On Equivalence and Inconsistency of Answer Set Programs with External Sources[*]

Christoph Redl
Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
redl@kr.tuwien.ac.at

Abstract

HEX-programs extend of answer-set programs (ASP) with external sources. In previous work, notions of equivalence of ASP programs under extensions have been developed. Most well-known are *strong equivalence*, which is given for programs P and Q if $P \cup R$ and $Q \cup R$ have the same answer sets for arbitrary programs R, and *uniform equivalence*, which is given if this is guaranteed for sets R of facts. More fine-grained approaches exist, which restrict the set of atoms in the added program R. In this paper we provide a characterization of equivalence of HEX-programs. Since well-known ASP extensions (e.g. constraint ASP) amount to special cases of HEX, the results are interesting beyond the particular formalism. Based on this, we further characterize inconsistency of programs wrt. program extensions. We then discuss possible applications of the results for algorithms improvements.

1 Introduction

Answer-Set Programming (ASP) is a declarative programming paradigm based on nonmonotonic programs and a multi-model semantics (Gelfond and Lifschitz 1991). In previous works, characterizations of equivalence of answer set programs under programs extensions have been developed. That is, two programs P and Q are considered to be equivalent if $P \cup R$ and $Q \cup R$ have the same answer sets for all programs R of a certain type, which depends on the notion of equivalence. Most importantly, for *strongly equivalent* programs we have that $P \cup R$ and $Q \cup R$ have the same answer sets for any program R (Lifschitz, Pearce, and Valverde 2001), while *uniformly equivalent* programs guarantee this only if R is a set of facts (Eiter and Fink 2003). Later, these notions were extended to the non-ground case (Eiter et al. 2005a). A more fine-grained approach is the one by Woltran [2007], where R can contain rules other than facts, but the sets of atoms which can occur in rule heads and bodies are restricted.

HEX-programs are an extension of ASP with external sources such as description logic ontologies and Web resources. So-called external atoms pass information from the logic program (given by predicate extensions and constants), to an external source, which in turn returns values to the program. For instance, the external atom $\&synonym[car](X)$ might be used to find the synonyms X of car, e.g. $automobile$. Also recursive data exchange between the program and external sources is supported, which leads to high expressiveness of the formalism.

In this paper, we present a characterization of *equivalence* and of *inconsistency* of HEX-programs. Since well-known ASP extensions (such as programs with aggregates (Faber, Leone, and Pfeifer 2011) and constraint ASP (Gebser, Ostrowski, and Schaub 2009)) amount to special cases of HEX-programs, the results are interesting beyond the specific formalism. Our approach is based on a recent technique for inlining of external atoms and a generalization of results by Woltran [2007] from ordinary ASP programs to HEX-programs. We are able to show that equivalence can be characterized similarly to ordinary ASP progams, which is convenient; however, due to the support for external atoms and the use of the FLP-reduct (Faber, Leone, and Pfeifer 2011) instead of the GL-reduct (Gelfond and Lifschitz 1988)) in the semantics of HEX-programs, this result is not immediate. Afterwards, we derive also a characterization of inconsistency of a program wrt. program extensions, which we call *persistent inconsistency*. While the main results are decision criteria based on programs and their reducts, we further derive a criterion for checking persistent inconsistency based on unfounded sets; this is driven by possible applications. We then give an outlook of an intended application of the results to algorithmic improvements.

The structure of the paper is as follows:

- In Section 2 we recapitulate HEX-programs and an approach for external atom inlining.
- Based on this inlining approach, Section 3 characterizes equivalence of HEX-programs, which generalizes results by Woltran [2007]. The generalization of strong (Lifschitz, Pearce, and Valverde 2001) and uniform equivalence (Eiter and Fink 2003) correspond to special cases thereof.
- In Section 4 we present a characterization of *inconsistency of* HEX-*programs wrt. program extensions*, which we call *persistent inconsistency*. This characterization is derived from the previously presented notion of equivalence.
- Section 5 presents envisaged applications of the results.
- In Section 6 we discuss related work and conclude.

For space reasons, proofs are outsourced to the

[*]This research has been supported by the Austrian Science Fund (FWF) project P27730.

extended version at http://www.kr.tuwien.ac.at/research/projects/inthex/hexequivalence-ext.pdf.

2 Preliminaries

A ground atom a is of form $p(c_1, \dots, c_\ell)$ with predicate symbol p and constant symbols $c_1, \dots, c_\ell$ from a finite set $\mathcal{C}$, abbreviated as $p(\mathbf{c})$; we write $c \in \mathbf{c}$ if $c = c_i$ for some $1 \leq i \leq \ell$. An *assignment* Y over the (finite) set A of atoms is a set $Y \subseteq A$; here $a \in Y$ expresses that a is true, also denoted $Y \models a$, and $a \notin Y$ that a is false, also denoted $Y \not\models a$. For a *default-literal* $\mathrm{not}\, a$ over atom a we let $Y \models \mathrm{not}\, a$ if $Y \not\models a$ and $Y \not\models \mathrm{not}\, a$ otherwise. An assignment satisfies a set S of atoms, denoted $Y \models S$, if $Y \models a$ for some $a \in S$; it does not satisfy S, denoted $Y \not\models S$, otherwise.

HEX-Programs.. We briefly recall HEX-programs, which generalize (disjunctive) logic programs under the answer set semantics (Gelfond and Lifschitz 1991); for more details and background, see Eiter et al. [2005b] and Eiter et al. [2014].

Syntax. HEX-programs extend ASP programs by *external atoms*, which enable a bidirectional interaction between a program and external sources of computation. An *external atom* is of the form $\&g[\mathbf{p}](\mathbf{c})$, where $\mathbf{p} = p_1, \dots, p_k$ is a list of input parameters (predicate names or object constants), called *input list*, and $\mathbf{c} = c_1, \dots, c_l$ are constant output terms.

