Eliminating Disjunction from Propositional Logic Programs under Stable Model Preservation[⋆]

Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran

Institut für Informationssysteme 184/3, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,michael,tompits,stefan}@kr.tuwien.ac.at

Abstract. In general, disjunction is considered to add expressive power to propositional logic programs under stable model semantics, and to enlarge the range of problems which can be expressed. However, from a semantical point of view, disjunction is often not really needed, in that an equivalent program without disjunction can be given. We thus consider the question, given a disjunctive logic program P, does there exist an equivalent normal (i.e., disjunction-free) logic program P'? In fact, we consider this issue for different notions of equivalence, namely for ordinary equivalence (regarding the collections of all stable models of the programs) as well as for the more restrictive notions of strong and uniform equivalence. We resolve the issue for propositional programs, and present a simple, appealing semantic criterion for the programs from which all disjunctions can be eliminated under strong equivalence; testing this criterion is coNP-complete. We also show that under ordinary and uniform equivalence, this elimination is always possible. In all cases, there are constructive methods to achieve this. Our results extend and complement recent results on simplifying logic programs under different notions of equivalence, and add to the foundations of improving implementations of Answer Set Solvers.

1 Introduction

Disjunctive logic programming is an extension to normal logic programming which is generally considered to add expressive power to logic programs under stable model semantics, and to enlarge the range of problems which can be expressed. This view is supported by results on the expressiveness of disjunctive logic programs (DLPs) over finite structures, which show that properties at the second level of the Polynomial Hierarchy (PH) can be expressed by inference from function-free (Datalog) DLPs [11], while by normal logic programs only properties at the first level can be expressed [28]. However, from a semantical point of view, disjunction is often not really needed, in that an equivalent normal logic program (NLP, i.e., without disjunction) can be given. For example, in [10], it was shown that in the presence of functions symbols, DLPs have over Herbrand models the same expressive power as NLPs, namely Π^1_1.

With the rise of Answer Set Programming as a program solving paradigm, in which solutions are computed in the answer sets resp. stable models of a logic program, attention has been directed to the expressiveness of logic programs in terms of the whole

[⋆] This work was partially supported by the Austrian Science Fund (FWF) under projects Z29-N04 and P15068-INF, as well as by the European Commission under projects FET-2001-37004 WASP and IST-2001-33570 INFOMIX.

collection of their answer sets per se rather than their intersection (resp. union) as in cautious and brave reasoning, respectively), cf. [20]; related to this is preliminary work on the expressiveness of formalisms such as default logic and circumscription [13, 19].

In particular, equivalence of logic programs in terms of their collections of stable models has been considered, as well as the refined notions of strong equivalence, cf. [16, 29, 30, 24, 17, 4], and uniform equivalence [7, 8, 25], which dates back to [27, 18]. Two DLPs P_1 and P_2 are strongly equivalent (resp., uniformly equivalent), if, for any set R of rules (resp., set of atoms R), the programs $P_1 \cup R$ and $P_2 \cup R$ are equivalent under the stable semantics, i.e., have the same set of stable models.

Strong and uniform equivalence can be utilized for program optimization, cf. [30, 22, 8], taking into account possible incompleteness of a program, where not all rules are known at the time of optimization, respectively varying input data given by atomic facts are respected. This is in particular helpful for optimizing components which are embedded into a more complex logic program. Note that as recently discussed by Pearce and Valverde [25], uniform and strong equivalence are essentially the only concepts of equivalence obtained by varying the logical form of the program extensions.

A natural issue in this context is the expressiveness of disjunction in rule heads, i.e., whether it really adds expressive power. This is indeed the case, as can be seen on the simple example of the program $P = \{a \vee b \leftarrow\}$: This program is not strongly equivalent to any normal logic program P' (cf. [30]). However, as easily seen P is equivalent to the NLP $P' = \{a \leftarrow \mathit{not}\, b,\ b \leftarrow \mathit{not}\, a\}$ since for both the stable models are $X_1 = \{a\}$ and $X_2 = \{b\}$, and furthermore P is also uniformly equivalent to P' (this is immediate from the result that rewriting a head-cycle free program to a normal logic program by standard shifting preserves uniform equivalence [7]). On the other hand, the enriched program $P = \{a \vee b \leftarrow\ ,\quad \leftarrow a, b\}$ is strongly equivalent to the program $P' = \{a \leftarrow \mathit{not}\, b,\ b \leftarrow \mathit{not}\, a,\quad \leftarrow a, b\}$.

This raises the question of a criterion which tells when disjunctions can be eliminated, and a method for deciding, given a disjunctive logic program P, does there exist an equivalent normal (i.e., disjunction-free) program P'? We study this issue for propositional programs, on which we focus here, and make the following contributions:

- We present a simple, appealing semantic characterization of the programs from which all disjunctions can be eliminated under strong equivalence. The characterization is based on the strong-equivalence models (SE-models) [29, 30] which rephrase models in the more general logic of here-and-there [16] in logic programming terms. In fact, we show that this property holds for a program P if and only if the collection of SE-models (X, Y) of P is closed under *here-intersection*, i.e., whenever (X, Y) and (X', Y) are SE-models of P, then also $(X \cap X', Y)$ is an SE-model of P. In more familiar terms, this condition is equivalent to the property that for each classical model Y of P, the Gelfond-Lifschitz reduct P^Y of P is semantically Horn if models X not contained in Y are disregarded.
- We further show that under ordinary and uniform equivalence, this elimination is always possible. In all three cases, we obtain a constructive method to rewrite a DLP P to an equivalent normal logic program P'. In general, the rewriting will be of exponential size (if it exists), but this will be unavoidable in practice.
- Finally, we show that testing whether for a given propositional DLP P a strongly equivalent NLP P' exists is coNP-complete.

Our results extend and complement recent results on simplifying logic programs under different notions of equivalence, cf. [22, 30, 8]. They might be utilized for deciding whether a given disjunctive problem representation for a system such as such as DLV [6] or GnT [14] can, in principle, be replaced by an equivalent non-disjunctive representation, as well as for (automated) rewriting of disjunctive problem representations.

2 Preliminaries

We deal with propositional disjunctive logic programs, containing rules r of form

$$a_1 \vee \cdots \vee a_l \leftarrow a_{l+1}, \ldots, a_m, \mathit{not}\, a_{m+1}, \ldots, \mathit{not}\, a_n,$$

$n \geq m \geq l \geq 0$, where all a_i are atoms from a finite set of propositional atoms, At, and not denotes default negation. The *head* of r is the set $H(r) = \{a_1, \ldots, a_l\}$, and the *body* of r is the set $B(r) = \{a_{l+1}, \ldots, a_m, \mathit{not}\, a_{m+1}, \ldots, \mathit{not}\, a_n\}$.We also use $B^+(r) = \{a_{l+1}, \ldots, a_m\}$ and $B^-(r) = \{a_{m+1}, \ldots, a_n\}$. Moreover, for a set of atoms $A = \{a_1, \ldots, a_n\}$, $\mathit{not}\, A$ denotes $\{\mathit{not}\, a_1, \ldots, \mathit{not}\, a_n\}$.

A rule r is *normal*, if $l \leq 1$; *positive*, if $n = m$; and *Horn*, if it is normal and positive. If $H(r) = \emptyset$ and $B(r) \neq \emptyset$, then r is a *constraint*; if $B(r) = \emptyset$, r is a *fact*, written as $a_1 \vee \cdots \vee a_l$ if $l > 0$, and as $\bot$ otherwise.

A *disjunctive logic program* (DLP) P is a finite set of rules. It is a *normal logic program* (NLP) (resp., positive, Horn), if every $r \in P$ is normal (resp., positive, Horn).

