Alternative Characterizations for Program Equivalence under Answer-Set Semantics based on Unfounded Sets*

Martin Gebser[1], Torsten Schaub[1], Hans Tompits[2], and Stefan Woltran[3]

[1] Institut für Informatik, Universität Potsdam,
August-Bebel-Straße 89, D-14482 Potsdam, Germany
{gebser,torsten}@cs.uni-potsdam.de
[2] Institut für Informationssysteme 184/3, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
tompits@kr.tuwien.ac.at
[3] Institut für Informationssysteme 184/2, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
woltran@dbai.tuwien.ac.at

Abstract. Logic programs under answer-set semantics constitute an important tool for declarative problem solving. In recent years, two research issues received growing attention. On the one hand, concepts like loops and elementary sets have been proposed in order to extend Clark's completion for computing answer sets of logic programs by means of propositional logic. On the other hand, different concepts of program equivalence, like strong and uniform equivalence, have been studied in the context of program optimization and modular programming. In this paper, we bring these two lines of research together and provide alternative characterizations for different conceptions of equivalence in terms of unfounded sets, along with the related concepts of loops and elementary sets. Our results yield new insights into the model theory of equivalence checking. We further exploit these characterizations to develop novel encodings of program equivalence in terms of standard and quantified propositional logic, respectively.

1 Introduction

Among the plethora of semantics that emerged during the nineties of the twentieth century for giving meaning to logic programs with nonmonotonic negation, two still play a major role today: firstly, the *answer-set semantics*, due to Gelfond and Lifschitz [1], and secondly, the *well-founded semantics*, due to Van Gelder, Ross, and Schlipf [2]. While the answer-set semantics adheres to a multiple intended models approach, representing the canonical instance of the *answer-set programming* (ASP) paradigm [3], the well-founded semantics is geared toward efficient query answering and can be seen as a skeptical approximation of the answer-set semantics. In this paper, our interest lies with the answer-set semantics of nonmonotonic logic programs. The results developed here can informally be described as linking two research issues in the context of the answer-set semantics by way of the central constituents of the well-founded semantics, viz. *unfounded sets* [2, 4]. Let us explain this in more detail.

* This work was partially supported by the Austrian Science Fund (FWF) under grant P18019.

An important concept in logic programming is *Clark's completion* [5], which associates logic programs with theories of classical logic. While every answer set of a logic program P is also a model of the completion of P, the converse does not hold in general. As shown by Fages [6], a one-to-one correspondence is obtained if P satisfies certain syntactic restrictions. In recent years, a large body of work was devoted to extensions of Fages' characterization in which the syntactic proviso is dropped at the expense of introducing additional formulas to Clark's completion, referred to as *loop formulas* [7]. Although exponentially many such loop formulas must be added in the worst case [8], implementations for the answer-set semantics based on this technique, like ASSAT [7] and Cmodels [9], exploiting solvers for classical logic as back-end inference engines, behave surprisingly well compared to dedicated answer-set tools like DLV [10] and Smodels [11]. While DLV exploits unfounded sets as an approximation technique [4], recent work also reveals relations between unfounded sets and the semantical concepts underlying loop formulas [12, 13].

Another issue extensively studied in the context of answer-set semantics are different notions of program equivalence. The main reason for dealing with varying forms of equivalence for logic programs is that ordinary equivalence, in the sense that two programs are equivalent if they have the same answer sets, does not satisfy a substitution principle similar to that of classical logic. That is to say, replacing a subprogram Q of an overall program P by an equivalent program R does not, in general, yield a program that is equivalent to P. This is of course undesirable for modular programming or program optimization when submodules should be replaced by other, more efficient ones. This led to the introduction of more robust notions of equivalence, notably of *strong equivalence* [14] and *uniform equivalence* [15], defined as follows: two programs, P and Q, are strongly equivalent iff, for every program R, $P \cup R$ and $Q \cup R$ have the same answer sets; and P and Q are uniformly equivalent iff the former condition holds for every set R of facts.

The interesting fact about strong equivalence is that it can be reduced to equivalence in the nonclassical logic of *here-and-there* (also known as *Gödel's three-valued logic*) [14], which is basically intuitionistic logic restricted to two worlds, "here" and "there". This characterization was subsequently adapted by Turner [16] by introducing *SE-models*: an SE-model of a program P is a pair (X, Y), where X, Y are interpretations such that $X \subseteq Y$, $Y \models P$, and $X \models P^Y$, where P^Y is the usual Gelfond-Lifschitz reduct [1] of P relative to Y. Two programs are then strongly equivalent iff they possess the same SE-models. Uniform equivalence, in turn, can be captured by certain *maximal* SE-models, termed *UE-models* [15].

We provide a new perspective on SE- and UE-models by relating them to unfounded sets. As it turns out, an explicit reference to the reduct is not required; for a model Y of a program, unfounded sets U with respect to Y allow us to characterize and distinguish SE- and UE-models of the form $(Y \setminus U, Y)$.[4] While UE-models are certain maximal SE-models, our new characterization of UE-models involves *minimal* unfounded sets.

[4] A similar relationship has been established by Eiter, Leone, and Pearce [17] with respect to the logic $N2$; a logic that later has been used as a first characterization of strong equivalence. Thus, certain connections between unfounded sets and strong equivalence already appear in their work, but only in an implicit manner.

Using our characterization of SE- and UE-models, we also derive novel characterizations of program equivalence in terms of unfounded sets. These can in turn be linked to loop formulas, and consequently to classical logic. Similar to the observation that expressing answer sets in terms of loop formulas yields an exponential blow-up in the worst case, our reductions into standard propositional logic are likewise exponentially sized. However, we can avoid this exponential increase by switching to *quantified propositional logic* as the target language, which extends standard propositional logic by admitting quantifications over atomic formulas. Our encodings not only provide us with new theoretical insights, but the availability of practicably efficient solvers for quantified propositional logic also gives an easy means to build implementations for equivalence checking in a straightforward way. Indeed, other axiomatizations of strong equivalence in terms of propositional logic and of ordinary and uniform equivalence in terms of quantified propositional logic already appeared in the literature [18–20] with that purpose in mind, but these differ significantly from ours as they were based upon different approaches. Finally, all of our encodings are adequate in the sense that the evaluation problems obtained are of the same complexity as the encoded equivalence problems.

The outline of this paper is as follows. In Section 2, we introduce the formal background, and in Section 3, we develop characterizations of models substantial for program equivalence, viz. answer sets, SE-models, and UE-models, based on unfounded sets. We further exploit these characterizations in Section 4 for providing novel specifications of program equivalence as well as encodings in standard and quantified propositional logic. Finally, we discuss our results in Section 5.

2 Background

A propositional *disjunctive logic program* is a finite set of rules of the form

$$a_1 \vee \cdots \vee a_k \leftarrow a_{k+1}, \ldots, a_m, \mathit{not}\, a_{m+1}, \ldots, \mathit{not}\, a_n, \tag{1}$$

where $1 \leq k \leq m \leq n$, every a_i $(1 \leq i \leq n)$ is a propositional atom from some universe $\mathcal{U}$, and not denotes default negation. A rule r of form (1) is called a *fact* if $k = n = 1$, and *positive* if $m = n$. Furthermore, $H(r) = \{a_1, \ldots, a_k\}$ is the *head* of r, $B(r) = \{a_{k+1}, \ldots, a_m, \mathit{not}\, a_{m+1}, \ldots, \mathit{not}\, a_n\}$ is the *body* of r, $B^+(r) = \{a_{k+1}, \ldots, a_m\}$ is the *positive body* of r, and $B^-(r) = \{a_{m+1}, \ldots, a_n\}$ is the *negative body* of r. We sometimes denote a rule r by $H(r) \leftarrow B(r)$.

