P, by iteratively eliminating the facts in A.

Moving on to the uniform case, the following holds.

Theorem 16. *Let P be a uniform A-simplifiable program. Then $P_{|\overline{A}}$ is a uniform A-simplification of P.*

Proof (Sketch). Knowing that P satisfies the necessary and sufficient conditions, we prove $UE_A(P) = UE^{\overline{A}}(P_{|\overline{A}})$. □

Example 14 (Ex. 7 (ctd))**.** *For $A = \{x, y\}$, $P_1|_{\overline{A}} = \emptyset$ and $P_2|_{\overline{A}} = \{a \leftarrow c.\}$ are uniform A-simplifications of P_1 and P_2.*

Proposition 15 and its extension to a set A of facts also applies for uniform simplifications. Note that grounders also do elimination of rules where the body contains a positive atom that does not appear in any rule head. Though if the original program gets extended, then the elimination need to be reconsidered, since the resulting program is actually not a simplification in our sense.

Example 15. *Grounders simplify the program $P = \{a \leftarrow c, b.\ ;\ c.\}$ to $P' = \{c.\}$ since b does not appear in any rule head. However this is not a uniform $\{b\}$-simplification, since $AS(P \cup \{b.\}) = \{c, a, b\}$ while $P' \cup \{\} = \{c\}$.*

5 Computational Complexity

We now turn to the complexity of deciding simplifiability and simplification testing. We assume familiarity with basic concepts of complexity theory. For comprehensive details we refer to (Papadimitriou 2003; Arora and Barak 2009).

Theorem 17. *Let P be a program over $\mathcal{U}$ and $A \subseteq \mathcal{U}$. Deciding whether P is uniform A-simplifiable is Π_3^P-complete.*

Proof. We first show Π_3^P membership by analysing the three conditions of Proposition 3 separately, and then show Π_3^P-hardness utilizing the effect we have seen in Example 5.

(Δ_{u_1}) Membership: We give Σ_3^P membership for the complementary problem: It suffices to guess an SE-interpretation $\langle X, Y\rangle$ and check whether $\langle X, Y\rangle \in SE(P)$ and $\forall Y' \supseteq X$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$ such that $\langle Y', Y'\rangle \in SE(P)$, $\exists M$ with $X \subseteq M \subset Y'$, $\langle M, Y'\rangle \in SE(P)$. Inspecting the quantifier structure of above condition together with the fact that SE-model checking is in P (Eiter, Fink, and Woltran 2007), yields the required membership.

(Δ_{u_2}) Membership: Again, we give Σ_3^P membership for the complementary problem: It suffices to guess interpretations X, Y, with $X \subset Y$ and check whether $\langle Y, Y\rangle \in SE(P)$, and (a) exists M with $X \subseteq M \subset Y$, $\langle M, Y\rangle \in SE(P)$, or (b) exists X' with $X_{|\overline{A}} = X'_{|\overline{A}}$, such that for all $\langle Y', Y'\rangle \in SE(P)$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$ such that $X' \subseteq Y'$ there is an M' with $X' \subseteq M' \subset Y'$, $\langle M', Y'\rangle \notin SE(P)$. Using the fact that SE-model checking is in P, we derive that (a) is in NP; for (b) we observe that the quantifier structure gives a Σ_3^P procedure; together with our overall guess for X, Y, it follows that the entire procedure remains in Σ_3^P.

(Δ_{u_3}) Membership: this condition is even easier to check; in fact it is in coNP.

Hardness: We reduce from $(3, \exists)$-QSAT, and let $\Phi = \exists U \forall V \exists W \phi$ with $\phi = \bigwedge_{i=1}^n (l_{i,1} \vee l_{i,2} \vee l_{i_3})$. W.l.o.g. we assume that U, V, W are all non-empty and that each clause $l_{i,1} \vee l_{i,2} \vee l_{i_3}$ contains at least one literal over $V \cup W$. We use copies of atoms, e.g., $\widetilde{U} = \{\widetilde{u} \mid u \in U\}$. Given Φ, we set $\overline{A} = U \cup \widetilde{U}$ and construct a program P_Φ over atoms $\mathcal{U} = U \cup \widetilde{U} \cup V \cup \widetilde{V} \cup W \cup \widetilde{W} \cup \{abcd\}$.

