Signed Logic Programs

Hudson Turner
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA
hudson@cs.utexas.edu

Abstract

In this paper we explore the notion of a "signing" of a logic program, in the framework of the answer set semantics. In particular, we generalize and extend the notion of a signing, and show that even for programs with classical negation and disjunction the existence of a signing is a simple syntactic criterion that can guarantee several different sorts of good behavior: consistency, coincidence of consequences under answer set and well-founded semantics, existence of "standard" answer sets expressible in terms of the well-founded model and a signing for the program, and a restricted monotonicity property. The key technical result in this paper is a theorem relating the consequences of a signed disjunctive program with classical negation to the consequences of the members of a closely related family of signed nondisjunctive programs. These nondisjunctive programs are the "covers" of the disjunctive program, where a cover is any program that can be obtained by removing all but one literal from the head of each rule in the disjunctive program. To illustrate the usefulness of these results, we apply them to a family of programs for reasoning about action.

1 Introduction

In this paper we explore the notion of a "signing" of a logic program, in the framework of the answer set (or stable model) semantics [Gelfond and Lifschitz, 1991]. In particular, we generalize and extend the notion of a signing, and show that even for programs with classical negation and disjunction the existence of a signing is a simple syntactic criterion that guarantees several different sorts of good behavior.

The notion of a signing for a normal logic program (that is, a program without classical negation and disjunction) was introduced by Kunen [1989], who used it as a tool in his proof that two-valued and three-valued completion semantics coincide on the class of "strict" normal programs. For Kunen, the notion was defined on the predicate dependency graph of a finite first-order program, so when Gelfond and Lifschitz [1993] recast the definition to apply directly to the rules of infinite propositional normal programs, the notion of a signing was made strictly more general. In [Turner, 1993] the definition was extended to the class of nondisjunctive programs with classical negation. In this paper it is generalized slightly and further extended to apply to programs with disjunction as well as classical negation.

For a normal program P, a signing S is a set of atoms such that for every rule in P either (i) the head and the positive atoms in the body belong to S and the negated atoms do not, or (ii) the head and the positive atoms in the body do not belong to S and the negated atoms do. From the perspective of answer sets for normal programs, signings are already known to be interesting for the following four reasons.

1. Signed normal programs are consistent. This is a special case of a more general theorem by Fages [1994], who has shown that "order-consistent" normal programs are consistent.

2. If S is a signing for a normal program P, then P has two "standard" answer sets that are expressible in terms of S and the well-founded model of P. [Turner, 1993]

3. The consequences of a signed normal program under the answer set semantics coincide with its consequences under the well-founded semantics. This is a special case of a more general result due to Dung [1992], who has shown that the answer set and well-founded semantics coincide for "bottom-stratified & top-strict" programs. Notice that this result shows that interpreters such as SLG [Chen and Warren, 1993], which compute the well-founded semantics, can also be used to compute the consequences of such programs under the answer set semantics.[1]

4. There is a monotonicity theorem for signed normal programs. [Turner, 1993]

In [Turner, 1993] we showed that some of these results can be extended to nondisjunctive programs with classical negation. In this paper we generalize these previous results slightly, and also extend them in various ways to signed programs with disjunction as well as classical negation.

For a disjunctive program P with classical negation, a signing S is a subset of the literals of the language of P, satisfying several simple syntactic conditions. (The precise definition appears in Section 3.) We show that under this extension of the notion of a signing, the following properties hold.

1. Signed disjunctive programs without classical negation are consistent. (Corollary 4.) A similar result is known for "locally stratified" disjunctive programs without classical negation.[2] Since some signed disjunctive programs without classical negation are not locally stratified, the consistency of such disjunctive programs is a new result.

2. If S is a signing for a consistent nondisjunctive program P with classical negation, then P has a "standard" answer set expressible in terms of S and a naive extension of the well-founded semantics.[3] (Theorem 1(iii).)

3. If S is a signing for a consistent nondisjunctive program P with classical negation, then the consequences of P in the complement of S are also expressible in terms of a naive extension of the well-founded semantics. (Corollary 1.) This result corresponds to Lemma 8 of [Turner, 1993]. Similarly, if S is a signing for a disjunctive program P with classical negation, then the consequences of P in the complement of S can be characterized in terms of a syntactically determined family of signed nondisjunctive programs with classical negation. (Theorem 2, Corollary 3.)