The (*positive*) *dependency graph* of a program P is the pair

$$(\mathcal{U}, \{(a, b) \mid r \in P, a \in H(r), b \in B^+(r)\}).$$

A nonempty set $U \subseteq \mathcal{U}$ is a *loop* of P if the subgraph of the dependency graph of P induced by U is strongly connected. Following Lee [12], we consider every singleton over $\mathcal{U}$ as a loop. A program P is *tight* [6, 21] if every loop of P is a singleton.

As usual, an interpretation Y is a set of atoms over $\mathcal{U}$. For a rule r, we write $Y \models r$ iff $H(r) \cap Y \neq \emptyset$, $B^+(r) \not\subseteq Y$, or $B^-(r) \cap Y \neq \emptyset$. An interpretation Y is a *model* of a program P, denoted by $Y \models P$, iff $Y \models r$ for every $r \in P$. The *reduct* of P with

respect to Y is $P^Y = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap Y = \emptyset\}$; Y is an *answer set* of P iff Y is a minimal model of P^Y.

Two programs, P and Q, are *ordinarily equivalent* iff their answer sets coincide. Furthermore, P and Q are *strongly equivalent* [14] (resp., *uniformly equivalent* [15]) iff, for every program (resp., set of facts) R, $P \cup R$ and $Q \cup R$ have the same answer sets. For interpretations X, Y, the pair (X, Y) is an *SE-interpretation* iff $X \subseteq Y$. An SE-interpretation (X, Y) is an *SE-model* [16] of a program P iff $Y \models P$ and $X \models P^Y$. The pair (X, Y) is a *UE-model* [15] of P iff it is an SE-model of P and there is no SE-model (Z, Y) of P such that $X \subset Z \subset Y$. The set of all SE-models (resp., UE-models) of P is denoted by $SE(P)$ (resp., $UE(P)$). Two programs, P and Q, are strongly (resp., uniformly) equivalent iff $SE(P) = SE(Q)$ (resp., $UE(P) = UE(Q)$) [16, 15].

Example 1. Consider $P = \{a \vee b \leftarrow\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$. Clearly, both programs are ordinarily equivalent as $\{a\}$ and $\{b\}$ are their respective answer sets. However, they are not strongly equivalent. Indeed, since P is positive, we have that $SE(P) = \{(a, a), (b, b), (ab, ab), (a, ab), (b, ab)\}$.[5] For Q, we have to take the reduct into account. In particular, we have $Q^{\{a,b\}} = \emptyset$, and so any interpretation is a model of $Q^{\{a,b\}}$. Hence, each pair (X, ab) with $X \subseteq \{a, b\}$ is an SE-model of Q. We thus have $SE(Q) = \{(a, a), (b, b), (ab, ab), (a, ab), (b, ab), (\emptyset, ab)\}$. That is, $SE(P) \neq SE(Q)$, so P and Q are not strongly equivalent. A witness for this is $R = \{a \leftarrow b;\ b \leftarrow a\}$, as $P \cup R$ has $\{a, b\}$ as its (single) answer set, while $Q \cup R$ has no answer set.

Concerning uniform equivalence, observe first that $UE(P) = SE(P)$. This is not the case for Q, where the SE-model $(\emptyset, ab)$ is not a UE-model since there exist further SE-models (Z, ab) of Q with $\emptyset \subset Z \subset \{a, b\}$, viz. (a, ab) and (b, ab). One can check that $(\emptyset, ab)$ is in fact the only pair in $SE(Q)$ that is not a UE-model of Q. So, $UE(Q) = SE(Q) \setminus \{(\emptyset, ab)\} = SE(P) = UE(P)$. Thus, P and Q are uniformly equivalent. $\Diamond$

We conclude this section with the following known properties. First, for any program P and any interpretation Y, the following statements are equivalent: (i) $Y \models P$; (ii) $Y \models P^Y$; (iii) $(Y, Y) \in SE(P)$; and (iv) $(Y, Y) \in UE(P)$. Second, if $Y \models P$, Y is an answer set of P iff, for each SE-model (resp., UE-model) (X, Y) of P, $X = Y$.

3 Model-Theoretic Characterizations by Unfounded Sets

In this section, we exploit the notion of an unfounded set [2, 4] and provide alternative characterizations of models for logic programs and program equivalence. Roughly speaking, the aim of unfounded sets is to collect atoms that cannot be derived from a program with respect to a fixed interpretation. Given the closed-world reasoning flavor of answer sets, such atoms are considered to be false. However, we shall relate here unfounded sets also to SE- and UE-models, and thus to concepts that do not fall under the closed-world assumption (since they implicitly deal with program extensions). For the case of uniform equivalence, we shall also employ the recent concept of elementarily unfounded sets [13], which via elementary sets decouple the idea of (minimal) unfounded sets from fixed interpretations. Finally, we shall link our results to loops.

[5] Whenever convenient, we use strings like ab as a shorthand for $\{a, b\}$. As a convention, we let universe $\mathcal{U}$ be the set of atoms occurring in the programs under consideration.

Given a program P and an interpretation Y, a set $U \subseteq \mathcal{U}$ is *unfounded* for P with respect to Y if, for each $r \in P$, at least one of the following conditions holds:

1. $H(r) \cap U = \emptyset$,
2. $H(r) \cap (Y \setminus U) \neq \emptyset$,
3. $B^+(r) \not\subseteq Y$ or $B^-(r) \cap Y \neq \emptyset$, or
4. $B^+(r) \cap U \neq \emptyset$.

Note that the empty set is unfounded for any program P with respect to any interpretation since the first condition, $H(r) \cap \emptyset = \emptyset$, holds for all $r \in P$.

Example 2. Consider the following program:

$$P = \left\{ \begin{array}{lll} r_1: \ a \vee b \leftarrow & r_3: \ c \leftarrow a & r_5: \ c \leftarrow b, d \\ r_2: \ b \vee c \leftarrow & r_4: \ d \leftarrow not\ b & r_6: \ d \leftarrow c, not\ a \end{array} \right\}.$$

Let $U = \{c, d\}$. We have $H(r_1) \cap U = \{a, b\} \cap \{c, d\} = \emptyset$, that is, r_1 satisfies Condition 1. For r_5 and r_6, $B^+(r_5) \cap U = \{b, d\} \cap \{c, d\} \neq \emptyset$ and $B^+(r_6) \cap U = \{c\} \cap \{c, d\} \neq \emptyset$. Hence, both rules satisfy Condition 4. Furthermore, consider the interpretation $Y = \{b, c, d\}$. We have $H(r_2) \cap (Y \setminus U) = \{b, c\} \cap \{b\} \neq \emptyset$. Thus, r_2 satisfies Condition 2. Finally, for r_3 and r_4, $B^+(r_3) = \{a\} \not\subseteq \{b, c, d\} = Y$ and $B^-(r_4) \cap Y = \{b\} \cap \{b, c, d\} \neq \emptyset$, that is, both rules satisfy Condition 3. From the fact that each rule in P satisfies at least one of the unfoundedness conditions, we conclude that $U = \{c, d\}$ is unfounded for P with respect to $Y = \{b, c, d\}$. $\Diamond$

The basic relation between unfounded sets and answer sets is as follows.