Before constructing P_Φ we give its intended SE-models. For the sake of presentation we sometimes write unions of sets as strings, i.e. we omit the $\cup$-symbols. We also use the following abbreviation for any set I of atoms over $U \cup V \cup W$: $I_U = (U \cap I) \cup (\widetilde{U} \setminus \widetilde{I})$; accordingly we use I_V and I_W. Finally we use $I_U^+ = I_U V \widetilde{V} W \widetilde{W} abcd$. Our program will satisfy

$$
\begin{aligned}
SE(P_\Phi) = & \{\langle I_U^+, I_U^+\rangle \mid I \subseteq U\} \cup \\
& \{\langle I_U, I_U^+\rangle \mid I \subseteq U\} \cup \\
& \{\langle I_U I_V W \widetilde{W} abc, I_U^+\rangle \mid I \subseteq UV\} \cup \\
& \{\langle I_U I_V W \widetilde{W} abc, I_U I_V W \widetilde{W} abc\rangle \mid I \subseteq UV\} \cup \\
& \langle I_U I_V W \widetilde{W} abd, I_U^+\rangle \mid I \subseteq UV\} \cup \\
& \langle I_U I_V W \widetilde{W} abd, I_U I_V W \widetilde{W} abd\rangle \mid I \subseteq UV\} \cup \\
& \{\langle I_U I_V W \widetilde{W} a, I_U^+\rangle \mid I \subseteq UV\} \cup \\
& \{\langle I_U I_V W \widetilde{W} a, I_U I_V W \widetilde{W} abc\rangle \mid I \subseteq UV\} \cup \\
& \{\langle I_U I_V I_W b, I_U^+\rangle \mid I \subseteq UVW \models \phi\} \cup \\
& \{\langle I_U I_V I_W b, I_U I_V W \widetilde{W} abd\rangle \mid I \subseteq UVW \models \phi\}
\end{aligned}
$$

Observe that independent of ϕ, Δ_{u_3} is always satisfied, due to the total models $\langle I_U^+, I_U^+\rangle$. Also note already at this point that for each $I \subseteq U$, the way the atoms $abcd$ are assigned follows the pattern of Example 5; however, we have such abc/abd pairs now for each $I \subseteq UV$.

We show that Φ is true iff P_Φ does not jointly satisfy Δ_{u_1} and Δ_{u_2}. By generalizing the reasoning from Example 5, we can reason that Φ is true iff P_Φ satisfies

(*) $\exists \langle X, Y\rangle \in SE(P_\Phi)$ with $X \subset Y$, $X_{|\overline{A}} = Y_{|\overline{A}}$,
$\forall Y' \supseteq X$ with $Y'_{|\overline{A}} = Y_{|\overline{A}}$ such that $\langle Y', Y'\rangle \in SE(P_\Phi)$,
$\exists M$ with $X \subseteq M \subset Y'$, $\langle M, Y'\rangle \in SE(P_\Phi)$.

Observation 1: the only candidates $\langle X, Y\rangle$ able to violate Condition (*) are those of form $\langle I_U, I_U^+\rangle$. (In fact, for $X = I_U I_V I_W b$, we have $Y' = I_U I_V W \widetilde{W} abd$, such that no M with $X \subseteq M \subset Y$ satisfies $\langle M, Y'\rangle \in SE(P_\Phi)$; for $I_U I_V W \widetilde{W} a$, we have $Y' = I_U I_V W \widetilde{W} abc$. For the remaining non-total SE-models $\langle X, Y\rangle$, we always have total SE-model $\langle X, X\rangle$.)

Observation 2: $\langle I_U, I_U^+\rangle$ violates Condition (*) exactly if there is $J \subseteq V$ such that for all $K \subseteq W$, IJK is not a model of ϕ (i.e., Φ is false under assignment I to U).