Definition 1. *A HEX-program P consists of rules*

$$a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m, \mathrm{not}\ b_{m+1}, \dots, \mathrm{not}\ b_n\ ,$$

where each a_i is an ordinary atom and each b_j is either an ordinary atom or an external atom.

For a rule r, its *head* is $H(r) = \{a_1, \dots, a_k\}$, its *body* $B(r) = \{b_1, \dots, b_m, \mathrm{not}\ b_{m+1}, \dots, \mathrm{not}\ b_n\}$, its *positive body* is $B^+(r) = \{b_1, \dots, b_m\}$ and its *negative body* is $B^-(r) = \{b_{m+1}, \dots, b_n\}$. For a program P we let $X(P) = \bigcup_{r \in P} X(r)$ for $X \in \{H, B, B^+, B^-\}$.

In this paper we discuss only variable-free programs.

Semantics. Assignments are over the set $A(P)$ of ordinary atoms that occur in the program P at hand. The semantics of an external atom $\&g[\mathbf{p}](\mathbf{c})$ wrt. an assignment Y is given by the value of a $1{+}k{+}l$-ary *two-valued (Boolean) oracle function* $f_{\&g}$ that is defined for all possible values of Y, $\mathbf{p}$ and $\mathbf{c}$. Thus, $\&g[\mathbf{p}](\mathbf{c})$ is true relative to Y iff $f_{\&g}(Y, \mathbf{p}, \mathbf{c}) = \mathbf{T}$, where the value of $f_{\&g}(Y, \mathbf{p}, \mathbf{c})$ depends only on atoms $p(\mathbf{c}) \in Y$ such that $p \in \mathbf{p}$ (i.e., only those atoms in Y influence the result, whose predicate occurs as input parameter). In practice, oracle functions are implemented as reasoner plugins in order to integrate external data or computation sources. Similarily to Gelfond and Lifschitz [1991], a rule r as by Definition 1 is true under Y, denoted $Y \models r$, if $Y \models h$ for some $h \in H(r)$ or $Y \not\models b$ for some $b \in B(r)$.

The answer sets of a HEX-program P are defined as follows. Let the *FLP-reduct* of P wrt. an assignment Y be the set $fP^Y = \{r \in P \mid Y \models b \text{ for all } b \in B(r)\}$ of all rules whose body is satisfied by Y (Faber, Leone, and Pfeifer 2011). Then

Definition 2. *An assignment Y is an answer set of a HEX-program P, if Y is a subset-minimal model of fP^Y.* [1]

[1]For ordinary Π, these are Gelfond & Lifschitz's answer sets.

We let $\mathcal{AS}(P)$ denote the set of all answer sets of P.

Example 1. *Consider the program $P = \{p \leftarrow \&id[p]()\}$, where $\&id[p]()$ is true iff p is true. Then P has the answer set $Y_1 = \emptyset$; indeed it is a subset-minimal model of $fP^{Y_1} = \emptyset$.*

External Source Inlining. We now recapitulate a rewriting which compiles HEX-programs into equivalent ordinary ASP programs (modulo auxiliary atoms); for details we refer to Redl [2017].

For an external atom e in a program P, we denote by $I_{e,P}$ the set of all ordinary atoms in P which are input to e, i.e., whose predicate occurs as a predicate parameter in e. Moreover, let $\mathcal{S}_{e,P} = \{A \mid A \subseteq I_{e,P}, A \models e\}$ be the set of all assignments over the input atoms of e in P which satisfy e (previously, such a characterization of an external atom was called a *complete family of support sets*). Note that the possible exponentiality of $\mathcal{S}_{e,P}$ is not relevant for our purposes as we use it only as a theoretical construct.

Definition 3 (External Atom Inlining)**.** *Let P be a HEX-program and e be an external atom which occurs only positively in P. We define:*

$$P_{[e]} = \{x_e \leftarrow S \cup \{\bar{a} \mid a \in I_{e,P} \setminus S\} \mid S \in \mathcal{S}_{e,P}\} \quad (1)$$

$$\cup \left\{ \begin{array}{l} \bar{a} \leftarrow \mathrm{not}\, a; \bar{a} \leftarrow x_e \\ a \vee \bar{a} \leftarrow \mathrm{not}\, \bar{x_e} \end{array} \;\middle|\; a \in I_{e,P} \right\} \quad (2)$$

$$\cup \{\bar{x_e} \leftarrow \mathrm{not}\, x_e\} \quad (3)$$

$$\cup\, P|_{e \to x_e}, \quad (4)$$

where $\bar{a}$ is a new atom for each a, x_e and $\bar{x_e}$ are new atoms for e, and $P|_{e\to x_e} = \bigcup_{r \in P} r|_{e \to x_e}$ where $r|_{e\to x_e}$ denotes rule r with every occurrence of e replaced by x_e.

Intuitively, atom x_e represents that the former external atom is true and $\bar{x_e}$ that it is false. Each rule in (1) represents one possibility to satisfy e. For an input atom a of e, the auxiliary atom $\bar{a}$ represents that a is false *or* that x_e (resp. e) is true, as formalized in (2), which amounts to a saturation encoding. The rule (3) enforces $\bar{x_e}$ to be true whenever x_e is false. and the rules (4) resemble the original program.

The following result states that external atoms can be inlined without changing the answer sets:

Proposition 1. *For all HEX-programs P, external atoms e in P, the answer sets of P are equivalent to $P_{[e]}$, modulo the atoms newly introduced in $P_{[e]}$.*

Obviously, multiple external atoms can be inlined by iterated application of the rewriting. For a program P and a set E of external atoms in P we denote by $P_{[E]}$ the program after all external atoms from E have been inlined.

3 Equivalence of HEX-Programs

In this section we present a notion for equivalence of HEX-programs P and Q under extensions by additional rules R. The possibly added rules are constrained such that their head and body atoms and input atoms to external atoms can only come from fixed sets.