Proposition 1 ([4, 17]). *Let P be a program and Y an interpretation. Then, Y is an answer set of P iff $Y \models P$ and no nonempty subset of Y is unfounded for P with respect to Y.*

Example 3. Program P in Example 2 has two answer sets: $\{a, c, d\}$ and $\{b\}$. For the latter, we just have to check that $\{b\}$ is not unfounded for P with respect to $\{b\}$ itself, which holds in view of either rule r_1 or r_2. To verify via unfounded sets that $Y = \{a, c, d\}$ is an answer set of P, we have to check all nonempty subsets of Y. For instance, take $U = \{c, d\}$. We have already seen that r_1, r_5, and r_6 satisfy Condition 1 or 4, respectively; but the remaining rules r_2, r_3, and r_4 violate all four unfoundedness conditions for U with respect to Y. Hence, $U = \{c, d\}$ is not unfounded for P with respect to $Y = \{a, c, d\}$. $\Diamond$

We next detail the relationship between unfounded sets and models of logic programs as well as of their reducts. First, we have the following relationships between models and unfounded sets.

Lemma 1. *Let P be a program and Y an interpretation. Then, the following statements are equivalent:*

(a) $Y \models P$;
(b) every set $U \subseteq \mathcal{U} \setminus Y$ is unfounded for P with respect to Y; and

(*c*) *every singleton $U \subseteq \mathcal{U} \setminus Y$ is unfounded for P with respect to Y.*

Proof. (*a*) $\Rightarrow$ (*b*): Assume that some set $U \subseteq \mathcal{U} \setminus Y$ is not unfounded for P with respect to Y. Then, for some rule $r \in P$, we have: (α) $H(r) \cap U \neq \emptyset$; ($\beta$) $H(r) \cap (Y \setminus U) = \emptyset$; ($\gamma$) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$; and ($\delta$) $B^+(r) \cap U = \emptyset$. Since $U \cap Y = \emptyset$ by hypothesis, we conclude from (β) that $H(r) \cap Y = \emptyset$. Since (γ) holds in addition, we have $Y \not\models r$ and thus $Y \not\models P$.

(*b*) $\Rightarrow$ (*c*): Trivial.

(*c*) $\Rightarrow$ (*a*): Assume $Y \not\models P$. Then, there is a rule $r \in P$ such that $Y \not\models r$, that is, $H(r) \cap Y = \emptyset$ and (γ) hold. By the definition of rules, $H(r) \neq \emptyset$. So, consider any $a \in H(r)$ and the singleton $U = \{a\}$. Clearly, (α) holds for r, and (β) holds by $H(r) \cap Y = \emptyset$. Finally, since $B^+(r) \subseteq Y$ and $a \notin Y$, (δ) holds as well. That is, there is a singleton $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for P with respect to Y. □

We further describe the models of a program's reduct by unfounded sets.

Lemma 2. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U) \models P^Y$ iff U is unfounded for P with respect to Y.*

Proof. ($\Rightarrow$) Assume that U is not unfounded for P with respect to Y. Then, for some rule $r \in P$, (α)–(δ) from the proof of Lemma 1 hold. Clearly, $B^-(r) \cap Y = \emptyset$ implies $(H(r) \leftarrow B^+(r)) \in P^Y$. From $B^+(r) \subseteq Y$ and (δ), we conclude $B^+(r) \subseteq (Y \setminus U)$. Together with ($\beta$), we obtain $(Y \setminus U) \not\models (H(r) \leftarrow B^+(r))$ and thus $(Y \setminus U) \not\models P^Y$.

($\Leftarrow$) Assume $(Y \setminus U) \not\models P^Y$. Then, there is a rule $r \in P$ such that $(Y \setminus U) \not\models \{r\}^Y$. We conclude that r satisfies (β), $B^+(r) \subseteq (Y \setminus U)$, and $B^-(r) \cap Y = \emptyset$. Since $B^+(r) \subseteq (Y \setminus U)$ immediately implies $B^+(r) \subseteq Y$, (γ) holds. Moreover, $B^+(r) \subseteq (Y \setminus U)$ also implies (δ). It remains to show (α). From (γ) and $Y \models r$ (which holds by the assumption $Y \models P$), we conclude $H(r) \cap Y \neq \emptyset$. Together with ($\beta$), this implies ($\alpha$). Since ($\alpha$), ($\beta$), ($\gamma$), and ($\delta$) jointly hold for some rule $r \in P$, we have that U is not unfounded for P with respect to Y. □

Example 4. For illustration, reconsider P from Example 2 and $Y = \{b, c, d\}$. For singleton $\{a\}$ and r_1, we have $H(r_1) \cap (Y \setminus \{a\}) = \{a, b\} \cap \{b, c, d\} \neq \emptyset$. Furthermore, $a \notin H(r)$ for all $r \in \{r_2, \ldots, r_6\}$. That is, $\{a\}$ is unfounded for P with respect to Y. From this, we can conclude by Lemma 1 that Y is a model of P, i.e., $Y \models P$.

As we have already seen in Example 2, $U = \{c, d\}$ is unfounded for P with respect to Y. Lemma 2 now tells us that $(Y \setminus U) = \{b\}$ is a model of $P^Y = \{r_1, r_2, r_3, r_5, (H(r_6) \leftarrow B^+(r_6))\}$. Moreover, one can check that $\{a, c, d\}$ is as well unfounded for P with respect to Y. ◊

The last observation in Example 4 stems from a more general side-effect of Lemma 2: for any program P, any interpretation Y such that $Y \models P$, and $U \subseteq \mathcal{U}$, U is unfounded for P with respect to Y iff $(U \cap Y)$ is unfounded for P with respect to Y. For models Y, this allows us to restrict our attention to unfounded sets $U \subseteq Y$.

We now are in a position to state an alternative characterization of SE-models.

Theorem 1. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U, Y)$ is an SE-model of P iff $(U \cap Y)$ is unfounded for P with respect to Y.*

The following result reformulates the definition of UE-models in view of Theorem 1.

Corollary 1. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$ such that $(U \cap Y)$ is unfounded for P with respect to Y. Then, $(Y \setminus U, Y)$ is a UE-model of P iff, for each V with $(Y \setminus U) \subset (Y \setminus V) \subset Y$, $(V \cap Y)$ is not unfounded for P with respect to Y.*

A simple reformulation of this result provides us with the following novel characterization of UE-models.

Theorem 2. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U, Y)$ is a UE-model of P iff $(U \cap Y)$ is unfounded for P with respect to Y and no nonempty proper subset of $(U \cap Y)$ is unfounded for P with respect to Y.*

Note that the inherent maximality criterion of UE-models is now reflected by a *minimality condition* on (nonempty) unfounded sets. Theorems 1 and 2 allow us to characterize strong and uniform equivalence in terms of unfounded sets, avoiding an explicit use of programs' reducts. This will be detailed in Section 4.

Example 5. Recall programs $P = \{a \vee b \leftarrow\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$ from Example 1. We have seen that the only difference in their SE-models is the pair $(\emptyset, ab)$, which is an SE-model of Q, but not of P. Clearly, $Y = \{a, b\}$ is a classical model of P and of Q, and in view of Theorem 1, we expect that Y is unfounded for Q with respect to Y, but not for P with respect to Y. The latter is easily checked since the rule $r = (a \vee b \leftarrow)$ yields (1) $H(r) \cap Y \neq \emptyset$; (2) $H(r) \cap (Y \setminus Y) = \emptyset$; (3) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$; and (4) $B^+(r) \cap Y = \emptyset$. Thus, none of the four unfoundedness conditions is met. However, for $r_1 = (a \leftarrow not\, b)$ and $r_2 = (b \leftarrow not\, a)$, we have $B^-(r_i) \cap Y \neq \emptyset$, for $i \in \{1, 2\}$, and thus Y is unfounded for Q with respect to Y.