Hence, the entire condition (*) is satisfied exactly when we find an $\langle I_U, I_U^+\rangle$ that does not violate it, i.e., exactly if Φ is true. It remains to show that given Φ we can construct P_Φ with $SE(P_\Phi)$ as outlined above in polynomial-time. The following encoding does the job:

$$
\begin{aligned}
P_\Phi = & \{b \leftarrow c.\ a \leftarrow d.\ b \leftarrow not\ c.\ a \leftarrow not\ d.\} \cup \\
& \{c \vee d \leftarrow a, b.\ a \leftarrow b, c.\ b \leftarrow a, d.\ \} \cup \\
& \{u \vee \widetilde{u} \leftarrow .\ t \leftarrow u, \widetilde{u}.\ \widetilde{t} \leftarrow u, \widetilde{u}.\ \mid u, t \in U\} \cup \\
& \{v \vee \widetilde{v} \leftarrow a.\ c \leftarrow v, \widetilde{v}.\ d \leftarrow v, \widetilde{v}.\ \mid v \in V\} \cup \\
& \{w \leftarrow a.\ \widetilde{w} \leftarrow a.\ a \leftarrow b, w, \widetilde{w}.\ \mid w \in W\} \cup \\
& \{z \vee \widetilde{z} \leftarrow b.\ a \vee b \leftarrow z.\ a \vee b \leftarrow \widetilde{z}.\ \mid z \in V \cup W\} \cup \\
& \{z \leftarrow a, b, c, d.\ \widetilde{z} \leftarrow a, b, c, d.\ \mid z \in V \cup W\} \cup \\
& \{a \leftarrow \widetilde{l}_{i,1}, \widetilde{l}_{i,2}, \widetilde{l}_{i_3}.\ \mid 1 \leq i \leq n\}
\end{aligned}
$$

where $\widetilde{l}_{i,j}$ is given by atom a if $l_{i,j}$ is $\neg a$ and by $\widetilde{a}$ if $l_{i,j}$ is a positive literal a. □

Theorem 18. *Let P be a program over $\mathcal{U}$ and $A \subseteq \mathcal{U}$. Deciding whether P is strong A-simplifiable is coNP-complete.*

It is easy to see that violation of any Δ_{s_i} can be checked in NP, since SE-model checking is in P. Hardness can be shown for each Δ_{s_i} separately by reducing from SAT where the program contains a "violating" SE-model iff the formula reduced from is satisfiable.

For the remaining results we give only upper bounds. Trivial lower bounds are obtained by utilizing observations from Section 3.3 where we have seen that for $A = \emptyset$, the conditions for simplification are always satisfied and checking whether Q is a strong (resp. uniform) A-simplification of some P amounts to strong (resp. uniform) equivalence between P and Q. We recall that checking uniform equivalence is Π_2^P-complete and deciding strong equivalence is coNP-complete (Eiter, Fink, and Woltran 2007). We anticipate that matching lower bounds can be obtained but leave this for future work.

Theorem 19. *Given $A \subseteq \mathcal{U}$, P a program which is uniform A-simplifiable, and program Q, checking whether Q is a uniform A-simplification of P is in Π_3^P and Π_2^P-hard.*

Proof (Sketch). Making use of the characterising equality (5), the key observation is that checking $\langle X, Y\rangle \in UE_A(P)$ is contained in Π_2^P which relies on the known result that UE-model checking is in coNP (also note that testing $\langle X, Y\rangle \in UE^{\overline{A}}(Q)$ amounts to UE-model checking). By that the complementary problem is then shown to be in Σ_3^P by guessing an SE-interpretation $\langle X, Y\rangle$ and checking containment either in $UE_A(P)$ or in $UE^{\overline{A}}(Q)$ but not in both. □

By essentially the same arguments and the fact that SE-model checking is easier we obtain our final result.