We recall the stable model semantics for DLPs [12, 26]. Let I be an interpretation, i.e., a subset of At. Then, I *satisfies* a rule r, denoted $I \models r$, iff $I \models H(r)$ whenever $I \models B(r)$, where $I \models H(r)$ iff $a \in I$ for some $a \in H(r)$, and $I \models B(r)$ iff (i) each $a \in B^+(r)$ is *true* in I, i.e., $a \in I$, and (ii) each $a \in B^-(p)$ is *false* in I, i.e., $a \notin I$. Furthermore, I is a model of a program P, denoted $I \models P$, iff $I \models r$, for all $r \in P$.

The *reduct*, r^I, of a rule r *relative to* a set of atoms I is the positive rule r' such that $H(r') = H(r)$ and $B(r') = B^+(r)$ if $I \cap B^-(r) = \emptyset$, and is void otherwise. The *Gelfond-Lifschitz reduct* of a program P is the positive program $P^I = \{r^I \mid r \in P\}$. An interpretation I is a *stable model* of a program P iff I is a minimal model (under set inclusion) of P^I. By $\mathcal{SM}(P)$ we denote the set of all stable models of P.

Lemma 1. *Let P be a DLP and $X \subseteq Y' \subseteq Y$. Then, $X \models P^{Y'}$ implies $X \models P^Y$.*

The result is seen by the observation that $Y' \subseteq Y$ implies $P^Y \subseteq P^{Y'}$. Thus, $X \models P^{Y'}$ implies $X \models P^Y$. In particular, for $X = Y'$, $X \models P$ implies $X \models P^Y$, for any $X \subseteq Y$.

Several notions of equivalence for logic programs have been considered in the literature (see, e.g., [16, 18, 27]). Under stable semantics, two DLPs P and Q are regarded as equivalent, denoted $P \equiv Q$, iff $\mathcal{SM}(P) = \mathcal{SM}(Q)$. The more restrictive forms of *strong equivalence* [16] and *uniform equivalence* [27, 18] are as follows:

Definition 1. *Let P and Q be two DLPs. Then,*

(*i*) *P and Q are* strongly equivalent, *denoted $P \equiv_s Q$, iff, for any set R of rules, the programs $P \cup R$ and $Q \cup R$ are equivalent, i.e., $P \cup R \equiv Q \cup R$.*

(*ii*) *P and Q are* uniformly equivalent, *denoted $P \equiv_u Q$, iff, for any set F of non-disjunctive facts, $P \cup F$ and $Q \cup F$ are equivalent, i.e., $P \cup F \equiv Q \cup F$.*

Obviously, $P \equiv_s Q$ implies $P \equiv_u Q$ but not vice versa. Both notions of equivalence, however, enjoy interesting semantical characterizations. As shown in [16], strong equivalence is closely related to the non-classical logic of here-and-there, which was adapted to logic-programming terms by Turner [29, 30]:

Definition 2. *A pair (X, Y) with $X, Y \subseteq At$ such that $X \subseteq Y$ is called an* SE-interpretation *(over At). By INT_{At} we denote the of all SE-interpretations over At. An SE-interpretation (X, Y) is an* SE-model *of a DLP P, if $Y \models P$ and $X \models P^Y$. By $M_s(P)$ we denote the set of all SE-models of P.*

Proposition 1 ([29, 30]). *For every DLP P and Q, $P \equiv_s Q$ iff $M_s(P) = M_s(Q)$.*

SE-models also can be used to determine the stable models of a program.

Proposition 2 ([23, 16]). *Let P be a DLP. Then, $Y \in \mathcal{SM}(P)$ iff $(Y, Y) \in M_s(P)$ and, for each $X \subset Y$, $(X, Y) \notin M_s(P)$.*

Recently, the following pendant to SE-models, characterizing uniform equivalence for (finite) logic programs has been defined [7].

Definition 3. *Let P be a DLP and $(X, Y) \in M_s(P)$. Then, (X, Y) is* UE-model *of P iff, for every $(X', Y) \in M_s(P)$, it holds that $X \subset X'$ implies $X' = Y$. By $M_u(P)$ we denote the set of all UE-models of P.*

Proposition 3 ([7]; cf. also [25]). *For any DLP P and Q, $P \equiv_u Q$ iff $M_u(P) = M_u(Q)$.*

This test can be reformulated as follows.

Proposition 4. *For DLPs P and Q, $P \equiv_u Q$ iff $M_u(P) \subseteq M_s(Q)$ and $M_u(Q) \subseteq M_s(P)$.*

Proof. From Proposition 3, $P \equiv_u Q$ iff $M_u(P) \subseteq M_u(Q)$ and $M_u(Q) \subseteq M_u(P)$. Clearly, $M_u(R) \subseteq M_s(R)$ holds for any DLP R, which immediately gives the only-if direction. For the if direction, suppose $P \not\equiv_u Q$. Hence, there exists an SE-interpretation (X, Y), such that either (i) $(X, Y) \in M_u(P)$ and $(X, Y) \notin M_u(Q)$; or (ii) $(X, Y) \in M_u(Q)$ and $(X, Y) \notin M_u(P)$. For (i), we have two cases, by definition of UE-models. First, $(X, Y) \notin M_s(Q)$. But then, $M_u(P) \subseteq M_s(Q)$ cannot hold. Second, there exists a set X' with $X \subset X' \subset Y$ such that $(X', Y) \in M_u(Q)$. But $(X', Y) \notin M_s(P)$ since $(X, Y) \in M_u(P)$, hence $M_u(Q) \subseteq M_s(P)$ cannot hold. The argument for (ii) is analogous. □

As a final result here, we characterize the set of SE-models of a disjunctive rule.

Proposition 5. *Let r be a disjunctive rule, and (X, Y) an SE-interpretation. Then, $(X, Y) \in M_s(r)$ holds iff one of the following conditions is satisfied: (i) $X \models H(r)$; (ii) $Y \not\models B(r)$; or (iii) $X \not\models B^+(r)$ and $Y \models H(r)$.*

Proof. By definition, $(X, Y) \in M_s(r)$ iff $Y \models r$, and $X \models r^Y$. The former holds iff $Y \models H(r)$, $Y \not\models B^+(r)$, or $Y \not\models B^-(r)$. The latter holds iff $X \models H(r)$, $X \not\models B^+(r)$, or $Y \not\models B^-(r)$. Hence, $(X, Y) \in M_s(r)$ iff $Y \not\models B^-(r)$ or

$$\Big(Y \models H(r) \lor Y \not\models B^+(r)\Big) \land \Big(X \models H(r) \lor X \not\models B^+(r)\Big). \qquad (1)$$

Clearly, $Y \not\models B^+(r)$ implies $X \not\models B^+(r)$ and, furthermore, $X \models H(r)$ implies $Y \models H(r)$. From this, it is easily verified that (X, Y) satisfies (1) iff either $Y \not\models B^+(r)$, $X \models H(r)$ (i.e., (i)), or jointly $X \not\models B^+(r)$ and $Y \models H(r)$ (i.e., (iii)), holds. Hence, we have that $(X, Y) \in M_s(P)$ iff either $Y \not\models B^-(r)$, $Y \not\models B^+(r)$, (i), or (iii) holds. Finally, $Y \not\models B^-(r)$ or $Y \not\models B^+(r)$ holds exactly iff $Y \not\models B(r)$ (i.e., (ii)) holds. □

3 Strong Equivalence

We start with some informal discussion. Consider the following logic programs, each of them having $r = a \lor b \leftarrow$ as its only disjunctive rule.