Recall that $(\emptyset, ab)$ is not a UE-model of Q. In view of Theorem 2, we thus expect that $Y = \{a, b\}$ is not a minimal nonempty unfounded set. As one can check, both nonempty proper subsets $\{a\}$ and $\{b\}$ are in fact unfounded for Q with respect to Y. $\Diamond$

In the remainder of this section, we provide a further characterization of UE-models that makes use of elementary sets [13]. For a UE-model (X, Y), this not only gives us a more intrinsic characterization of the difference $U = (Y \setminus X)$ than stated in Theorem 2, but it also yields a further direct relation to loops. We make use of this fact and provide a new result for the UE-models of tight programs.

We define a nonempty set $U \subseteq \mathcal{U}$ as *elementary* for a program P if, for each V such that $\emptyset \subset V \subset U$, there is some $r \in P$ jointly satisfying

1. $H(r) \cap V \neq \emptyset$,
2. $H(r) \cap (U \setminus V) = \emptyset$,
3. $B^+(r) \cap V = \emptyset$, and
4. $B^+(r) \cap (U \setminus V) \neq \emptyset$.

Due to Conditions 1 and 4, every elementary set is also a loop of P, but the converse does not hold in general. Similar to loops, however, elementary sets can be used to characterize answer sets in terms of propositional logic (cf. Proposition 3).

Example 6. Consider the following program:

$$P = \left\{ \begin{array}{lll} r_1: & a \vee d \leftarrow c & r_3: \quad b \leftarrow a, c \qquad r_5: \quad b \vee c \leftarrow d, e \\ r_2: & a \leftarrow b, d & r_4: \quad c \leftarrow a \qquad r_6: \quad d \vee e \leftarrow \end{array} \right\}.$$

As one can check, the subgraph of the dependency graph of P induced by $\{a, c, d\}$ is strongly connected, hence, $\{a, c, d\}$ is a loop of P. However, looking at subset $\{d\}$, we can verify that there is no rule in P satisfying all four elementary set conditions: only for $r \in \{r_1, r_6\}$, we have $H(r) \cap \{d\} \neq \emptyset$, but $H(r_1) \cap (\{a, c, d\} \setminus \{d\}) = \{a, d\} \cap \{a, c\} \neq \emptyset$ and $B^+(r_6) \cap (\{a, c, d\} \setminus \{d\}) = \emptyset \cap \{a, c\} = \emptyset$. This shows that the loop $\{a, c, d\}$ is not an elementary set of P.

Next, consider $U = \{a, b, c\}$, which is again a loop of P. In contrast to $\{a, c, d\}$, U is as well elementary for P. For instance, taking $V = \{a, b\}$ and r_1, we have $H(r_1) \cap V = \{a, d\} \cap \{a, b\} \neq \emptyset$, $H(r_1) \cap (U \setminus V) = \{a, d\} \cap \{c\} = \emptyset$, $B^+(r_1) \cap V = \{c\} \cap \{a, b\} = \emptyset$, and $B^+(r_1) \cap (U \setminus V) = \{c\} \cap \{c\} \neq \emptyset$. $\Diamond$

To link elementary sets and unfounded sets together, for a program P, an interpretation Y, and $U \subseteq \mathcal{U}$, we define

$$P_{Y,U} = \{r \in P \mid H(r) \cap (Y \setminus U) = \emptyset,\ B^+(r) \subseteq Y,\ B^-(r) \cap Y = \emptyset\}.$$

Provided that $H(r) \cap U \neq \emptyset$, a rule $r \in P_{Y,U}$ supports U with respect to Y, while no rule in $(P \setminus P_{Y,U})$ supports U with respect to Y. Analogously to Gebser, Lee, and Lierler [13], we say that U is *elementarily unfounded* for P with respect to Y iff (i) U is unfounded for P with respect to Y and (ii) U is elementary for $P_{Y,U}$. Any elementarily unfounded set of P with respect to Y is also elementary for P, but an elementary set U that is unfounded for P with respect to Y is not necessarily elementarily unfounded because U might not be elementary for $P_{Y,U}$ [13].

Elementarily unfounded sets coincide with minimal nonempty unfounded sets.

Proposition 2 ([13]). *Let P be a program, Y an interpretation, and $U \subseteq \mathcal{U}$. Then, U is a minimal nonempty unfounded set of P with respect to Y iff U is elementarily unfounded for P with respect to Y.*

Example 7. Recall that $U = \{a, b, c\}$ is elementary for Program P from Example 6. For $Y = \{a, b, c, d\}$, we have $P_{Y,U} = \{r_2, r_3, r_4\}$ because $H(r_1) \cap (Y \setminus U) = H(r_6) \cap (Y \setminus U) = \{d\} \neq \emptyset$ and $B^+(r_5) = \{d, e\} \not\subseteq \{a, b, c, d\} = Y$. Since $B^+(r_i) \cap U \neq \emptyset$, for $i \in \{2, 3, 4\}$, we further conclude that U is unfounded for P with respect to Y. Although U is an elementary set of P and unfounded for P with respect to Y, it is not elementary for $P_{Y,U}$ and thus not an elementarily unfounded set of P with respect to Y. This is verified by taking $V = \{a, b\}$, where we have $B^+(r_i) \cap V \neq \emptyset$, for $i \in \{2, 3, 4\}$. As one can check, V itself is unfounded for P with respect to Y and elementary for $P_{Y,V} = \{r_2, r_3\}$. Therefore, $V = \{a, b\}$ is elementarily unfounded for P with respect to Y. Finally, observe that neither $\{a\}$ nor $\{b\}$ is unfounded for P with respect to Y. Thus, V is indeed a minimal nonempty unfounded set of P with respect to Y. $\Diamond$

The fact that every nonempty unfounded set contains some elementarily unfounded set, which by definition is an elementary set, allows us to derive some properties of

the difference $U = (Y \setminus X)$ for SE-interpretations (X, Y). For instance, we can make use of the fact that every elementary set is also a loop, while it is not that obvious to conclude the same for minimal nonempty unfounded sets, only defined with respect to interpretations.

Formally, we derive the following properties for UE-models (resp., SE-models).

Corollary 2. *Let P be a program and (X, Y) a UE-model (resp., SE-model) of P. If $X \neq Y$, then $(Y \setminus X)$ is (resp., contains) (a) an elementarily unfounded set of P with respect to Y; (b) an elementary set of P; and (c) a loop of P.*

For tight programs, i.e., programs such that every loop is a singleton, we obtain the following property.

Corollary 3. *Let P be a tight program and (X, Y) an SE-model of P. Then, (X, Y) is a UE-model of P iff $X = Y$ or $(Y \setminus X)$ is a singleton that is unfounded for P with respect to Y.*

Proof. Given that P is tight, every loop of P is a singleton. Thus, the fact that any elementarily unfounded set is a loop of P implies that only singletons can be elementarily unfounded. From Theorem 2 and Proposition 2, we conclude that (X, Y) is a UE-model of P iff $X = Y$ or $Y \setminus X$ is a singleton that is unfounded for P with respect to Y. □

Example 8. Recall the SE-model $(\emptyset, ab)$ of $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$. The loops of Q are $\{a\}$ and $\{b\}$, thus, Q is tight. This allows us to immediately conclude that $(\emptyset, ab)$ is not a UE-model of Q, without looking for any further SE-model to rebut it. ◊

Corollary 3 shows that, for tight programs, the structure of UE-models is particularly simple, i.e., they are always of the form (Y, Y) or $(Y \setminus \{a\}, Y)$, for some $a \in Y$. As we will see in the next section, this also allows for simplified encodings.