We proceed as follows. In the first step, only the programs P and Q can be HEX-programs, but the added program R must be ordinary. This amounts to a generalization of the results by Woltran [2007] from ordinary ASP to HEX-programs.

Note that because of external atoms in P and Q, which can be nonmonotonic, and the use of the FLP- instead of the GL-reduct, it is not immediate that this generalization goes through. In the second step, we allow also the added program R to contain external atoms. For this purpose, we exploit the possibility to inline external atoms.

Generalizing Equivalence Results

In the following, for sets $\mathcal{H}$ and $\mathcal{B}$ of atoms we let $\mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle} = \{P \text{ is an ASP program} | H(P) \subseteq \mathcal{H}, B^+(P) \cup B^-(P) \subseteq \mathcal{B}\}$ be the set of ordinay programs whose head and body atoms come only from $\mathcal{H}$ and $\mathcal{B}$, respectively.

Ordinary ASP programs P and Q are called $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent, if the answer sets of $P \cup R$ and $Q \cup R$ are the same whenever the ordinary ASP program R uses only head atoms from $\mathcal{H}$ and body atoms from $\mathcal{B}$. We first lift this result to the case where P and Q are general HEX-programs which possibly contain external atoms; in the first step R remains an ordinary ASP program. Formally:

Definition 4. *HEX-programs P and Q are* equivalent wrt. a pair $\langle\mathcal{H},\mathcal{B}\rangle$ of sets of atoms*, or* $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent*, denoted $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, if $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$ for all $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$.*

Similarily, we write $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ if $\mathcal{AS}(P \cup R) \subseteq \mathcal{AS}(Q \cup R)$ for all $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$.

Towards a characterization of equivalence, one can first show that if there is a counterexample R for $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, i.e., an $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$ such that $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$, then there is also a 'simple' counterexample in form of a positive program $R' \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$.

Proposition 2. *Let P and Q be HEX-programs, R be an ordinary ASP program, and Y be an interpretation s.t. $Y \in \mathcal{AS}(P \cup R)$ but $Y \notin \mathcal{AS}(Q \cup R)$. Then there is also a positive ordinary ASP program R' such that $Y \in \mathcal{AS}(P \cup R')$ but $Y \notin \mathcal{AS}(Q \cup R')$ and $B(R') \subseteq B(R)$ and $H(R') \subseteq H(R)$.*

Next, we show that the concepts on equivalence generalize from ordinary ASP to HEX-programs. In the following, for an interpretation Y and a set of atoms A we write $Y|_A$ for $Y \cap A$. Moreover, for sets of atoms X, Y we write $X \leq^{\mathcal{B}}_{\mathcal{H}} Y$ if $X|_{\mathcal{H}} \subseteq Y|_{\mathcal{H}}$ and $X|_{\mathcal{B}} \supseteq Y|_{\mathcal{B}}$. Intuitively, if $X \leq^{\mathcal{B}}_{\mathcal{H}} Y$ then Y satisfies all positive programs from $\mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$ which are also satisfied by X because it satisfies no fewer heads and no more bodies than X. We write $X <^{\mathcal{B}}_{\mathcal{H}} Y$ if $X \leq^{\mathcal{B}}_{\mathcal{H}} Y$ and $X|_{\mathcal{H}\cup\mathcal{B}} \neq Y|_{\mathcal{H}\cup\mathcal{B}}$.

We use the following concept for witnessing that $\mathcal{AS}(P \cup R) \subseteq \mathcal{AS}(Q \cup R)$ does *not* hold.

Definition 5. *A witness for $P \not\subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ is a pair (X, Y) of interpretations with $X \subseteq Y$ such that*[2]*:*

(i) $Y \models P$ and for each $Y' \subsetneq Y$ with $Y' \models fP^Y$ we have $Y'|_{\mathcal{H}} \subsetneq Y|_{\mathcal{H}}$; and

(ii) if $Y \models Q$ then $X \subsetneq Y$, $X \models fQ^Y$ and for all X' with $X \leq^{\mathcal{B}}_{\mathcal{H}} X' \subsetneq Y$ we have $X' \not\models fP^Y$.

[2]Note that Woltran [2007] called this *a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$*, but since it is actually a witness for the violation of the containment, we change the terminology.

The idea is that a witness represents a counterexample to the containment, where X characterizes a program R such that Y is an answer set of $P \cup R$ but not of $Q \cup R$. One can show that the existence of a witness and the violation of the containment are equivalent.

Because some steps in the according considerations for ordinary ASP depend on the fact that GL-reducts of programs wrt. assignments are positive programs (cf. $\leq^{\mathcal{B}}_{\mathcal{H}}$), it is an interesting result that the following propositions still hold in its generalized form. Because we use FLP-reducts instead, and P and Q might even contain nonmonotonic external atoms, the results do not automatically carry over. However, a closer analysis reveals that the property of being a positive program is only required from reducts of R but not of P or Q. Since we restricted R to ordinary ASP programs for now, and Proposition 2 allows us to further restrict it to positive programs, the use of the FLP-reduct does not harm: if R is positive from the beginning, then also its FLP-reduct (wrt. any assignment) is positive.

Proposition 3. *For HEX-programs P and Q and sets $\mathcal{H}$ and $\mathcal{B}$ of atoms, there is a program $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$ with $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$ iff there is a witness for $P \not\subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$.*

While witnesses compare the sets of answer sets of two programs directly, the next concept of $\langle\mathcal{H},\mathcal{B}\rangle$-models can be used to characterize a single program. In the following, for two sets of atoms $\mathcal{H}$ and $\mathcal{B}$, a pair (X, Y) of interpretations is called $\leq^{\mathcal{B}}_{\mathcal{H}}$-maximal for P if $X \models fP^Y$ and for all X' with $X <^{\mathcal{B}}_{\mathcal{H}} X' \subsetneq Y$ we have $X' \not\models fP^Y$.