Theorem 20. *Given $A \subseteq \mathcal{U}$, P a program which is strong A-simplifiable, and program Q, checking whether Q is a strong A-simplification of P is in Π_2^P and coNP-hard.*

6 Discussion on Forgetting

We refer to (Gonçalves, Knorr, and Leite 2016a; Delgrande 2017) for recent surveys on forgetting and just shortly summarize the notions needed here. Below are two of the properties considered in forgetting that are relevant for our purposes, where F is a class of forgetting operators and $\mathcal{C}$ a class of programs:

(SP) F satisfies *Strong Persistence* if, for each $f \in F$, $P \in \mathcal{C}$ and $A \subseteq \mathcal{U}$, we have $AS(f(P,A) \cup R) = AS(P \cup R)_{|\overline{A}}$ for all programs $R \in \mathcal{C}$ over $\overline{A}$.

(UP) F satisfies *Uniform Persistence* if, for each $f \in F$, $P \in \mathcal{C}$ and $A \subseteq \mathcal{U}$, we have $AS(f(P,A) \cup R) = AS(P \cup R)_{|\overline{A}}$ for all sets of facts $R \in \mathcal{C}$ over $\overline{A}$.

$f(P,A)$ denotes the result of forgetting about A from P. Strong persistence and uniform persistence are also considered for a particular forgetting instance $\langle P, A\rangle$, for $P \in \mathcal{C}$ and $A \subseteq \mathcal{U}$, denoted by **(SP)**$_{\langle P,A\rangle}$ and **(UP)**$_{\langle P,A\rangle}$, respectively (Gonçalves, Knorr, and Leite 2016b; Gonçalves et al. 2019). The following is then easy to observe, as strong (resp. uniform) simplification notions consider R to be over $\mathcal{U}$.

Proposition 21. *Let f be a forgetting operator. For a program P and a set A of atoms, if $f(P,A)$ is a strong (resp. uniform) A-simplification of P then f satisfies* **(SP)**$_{\langle P,A\rangle}$ *(resp.* **(UP)**$_{\langle P,A\rangle}$*).*

Though the other direction might not always hold.

Example 16 (Ex. 9 (ctd))**.** *From (Gonçalves, Knorr, and Leite 2016b) we know that $A = \{p,q,a\}$ is forgettable from P while preserving* **(SP)**. *The program below, $f(P,A)$, would be the result of forgetting A.*

$$b \leftarrow not\ not\ b.$$

Though for $R = \{p; q \leftarrow p\}$, we have $AS(P \cup R) = \{\{p,q,a,b\}\}$, while $AS(f(P,A)) \cup \{\}) = \{\{\},\{b\}\}$. Thus $f(P,A)$ is not a strong A-simplification of P.

Example 17 (Ex. 2 (ctd))**.** *Applying the syntactic* **(UP)** *achieving operator f_u (Gonçalves et al. 2021) to program P to forget $\{b\}$ would give us $f_u(P,\{b\})$ as*

$$a \leftarrow not\ not\ a.$$

Though for $R = \{b\}$, we have $AS(P \cup \{b\}) = \{\{b\}\}$, while $AS(f_u(P,\{b\}) \cup \{\}) = \{\{\},\{a\}\}$. Thus $f_u(P,\{b\})$ is not a uniform $\{b\}$-simplification of P.

(Gonçalves, Knorr, and Leite 2016b) introduces the following criterion Ω to characterize the instances for which an operator achieving **(SP)**$_{\langle P,A\rangle}$ is impossible.

Definition 6. *(Gonçalves, Knorr, and Leite 2016b) Let P be a program over $\mathcal{U}$ and $A \subseteq \mathcal{U}$. An instance $\langle P, A\rangle$ satisfies criterion Ω if there exists $Y \subseteq \mathcal{U} \setminus A$ such that the set of sets*

$$\mathcal{R}^{Y}_{\langle P,A\rangle} = \{R^{Y,A'}_{\langle P,A\rangle} \mid A' \in Rel^{Y}_{\langle P,A\rangle}\}$$

is non-empty and has no least element, where

$$\begin{aligned} R^{Y,A'}_{\langle P,A\rangle} &= \{X \setminus A \mid \langle X, Y \cup A'\rangle \in SE(P)\} \\ Rel^{Y}_{\langle P,A\rangle} &= \{A' \subseteq A \mid \langle Y \cup A', Y \cup A'\rangle \in SE(P) \text{ and} \\ &\quad \nexists A'' \subset A' \text{ s.t. } \langle Y \cup A'', Y \cup A'\rangle \in SE(P)\}. \end{aligned}$$