4 Characterizations for Program Equivalence

In this section, we further exploit unfounded sets to characterize different notions of program equivalence. We start by comparing two programs, P and Q, regarding their unfounded sets for deriving conditions under which P and Q are ordinarily, strongly, and uniformly equivalent, respectively. Based on these conditions, we then provide novel encodings in standard and quantified propositional logic.

4.1 Characterizations based on Unfounded Sets

Two programs are ordinarily equivalent if they possess the same answer sets. As Proposition 1 shows, answer sets are precisely the models of a program that do not contain any nonempty unfounded set. Hence, ordinary equivalence can be described as follows.

Theorem 3. *Let P and Q be programs. Then, P and Q are ordinarily equivalent iff, for every interpretation Y, the following statements are equivalent:*

(a) *$Y \models P$ and no nonempty subset of Y is unfounded for P with respect to Y; and*

(*b*) $Y \models Q$ *and no nonempty subset of* Y *is unfounded for* Q *with respect to* Y.

Note that ordinarily equivalent programs are not necessarily classically equivalent, as is for instance witnessed by programs $P = \{a \vee b \leftarrow\}$ and $Q = \{a \vee b \leftarrow;\ a \leftarrow c\}$ possessing the same answer sets: $\{a\}$ and $\{b\}$. However, $\{b, c\}$ is a model of P, but not of Q. In turn, for strong and uniform equivalence, classical equivalence is a necessary (but, in general, not a sufficient) condition. This follows from the fact that every model of a program participates in at least one SE-model (resp., UE-model) and is thus relevant for testing strong (resp., uniform) equivalence. Therefore, the following characterization of strong equivalence considers all classical models.

Theorem 4. *Let* P *and* Q *be programs. Then,* P *and* Q *are strongly equivalent iff, for every interpretation* Y *such that* $Y \models P$ *or* $Y \models Q$, P *and* Q *possess the same unfounded sets with respect to* Y.

Proof. ($\Rightarrow$) Assume that P and Q are strongly equivalent. Fix any interpretation Y such that $Y \models P$ (or $Y \models Q$). Then, (Y, Y) is an SE-model of P (or Q), and since P and Q are strongly equivalent, (Y, Y) is also an SE-model of Q (or P). That is, both $Y \models P$ and $Y \models Q$ hold. Fix any set $U \subseteq \mathcal{U}$. By Lemma 2, U is unfounded for P with respect to Y iff $(Y \setminus U, Y)$ is an SE-model of P. Since P and Q are strongly equivalent, the latter holds iff $(Y \setminus U, Y)$ is an SE-model of Q, which in turn holds iff U is unfounded for Q with respect to Y.

($\Leftarrow$) Assume that P and Q are not strongly equivalent. Then, without loss of generality, there is an SE-model (X, Y) of P that is not an SE-model of Q (the other case is symmetric). By the definition of SE-models, we have $Y \models P$, and by Lemma 2, $(Y \setminus X)$ is unfounded for P with respect to Y, but either $Y \not\models Q$ or $(Y \setminus X)$ is not unfounded for Q with respect to Y. If $(Y \setminus X)$ is not unfounded for Q with respect to Y, then P and Q do not possess the same unfounded sets with respect to Y. Otherwise, if $Y \not\models Q$, by Lemma 1, there is a set $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for Q with respect to Y, but U is unfounded for P with respect to Y. □

Theorem 4 shows that strong equivalence focuses primarily on the unfounded sets admitted by the compared programs. In the setting of uniform equivalence, the consideration of unfounded sets is further restricted to minimal ones (cf. Theorem 2), and by Proposition 2, these are exactly the elementarily unfounded sets.

Theorem 5. *Let* P *and* Q *be programs. Then,* P *and* Q *are uniformly equivalent iff, for every interpretation* Y *such that* $Y \models P$ *or* $Y \models Q$, P *and* Q *possess the same elementarily unfounded sets with respect to* Y.

Proof. ($\Rightarrow$) Assume that P and Q are uniformly equivalent. Fix any interpretation Y such that $Y \models P$ (or $Y \models Q$). Then, (Y, Y) is a UE-model of P (or Q), and since P and Q are uniformly equivalent, (Y, Y) is also a UE-model of Q (or P). That is, both $Y \models P$ and $Y \models Q$ hold. Fix any elementarily unfounded set U of P (or Q) with respect to Y. If $U \subseteq \mathcal{U} \setminus Y$, by Lemma 1 and Proposition 2, U is a singleton that is unfounded for both P and Q with respect to Y, which implies that U is elementarily unfounded for Q (or P) with respect to Y. Otherwise, if $U \cap Y \neq \emptyset$, then Lemma 1 and Proposition 2 imply $U \subseteq Y$. By Theorem 2 and Proposition 2, $(Y \setminus U, Y)$ is a

UE-model of P (or Q), and since P and Q are uniformly equivalent, $(Y \setminus U, Y)$ is as well a UE-model of Q (or P). Since $\emptyset \neq U \subseteq Y$, by Theorem 2 and Proposition 2, we conclude that U is elementarily unfounded for Q (or P) with respect to Y.

($\Leftarrow$) Assume that P and Q are not uniformly equivalent. Then, without loss of generality, there is a UE-model (X, Y) of P that is not a UE-model of Q (the other case is symmetric). Since (X, Y) is also an SE-model of P, we have $Y \models P$. If $Y \not\models Q$, by Lemma 1, there is a singleton $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for Q with respect to Y, but U is unfounded for P with respect to Y. That is, U is elementarily unfounded for P with respect to Y, but not for Q with respect to Y. Otherwise, if $Y \models Q$, (Y, Y) is a UE-model both of P and of Q. Hence, $X \subset Y$, and by Theorem 2 and Proposition 2, $(Y \setminus X)$ is elementarily unfounded for P with respect to Y. Furthermore, the fact that (X, Y) is not a UE-model of Q, by Theorem 2 and Proposition 2, implies that $(Y \setminus X)$ is not elementarily unfounded for Q with respect to Y. □

In contrast to arbitrary unfounded sets, elementarily unfounded sets exhibit a certain structure as they are loops or, even more accurately, elementary sets (cf. Corollary 2). Theorem 5 tells us that such structures alone are material to uniform equivalence.

4.2 Characterizations in (Standard) Propositional Logic

We now exploit the above results on unfounded sets to encode program equivalence in propositional logic. For ordinary equivalence, we use the well-known concept of loop formulas, while for strong and uniform equivalence, we refer directly to unfounded sets.

In what follows, we write for a set of default literals, like $B(r)$, and a set of atoms, like $H(r)$, $B(r) \rightarrow H(r)$ as a shorthand for

$$\big(\bigwedge_{a \in B^+(r)} a \wedge \bigwedge_{a \in B^-(r)} \neg a\big) \rightarrow \bigvee_{a \in H(r)} a,$$

where, as usual, empty conjunctions (resp., disjunctions) are understood as $\top$ (resp., $\bot$). For instance, for a rule r of form (1), $B(r) \rightarrow H(r)$ stands for

$$a_{k+1} \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n \rightarrow a_1 \vee \cdots \vee a_k.$$

Furthermore, within the subsequent encodings, an occurrence of a program P is understood as $\bigwedge_{r \in P}(B(r) \rightarrow H(r))$.