Definition 6. *Given sets $\mathcal{H}$, $\mathcal{B}$ of atoms, a pair (X, Y) of interpretations is an $\langle\mathcal{H},\mathcal{B}\rangle$-model of a program P if*

(i) $Y \models P$ and for each $Y' \subsetneq Y$ with $Y' \models fP^Y$ we have $Y'|_{\mathcal{H}} \subsetneq Y|_{\mathcal{H}}$; and

(ii) if $X \subsetneq Y$ then there exists an $X' \subsetneq Y$ with $X'|_{\mathcal{H}\cup\mathcal{B}} = X$ such that (X', Y) is $\leq^{\mathcal{B}}_{\mathcal{H}}$-maximal for P.

One can show that $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence of two programs can be reduced to a comparison of their $\langle\mathcal{H},\mathcal{B}\rangle$-models. We denote the set of all $\langle\mathcal{H},\mathcal{B}\rangle$-models of a program P by $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$.

Proposition 4. *For sets $\mathcal{H}$ and $\mathcal{B}$ of atoms and HEX-programs P and Q, we have $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ iff $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) = \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$.*

We demonstrate the lifted results using two examples.

Example 2. *Consider the programs $P = \{a \leftarrow \&neg[b](); \ b \leftarrow \&neg[a](); \ a \leftarrow b\}$ and $Q = \{a \vee b; \ a \leftarrow b\}$ where $\&neg[x]()$ evaluates to true whenever x is false and to true otherwise.*

For $\mathcal{H} = \{a, b\}$ and $\mathcal{B} = \{b\}$ we have that $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) = \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q) = \{(\{a\}, \{a\}), (\{a\}, \{a, b\}), (\{a, b\}, \{a, b\})\}$, and thus the programs are $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent. The most interesting candidate which fails to be an $\langle\mathcal{H},\mathcal{B}\rangle$-model of either progam is $(\emptyset, \{a, b\})$. For P we have that $fP^{\{a,b\}} = \{a \leftarrow b\}$, of which $\emptyset$ is a model, but for $\{a\}$ we have $\emptyset \leq^{\mathcal{B}}_{\mathcal{H}} \{a\} \subsetneq Y$ and $\{a\} \models fP^{\{a,b\}}$, thus $\emptyset$ is not $\leq^{\mathcal{B}}_{\mathcal{H}}$-maximal for P. For Q we have that $fQ^{\{a,b\}} = \{a \vee b; a \leftarrow b\}$, which is unsatisfied under $\emptyset$.

Example 3. *Consider the programs P and Q from Example 2 and $\mathcal{H} = \{a, b\}$ and $\mathcal{B} = \{a, b\}$. We have that $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) = \{(\{a\}, \{a\}), (\emptyset, \{a, b\}), (\{a\}, \{a, b\}), (\{a, b\}, \{a, b\})\}$. Note that $(\emptyset, \{a, b\})$ is now an $\langle\mathcal{H},\mathcal{B}\rangle$-model of P because $\emptyset$ is a model of $fP^{\{a,b\}} = \{a \leftarrow b\}$ and there is no X' with $\emptyset \leq^{\mathcal{B}}_{\mathcal{H}} X' \subsetneq Y$ with $X' \models fP^{\{a \leftarrow b\}}$ (because now $\emptyset \not\leq^{\mathcal{B}}_{\mathcal{H}} \{a\}$); thus $\emptyset$ is $\leq^{\mathcal{B}}_{\mathcal{H}}$-maximal for P. On the other hand, $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q) = \{(\{a\}, \{a\}), (\{a\}, \{a, b\}), (\{a, b\}, \{a, b\})\}$. That is, $(\emptyset, \{a, b\})$ is still not an $\langle\mathcal{H},\mathcal{B}\rangle$-model of Q because $\emptyset$ is not a model of $fQ^{\{a,b\}} = \{a \vee b;\ a \leftarrow b\}$. And thus the programs are not $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent.*

Indeed, for $R = \{b \leftarrow a\} \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$ we have that $Y = \{a, b\}$ is an answer set of $Q \cup R$ but not of $P \cup R$.

Adding General HEX-Programs

Until now we allow only the addition of ordinary ASP programs $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$. As a preparation for the addition of HEX-programs, we show that if programs P and Q are $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent, then sets $\mathcal{B}$ and $\mathcal{H}$ can be extended by atoms that do not appear in P and Q and the programs are still equivalent wrt. the expanded sets. Intuitively, this allows introducing auxiliary atoms without harming their equivalence. Afterwards, we extend the above results to the case where R is a general HEX-program.

Expanding Sets $\mathcal{B}$ and $\mathcal{H}$. If programs P and Q are $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent, then they are also $\langle\mathcal{H}',\mathcal{B}'\rangle$-equivalent whenever $\mathcal{H}' \setminus \mathcal{H}$ and $\mathcal{B}' \setminus \mathcal{B}$ contain only atoms which do not appear in P or Q. This is intuitively the case because such atoms cannot interfere with atoms which are already in the program.

Formally, one can show the following result:

Proposition 5. *For sets $\mathcal{H}$ and $\mathcal{B}$ of atoms, HEX-programs P and Q, and an atom a which does not occur in P or Q, the following holds:*

(i) $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ iff $P \equiv_{\langle\mathcal{H}\cup\{a\},\mathcal{B}\rangle} Q$; and

(ii) $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ iff $P \equiv_{\langle\mathcal{H},\mathcal{B}\cup\{a\}\rangle} Q$.