$P_1 = \{a \lor b \leftarrow\}$ $\quad$ $P_2 = \{a \lor b \leftarrow; a \leftarrow\}$
$P_3 = \{a \lor b \leftarrow; a \leftarrow b\}$ $\quad$ $P_4 = \{a \lor b \leftarrow; a \leftarrow; \leftarrow \mathit{not}\, b\}$
$P_5 = \{a \lor b \leftarrow; a \leftarrow b; \leftarrow \mathit{not}\, b\}$ $\quad$ $P_6 = \{a \lor b \leftarrow; a \leftarrow; b \leftarrow\}$
$P_7 = \{a \lor b \leftarrow; a \leftarrow b; b \leftarrow a\}$ $\quad$ $P_8 = \{a \lor b \leftarrow; \leftarrow a, b\}$
$P_9 = \{a \lor b \leftarrow; \leftarrow \mathit{not}\, a; \leftarrow \mathit{not}\, b\}$

Let us first compute the SE-models (over $At = \{a, b\}$) of these programs:[1]

$$\begin{aligned}
M_s(P_1) &= \{\ (ab, ab),\ (a, ab),\ (b, ab),\ (a, a),\ (b, b)\ \};\\
M_s(P_2) = M_s(P_3) &= \{\ (ab, ab),\ (a, ab),\ (a, a)\ \};\\
M_s(P_4) = M_s(P_5) &= \{\ (ab, ab),\ (a, ab)\ \};\\
M_s(P_6) = M_s(P_7) &= \{\ (ab, ab)\ \};\\
M_s(P_8) &= \{\ (a, a),\ (b, b)\ \};\\
M_s(P_9) &= \{\ (ab, ab)\ (a, ab),\ (b, ab)\ \}.
\end{aligned}$$

A good approximation to derive corresponding strongly equivalent *normal* logic programs is to replace $a \lor b \leftarrow$ by the rules $a \leftarrow \mathit{not}\, b; b \leftarrow \mathit{not}\, a$, i.e., by the usual shifting technique. It is left to the reader to verify that this replacement works for P_2, P_4, P_6, and P_8, but not for P_1, P_3, P_5, P_7, and P_9. In fact, for the latter programs this replacement yields an additional SE-model $(\emptyset, ab)$. In some of these cases we can circumvent this problem by adding further rules. As is easily seen, adding $a \leftarrow$ to P_3, P_5, and P_7, respectively, solves this problem, since $(\emptyset, ab) \notin M_s(a \leftarrow)$, and, for each $(X, Y) \in M_s(P_i)$, $(X, Y) \in M_s(a \leftarrow)$ holds ($i \in \{3, 5, 7\}$). For P_1 and P_9 this does not work. As we will see soon, there is no normal logic program strongly equivalent to P_1 or P_9.

Let us have a closer look at the difference between the SE-models of a disjunctive rule and its corresponding shifting rules.

Proposition 6. *For a disjunctive rule, r, define*

$r^{\rightarrow} = \{p \leftarrow B(r), \mathit{not}\,(H(r) \setminus \{p\}) \mid p \in H(r)\}$; *and*
$S_r = \{(X, Y) \in \mathit{INT}_{At} \mid X \not\models H(r), X \models B^+(r), Y \models B^-(r), |Y \cap H(r)| > 1\}$.

Then, $M_s(r^{\rightarrow}) = M_s(r) \cup S_r$.

[1] We write ab instead of $\{a, b\}$, a instead of $\{a\}$, etc.

Proof. In what follows let r_p denote that rule in $r^{\rightarrow}$ with $H(r_p) = p$.

Clearly, we have $S_r \subseteq M_s(r^{\rightarrow})$ and $M_s(r) \subseteq M_s(r^{\rightarrow})$. In particular, the former relation can be seen by the fact that, for each $r_p \in r^{\rightarrow}$, $Y \not\models B^-(r_p)$ holds, since $|Y \cap H(r)| > 1$ and $(H(r) \setminus \{p\}) \subseteq B^-(r_p)$.

It remains to show $M_s(r^{\rightarrow}) \subseteq M_s(r) \cup S_r$. Therefore, let $(X, Y) \in M_s(r^{\rightarrow})$ and suppose $(X, Y) \notin M_s(r)$. We show that $(X, Y) \in S_r$. Towards a contradiction, suppose $(X, Y) \notin S_r$. Since $(X, Y) \notin M_s(r)$, we get by Proposition 5 that $X \not\models H(r)$, $Y \models B(r)$, and either $X \models B^+(r)$ or $Y \not\models H(r)$. We have two cases.

First, if $Y \not\models H(r)$, i.e., $|Y \cap H(r)| = 0$, this clearly contradicts $(X, Y) \in M_s(r^{\rightarrow})$. Otherwise, if $Y \models H(r)$, we have $X \models B^+(r)$, and we get $|Y \cap H(r)| = 1$, otherwise $(X, Y) \in S_r$. Let $Y \cap H(r) = \{p\}$ and consider the rule r_p. Obviously, $Y \models B^-(r_p)$. Moreover, we have $X \models B^+(r_p) = B^+(r)$. But then, $X \not\models H(r)$, i.e., $X \cap H(r) = \emptyset$, yields $p \notin X$, and thus contradicts $(X, Y) \in M_s(r^{\rightarrow})$. □

In other words, S_r contains all SE-interpretations (X, Y) such that $X \not\models r^Y$ with $Y \models B(r)$ and $|Y \cap H(r)| > 1$. Thus, $M_s(r)$ and S_r are disjoint, although, for each $(X, Y) \in S_r$, $Y \models r$ holds.

Hence, if we have a disjunctive program P with a disjunctive rule $r \in P$; we can replace r by $r^{\rightarrow}$ but this may yield additional SE-models from S_r. (The same observation can be found in [7], see Theorem 4.3). In particular, this is the case if $T = M_s(P \setminus \{r\}) \cap S_r \neq \emptyset$. So, in addition to $r^{\rightarrow}$, we add the following rules which under certain circumstances eliminate all elements from T but none from $M_s(P)$.

Proposition 7. *For any sets, $X, Y, Z \subseteq At$, define the set of rules*

$$r_{X,Y,Z} = \{p \leftarrow X, \mathit{not}\,(At \setminus Z) \mid p \in Y\}.$$

Then, $(X', Y') \in INT_{At}$ is SE-model of $r_{X,Y,Z}$ iff one of the following conditions holds: (1) $Y \subseteq X'$; (2) $X \not\subseteq Y'$; (3) $Y' \not\subseteq Z$; or (4) $X \not\subseteq X'$ and $Y \subseteq Y'$.

Proof. Let $p \in Y$ and r_p the corresponding rule in $r_{X,Y,Z}$ with $H(r_p) = p$. By Proposition 5, $(X', Y') \in M_s(r_p)$ iff one of the following conditions hold: (i) $X' \models H(r_p)$ (i.e., $p \in X'$); (ii) $Y' \not\models B(r_p)$ (i.e., either $X \not\subseteq Y'$ or $Y' \not\subseteq Z$); or (iii) $X' \not\models B^+(r_p)$ and $Y' \models H(r_p)$ (i.e., $X \not\subseteq X'$ and $p \in Y'$). For any two rules $r_p, r_q \in r_{X,Y,Z}$, we have $B(r_p) = B(r_q)$. Thus, $(X', Y') \in M_s(r_{X,Y,Z})$ iff either (ii), or, for any $r_p \in r_{X,Y,Z}$, (i) or (iii) holds. For the latter relations consider two cases. If $X \subseteq X'$, the relation holds iff $p \in X'$ for each $r_p \in r_{X,Y,Z}$, i.e., for each $p \in Y$. Hence, the relation holds iff $Y \subseteq X'$. If $X \not\subseteq X'$ and $Y \not\subseteq X'$ then from (iii) $p \in Y'$ for each $p \in Y$. Hence, $Y \subseteq Y'$ has then to hold. □

The rules $r_{X,Y,Z}$ play a central role and will be subsequently applied in several ways. However, we mention that $r_{X,Y,Z}$ may contain redundant rules, for instance if we have $X \subseteq Y$. It can be shown that then, $r_{X,Y,Z} \equiv_s r_{X,Y \setminus X,Z}$ which reduces the number of rules. However, for technical reasons we subsequently do not pay attention to this potential optimization.