As a basis for the encodings, we use the following concept. Following Lee [12], for a program P and $U \subseteq \mathcal{U}$, the *external support formula* of U for P is

$$ES_P(U) = \bigvee_{r \in P, H(r) \cap U \neq \emptyset, B^+(r) \cap U = \emptyset} \neg\big(B(r) \rightarrow (H(r) \setminus U)\big). \tag{2}$$

Intuitively, the models of $ES_P(U)$ are those interpretations Y such that U is externally supported by P with respect to Y. That is, there is a rule $r \in P$ that *supports* U with respect to Y, i.e., $H(r) \cap U \neq \emptyset$, $H(r) \cap (Y \setminus U) = \emptyset$, $B^+(r) \subseteq Y$, and $B^-(r) \cap Y = \emptyset$ jointly hold. In addition, to make U *externally* supported, one requires $B^+(r) \cap U = \emptyset$, expressing that the support comes from "outside" U.

The relationship between unfounded sets and external support formulas is as follows.

Lemma 3. *Let P be a program, Y an interpretation, and $U \subseteq \mathcal{U}$. Then, U is unfounded for P with respect to Y iff $Y \not\models ES_P(U)$.*

Proof. ($\Rightarrow$) Assume $Y \models ES_P(U)$. Then, there is a rule $r \in P$ such that (α) $H(r) \cap U \neq \emptyset$; ($\beta$) $B^+(r) \cap U = \emptyset$; ($\gamma$) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$; and ($\delta$) $(H(r) \setminus U) \cap Y = H(r) \cap (Y \setminus U) = \emptyset$. That is, U is not unfounded for P with respect to Y.

($\Leftarrow$) Assume that U is not unfounded for P with respect to Y. Then, there is a rule $r \in P$ for which (α), (β), (γ), and (δ) hold. From (γ) and (δ), we conclude $Y \models \neg\big(B(r) \rightarrow (H(r) \setminus U)\big)$, which together with ($\alpha$) and ($\beta$) implies $Y \models ES_P(U)$. □

For a program P and $U \subseteq \mathcal{U}$, the (conjunctive) *loop formula* [12] of U for P is

$$LF_P(U) = \big(\bigwedge_{p \in U} p\big) \rightarrow ES_P(U). \tag{3}$$

With respect to an interpretation Y, the loop formula of U is violated if Y contains U as an unfounded set, otherwise, the loop formula of U is satisfied.

Proposition 3 ([12, 13]). *Let P be a program and Y an interpretation such that $Y \models P$. Then, the following statements are equivalent:*

(*a*) *Y is an answer set of P;*
(*b*) *$Y \models LF_P(U)$ for every nonempty subset U of $\mathcal{U}$;*
(*c*) *$Y \models LF_P(U)$ for every loop U of P;*
(*d*) *$Y \models LF_P(U)$ for every elementary set U of P.*

For ordinary equivalence, the following encodings (as well as different combinations thereof) can thus be obtained.

Theorem 6. *Let P and Q be programs, and let $\mathcal{L}$ and $\mathcal{E}$ denote the set of all loops and elementary sets, respectively, of P and Q. Then, the following statements are equivalent:*

(*a*) *P and Q are ordinarily equivalent;*
(*b*) *$\big(P \wedge \bigwedge_{\emptyset \neq U \subseteq \mathcal{U}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{\emptyset \neq U \subseteq \mathcal{U}} LF_Q(U)\big)$ is a tautology;*
(*c*) *$\big(P \wedge \bigwedge_{U \in \mathcal{L}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{U \in \mathcal{L}} LF_Q(U)\big)$ is a tautology;*
(*d*) *$\big(P \wedge \bigwedge_{U \in \mathcal{E}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{U \in \mathcal{E}} LF_Q(U)\big)$ is a tautology.*

Recall that, for tight programs, each loop (and thus each elementary set) is a singleton. In this case, the encodings in (*c*) and (*d*) are therefore polynomial in the size of the compared programs. Moreover, one can verify that they amount to checking whether the completions [5] of the compared programs are equivalent in classical logic.

For strong and uniform equivalence between P and Q, the models of P and Q along with the corresponding unfounded sets are compared, as Theorems 4 and 5 show. We thus directly consider external support formulas, rather than loop formulas.

Theorem 4 and Lemma 3 yield the following encoding for strong equivalence.

Theorem 7. *Let P and Q be programs. Then, P and Q are strongly equivalent iff $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \subseteq \mathcal{U}} \big(ES_P(U) \leftrightarrow ES_Q(U)\big)\big)$ is a tautology.*

Proof. By Theorem 4, P and Q are strongly equivalent iff, for every interpretation Y such that $Y \models P$ or $Y \models Q$, P and Q possess the same unfounded sets with respect to Y. By Lemma 3, a set $U \subseteq \mathcal{U}$ is unfounded for P (resp., Q) with respect to Y iff $Y \not\models ES_P(U)$ (resp., $Y \not\models ES_Q(U)$). From this, the statement follows. □

In order to encode also uniform equivalence, we have to single out elementarily unfounded sets. To this end, we modify the definition of the external support formula, $ES_P(U)$, and further encode the case that U is (not) a minimal nonempty unfounded set. For a program P and $U \subseteq \mathcal{U}$, we define the *minimality* external support formula as

$$ES^\star_P(U) = ES_P(U) \vee \neg\big(\textstyle\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big). \tag{4}$$

Similar to external support formulas and unfounded sets, minimality external support formulas correspond to elementarily unfounded sets as follows.

Lemma 4. *Let P be a program, Y an interpretation, and $\emptyset \subset U \subseteq \mathcal{U}$. Then, U is elementarily unfounded for P with respect to Y iff $Y \not\models ES^\star_P(U)$.*

Proof. ($\Rightarrow$) Assume $Y \models ES^\star_P(U)$. We have two cases. First, $Y \models ES_P(U)$: By Lemma 3, U is not unfounded for P with respect to Y, which implies that U is not elementarily unfounded for P with respect to Y. Second, $Y \not\models \big(\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big)$: For some V such that $\emptyset \subset V \subset U$, we have $Y \not\models ES_P(V)$. By Lemma 3, V is unfounded for P with respect to Y. We conclude that U is not a minimal nonempty unfounded set of P with respect to Y, and by Proposition 2, U is not elementarily unfounded for P with respect to Y.

($\Leftarrow$) Assume $Y \not\models ES^\star_P(U)$. Then, $Y \not\models ES_P(U)$, and by Lemma 3, U is unfounded for P with respect to Y. Furthermore, $Y \models \big(\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big)$, and thus no set V such that $\emptyset \subset V \subset U$ is unfounded for P with respect to Y (again by Lemma 3). That is, U is a minimal nonempty unfounded set of P with respect to Y, and by Proposition 2, U is elementarily unfounded for P with respect to Y. □

Theorem 5 and Lemma 4 allow us to encode uniform equivalence as follows.