By iterative applications of this result we get further:

Corollary 1. *For sets $\mathcal{H}$ and $\mathcal{B}$ of atoms, programs P and Q, and an sets of atoms $\mathcal{H}'$ and $\mathcal{B}'$ which does not occur in P or Q, we have $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ iff $P \equiv_{\langle\mathcal{H}\cup\mathcal{H}',\mathcal{B}\cup\mathcal{B}'\rangle} Q$.*

Addition of General HEX-Programs. In the following, for sets $\mathcal{H}$, $\mathcal{B}$ of atoms we let $\mathcal{P}^e_{\langle\mathcal{H},\mathcal{B}\rangle}$ denote the set

$$\left\{ \text{HEX-program } P \;\middle|\; \begin{array}{l} H(P) \subseteq \mathcal{H}, B^+(P) \cup B^-(P) \subseteq \mathcal{B}, \\ \text{only } \mathcal{B} \text{ are input to external atoms} \end{array} \right\}$$

of general HEX-programs whose head atoms come only from $\mathcal{H}$, and whose body atoms and input atoms to external atoms come only from $\mathcal{B}$ (the latter restriction is necessary since these atoms appear in bodies of our rewriting in Lemma 1 below), respectively. We then extend Definition 4 as follows.

Definition 7. *HEX-programs P and Q are* e-equivalent wrt. a pair $\langle\mathcal{H},\mathcal{B}\rangle$ of sets of atoms, *or $\langle\mathcal{H},\mathcal{B}\rangle^e$-equivalent, denoted $P \equiv^e_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, if $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$ for all $R \in \mathcal{P}^e_{\langle\mathcal{H},\mathcal{B}\rangle}$.*

Towards a characterization of $\langle\mathcal{H},\mathcal{B}\rangle^e$-equivalence, we make use of external atom inlining as by Definition 3 without changing the answer sets of a program, cf. Proposition 1.

We start with a technical result which allows for renaming a predicate input parameter $p_i \in \mathbf{p}$ of an external atom $e = \&g[\mathbf{p}](\mathbf{c})$ in a program P to a new predicate q which does not occur in P. This allows us to rename predicates such that inlining does not introduce rules which derive atoms other than auxiliaries, which is advantageous in the following.

The idea of the renaming is to add auxiliary rules which define q such that its extension represents exactly the former atoms over p_i, i.e., each atom $p_i(\mathbf{d})$ is represented by $q(p_i, \mathbf{d})$. Then, external predicate $\&g$ is replaced by a new $\&g'$ whose semantics is adopted to this encoding of the input atoms.

For the formalization of this idea, let $\mathbf{p}|_{p_i \to q}$ be vector $\mathbf{p}$ after replacement of its i-th element p_i be q. Moreover, for an assignment A let $A^q = A \cup \{p_i(\mathbf{d}) \mid q(p_i, \mathbf{d}) \in A\}$ be the extended assignment which 'extracts' from each atom $q(p_i, \mathbf{d}) \in A$ the original atom $p_i(\mathbf{d})$. Then one can show:

Lemma 1. *For an external atom $e = \&g[\mathbf{p}](\mathbf{c})$ in program P, $p_i \in \mathbf{p}$, a new predicate q, let $e' = \&g'[\mathbf{p}|_{p_i \to q}](\mathbf{c})$ s.t. $f_{\&g'}(A, \mathbf{p}|_{p_i \to q}, \mathbf{c}) = f_{\&g}(A^q, \mathbf{p}, \mathbf{c})$ for all assignments A.*

For $P' = P|_{e \to e'} \cup \{q(p_i, \mathbf{d}) \leftarrow p_i(\mathbf{d}) \mid p_i(\mathbf{d}) \in A(P)\}$, $\mathcal{AS}(P)$ and $\mathcal{AS}(P')$ coincide, modulo atoms $q(\cdot)$.

We now come to the actual inlining. Observe that Definition 3 is *modular* in the sense that inlining external atoms E in a program P affects only the rules of P containing some of E and adds additional rules, but does not change the remaining rules. If $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$ denotes the symmetric difference between sets X and Y, one can formally show:

Lemma 2. *For a HEX-program P and a set of external atoms E in P, we have $P \triangle P_{[E]} = \{r \in P \mid$ none of E occur in $r\}$.*

This equips us to turn to our main goal of characterizing equivalence of HEX-programs. If programs P and Q are $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent, then $P \cup R$ and $Q \cup R$ have the same answer sets for all ordinary ASP-programs $R \in \mathcal{P}_{\langle\mathcal{H},\mathcal{B}\rangle}$. We will show that equivalence holds in fact even for HEX-programs $R \in \mathcal{P}^e_{\langle\mathcal{H},\mathcal{B}\rangle}$. To this end, assume that P and Q are $\langle\mathcal{H},\mathcal{B}\rangle$-equivalent for some $\mathcal{H}$ and $\mathcal{B}$ and let $R \in \mathcal{P}^e_{\langle\mathcal{H},\mathcal{B}\rangle}$.

We apply the following transformation to all external atoms E in $P \cup R$ and $Q \cup R$ which appear in the R part, but not in the P part or Q part[3]:

(1) rename their input parameters using Lemma 1; and

(2) subsequently inline them by applying Definition 3.