[1] In [Lifschitz *et al.*, 1993] we show that SLG can also be used, under certain qualifications, to correctly compute the consequences of signed nondisjunctive programs with classical negation.

[2] The definition of "local stratification" is due to Przymusinski [1988]. The consistency of locally stratified disjunctive programs without classical negation under the answer set semantics is clear from Przymusinski's similar result under the perfect model semantics.

[3] This result is implicit in [Turner, 1993].

4. A generalization of the monotonicity theorem from [Turner, 1993] applies to all signed programs.[4] (Theorem 4.)

The key technical result in this paper is a theorem (Theorem 2) relating the consequences of a signed disjunctive program to the consequences of the members of a closely related family of signed nondisjunctive programs. These nondisjunctive programs are the "covers" of the disjunctive program, where a cover is any program that can be obtained by removing all but one literal from the head of each rule in the disjunctive program. The notion of a cover was introduced in the paper "Disjunctive Defaults" [Gelfond *et al.*, 1991] in order to explore the possibility of reducing a "disjunctive default theory" to a family of (nondisjunctive) default theories. The authors showed by counterexample that in general this cannot be done in any straightforward manner. On the other hand, it follows from their results (Theorems 6.2, 6.3 and 7.2) that for any disjunctive logic program P with classical negation: (i) every answer set for P is a minimal member of the set of answer sets for covers of P, and (ii) if P is positive (that is, includes no negation as failure), then X is an answer set for P if and only if X is a minimal member of the set of answer sets for covers of P. From (ii) it follows that the consequences of a positive disjunctive program P with classical negation coincide with the intersection of the consequences of all covers of P. In this paper we extend this result by showing that if S is a signing for a disjunctive program P with classical negation, then the consequences of P in the complement of S coincide with the intersection of the consequences, in the complement of S, of the covers of P.

To illustrate the usefulness of these results, we apply them to a family of programs for reasoning about action. Gelfond and Lifschitz [1993] defined a high-level language $\mathcal{A}$ for reasoning about action, and a sound translation from $\mathcal{A}$ to nondisjunctive logic programs with classical negation. We define in this paper a slight extension $\mathcal{A}_d$ of the language $\mathcal{A}$, in which disjunctive information about the values of fluents can be expressed, and we specify a translation from $\mathcal{A}_d$ into signed disjunctive logic programs with classical negation. We use our results on the properties of signed programs, along with the Splitting Sequence Theorem from [Lifschitz and Turner, 1994], to prove this translation sound and complete.

Section 2 consists of preliminary definitions and observations, after which we define the notion of a signing and give some examples (Section 3), and characterize useful properties of nondisjunctive programs with signings (Section 4). The theorem relating a signed disjunctive program to its signed nondisjunctive covers is discussed in Section 5, along with results on the existence of consistent answer sets for signed disjunctive programs. In Section 6 we discuss the restricted monotonicity theorem for signed programs. These various results are applied to a family of signed programs for reasoning about actions (Section 7), and finally we conclude with a few additional remarks (Section 8). Proofs are omitted due to lack of space.

2 Answer Sets

We begin with a brief review of the syntax and semantics of logic programs.

To specify a language $\mathcal{L}$ for logic programs, we can begin with a nonempty set of symbols called *atoms*. A *literal* of $\mathcal{L}$ is an atom of $\mathcal{L}$ possibly preceded by the classical negation symbol $\neg$. A *rule* in $\mathcal{L}$ is determined by three finite subsets of

[4] In [Turner, 1993] the monotonicity theorem is applied to nondisjunctive programs only, and even in this special case it is slightly less general than the monotonicity theorem in this paper.

the literals of $\mathcal{L}$—the set of *head literals*, the set of *positive subgoals* and the set of *negated subgoals*. The rule with the head literals $L_1, \ldots, L_l$, the positive subgoals $L_{l+1}, \ldots, L_m$ and the negated subgoals $L_{m+1}, \ldots, L_n$ is written as

$$L_1 \mid \ldots \mid L_l \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n\ .$$

We will denote the three parts of a rule r by $\mathit{head}(r)$, $\mathit{pos}(r)$ and $\mathit{neg}(r)$. A *program* is a set of rules in a language $\mathcal{L}$. For convenience, we often use $\mathcal{L}_P$ to denote the set of all literals of the language of a program P.