It is not possible to forget about A from P while satisfying strong persistence exactly when $\langle P, A\rangle$ satisfies criterion Ω. Thus from Proposition 21 we get the following.

Corollary 22. *If P is a strong A-simplifiable, then $\langle P, A\rangle$ does not satisfy Ω.*

As expected, Ω is not sufficient to determine strong A-simplifiability, since the definitions of $R^{Y,A'}_{\langle P,A\rangle}$ and $Rel^{Y}_{\langle P,A\rangle}$ are too weak for checking the simplifiability conditions. Below are observations on this for conditions Δ_{s_2} and Δ_{s_3}.

Proposition 23. *The following holds.*

- *For any $Y \subseteq \mathcal{U} \setminus A$, $A' \subseteq A$, $\langle Y \cup A', Y \cup A'\rangle \in SE(P)$ implies $\{A'\} \in Rel^{Y}_{\langle P,A\rangle}$ iff P satisfies Δ_{s_2} for A.*
- *For any $Y \subseteq \mathcal{U} \setminus A$, $X \subseteq \mathcal{U}$, and $A' \subseteq A$, $\{X \setminus A\} \in R^{Y,A'}_{\langle P,A\rangle}$ implies $\langle X \cup A', Y \cup A'\rangle \in SE(P)$ iff P satisfies Δ_{s_3} for A.*

It was shown that it is always possible to forget from stratified programs while preserving **(SP)**, thus also **(UP)** (Gonçalves et al. 2021). Investigating the properties of stratified programs for our notions remains as future work.

7 Conclusion

We introduced a novel equivalence notion to capture the irrelevancy of atoms so that they can be disregarded from the original program and also the context program, and that the simplified program can reason over the reduced vocabulary while ensuring that the semantics of the original program is preserved w.r.t. the modified signature. We provided the necessary and sufficient conditions for a program to be strong/uniform simplifiable for a set of atoms, and showed that the simplifications can actually be achieved by projecting away those atoms from the program.

To continue this line of research, finding ways to detect the atoms that can be projected away would be an interesting next step. The simplifiability conditions hint on such potential atoms, and we already observed some examples of program structures where also a set of non-fact atoms can be simplified. We are also planning to consider the relativized case, which restricts the alphabet of the context programs. This would give us the ability to capture **(SP)**/**(UP)**. Furthermore, considering a novel notion of relativization, that restricts the alphabet of F in uniform simplification to a set of sets, would bring us closer to applications, such as the planning problem we mentioned in the introduction. For example, for the program $\{a \leftarrow b, c\ ;\ a \leftarrow b, d\ ;\ b.\}$ when the facts are restricted to $F \in \{\{c\},\{d\}\}$, it is expected that the program can be simplified by projecting away c and d, since it is not of relevant which of them occur in R. Lastly, in this work, we relied on the key property of DLPs. Investigating conditions needed for more general but also more restricted classes of programs remains as future work.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) projects T-1315, P32830, and by the Vienna Science and Technology Fund (WWTF) under grant ICT19-065. We thank the anonymous reviewers for their valuable feedback.

References

Arora, S., and Barak, B. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

Brass, S., and Dix, J. 1997. Characterizations of the disjunctive stable semantics by partial evaluation. *Journal of Logic Programming* 32(3):207–228.

Delgrande, J. P. 2017. A knowledge level account of forgetting. *Journal of Artificial Intelligence Research* 60:1165–1213.

Eiter, T., and Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, volume 2916 of *Lecture Notes in Computer Science*, 224–238. Springer.