Definition 4 (here-intersection). [2] *For any pair of SE-interpretations, (X, Y) and (X', Y) of INT_{At}, their* here-intersection *is the SE-interpretation $(X \cap X', Y)$.*

[2] The first component of an SE-interpretation refers to the world of 'here' when logically interpreted in the logic of here-and-there, cf. [23].

Recall our examples. The difference between P_1 and P_9 on the one side, and P_3, P_5, and P_7 on the other side, is captured by the following property.

Definition 5. *For any logic program P, we say that P is closed under here-intersection iff $M_s(P)$ satisfies this property.*

Lemma 2. *Each normal LP is closed under here-intersection.*

Proof. Since P is normal, P^Y is Horn. Then, $X \models P^Y$ and $X' \models P^Y$ immediately implies $X \cap X' \models P^Y$ since, each Horn program P', satisfies the intersection property: $X \models P'$ and $X' \models P'$ implies $X \cap X' \models P'$. □

This leads us directly to our first central theorem.

Theorem 1. *Let P be a DLP over At. Then, there exists a NLP Q over At, such that P and Q are strongly equivalent iff P is closed under here-intersection.*

Proof. The only-if direction is immediate by Lemma 2. For the if direction, assume P is closed under here-intersection, let $r \in P$ be disjunctive, let $P_r^- = P \setminus \{r\}$, and consider the logic program

$$P_s \quad = \quad P_r^- \cup r^{\rightarrow} \cup \hat{r}_{P_s} \quad \text{with} \quad \hat{r}_{P_s} = \bigcup_{(X,Y,Z) \in S_r^{\uparrow}(P)} r_{X,Y,Z} \,,$$

where

$$S_r^{\uparrow}(P) \;=\; \{(X,Y,Z) \mid (X,Z) \in S_r \cap M_s(P_r^-), X \subseteq Y, (Y,Z) \in M_s(P), \\ \forall Y' : X \subseteq Y' \subset Y \;\Rightarrow\; (Y',Z) \notin M_s(P)\}.$$

Intuitively, $S_r^{\uparrow}(P)$ collects, for each $(X,Z) \in S_r$ which is also SE-model of P_r^-, the minimal SE-models (Y,Z) of P above X (with fixed Z). Note that by definition of S_r, for any $(X,Z) \in INT_{At}$, $(X,X,Z) \notin S_r^{\uparrow}(P)$ but $(X,Z) \in S_r$ implies existence of an Y with $X \subseteq Y \subseteq Z$ such that $(X,Y,Z) \in S_r^{\uparrow}(P)$, since for $Y = Z$, $(Y,Z) \in M_s(P)$ holds (again by definition of S_r).

We proceed with the proof and show $M_s(P) = M_s(P_s)$. Clearly, if this holds, the above transformation sequentially applied to all disjunctive rules in P yields a normal logic program strongly equivalent to P. First observe that, by Proposition 6, we have

$$\begin{aligned}
M_s(P_s) &= M_s(P_r^- \cup r^{\rightarrow} \cup \hat{r}_{P_s}) \\
&= M_s(P_r^-) \cap M_s(r^{\rightarrow}) \cap M_s(\hat{r}_{P_s}) \\
&= M_s(P_r^-) \cap (M_s(r) \cup S_r) \cap M_s(\hat{r}_{P_s}) \\
&= \Big(M_s(P_r^-) \cap M_s(r) \cap M_s(\hat{r}_{P_s})\Big) \cup \Big(M_s(P_r^-) \cap S_r \cap M_s(\hat{r}_{P_s})\Big) \\
&= \Big(M_s(P) \cap M_s(\hat{r}_{P_s})\Big) \cup \Big(M_s(P_r^-) \cap S_r \cap M_s(\hat{r}_{P_s})\Big).
\end{aligned}$$

The strategy for the remainder of the proof is as follows. We first show that $T = M_s(P_r^-) \cap S_r \cap M_s(\hat{r}_{P_s}) = \emptyset$. Then, it remains to show $M_s(P_s) = M_s(P) \cap M_s(\hat{r}_{P_s})$.

We show $T = \emptyset$. Let $(X,Z) \in S_r$. If $(X,Z) \notin M_s(P_r^-)$, we immediately have $(X,Z) \notin T$. Otherwise, $(X,Z) \in M_s(P_r^-)$. From above we know that then there exists a triple $(X, Y, Z) \in S_r^{\uparrow}(P)$ with $X \subset Y \subseteq Z$. Thus, we have $r_{X,Y,Z} \subseteq \hat{r}_{P_s}$. We now show that $(X, Z) \notin M_s(r_{X,Y,Z})$. Assume the contrary, i.e., $(X, Z) \in M_s(r_{X,Y,Z})$. Then, by Proposition 7 one of the following conditions has to hold: (i) $Y \subseteq X$, (ii) $X \not\subseteq Z$, (iii) $Z \not\subseteq Z$, or (iv) $X \not\subseteq X$ and $Y \subseteq Z$. (i) does not hold since we have $X \subset Y$; (ii) since $X \subset Z$, which follows from $X \subset Y \subseteq Z$; (iii+iv) do not hold trivially. Contradiction.

It remains to show that $M_s(P) = M_s(P) \cap M_s(\hat{r}_{P_s})$, i.e., that $M_s(P) \subseteq M_s(\hat{r}_{P_s})$ holds. Clearly, if $\hat{r}_{P_s}$ is empty we are done. Hence, suppose $\hat{r}_{P_s} \neq \emptyset$. We show that, for each $r_{X,Y,Z} \subseteq \hat{r}_{P_s}$ and each $(X', Z') \in M_s(P)$, $(X', Z') \in M_s(r_{X,Y,Z})$ holds. Towards a contradiction, let $r_{X,Y,Z} \subseteq \hat{r}_{P_s}$, $(X', Z') \in M_s(P)$, and suppose $(X', Z') \notin M_s(r_{X,Y,Z})$. On the one hand, from $r_{X,Y,Z} \subseteq \hat{r}_{P_s}$, we have $(X, Y, Z) \in S_r^{\uparrow}(P)$, which implies that (a) $(X, Z) \in S_r \cap M_s(P_r^-)$; (b) $(Y, Z) \in M_s(P)$; and (c) $X \subset Y \subseteq Z$ hold. On the other hand, by Proposition 7, (X', Z') satisfies the following conditions for being a counter SE-model of $r_{X,Y,Z}$: (1) $Y \not\subseteq X'$, (2) $X \subseteq Z'$, (3) $Z' \subseteq Z$, and (4) $X \subseteq X'$ or $Y \not\subseteq Z'$.

By assumption, $(X', Z') \in M_s(P)$. Hence, $X' \models P^{Z'}$ and $Z' \models P^{Z'}$. Moreover, by (3), $Z' \subseteq Z$ holds. By Lemma 1, we get $X' \models P^Z$ and $Z' \models P^Z$, and from (b) we get $(Y, Z) \in M_s(P)$, and thus $Z \models P$. Hence, $(X', Z) \in M_s(P)$ and $(Z', Z) \in M_s(P)$. Now P is closed under here-intersection yielding $(Y \cap X', Z) \in M_s(P)$, and $(Y \cap Z', Z) \in M_s(P)$. We use (4) to distinguish between the following two cases.