A program P is *positive* if, for every rule $r \in P$, $\mathit{neg}(r)$ is empty. The notion of an answer set is first defined for positive programs, as follows. Let P be a positive program and let X be a subset of $\mathcal{L}_P$. We say that X is *closed* under P if, for every rule $r \in P$ such that $\mathit{pos}(r) \subset X$, $\mathit{head}(r) \cap X$ is nonempty. (We write $X \subset Y$ when X is a subset of Y, not necessarily proper.) We say that X is *logically closed* (with respect to $\mathcal{L}_P$) if X is consistent or $X = \mathcal{L}_P$. An *answer set* for P is a minimal set of literals that is both closed under P and logically closed (with respect to $\mathcal{L}_P$).

Now let P be an arbitrary program, with X a subset of $\mathcal{L}_P$. For each rule $r \in P$ such that $\mathit{neg}(r) \cap X$ is empty, consider the rule r' defined by $\mathit{head}(r') = \mathit{head}(r)$, $\mathit{pos}(r') = \mathit{pos}(r)$, $\mathit{neg}(r') = \emptyset$. The positive program consisting of all rules r' obtained in this way is the *reduct* of P relative to X, denoted by P^X. We say that X is an *answer set* for P if X is an answer set for P^X.

A program P *entails* exactly those literals from $\mathcal{L}_P$ that are included in every answer set for P. By $\mathit{Cn}(P)$ we denote the set of literals entailed by program P. Finally, P is *consistent* if $\mathit{Cn}(P)$ is consistent, and *inconsistent* otherwise.

We will at times be interested in the following classes of programs. A program P is *constraint-free* if for every rule $r \in P$, $\mathit{head}(r)$ is nonempty. A program P is *nondisjunctive* if for every rule $r \in P$, $\mathit{head}(r)$ is a singleton.[5] Notice that nondisjunctive programs are constraint-free. A program is *basic* if it is positive and nondisjunctive.

Traditionally, programs without classical negation have been of great interest. In particular, nondisjunctive programs without classical negation (that is, "normal" programs) have been extensively studied. In this paper, instead of making stipulations about the presence of classical negation, we'll generally prefer the following more general condition.

Definition. For any program P, $\mathit{Head}(P) = \bigcup_{r \in P} \mathit{head}(r)$. We say that P is *head-consistent* if $\mathit{Head}(P)$ is a consistent set.

Observe that if a set X of literals is closed under a positive program P, then so is $X \cap \mathit{Head}(P)$. It follows that every minimal set closed under a head-consistent program is consistent. For our purposes, this is the most salient property possessed by programs without classical negation but not possessed by programs in general. So we'll usually speak about head-consistent programs instead of programs without classical negation, and about head-consistent nondisjunctive programs instead of normal programs.

[5]Nondisjunctive programs are also known as *extended* programs [Gelfond and Lifschitz, 1990]. The objects we here call logic programs are also known as *extended disjunctive* programs [Gelfond and Lifschitz, 1991].

3 Signings

Definition. Let P be a constraint-free program, with S a subset of $\mathcal{L}_P$ such that no literal in $S \cap Head(P)$ appears complemented in $Head(P)$. We say that S is a *signing* for P if each rule $r \in P$ satisfies the following two conditions:

- $head(r) \cup pos(r) \subset S$ and $neg(r) \subset \overline{S}$, or
  $head(r) \cup pos(r) \subset \overline{S}$ and $neg(r) \subset S$,
- if $head(r) \subset S$, then $head(r)$ is a singleton,

where $\overline{S} = \mathcal{L}_P \setminus S$. If a program has a signing, we say that it is *signed*.[6]

Notice that every constraint-free positive program has the signing $S = \emptyset$. We also observe that for programs without classical negation, the class of signed programs and the class of locally stratified programs overlap, and neither contains the other.