Let R' denote the result after applying the two steps to R. Note that neither of the two steps modifies P or Q: for (1) this is by construction of the modified program in Lemma 1, for (2) this follows from Lemma 2. As observable from Lemma 1 and Definition 3, head atoms $H(R')$ in R' come either from $H(R)$ or are newly introduced auxiliary atoms; the renaming as by Lemma 1 prohibits that $H(R')$ contains input atoms to external atoms in R. Body atoms $B(R')$ in R' come either from $B(R)$, from input atoms to external atoms

[3]If the same external atom e is shared between P and R resp. Q and R, the restriction of the inlining to the R part is still possible by standardizing external atoms in R apart from those in P and Q, e.g., by introducing a copy of the external predicate.

in R (see rules (2)), or are newly introduced auxiliary atoms. Since $R \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$, this implies that $H(R') \subseteq \mathcal{H} \cup \mathcal{H}'$ and $B(R') \subseteq \mathcal{B} \cup \mathcal{B}'$, where $\mathcal{H}'$ and $\mathcal{B}'$ are newly introduced auxiliary atoms. Since the auxiliary atoms do not occur in P and Q, by Corollary 1 they do not harm equivalence, i.e., $\langle \mathcal{H},\mathcal{B}\rangle$-equivalence implies $\langle \mathcal{H}\cup\mathcal{H}', \mathcal{B}\cup\mathcal{B}'\rangle$-equivalence. Thus, $\langle \mathcal{H},\mathcal{B}\rangle$-equivalence of P and Q implies that $P \cup R'$ and $Q \cup R'$ have the same answer sets.

The claim follows then from the observation that, due to Lemma 1 and soundness and completeness of inlining (cf. Proposition 1), $P \cup R$ and $Q \cup R$ have the same answer sets whenever $P \cup R'$ and $Q \cup R'$ have the same answer sets.

Example 4. *Consider the programs*

$$P = \{a \leftarrow \&neg[b](); \; b \leftarrow \&neg[a](); \; a \leftarrow b\}$$
$$Q = \{a \vee b; \; a \leftarrow b\}$$

and let $\mathcal{H} = \{a, c\}$ and $\mathcal{B} = \{b\}$. Note that $P \equiv_{\langle \mathcal{H},\mathcal{B}\rangle} Q$.[4]

Let $R = \{c \leftarrow \&neg[b]()\} \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$. Renaming the input predicate of $\&neg[b]()$ by step (1) yields the program $\{q(b) \leftarrow b; \; c \leftarrow \&neg'[q]()\}$. After step (2) we have:

$$R' = \{q(b) \leftarrow b; \; c \leftarrow x_e; \; x_e \leftarrow \bar{q(b)}; \; \bar{x_e} \leftarrow \text{not}\, x_e$$
$$\bar{q(b)} \leftarrow \text{not}\, q(b); \; \bar{q(b)} \leftarrow x_e; \; q(b) \vee \bar{q(b)} \leftarrow x_e\}$$

Here, rule $q(b) \leftarrow b$ comes from step (1), $c \leftarrow x_e$ represents the rule in R, and the remaining rules from inlining in step (2). Except for new auxiliary atoms, we have that $H(R')$ use only atoms from $\mathcal{H}$ and $B(R')$ only atoms from $B(R')$. One can check that $P \cup R'$ and $Q \cup R'$ have the same (unique) answer set $\{a, c, \bar{x_e}, \bar{q(b)}\}$, which corresponds to the (same) unique answer set $\{a, c\}$ of $P \cup R$ and $Q \cup R$, respectively.

One can then show that equivalence wrt. program extensions that contain external atoms is characterized by the same criterion as extensions with ordinary ASP programs only.

Proposition 6. *For sets $\mathcal{H}$ and $\mathcal{B}$ of atoms and* HEX-*programs P and Q, we have $P \equiv^{e}_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ iff $\sigma_{\langle \mathcal{H},\mathcal{B}\rangle}(P) = \sigma_{\langle \mathcal{H},\mathcal{B}\rangle}(Q)$.*

Finally, for the Herbrand base $HB_{\mathcal{C}}(P)$ of all atoms constructable from the predicates in P and the constants $\mathcal{C}$, strong equivalence (Lifschitz, Pearce, and Valverde 2001) corresponds to the special case of $\langle HB_{\mathcal{C}}(P), HB_{\mathcal{C}}(P)\rangle$-equivalence, and uniform equivalence (Eiter and Fink 2003) corresponds to $\langle HB_{\mathcal{C}}(P), \emptyset\rangle$-equivalence; this follows directly from definition of strong resp. uniform equivalence. One may instantiate the sets $\mathcal{H}$ and $\mathcal{B}$ directly in Definition 6 to get simplification of the conditions. However, this is akin to ordinary ASP and we thus refer to Woltran [2007].

4 Inconsistency of HEX-Programs

We turn now to inconsistency of HEX-programs. Similarly to equivalence, we want to characterize inconsistency wrt. program extensions. Akin to equivalence, sets $\mathcal{H}$ and $\mathcal{B}$ constrain the atoms that may be occur in rule heads, rule bodies and input atoms to external atoms of the added program, respectively. In contrast to equivalence, the criterion naturally concerns only a single program. However, we are still able to derive the criterion from the above results.

Deriving a Criterion for Inconsistency. We define:

Definition 8. *A* HEX-*program P is called* persistently inconsistent *wrt. sets of atoms $\mathcal{H}$ and $\mathcal{B}$, if $P \cup R$ is inconsistent for all $R \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$.*

Example 5. *The program $P = \{p \leftarrow \&neg[p]()\}$ is persistently inconsistent wrt. all $\mathcal{H}$ and $\mathcal{B}$ such that $p \notin \mathcal{H}$. This is because any model Y of P, and thus of $P \cup R$ for some $R \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$, must set p to true due to the rule $p \leftarrow \&neg[p]()$. However, $Y \setminus \{p\}$ is a model of $f(P \cup R)^{Y}$ if no rule in R derives p, hence Y is not a subset-minimal model of $f(P \cup R)^{Y}$.*

We start deriving a criterion by observing that a program $P_{\perp}$ is persistently inconsistent wrt. any $\mathcal{H}$ and $\mathcal{B}$ whenever it is classically inconsistent. Then $P_{\perp} \cup R$ does not even have classical models for any $R \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$, and thus it cannot have answer sets. For such a $P_{\perp}$, another program P is persistently inconsistent wrt. $\mathcal{H}$ and $\mathcal{B}$ iff it is $\langle \mathcal{H},\mathcal{B}\rangle^{e}$-equivalent to $P_{\perp}$; the latter can by Proposition 4 be checked by comparing their $\langle \mathcal{H},\mathcal{B}\rangle$-models. This allows us to derive the desired criterion in fact as a special case of the one for equivalence.