$X \subseteq X'$: By (c), $X \subset Y$, and thus $X \subseteq (Y \cap X')$. On the other hand, from (1), we have $Y \not\subseteq X'$. This implies $(Y \cap X') \subset Y$. Hence, we have $X \subseteq (Y \cap X') \subset Y$. Together with (a) and $(Y \cap X', Z) \in M_s(P)$ this clearly contradicts $(X, Y, Z) \in S_r^{\uparrow}(P)$, since Y is not minimal anymore.

$X \not\subseteq X'$: We have a similar argumentation. By (4), $Y \not\subseteq Z'$ holds, yielding $(Y \cap Z') \subset Y$. Moreover, by (2) $X \subseteq Z'$ and by (c) $X \subset Y$ hold. Thus, $X \subseteq (Y \cap Z')$. Again, we have $X \subseteq (Y \cap Z') \subset Y$, and by (a) and $(Y \cap Z', Z) \in M_s(P)$, this contradicts $(X, Y, Z) \in S_r^{\uparrow}(P)$. □

Let us apply the construction of P_s as used in the proof to the examples discussed in the beginning of the section (except P_1 and P_9 which are not closed under here-intersection). They all have $r = a \vee b \leftarrow$ as their only disjunctive rule. Hence, $S_r = \{(\emptyset, ab)\}$. Consider now $S_r^{\uparrow}(P_i)$. Clearly, if $S_r \cap M_s((P_i)_r^-) = \emptyset$, r is just replaced by $r^{\rightarrow}$. This applies to $i \in \{2, 4, 6, 8\}$. For programs $i \in \{3, 5, 7\}$, $S_r \cap M_s((P_i)_r^-) = \{(\emptyset, ab)\}$, and by the SE-models of the respective programs we get $S_r^{\uparrow}(P_3) = S_r^{\uparrow}(P_5) = \{(\emptyset, a, ab)\}$ and $S_r^{\uparrow}(P_7) = \{(\emptyset, ab, ab)\}$. Thus in P_3 and P_5, exchange r by $r^{\rightarrow} \cup r_{\emptyset,a,ab}$, with $r_{\emptyset,a,ab} = \{a \leftarrow\}$ (under assumption $At = \{a, b\}$). This goes well conform from our informal analysis in the beginning of the section. Finally, for P_7, we use $r_{\emptyset,ab,ab}$ instead of $r_{\emptyset,a,ab}$, yielding the following normal program $\{a \leftarrow \mathit{not}\, b; b \leftarrow \mathit{not}\, a; a \leftarrow; b \leftarrow; a \leftarrow b; b \leftarrow a;\}$ which is obviously strongly equivalent to $\{a \leftarrow; b \leftarrow;\}$. The latter has (ab, ab) as its only SE-model and thus is strongly equivalent to P_7.

4 Uniform Equivalence

The intuitive problems in constructing a uniform equivalent normal logic program from a given disjunctive program are very similar to those observed in the case of strong equivalence. Consider a program having as its only disjunctive rule r. Again, a good starting point is to replace r by its corresponding shifting rules $r^{\rightarrow}$. But now, an SE-model (X, Y) from S_r only comes into play, iff it is also UE-model of the rest program, i.e., for each X' with $X \subset X' \subseteq Y$, $(X', Y) \in M_s(P \setminus \{r\})$ implies $X' = Y$. Thus, if we want to eliminate such an SE-model, the problem of eliminating further SE-models, which should be retained, is less complicated compared to the case of strong equivalence. Roughly speaking, because of this difference we are *always* able to construct a uniformly equivalent normal program. For instance, all our example programs P_1–P_9 except P_7 are uniformly equivalent to the program resulting from P_i with r replaced by $r^{\rightarrow}$. P_7, however, matches exactly the case where $(\emptyset, ab)$ is UE-model of the rest program $P_7 \setminus \{r\}$. Adding rules as $a \leftarrow$ or $b \leftarrow$ (or both of them) circumvents the problem. Hence, in some (but less) cases we again have to add further rules, but as is seen in proof below, the difference to the construction of P_s is very subtle.

Theorem 2. *For each DLP there exists a uniform equivalent NLP.*

Proof. Again, we give a step-by-step transformation. So, let r be a disjunctive rule in a DLP P, $P_r^- = P \setminus \{r\}$, and consider the program

$$P_u \quad = \quad P_r^- \cup r^{\rightarrow} \cup \hat{r}_{P_u} \quad \text{with} \quad \hat{r}_{P_u} = \bigcup_{(X,Z,Z) \in S_r^{\uparrow}(P)} r_{X,Z,Z} \,,$$

where $S_r^{\uparrow}(P)$ is defined as in the proof of Theorem 1. Note that the only difference between P_s and P_u is that here we just consider those triples (X, Y, Z) from $S_r^{\uparrow}(P)$ where $Y = Z$. To proceed with the proof, observe that analogously to the proof of Theorem 1, we get

$$M_s(P_u) = \Big(M_s(P) \cap M_s(\hat{r}_{P_u})\Big) \cup \Big(M_s(P_r^-) \cap S_r \cap M_s(\hat{r}_{P_u})\Big). \tag{2}$$

We show $M_u(P) = M_u(P_u)$. By Proposition 4, this holds iff both $M_u(P) \subseteq M_s(P_u)$ and $M_u(P_u) \subseteq M_s(P)$ hold.

We first show $M_s(P) \subseteq M_s(P_u)$, which clearly implies $M_u(P) \subseteq M_s(P_u)$ immediately. Note that if $\hat{r}_{P_u}$ is empty we are done, since then $M_s(P) \subseteq M_s(P) \cap M_s(\hat{r}_{P_u})$ holds trivially. So consider $\hat{r}_{P_u} \neq \emptyset$. We show that, for each $r_{X,Y,Y} \subseteq \hat{r}_{P_u}$ and each $(X', Y') \in M_s(P)$, $(X', Y') \in M_s(r_{X,Y,Y})$ holds. Towards a contradiction, let $r_{X,Y,Y} \subseteq \hat{r}_{P_u}$, $(X', Y') \in M_s(P)$, and suppose $(X', Y') \notin M_s(r_{X,Y,Y})$. On the one hand, since $r_{X,Y,Y} \subseteq \hat{r}_{P_u}$, we have (a) $(X, Y) \in S_r \cap M_s(P_r^-)$, (b) $X \subset Y$, and (c) for each SE-interpretation (U, Y) with $X \subseteq U$, $(U, Y) \in M_s(P)$ implies $U = Y$. On the other hand, (X', Y') satisfies the following conditions for being a counter SE-model of $r_{X,Y,Y}$, by Proposition 7: (i) $Y \not\subseteq X'$, (ii) $X \subseteq Y'$, (iii) $Y' \subseteq Y$, and (iv) $X \subseteq X'$ or $Y \not\subseteq Y'$. We use (iv) for distinguishing between the following two cases:

First, assume $X \subseteq X'$. Clearly, $X' \subseteq Y'$ holds and by (i) $X' \neq Y$ and by (iii) $Y' \subseteq Y$ hold. We get $X \subseteq X' \subset Y$. Moreover, $X' \models P^{Y'}$ holds, since $(X', Y') \in M_s(P)$. By Lemma 1, we get $X' \models P^{Y}$ and $Y \models P$ holds by (a), i.e., since $(X, Y) \in S_r$. Hence $(X', Y) \in M_s(P)$, which clearly is in contradiction to (c).

Second, assume $X \not\subseteq X'$. By (iv), then $Y \not\subseteq Y'$. Together with (iii) we have $Y' \subset Y$. Moreover, $X \subseteq Y'$ holds by (ii). Since $(X', Y') \in M_s(P)$, $Y' \models P^{Y'}$ holds.