Theorem 8. *Let P and Q be programs, and let $\mathcal{L}$ and $\mathcal{E}$ denote the set of all loops and elementary sets, respectively, of P and Q. Then, the following statements are equivalent:*

(*a*) *P and Q are uniformly equivalent;*
(*b*) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \subseteq \mathcal{U}} \big(ES^\star_P(U) \leftrightarrow ES^\star_Q(U)\big)\big)$ *is a tautology;*
(*c*) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \in \mathcal{L}} \big(ES^\star_P(U) \leftrightarrow ES^\star_Q(U)\big)\big)$ *is a tautology;*
(*d*) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \in \mathcal{E}} \big(ES^\star_P(U) \leftrightarrow ES^\star_Q(U)\big)\big)$ *is a tautology.*

Proof. By Theorem 5, P and Q are uniformly equivalent iff, for every interpretation Y such that $Y \models P$ or $Y \models Q$, P and Q possess the same elementarily unfounded sets with respect to Y. Clearly, any elementarily unfounded set of P or Q belongs to the set $\mathcal{E}$ of all elementary sets of P and Q, which is a subset of the set $\mathcal{L}$ of all loops of P and Q, and every element of $\mathcal{L}$ is a subset of $\mathcal{U}$. By Lemma 4, a set $\emptyset \subset U \subseteq \mathcal{U}$ is elementarily unfounded for P (resp., Q) with respect to Y iff $Y \not\models ES^\star_P(U)$ (resp., $Y \not\models ES^\star_Q(U)$). Finally, we have $ES^\star_P(\emptyset) = ES^\star_Q(\emptyset) = \bot \vee \neg\top \equiv \bot$, so that $Y \models \big(ES^\star_P(\emptyset) \leftrightarrow ES^\star_Q(\emptyset)\big)$ for any interpretation Y. From this, the statement follows. □

Again, we exploit the fact that, for tight programs, all loops and elementary sets are singletons. It is thus sufficient to consider only the external support formulas of singletons. To the best of our knowledge, this provides a novel technique to decide uniform equivalence between tight programs. In fact, the following result is an immediate consequence of (*c*), or likewise (*d*), in Theorem 8.

Corollary 4. *Let P and Q be tight programs. Then, P and Q are uniformly equivalent iff $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{a \in \mathcal{U}} \big(ES_P(\{a\}) \leftrightarrow ES_Q(\{a\})\big)\big)$ is a tautology.*

Indeed, for singletons $\{a\}$, $\neg\big(\bigwedge_{\emptyset \subset V \subset \{a\}} ES_P(V)\big)$ (resp., $\neg\big(\bigwedge_{\emptyset \subset V \subset \{a\}} ES_Q(V)\big)$) can be dropped from $ES^\star_P(\{a\})$ (resp., $ES^\star_Q(\{a\})$) because it is equivalent to $\bot$.

Except for ordinary and uniform equivalence between tight programs, all of the above encodings are of exponential size. As with the well-known encodings for answer sets (cf. Proposition 3), we do not suggest to a priori reduce the problem of deciding program equivalence to propositional logic. Rather, our encodings provide an alternative view on the conditions underlying program equivalence; similar characterizations have already been successfully exploited in answer-set solving [7, 9].

4.3 Characterizations in Quantified Propositional Logic

We now provide encodings that avoid the exponential blow-ups encountered above. Except for strong equivalence, this requires encodings to be stated in quantified propositional logic rather than in (standard) propositional logic.

We briefly recall the basic elements of quantified propositional logic. Syntactically, quantified propositional logic extends standard propositional logic by permitting quantifications over propositional variables. Formulas of quantified propositional logic are also referred to as *quantified Boolean formulas* (QBFs). An occurrence of an atom p is *free* in a QBF Φ if it is not in the scope of a quantifier $\mathsf{Q}p$, for $\mathsf{Q} \in \{\forall, \exists\}$. Given a finite set $P = \{p_1, \ldots, p_n\}$ of atoms, $\mathsf{Q}P\,\Psi$ stands for any QBF of the form $\mathsf{Q}p_1 \ldots \mathsf{Q}p_n\Psi$. For an atom p and a formula Ψ, $\Phi[p/\Psi]$ denotes the QBF resulting from Φ by replacing each free occurrence of p in Φ by Ψ. For (indexed) sets $P = \{p_1, \ldots, p_n\}$ and $S = \{\Psi_1, \ldots, \Psi_n\}$, $\Phi[P/S]$ is a shorthand for $(\cdots(\Phi[p_1/\Psi_1])\cdots)[p_n/\Psi_n]$.

For an interpretation I and a QBF Φ, the relation $I \models \Phi$ is defined analogously to standard propositional logic, with the additional cases that, if $\Phi = \forall p\,\Psi$, then $I \models \Phi$ iff $I \models \Psi[p/\top]$ and $I \models \Psi[p/\bot]$, and if $\Phi = \exists p\,\Psi$, then $I \models \Phi$ iff $I \models \Psi[p/\top]$ or $I \models \Psi[p/\bot]$. We say that a QBF Φ is *valid* if $I \models \Phi$ for every interpretation I.

Given a universe $\mathcal{U}$, we use further copies $\mathcal{U}' = \{p' \mid p \in \mathcal{U}\}$, $\mathcal{U}'' = \{p'' \mid p \in \mathcal{U}\}$, etc. of mutually disjoint sets of new atoms. Moreover, we adopt the following abbreviations to compare interpretations over $\mathcal{U}$ (via copies):

$$\mathcal{U}' \leq \mathcal{U} \text{ stands for } \bigwedge_{p \in \mathcal{U}}(p' \rightarrow p); \text{ and}$$
$$\mathcal{U}' < \mathcal{U} \text{ stands for } \bigwedge_{p \in \mathcal{U}}(p' \rightarrow p) \wedge \neg\big(\bigwedge_{p \in \mathcal{U}}(p \rightarrow p')\big).$$

These building blocks allow us to check containedness between $Y, U \subseteq \mathcal{U}$ as follows. Let I be an interpretation over $\mathcal{U} \cup \mathcal{U}'$, $Y = I \cap \mathcal{U}$, and $U = \{p \mid p' \in (I \cap \mathcal{U}')\}$. Then, $I \models (\mathcal{U}' \leq \mathcal{U})$ iff $U \subseteq Y$, and $I \models (\mathcal{U}' < \mathcal{U})$ iff $U \subset Y$.

Making use of quantifiers, we next provide polynomial encodings for program equivalence. First, we introduce a module representing $ES_P(U)$, as given in (2), but without explicitly referring to certain sets U. Rather, a particular U is determined by the true atoms from a copy $\mathcal{U}'$ of $\mathcal{U}$. Given a rule r, we denote by r' the copy of r obtained by replacing each atom p with p', i.e., $r' = r[\mathcal{U}/\mathcal{U}']$.

Given the above notation, for a program P, we define

$$ES_P = \bigvee_{r\in P}\big(H(r') \wedge \bigwedge_{p\in H(r)}(p \rightarrow p') \wedge B(r) \wedge \bigwedge_{p\in B^+(r)}\neg p'\big).$$

For an interpretation Y over $\mathcal{U}$ and $U \subseteq \mathcal{U}$, we have $(Y \cup \{p' \mid p \in U\}) \models ES_P$ iff U is not unfounded for P with respect to Y. Regarding $ES_P(U)$, the following holds.

Lemma 5. *Let P be a program over $\mathcal{U}$, I an interpretation over $\mathcal{U} \cup \mathcal{U}'$, $Y = I \cap \mathcal{U}$, and $U = \{p \mid p' \in (I \cap \mathcal{U}')\}$. Then, $I \models ES_P$ iff $Y \models ES_P(U)$.*

Similar to $ES_P(U)$ and ES_P, we reformulate loop formulas $LF_P(U)$, as given in (3), without explicit reference to U:

$$LF_P = \forall\mathcal{U}'\big(\big((\bigvee_{p\in\mathcal{U}}p') \wedge (\mathcal{U}' \leq \mathcal{U})\big) \rightarrow ES_P\big).$$

We obtain the following counterpart of Proposition 3 in quantified propositional logic and afterwards reformulate the test for ordinary equivalence (as given by Theorem 6).