Eiter, T.; Fink, M.; Tompits, H.; and Woltran, S. 2004. Simplifying logic programs under uniform and strong equivalence. In Lifschitz, V., and Niemelä, I., eds., *Logic Programming and Nonmonotonic Reasoning, 7th International Conference, LPNMR 2004, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings*, volume 2923 of *Lecture Notes in Computer Science*, 87–99. Springer.

Eiter, T.; Fink, M.; Pührer, J.; Tompits, H.; and Woltran, S. 2013. Model-based recasting in answer-set programming. *Journal of Applied Non-Classical Logics* 23(1-2):75–104.

Eiter, T.; Fink, M.; and Woltran, S. 2007. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic* 8(3):17.

Eiter, T.; Tompits, H.; and Woltran, S. 2005. On solution correspondences in answer-set programming. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, 97–102. Professional Book Center.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2008. Advanced preprocessing for answer set solving. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 15–19. IOS Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3):365–385.

Gonçalves, R.; Janhunen, T.; Knorr, M.; Leite, J.; and Woltran, S. 2019. Forgetting in modular answer set programming. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2843–2850. AAAI Press.

Gonçalves, R.; Janhunen, T.; Knorr, M.; and Leite, J. 2021. On syntactic forgetting under uniform equivalence. In Faber, W.; Friedrich, G.; Gebser, M.; and Morak, M., eds., *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12678 of *Lecture Notes in Computer Science*, 297–312. Springer.

Gonçalves, R.; Knorr, M.; and Leite, J. 2016a. The ultimate guide to forgetting in answer set programming. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, 135–144. AAAI Press.

Gonçalves, R.; Knorr, M.; and Leite, J. 2016b. You can't always forget what you want: On the limits of forgetting in answer set programming. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 957–965. IOS Press.

Janhunen, T.; Niemelä, I.; Seipel, D.; Simons, P.; and You, J.-H. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic* 7(1):1–37.

Leite, J. 2017. A bird's-eye view of forgetting in answer-set programming. In Balduccini, M., and Janhunen, T., eds., *Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings*, volume 10377 of *Lecture Notes in Computer Science*, 10–22. Springer.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2(4):526–541.

Maher, M. J. 1986. Equivalences of logic programs. In Shapiro, E., ed., *Third International Conference on Logic Programming*, 410–424. Berlin, Heidelberg: Springer Berlin Heidelberg.

Osorio, M.; Navarro, J. A.; and Arrazola, J. 2002. Equivalence in answer set programming. In Pettorossi, A., ed., *Logic Based Program Synthesis and Transformation*, 57–75. Springer Berlin Heidelberg.

Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

Pearce, D., and Valverde, A. 2004. Synonymous theories in answer set programming and equilibrium logic. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, 388–392. IOS Press.

Pearce, D. 2004. Simplifying logic programs under answer set semantics. In Demoen, B., and Lifschitz, V., eds., *Logic Programming, 20th International Conference, ICLP*

2004, Saint-Malo, France, September 6-10, 2004, Proceedings, volume 3132 of *Lecture Notes in Computer Science*, 210–224. Springer.

Sagiv, Y. 1987. Optimizing datalog programs. In *Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '87, 349–362. New York, NY, USA: ACM.

Saribatur, Z. G., and Eiter, T. 2021. Omission-based abstraction for answer set programs. *Theory and Practice of Logic Programming* 21(2):145–195.

Saribatur, Z. G.; Eiter, T.; and Schüller, P. 2021. Abstraction for non-ground answer set programs. *Artificial Intelligence* 300:103563.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Logic Programming and Nonmotonic Reasoning: 6th International Conference, LPNMR 2001 Vienna, Austria, September 17–19, 2001 Proceedings 6*, 81–92. Springer.

Turner, H. 2003. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming* 3(4–5):609–622.

Woltran, S. 2004. Characterizations for relativized notions of equivalence in answer set programming. In Alferes, J. J., and Leite, J. A., eds., *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings*, volume 3229 of *Lecture Notes in Computer Science*, 161–173. Springer.