Lemma 1 yields $Y' \models P^Y$, and since $Y \models P$, we have $(Y',Y) \in M_s(P)$ with $X \subseteq Y' \subset Y$. Again this is in contradiction to (c).

It remains to show $M_u(P_u) \subseteq M_s(P)$. In fact we show $M_u(P_u) \cap T = \emptyset$ with $T = M_s(P_r^-) \cap S_r \cap M_s(\hat{r}_{P_u})$. By inspecting (2), it is seen that $M_u(P_u) \cap T = \emptyset$ implies $M_u(P_u) \subseteq M_s(P) \cap M_s(\hat{r}_{P_u})$ which shows the claim since $M_s(P) \cap M_s(\hat{r}_{P_u}) \subseteq M_s(P)$ holds trivially. To derive $M_u(P_u) \cap T = \emptyset$, we show, for any $(X,Y) \in S_r \cap M_s(P_r^-)$, that either $(X,Y) \notin M_u(P_u)$ or $(X,Y) \notin M_s(\hat{r}_{P_u})$. So, fix some $(X,Y) \in S_r \cap M_s(P_r^-)$. We consider two cases.

Assume $(X,Y,Y) \notin S_r^{\uparrow}(P)$. Hence, there exists a set $X \subseteq X' \subset Y$ such that $(X',Y) \in M_s(P)$. We know that $(X,X,Y) \notin S_r^{\uparrow}(P)$. Thus, $X \subset X'$. We already have shown $M_s(P) \subseteq M_s(P_u)$, yielding $(X',Y) \in M_s(P_u)$. But then, $(X,Y) \notin M_u(P_u)$, since $X \subset X' \subset Y$.

So assume $(X,Y,Y) \in S_r^{\uparrow}(P)$ and thus $r_{X,Y,Y} \subseteq \hat{r}_{P_u}$. However, we have $(X,Y) \notin M_s(r_{X,Y,Y})$, since none of the following conditions from Proposition 7 is satisfied: (i) $Y \subseteq X$, (ii) $X \not\subseteq Y$, (iii) $Y \not\subseteq Y$, or (iv) $X \not\subseteq X$ and $Y \subseteq Y$. For (i+ii) this is seen by the fact that $(X,Y) \in S_r$ and thus $X \subset Y$. (iii+iv) trivially fail. Hence, $(X,Y) \notin M_s(\hat{r}_{P_u})$. □

As already discussed above, the only program from our examples P_1–P_9 which is not uniformly equivalent after replacing r by $r^{\rightarrow}$ is P_7. However, since P_7 is closed under here-intersection, we already know how to derive a strongly (and thus uniformly) equivalent normal logic program from the proof of Theorem 1. In fact, one can verify that $(P_7)_s = (P_7)_u$. For an example program P which is not closed under here-intersection and has $\hat{r}_{P_u} \neq \emptyset$, with $r \in P$ disjunctive, consider $P = \{a \vee b \leftarrow; a \leftarrow c, b; b \leftarrow c, a\}$ over $At = \{a, b, c\}$. The SE-models of P are given as follows:

$$M_s(P) = \{(abc, abc), (ab, abc), (ab, ab), \\ (a, abc), (a, ab), (a, a), (b, abc), (b, ab), (b, b)\}.$$

Indeed, P is not closed under here-intersection. Let $r = a \vee b$. We have that S_r is given by $\{(\emptyset, ab), (\emptyset, abc), (c, abc)\}$ and as can be verified, $S_r \cap M_s(P_r^-) = S_r$. Moreover,

$$S_r^{\uparrow}(P) = \{(\emptyset, a, ab), (\emptyset, b, ab), (\emptyset, a, abc), (\emptyset, b, abc), (c, abc, abc)\}.$$

Only the last triple, (c, abc, abc), is applied in the construction of $\hat{r}_{P_u}$. In fact, we have to add $r_{c,abc,abc}$ (which is given by $\{a \leftarrow c; b \leftarrow c; c \leftarrow c\}$) to $P_r^- \cup r^{\rightarrow}$. For the resulting program P_u, we then have $M_s(P_u) = M_s(P) \cup \{(\emptyset, ab), (\emptyset, abc)\}$, but the “critical” SE-interpretation, (c, abc), has been eliminated. In fact, $M_u(P) = M_u(P_u)$ holds, since neither $(\emptyset, ab)$ nor $(\emptyset, abc)$ is UE-model of P_u.

5 Ordinary Equivalence

Finally, we discuss how to derive normal logic programs which are ordinary equivalent to given disjunctive ones. By Theorem 2, such programs always exist and, for a given program P, P_u clearly does the job, since uniform equivalence implies ordinary equivalence. However, we give two further alternatives. The first is motivated by an *enumeration* of stable models, while the second one *optimizes* P_u. To start with, we state the following result which is easily seen from Proposition 7.

Proposition 8. *$M_s(r_{\emptyset,Y,Y}) = \{(X',Y') \in INT_{At} \mid Y \subseteq X'$ or $Y' \not\subseteq Y\}$, for any $Y \subseteq At$.*

Theorem 3. *Let P be DLP. Then, $\mathcal{SM}(P) = \mathcal{SM}(\tilde{P})$ with $\tilde{P} = \bigcup_{Y \in \mathcal{SM}(P)} r_{\emptyset,Y,Y}$.*

Proof. Let $Y \in \mathcal{SM}(P)$. Then, for each $Y' \in \mathcal{SM}(P)$, either $Y \not\subseteq Y'$ or $Y = Y'$. By Proposition 8, $(Y,Y) \in M_s(\tilde{P})$. Towards a contradiction, suppose $Y \notin \mathcal{SM}(\tilde{P})$. By Proposition 2, there exists a $X \subset Y$ such that $(X,Y) \in M_s(\tilde{P})$. In particular, we must have $(X,Y) \in r_{\emptyset,Y,Y}$. But by Proposition 8, it is easily seen that this is contradicted by $X \subset Y$. Thus, $Y \in \mathcal{SM}(\tilde{P})$.

Conversely, suppose $Y \in \mathcal{SM}(\tilde{P})$ and $Y \notin \mathcal{SM}(P)$. That is, $(X,Y) \in M_s(P)$ for some $X \subset Y$. Then, $Y \not\subseteq Y'$ for each $Y' \in \mathcal{SM}(P)$. We have $(X,Y) \in r_{\emptyset,Y',Y'}$, for each $Y' \in \mathcal{SM}(P)$ by Proposition 8. Thus, $(X,Y) \models \tilde{P}$ with $X \subset Y$. But this contradicts $Y \in \mathcal{SM}(\tilde{P})$. □

The following construction would be more "structure-preserving" and, to some extend, "optimizes" the program P_u.