Lemma 6. *Let P be a program over $\mathcal{U}$ and Y an interpretation over $\mathcal{U}$ such that $Y \models P$. Then, Y is an answer set of P iff $Y \models LF_P$.*

Theorem 9. *Let P and Q be programs over $\mathcal{U}$. Then, P and Q are ordinarily equivalent iff $\forall\mathcal{U}\big((P \wedge LF_P) \leftrightarrow (Q \wedge LF_Q)\big)$ is valid.*

Modifying the encoding in Theorem 7 by using ES_P and ES_Q yields that deciding strong equivalence between P and Q amounts to checking whether

$$\forall\mathcal{U}\big((P \vee Q) \rightarrow \forall\mathcal{U}'(ES_P \leftrightarrow ES_Q)\big)$$

is valid. However, $\forall\mathcal{U}'$ can safely be placed in front of the formula, yielding

$$\forall\mathcal{U}\forall\mathcal{U}'\big((P \vee Q) \rightarrow (ES_P \leftrightarrow ES_Q)\big). \tag{5}$$

Verifying the validity of (5) can be done by checking whether the quantifier-free part in (5) is a tautology in standard propositional logic. We can thus state the following.

Theorem 10. *Let P and Q be programs over $\mathcal{U}$. Then, P and Q are strongly equivalent iff $(P \vee Q) \rightarrow (ES_P \leftrightarrow ES_Q)$ is a tautology.*

Finally, we consider uniform equivalence, where we first reformulate $ES^\star_P(U)$, as given in (4), in the same manner as we did above for $ES_P(U)$. To this end, define

$$ES^\star_P = ES_P \vee \exists\mathcal{U}''\big((\bigvee_{p\in\mathcal{U}}p'') \wedge (\mathcal{U}'' < \mathcal{U}') \wedge \neg ES_P[\mathcal{U}'/\mathcal{U}'']\big).$$

Similar to Lemma 5 above, we have the following.

Lemma 7. *Let P be a program over $\mathcal{U}$, I an interpretation over $\mathcal{U} \cup \mathcal{U}'$, $Y = I \cap \mathcal{U}$, and $U = \{p \mid p' \in (I \cap \mathcal{U}')\}$. Then, $I \models ES^\star_P$ iff $Y \models ES^\star_P(U)$.*

By replacing $ES^\star_P(U)$ with $ES^\star_P$, we obtain the following counterpart of Theorem 8.

Theorem 11. *Let P and Q be programs over $\mathcal{U}$. Then, P and Q are uniformly equivalent iff $\forall\mathcal{U}\forall\mathcal{U}'\big((P \vee Q) \rightarrow (ES^\star_P \leftrightarrow ES^\star_Q)\big)$ is valid.*

This encoding is very similar in shape to the one for strong equivalence in (5). In contrast to ES_P and ES_Q, $ES^\star_P$ and $ES^\star_Q$, however, contain additional quantifiers, which are needed to keep the size of the encoding polynomial with respect to P and Q. In fact, all of the above QBFs are constructible in polynomial time from P and Q. To the best of our knowledge, they are novel and differ from the encodings proposed in previous work [18–20].

5 Discussion

We have provided novel characterizations for program equivalence in terms of unfounded sets and related concepts, like loops and elementary sets. This allowed us to identify close relationships between these central concepts. While answer sets, and thus ordinary equivalence, rely on the absence of (nonempty) unfounded sets, we have shown that potential extensions of programs, captured by SE- and UE-models, can also be characterized directly by appeal to unfounded sets, thereby avoiding any reference to reducts of programs.

We have seen that uniform equivalence is located between ordinary and strong equivalence, in the sense that it considers all models, similar to strong equivalence, but only minimal unfounded sets, which also are sufficient to decide whether a model is an answer set. This allowed us to develop particularly simple characterizations for uniform equivalence in the case of tight programs. In fact, our results offer novel encodings for program equivalence in terms of (quantified) propositional logic, which are different from the ones found in the literature [18–20]. The respective encodings reflect the complexity of checking program equivalence: while checking strong equivalence is in coNP, in general, the checks for ordinary and uniform equivalence are Π^P_2-hard. For tight programs, however, our simpler characterization of uniform equivalence is reflected by the corresponding encoding (in standard propositional logic) and makes the complexity of checks drop into coNP, as with ordinary equivalence.

The relationship between (exponential size) propositional encodings and (polynomial size) QBF encodings has also (implicitly) been investigated by Ferraris, Lee, and Lifschitz [22], where they transform a QBF encoding [20] for answer sets into a propositional formula by eliminating quantifiers, in order to develop a general theory of loop formulas. In fact, Ferraris, Lee, and Lifschitz [22] address answer sets (in a more general setting than ours), but neither strong nor uniform equivalence.

References

1. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3-4) (1991) 365–385

2. Van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM **38**(3) (1991) 620–650
3. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
4. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135**(2) (1997) 69–112
5. Clark, K.: Negation as failure. In Gallaire, H., Minker, J., eds.: Logic and Data Bases. Plenum Press (1978) 293–322
6. Fages, F.: Consistency of Clark's completion and the existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
7. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157**(1-2) (2004) 115–137
8. Lifschitz, V., Razborov, A.: Why are there so many loop formulas? ACM Transactions on Computational Logic **7**(2) (2006) 261–268
9. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning **36**(4) (2006) 345–377
10. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7**(3) (2006) 499–562
11. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234
12. Lee, J.: A model-theoretic counterpart of loop formulas. In Kaelbling, L., Saffiotti, A., eds.: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05). Professional Book Center (2005) 503–508
13. Gebser, M., Lee, J., Lierler, Y.: Elementary sets for logic programs. In Gil, Y., Mooney, R., eds.: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06). AAAI Press (2006)
14. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4) (2001) 526–541
15. Eiter, T., Fink, M.: Uniform equivalence of logic programs under the stable model semantics. In Palamidessi, C., ed.: Proceedings of the 19th International Conference on Logic Programming (ICLP'03). Volume 2916 of LNCS, Springer (2003) 224–238
16. Turner, H.: Strong equivalence made easy: Nested expressions and weight constraints. Theory and Practice of Logic Programming **3**(4-5) (2003) 602–622
17. Eiter, T., Leone, N., Pearce, D.: Assumption sets for extended logic programs. In Gerbrandy, J., Marx, M., de Rijke, M., Venema, Y., eds.: JFAK. Essays Dedicated to Johan van Benthem on the Occasion of his 50th Birthday. Amsterdam University Press (1999)
18. Lin, F.: Reducing strong equivalence of logic programs to entailment in classical propositional logic. In Fensel, D., Giunchiglia, F., McGuinness, D., Williams, M., eds.: Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR'02). Morgan Kaufmann (2002) 170–176
19. Pearce, D., Tompits, H., Woltran., S.: Encodings for equilibrium logic and logic programs with nested expressions. In Brazdil, P., Jorge, A., eds.: Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA'01). Volume 2258 of LNCS, Springer (2001) 306–320
20. Pearce, D., Tompits, H., Woltran, S.: Characterising equilibrium logic and nested logic programs: Reductions and complexity. Technical Report GIA-TR-2007-12-01, Universidad Rey Juan Carlos (2007)
21. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming **3**(4-5) (2003) 499–518

22. Ferraris, P., Lee, J., Lifschitz, V.: A generalization of the Lin-Zhao theorem. Annals of Mathematics and Artificial Intelligence **47**(1-2) (2006) 79–101