Consider the following program P_1.

$$\begin{array}{rcl} a & \leftarrow & not\ b \\ b & \leftarrow & not\ a \\ \neg a & \leftarrow & \end{array}$$

Program P_1 has a signing $S = \{b\}$. Observe that neither $\{a, \neg a\}$ nor $\{a\}$ is a signing for P, since we have $a, \neg a \in Head(P)$. So we see that the definition of a signing is asymmetric. That is, if a program P with signing S is not head-consistent, then $\overline{S}$ is not a signing for P, because there is a literal in $\overline{S} \cap Head(P)$ that appears complemented in $Head(P)$.

We will want the following definition in order to describe a second asymmetry in the definition of a signing.

Definition. Let P be a program. If S is a signing for P, then

$$\begin{array}{rcl} h_S(P) & = & \{r \in P : head(r) \subset S\} \ , \\ h_{\overline{S}}(P) & = & \{r \in P : head(r) \subset \overline{S}\} \ . \end{array}$$

Let r be a rule in a program P with signing S. Since signed programs are constraint-free, the set $head(r)$ is nonempty. Thus, either $head(r) \subset S$ or $head(r) \subset \overline{S}$, but not both. It follows that a signing S divides a program P into a pair of disjoint subprograms: $h_S(P)$ and $h_{\overline{S}}(P)$.

Consider the following program P_2.

$$\begin{array}{rcl} a \mid b & \leftarrow & \\ b & \leftarrow & not\ c \\ c & \leftarrow & not\ a \end{array}$$

Program P_2 has a signing $S = \{c\}$. So the last rule of P_2 belongs to $h_S(P_2)$, and $h_{\overline{S}}(P_2)$ consists of the first two rules of P_2. Because of the second condition in the

[6]Even in the special case of nondisjunctive programs, this definition is more general than the one proposed in [Turner, 1993]. There, a signing for a nondisjunctive program P is defined as a set S of atoms that satisfies condition (i) above and also includes no atom whose complement appears in P.

definition of a signing, the set $\{a, b\}$ is not a signing for P_2. And in general, if S is a signing for a program P, then $h_S(P)$ is a nondisjunctive program. So we see a second asymmetry in the definition of a signing. That is, if a program P with signing S is not nondisjunctive, then $\overline{S}$ is not a signing for P, because $h_{\overline{S}}(P)$ is not a nondisjunctive program.

In later discussion we will indicate why these asymmetries in the definition of a signing are necessary.

4 Signed Nondisjunctive Programs

For the most part, the results in this section can also be found, in a slightly less general form, in [Turner, 1993], either explicitly or implicitly.

Definition. Let P be a basic program. By $\alpha(P)$ we denote the least subset of $\mathcal{L}_P$ that is closed under P.

For any basic program P, it's clear that if $\alpha(P)$ is a consistent set, then $\alpha(P)$ is the unique answer set for P and $\alpha(P) = Cn(P)$. On the other hand, if $\alpha(P)$ is inconsistent, then $\mathcal{L}_P$ is the unique answer set for P. Observe that $\alpha(P) \subset Head(P)$.

Definition. Let P be a nondisjunctive program. For every $X \subset \mathcal{L}_P$, $\Gamma_P X = \alpha(P^X)$.

Observe that the answer sets for a nondisjunctive program P are the fixpoints of Γ_P. It is easy to verify that Γ_P is anti-monotone. Consequently, Γ_P^2 is monotone. Because Γ_P^2 is monotone, we know by the Knaster–Tarski theorem [Tarski, 1955] that Γ_P^2 has a least and a greatest fixpoint.

Definition. Let P be a nondisjunctive program. By $WF_\perp(P)$ we denote the least fixpoint of Γ_P^2, and by $WF_\top(P)$ the greatest.

If P is a normal program, then these two sets — $WF_\perp(P)$ and $WF_\top(P)$ — capture essential information about the well-founded semantics of P [Van Gelder *et al.*, 1990]. That is, under the well-founded semantics, when an atom $L \in \mathcal{L}_P$ is submitted as a query: the answer should be "yes" when $L \in WF_\perp(P)$; "unknown" when $L \in WF_\top(P) \setminus WF_\perp(P)$; and "no" when $L \notin WF_\top(P)$.