Classically inconsistent program do not have $\langle \mathcal{H},\mathcal{B}\rangle$-models due to violation of Property (i) of Definition 6. Therefore, checking for persistent inconsistency works by checking if P does not have $\langle \mathcal{H},\mathcal{B}\rangle$-models either. To this end, it is necessary that each classical model Y of P violates Property (i) of Definition 6, otherwise (Y, Y) (and possibly (X, Y) for some $X \subsetneq Y$) would be $\langle \mathcal{H},\mathcal{B}\rangle$-models of P. Formally:

Proposition 7. *Let P be a* HEX-*program. Then $P \cup R$ is inconsistent for all $R \in \mathcal{P}^{e}_{\langle \mathcal{H},\mathcal{B}\rangle}$ iff for each model Y of P there is an $Y' \subsetneq Y$ such that $Y' \models fP^{Y}$ and $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$.*

Example 6 (cont'd)**.** *For the program P from Example 5 we have for each classical model $Y \supseteq \{p\}$ that $Y' = Y \setminus \{p\}$ is a model of fP^{Y}, $Y' \subsetneq Y$ and $Y|_{\mathcal{H}} = Y'|_{\mathcal{H}}$.*

Example 7. *Consider the program $P = \{a \leftarrow \&aOrNotB[a, b](); \; \leftarrow a\}$. It is persistently inconsistent wrt. all $\mathcal{H}$ and $\mathcal{B}$ such that $b \notin \mathcal{H}$. This is the case because the rule $a \leftarrow \&aOrNotB[a, b]()$ derives a whenever b is false, which violates the constraint $\leftarrow a$.*

Formally, one can observe that we have $a \notin Y$ and $b \in Y$ for each classical model Y of P. But then $Y' = Y \setminus \{b\}$ is a model of fP^{Y}, $Y' \subsetneq Y$ and $Y|_{\mathcal{H}} = Y'|_{\mathcal{H}}$.

Applying the Criterion using Unfounded Sets. Proposition 7 formalizes a condition for deciding persistent inconsistency based on models of the program's reduct. However, practical implementations do usually not explicitly generate the reduct, but are often based on *unfounded sets* (Faber 2005). For a model Y of a program P, smaller models $Y' \subsetneq Y$ of the reduct fP^{Y} and unfounded sets of P wrt. Y correspond to each other one-by-one. This allows us to transform the above decision criterion such that it can be directly checked using unfounded sets.

We use unfounded sets for logic programs as introduced by Faber [2005] for programs with arbitrary aggregates.

[4]We know $P \equiv_{\langle \{a,b\},\{b\}\rangle} Q$ from Example 2, which implies $P \equiv_{\langle \{a\},\{b\}\rangle} Q$. As $c \notin A(P)$, $c \notin A(Q)$, Proposition 5 implies further $P \equiv_{\langle \{a,c\},\{b\}\rangle} Q$.

Definition 9 (Unfounded Set). *Given a program P and an assignment A, let U be any set of atoms appearing in P. Then, U is an* unfounded set for P wrt. A *if, for each $r \in P$ with $H(r) \cap U \neq \emptyset$, at least one of the following holds:*

(i) some literal of $B(r)$ is false wrt. A; or

(ii) some literal of $B(r)$ is false wrt. $A \setminus U$; or

(iii) some atom of $H(r) \setminus U$ is true wrt. A.

Lemma 3. *For a* HEX*-program P and a model A of P, a set of atoms U is an unfounded set of P wrt. A iff $A \setminus U \models fP^A$.*

By contraposition, the lemma implies that for a model A of P and a model $A' \subseteq A$ of fP^A we have that $A \setminus A'$ is an unfounded set of P wrt. A. This allows us to restate our decision criterion as follows:

Corollary 2. *For a* HEX*-program P, $P \cup R$ is inconsistent for all $R \in \mathcal{P}^e_{\langle \mathcal{H},\mathcal{B} \rangle}$ iff for each classical model Y of P there is a nonempty unfounded set U of P wrt. Y s.t. $U \cap Y \neq \emptyset$ and $U \cap \mathcal{H} = \emptyset$.*

Example 8 (cont'd). *For the program P from Example 7 we have that $U = \{b\}$ is an unfounded set of P wrt. any classical model Y of P; by assumption $b \notin \mathcal{H}$ we have $U \cap \mathcal{H} = \emptyset$.*

5 Applications

Generally, a characterization of equivalence paves the way for program transformations. Such transformations might be applied, for instance, for optimization purposes, or for compiling programs to a syntactically simplified form.

We want to discuss a specific use-case in more detail. The state-of-the-art evaluation approach for HEX-programs is based on *program splitting*. That is, the overall program is partitioned into components which are arranged in an *acyclic graph*. Then, beginning from the components without predecessors, each component is separately grounded and solved, and each answer set is added as facts to the successor components. The process is repeated in a recursive manner such that eventually the leaf components will yield the final answer sets; for details we refer to Eiter et al. [2015].

The main reason for program splitting is that *value invention* may lead to a grounding bottleneck if the program is evaluated as monolithic program. This is because the grounder needs to evaluate external atoms under all possible inputs in order to ensure that all possible outputs are respected in the grounding, as demonstrated by the following example.