Theorem 4. *Let P be DLP and $r \in P$. Then, $\mathcal{SM}(P) = \mathcal{SM}(P_e)$ with*

$$P_e = P_r^- \cup r^{\rightarrow} \cup \hat{r}_{P_e} \quad \textit{with} \quad \hat{r}_{P_e} = \{r_{\emptyset,Z,Z} \mid r_{X,Z,Z} \subseteq \hat{r}_{P_u},\ Z \in \mathcal{SM}(P)\}.$$

Proof. Let $Y \in \mathcal{SM}(P)$. Obviously, $Y \models P^-$ and, by Proposition 6, $Y \models r^{\rightarrow}$. Moreover, $\hat{r}_{P_e} \subseteq \tilde{P}$, which implies, by Theorem 3, $Y \models \hat{r}_{P_e}$. Thus, $Y \models P_e$. It remains to show that no $X \subset Y$, yields an SE-model (X,Y) of P_e. Towards a contradiction, suppose some $X \subset Y$ such that $(X,Y) \in M_s(P_e)$. Clearly, $(X,Y) \in M_s(P_r^-)$, hence, since $Y \in \mathcal{SM}(P)$, $(X,Y) \notin M_s(r)$ must hold. Then, by Proposition 6, $(X,Y) \in S_r$. We have $(X,Y) \in S_r \cap M_s(P_r^-)$ and $Y \in \mathcal{SM}(P)$, and thus get $r_{\emptyset,Y,Y} \in P_e$ by construction. By Proposition 8, $(X,Y) \notin M_s(r_{\emptyset,Y,Y})$ which contradicts $(X,Y) \in M_s(P_e)$. Thus, no $(X,Y) \in M_s(P_e)$ with $X \subset Y$ exists. This means $Y \in \mathcal{SM}(P_e)$.

Conversely, let $Y \in \mathcal{SM}(P_e)$. This implies $Y \models P$. Towards a contradiction, suppose $Y \notin \mathcal{SM}(P)$, i.e., there exists $X \subset Y$, such that $(X,Y) \in M_s(P)$. By Proposition 6, $(X,Y) \in M_s(P_r^- \cup r^{\rightarrow})$. Since $Y \notin \mathcal{SM}(P)$, we have $Y \not\subseteq Y'$, for each $Y' \in \mathcal{SM}(P)$. By Proposition 8, $(X,Y) \in M_s(r_{\emptyset,Y',Y'})$ for each Y' with $Y \not\subseteq Y'$. Hence, $(X,Y) \in M_s(P_e)$. By Proposition 2, this contradicts $Y \in \mathcal{SM}(P_e)$. □

To summarize, given a DLP P with $r \in P$ disjunctive, we are able to construct (via a replacement of r by normal rules) a logic program P_e which is ordinary equivalent to P; a program P_u which is uniformly equivalent to P; and, whenever P is closed under here-intersection, a program P_s which is strongly equivalent to P. All these programs are of the form

$$P_r^- \cup r^{\rightarrow} \cup \hat{r}_{P_\alpha} \text{ for } \alpha \in \{e,u,s\}.$$

By definition of $\hat{r}_{P_\alpha}$, we furthermore can state that $|P_e| \leq |P_u|$ and $P_u \subseteq P_s$. Hence, our method can be seen as a uniform framework for all important notions of equivalence. Moreover, our results extend and generalize methods based on shifting disjunctions, since the outcome of these methods coincides with the presented rewriting P_α, whenever $\hat{r}_{P_\alpha}$ is empty. In particular, concerning equivalence in terms of stable models, we present a semantic criterion (in contrast to the syntactic criterion of head-cycle free programs, cf. [1]) which allows for shifting. Concerning equivalence in terms of UE-models, we generalize an observation made in [7] (see Theorem 4.3).

6 Complexity Issues

Finally, we deal with some related complexity issues. We can express the test for a DLP P of being closed under here-intersection via the following normal logic program, which is linear in the size of P.

Definition 6. *Let P be a DLP over atoms V and let $\bar{V}$, V_1', $\bar{V}_1'$, V_2', $\bar{V}_2'$, V_3', and u be disjoint new atoms. Define*

$$\begin{aligned} P_Q = & \{v \leftarrow \mathit{not}\,\bar{v};\ \bar{v} \leftarrow \mathit{not}\,v \mid v \in V\} & (3)\\ & \cup \{v_i' \leftarrow v, \mathit{not}\,\bar{v}_i';\ \bar{v}_i' \leftarrow \mathit{not}\,v_i' \mid v \in V, i \in \{1,2\}\} & (4)\\ & \cup \{\leftarrow B(r), \mathit{not}\,H(r) \mid r \in P\} & (5)\\ & \cup \{\leftarrow B^+(r_i'), \mathit{not}\,B^-(r), \mathit{not}\,H(r_i') \mid r \in P, i \in \{1,2\}\} & (6)\\ & \cup \{v_3' \leftarrow v_1', v_2' \mid v \in V\} & (7)\\ & \cup \{u \leftarrow B^+(r_3'), \mathit{not}\,B^-(r), \mathit{not}\,H(r_3') \mid r \in P\} & (8)\\ & \cup \{\leftarrow \mathit{not}\,u\}. & (9) \end{aligned}$$

Intuitively, the program P_Q works as follows. Rules (3) guess an interpretation Y of P and rules (5) check that Y is a model of P. Similarly, rules (4) guess subsets X_1 and X_2 of Y such that both are models of P^Y, which is enforced by (6). Hence, (3)–(6) 'compute' all pairs of SE-models (X_1, Y) and (X_2, Y) of P.

Now, rules (7) compute the intersection $X_1 \cap X_2$ and via rules (8) the new atom u can be derived iff the intersection does not model P^Y, i.e. iff $(X_1 \cap X_2, Y)$ is no SE-model of P. The constraint (9) kills all models of P_Q for which this is not the case, i.e. for which $(X_1 \cap X_2, Y)$ is an SE-model of P. Thus, (7)–(9) ensure that P_Q has no stable model iff P is closed under here-intersection. Formally, we have:

Theorem 5. *A DLP P is closed under here-intersection iff P_Q has no stable model.*

Based on this, we derive the following complexity result.

Theorem 6. *Let P be a DLP. Then, checking whether there exists a normal logic program Q strongly equivalent to P is complete for* coNP.

Proof. By Theorem 5, and the linear encoding from Definition 6 we get that closedness under here-intersection is in coNP. By Theorem 1, we immediately get the coNP-membership part.

We show coNP-hardness of this problem by a reduction to the coNP-complete problem of deciding whether At is the unique model of a positive DLP.

Let P be a positive DLP over the alphabet At, let v, v' be new atoms, and consider the program

$$Q = P \cup \{v \vee v' \leftarrow;\ v \leftarrow At;\ v' \leftarrow At\}.$$

We prove that Q is closed under here-intersection iff At is the unique model of P.

The if direction is straight forward: If At is the unique model of P then, by construction, $At \cup \{v, v'\}$ is the unique model of Q, and since Q is positive – and hence constant under reduction – Q is trivially closed under here-intersection.

For the only-if direction assume Q is closed under here-intersection and, towards a proof by contradiction, suppose there exists a model $M \subset At$ of P. Then both, $M \cup \{v\}$ and $M \cup \{v'\}$ are models of Q and thus also both, $(M \cup \{v\}, At \cup \{v, v'\})$ and $(M \cup \{v'\}, At \cup \{v, v'\})$ are SE-models of Q. However, $(M, At \cup \{v, v'\})$ is not an SE-model of Q, since $M \not\models v \vee v' \leftarrow$. This contradicts the assumption that Q is closed under here-intersection, hence At is the unique model of P.

We have shown coNP-hardness of deciding whether a DLP P is closed under here-intersection which immediately implies coNP-hardness of checking whether there exists a normal logic program Q strongly equivalent to P by Theorem 1. □

Another interesting issue is the size of the rewriting of a given DLP P into an equivalent NLP Q (if it exists). Using the constructive methods presented in this paper the rewriting is of exponential size, in general. However there is strong evidence that this is unavoidable. Let $\mathcal{DLP}$ and $\mathcal{NLP}$ denote the classes of all DLPs and NLPs, respectively (over atoms At).

Theorem 7. *For each Π_2^P-hard family $\mathcal{F}$ of DLPs such that there exist e-equivalent NLPs, there does not exist a polynomial rewriting $f : \mathcal{F} \to \mathcal{NLP}$ such that $P \equiv_e f(P)$, $e \in \{u, s\}$, for every DLP P of $\mathcal{F}$, unless the Polynomial Hierarchy (PH) collapses.*

Proof. (Sketch) Towards a contradiction, assume that for a positive DLP P, a polynomial rewriting $f : P \to \mathcal{NLP}$ exists and consider the Π_2^P-hard problem of checking whether, for some atom A, $not\,A$ is a cautious consequence of P ([9]).

Then, we could guess an NLP $f(P) =: P'$ in nondeterministic polynomial time. Furthermore, checking $P \equiv_e P'$ is in coNP (even in case of uniform equivalence, since P is positive and P' is normal, i.e. head-cycle free [7]). As well, checking whether $P' \models_c not\,A$ is in coNP (since P' is normal). Thus, the Π_2^P-hard problem of deciding $P \models_c not\,A$ would be in Σ_2^P, a contradiction unless PH collapses. □

Also for rewritings under ordinary equivalence, we cannot avoid an exponential blow up unless PH collapses.

Theorem 8. *There is no polynomial rewriting $f : \mathcal{DLP} \to \mathcal{NLP}$ such that $P \equiv f(P)$ for every $P \in \mathcal{DLP}$, unless PH collapses.*

Proof. (Sketch) This can be shown by using nonuniform complexity classes as in [3, 2, 13], following closely the line of proof there that the existence of a polynomial model-preserving mapping $g : CIRC \to PL$ from propositional circumscription $CIRC$ to classical propositional logic PL would imply that coNP $\subseteq$ P/poly, i.e., the class of problems decidable in polynomial time with polynomial advice. This inclusion implies a collapse of PH. The proof can be easily adapted, where the mapping g, $CIRC$, and PL are replaced with f, (positive) $\mathcal{DLP}$, and $\mathcal{NLP}$, respectively. □

Clearly Theorem 7 implies Theorem 8, but the proof of the latter does not refer to non-uniform complexity classes, and is thus more accessible. We remark that in terms of [13], $\mathcal{DLP}$ is a stronger formalism than $\mathcal{NLP}$ unless PH collapses. Furthermore, Theorem 8 remains true for generalized rewritings f which admit projective extra variables, i.e., $P \equiv f(P)|_{At}$, where $f(P)$ is on atoms $At' \supseteq At$ and $f(P)_{At}$ denotes the restriction of the stable models of $f(P)$ to the original atoms At. Indeed, such an f would imply coNP$\subseteq$NP/poly, which again means that PH collapses.

7 Conclusion

In this work, we derived new results on the relationship between propositional disjunctive and normal logic programs under the stable model semantics, by investigating whether disjunctions in a given program P can be replaced by normal rules, in such a way that this modification does not change the set of SE-models (resp. UE-models, stable models) of P. In a bigger picture, such a rewriting technique allows to obtain a strongly (resp. uniformly, ordinary) equivalent normal logic program from a given disjunctive logic program.

Our results show that under ordinary and uniform equivalence, this rewriting is always possible. In the case of strong equivalence we identified an appealing semantic criterion based on so-called here-intersection. The rewriting itself is based on the well-known (local) shifting of disjunctive rules, but extends this method by an addition of further rules, which take the semantic of the entire program into account. Hence, this rewriting is in general hard to obtain, and has to be exponential in the worst case under widely accepted complexity theoretic assumptions.

These results complement recent considerations on simplification techniques under different notions of equivalence, cf. [22, 30, 8], and thus add to the foundations of improving implementations of Answer Set Solvers.

Moreover, we showed that the problem of deciding whether a DLP is closed under here-intersection, i.e., deciding whether there exists a strongly equivalent NLP, is complete for coNP, answering a question left open in [8]. As a by-product, we presented an encoding of this test via a normal logic program. Note, however, that this test may also be treated by SAT-solvers. Thus, we also contributed to a line of research in ASP, which relies on the application of classical propositional logic (or QBFs) to deal with certain problems in logic programming [15, 5, 24].

Further issues concern a closer investigation of adding extra variables, as well as of the newly derived class of DLPs closed under here-intersection. One the one hand, we are interested in the expressibility of such programs. On the other hand, it is an open question whether this class allows for optimizations of algorithms used in disjunctive logic programming engines such as DLV.

References

1. R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics & Artificial Intelligence*, 12:53–87, 1994.
2. M. Cadoli, F. Donini, and M. Schaerf. Space efficiency of propositional knowledge representation formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.
3. M. Cadoli, F. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. *Theoretical Computer Science*, 182(1-2):183–202, 2000.
4. D. J. de Jongh and L. Hendriks. Characterizations of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, 3(3):259–270, 2003.
5. U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified Boolean formulas. In *Proc. AAAI-00*, pp. 417–422. AAAI / MIT Press, 2000.
6. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative problem-solving using the DLV system. In J. Minker, ed., *Logic-Based Artificial Intelligence*, pp. 79–103. Kluwer, 2000.

7. T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. Tech. Rep. INFSYS RR-1843-03-08, Inst. für Informationssysteme, TU Wien, 2003. Preliminary Report. Short version in *Proc. ICLP-03*, to appear.
8. T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. Manuscript, submitted, July 2003.
9. T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics & Artificial Intelligence*, 15(3/4):289–323, 1995.
10. T. Eiter and G. Gottlob. Expressiveness of stable model semantics for disjunctive logic programs with functions. *Journal of Logic Programming*, 33(2):167–178, 1997.
11. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM TODS*, 22(3):364-417, 1997.
12. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
13. G. Gogic, H. Kautz, C. H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proc. IJCAI-95*, pp. 862–869. Morgan Kaufmann, 1995.
14. T. Janhunen, I. Niemelä, P. Simons, and J.-H. You. Partiality and disjunctions in stable model semantics. In *Proc. KR-00*, pp. 411–419. Morgan Kaufmann, 2000.
15. C. Koch and N. Leone. Stable model checking made easy. In *Proc. IJCAI-99*, pp. 70–75. Morgan Kaufmann, 1999.
16. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
17. F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. KR-02*, pp. 170–176. Morgan Kaufmann, 2002.
18. M. J. Maher. Equivalences of logic programs. In Minker [21], pp. 627–658.
19. V. W. Marek, J. Treur, and M. Truszczyński. Representation theory for default logic. *Annals of Mathematics & Artificial Intelligence*, 21(2–4):343–358, 1997.
20. W. Marek and J. Remmel. On the expressibility of stable logic programming. *Theory and Practice of Logic Programming*, 3(4-5):551–567, 2003.
21. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufman, Washington DC, 1988.
22. M. Osorio, J. Navarro, and J. Arrazola. Equivalence in answer set programming. In *Proc. LOPSTR-01*, LNCS 2372, pp. 57–75. Springer, 2001.
23. D. Pearce. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming (NMELP 1996)*, LNCS 1216, pp. 57–70. Springer, 1997.
24. D. Pearce, H. Tompits, and S. Woltran. Encodings for equilibrium logic and logic programs with nested expressions. In *Proc. EPIA-01*, LNCS 2258, pp. 306–320. Springer, 2001.
25. D. Pearce and A. Valverde. Some types of equivalence for logic programs and equilibrium logic. In *Proc. Joint Conf. on Declarative Programming (APPIA-GULP-PRODE)*, 2003.
26. T. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
27. Y. Sagiv. Optimizing Datalog programs. In Minker [21], pp. 659–698.
28. J. Schlipf. The expressive powers of logic programming semantics. *Journal of Computer and System Sciences*, 51(1):64–86, 1995. Abstract in Proc. PODS 90, pp. 196–204.
29. H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proc. LPNMR-01*, LNCS 2173, pp. 81–92. Springer, 2001.
30. H. Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4–5):609–622, 2003.