Example 9. *Consider the program*

$$\begin{aligned} P = \{ & r_1\colon \mathit{in}(X) \vee \mathit{out}(X) \leftarrow \mathit{node}(X) \\ & r_2\colon \leftarrow \mathit{in}(X), \mathit{in}(Y), \mathit{edge}(X,Y) \\ & r_3\colon \mathit{size}(S) \leftarrow \&\mathit{count}[\mathit{in}](S) \} \end{aligned}$$

where facts over $\mathit{node}(\cdot)$ and $\mathit{edge}(\cdot)$ define a graph. Then r_1 and r_2 guess an independent set, and r_3 computes its size. The grounder must evaluate &count under all exponentially many possible extensions of in in order to instantiate rule r_3 for all relevant values of variable S.

In the previous example, program splitting allows for avoiding unnecessary evaluations. The program might be split into $P_1 = \{r_1, r_2\}$ and $P_2 = \{r_3\}$. Then the state-of-the-art algorithm would ground and solve P_1, which computes all independent sets, and for each of them P_2 is grounded and solved. Since the number of independent sets can be exponentially smaller than the set of all node selections, the grounding bottleneck can be avoided. However, program splitting has the disadvantage that nogoods learned from conflict-driven algorithms (Gebser, Kaufmann, and Schaub 2012) cannot effectively propagated through the whole program, but only within a component.

We envisage to use the results from Section 4 for deciding when a component remains inconsistent for all possible inputs, and propagating this information to predecessor components to eliminate irrelevant models earlier.

6 Discussion and Conclusion

We summarize and discuss related work, and give an outlook on possible future work.

Summary and Discussion of Related Work. We provided a characterization of equivalence of HEX-programs. The criteria generalize previous results for ordinary ASP by Woltran [2007]; this is a convenient result, but it is not immediate due to possibly nonmonotonic external atoms and the use of the FLP- instead of the GL-reduct. Strong (Lifschitz, Pearce, and Valverde 2001) and uniform equivalence (Eiter and Fink 2003) are special cases thereof and carry over as well. The work is also related to the one by Truszczyski [2010], who extended strong equivalence to propositional theories under FLP-semantics. However, $\langle \mathcal{H}, \mathcal{B} \rangle$-equivalence and external sources were not considered. A recent alternative notion of equivalence is *rule equivalence* (Bliem and Woltran 2016). Here, not the set of atoms which can occur in the added program is constrained, but the type of the rules. In particular, proper rules may be added, while the addition of facts is limited to certain atoms.

We note that the results are interesting beyond HEX-programs. Well-known ASP extensions, such as programs with aggregates (Faber, Leone, and Pfeifer 2011) or with specific external atoms such as constraint atoms (Gebser, Ostrowski, and Schaub 2009), amount to special cases of HEX, and thus the results are applicable in such cases.

Outlook. Future work includes the extension of the results to non-ground programs, cf. Eiter et al. [2005a]. This might be necessary in the context of an envisaged application of the results for improving HEX-program evaluation. In particular, we plan to exploit the results on inconsistency characterization for detecting inconsistent program components in an existing model-building framework based an program decomposition. Based on this, knowledge about the inconsistency shall be propagated into other program components.

As another possible starting point, alternative notions of equivalence, such as rule equivalence for HEX-programs, might be investigated. Moreover, currently we do not distinguish between body atoms and input atoms to external atoms when we define which programs are allows to be added. A more fine-grained approach which supports this distinction may allow for identifying programs as equivalent which are not equivalent wrt. to the current notion. Also allowing only external atoms with specific properties, such as monotonicity, may lead to more fine-grained criteria.

References

Bliem, B., and Woltran, S. 2016. Equivalence between answer-set programs under (partially) fixed input. In *FoIKS*, volume 9616 of *Lecture Notes in Computer Science*, 95–111. Springer.

Eiter, T., and Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In Palamidessi, C., ed., *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. 2005a. Strong and uniform equivalence in answer-set programming: Characterizations and complexity results for the non-ground case. In Veloso, M. M., and Kambhampati, S., eds., *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 695–700. AAAI Press / The MIT Press.

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005b. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In Kaelbling, L. P., and Saffiotti, A., eds., *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 90–96. Denver, USA: Professional Book Center.

Eiter, T.; Fink, M.; Krennwallner, T.; Redl, C.; and Schüller, P. 2014. Efficient HEX-program evaluation based on unfounded sets. *Journal of Artificial Intelligence Research* 49:269–321.

Eiter, T.; Fink, M.; Ianni, G.; Krennwallner, T.; Redl, C.; and Schüller, P. 2015. A model building framework for answer set programming with external computations. *Theory and Practice of Logic Programming*.

Faber, W.; Leone, N.; and Pfeifer, G. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence* 175(1):278–298.

Faber, W. 2005. Unfounded sets for disjunctive logic programs with arbitrary aggregates. In *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005), Diamante, Italy, September 5-8, 2005*, volume 3662, 40–52. Springer.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187-188:52–89.

Gebser, M.; Ostrowski, M.; and Schaub, T. 2009. Constraint answer set solving. In Hill, P., and Warren, D., eds., *Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09)*, volume 5649 of *Lecture Notes in Computer Science*, 235–249. Springer-Verlag.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In Kowalski, R., and Bowen, K., eds., *Logic Programming: Proceedings of the 5th International Conference and Symposium*, 1070–1080. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9(3–4):365–386.

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

Redl, C. 2017. Efficient evaluation of answer set programs with external sources using inlining. In *Proceedings of the Thirty-First AAAI Conference (AAAI 2017), February, 2017, San Francisco, California, USA*. AAAI Press. Accepted for publication.

Truszczyski, M. 2010. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence* 174(16):1285 – 1306.

Woltran, S. 2007. A common view on strong, uniform, and other notions of equivalence in answer-set programming. In Pearce, D.; Polleres, A.; Valverde, A.; and Woltran, S., eds., *Proceedings of the LPNMR'07 Workshop on Correspondence and Equivalence for Nonmonotonic Theories (CENT2007), Tempe, AZ, May 14, 2007*, volume 265 of *CEUR Workshop Proceedings*. CEUR-WS.org.