Even when a nondisjunctive program P includes classical negation, the sets $WF_\perp(P)$ and $WF_\top(P)$ provide lower and upper bounds on the consistent answer sets for P. Thus, if X is a consistent answer set for P, then

$$WF_\perp(P) \subset X \subset WF_\top(P) ,$$

because each fixpoint of Γ_P is also a fixpoint of Γ_P^2.[7]

Theorem 1 *Let P be a nondisjunctive program with signing S. The following three conditions are equivalent.*

(i) P is a consistent program.

(ii) $WF_\perp(P) \cap \overline{S}$ is a consistent set.

(iii) $WF_\perp(P) \cup (WF_\top(P) \cap S)$ is a consistent answer set for P.

[7]The reader may notice that the sets $WF_\perp(P)$ and $WF_\top(P)$ correspond, essentially, to a proposal by Przymusinski [1990], extending the well-founded semantics to nondisjunctive programs with classical negation. Further work in this direction can be found in [Pereira and Alferes, 1992].

This theorem shows, for instance, that if a nondisjunctive program P with signing S is consistent, then P has a "standard" answer set expressible in terms of $WF_\perp(P)$, $WF_\top(P)$ and S.

We also have the following corollary.

Corollary 1 *Let P be a nondisjunctive program with signing S. If P is consistent, then $Cn(P) \cap \overline{S} = WF_\perp(P) \cap \overline{S}$.*

The right to left direction is clear, since we've seen that every answer set for a nondisjunctive program P contains $WF_\perp(P)$. For the other direction, we know by Theorem 1(iii) that $WF_\perp(P) \cup (WF_\top(P) \cap S)$ is an answer set for P. The intersection of this answer set with $\overline{S}$ coincides with $WF_\perp(P) \cap \overline{S}$, and it follows that $Cn(P) \cap \overline{S} \subset WF_\perp(P) \cap \overline{S}$.

Consider, for example, program P_1 from the previous section. Since P_1 is finite, and very small, we can conveniently calculate $WF_\perp(P_1)$ as follows.

$$\begin{aligned}
\Gamma_{P_1}\emptyset &= \{\neg a, a, b\} \\
\Gamma^2_{P_1}\emptyset &= \{\neg a\} \\
\Gamma^3_{P_1}\emptyset &= \{\neg a, a, b\}
\end{aligned}$$

We see that $\Gamma^2_{P_1}\emptyset$ is the least fixpoint of $\Gamma^2_{P_1}$, so $WF_\perp(P_1) = \{\neg a\}$. Recall that $S = \{b\}$ is a signing for P_1, so $WF_\perp(P_1) \cap \overline{S} = \{\neg a\}$. Since $WF_\perp(P_1) \cap \overline{S}$ is a consistent set, we can conclude by Theorem 1 that P_1 is a consistent program. For any nondisjunctive program P, it follows from the anti-monotonicity of Γ_P that $WF_\top(P) = \Gamma_P(WF_\perp(P))$, so we have $WF_\top(P_1) = \Gamma^3_{P_1}\emptyset = \{\neg a, a, b\}$. Thus $WF_\top(P_1) \cap S = \{b\}$, and we can conclude by Theorem 1 that $\{\neg a, b\}$ is an answer set for P_1.[8] Finally, since program P_1 is consistent and $WF_\perp(P_1) \cap \overline{S} = \{\neg a\}$, we can conclude by Corollary 1 that $Cn(P_1) \cap \overline{S} = \{\neg a\}$.

Observe that for any head-consistent basic program P, the set $\alpha(P)$ is consistent, since $\alpha(P) \subset Head(P)$. From this it follows easily that if P is a head-consistent nondisjunctive program, then $WF_\top(P)$ is consistent. Furthermore, if such a program P has a signing S, then $\overline{S}$ is also a signing for P. These observations yield the following corollary to the previous results.

Corollary 2 *Let P be a head-consistent nondisjunctive program with signing S. The following three conditions hold:*

(i) P is a consistent program;

(ii) $WF_\perp(P) \cup (WF_\top(P) \cap S)$ and $WF_\perp(P) \cup (WF_\top(P) \cap \overline{S})$ are consistent answer sets for P;

(iii) $Cn(P) = WF_\perp(P)$.

In particular, Corollary 2 applies when P is a normal program with signing S.

[8] Note that the set $WF_\perp(P_1) \cup (WF_\top(P_1) \cap \overline{S}) = \{\neg a\} \cup \{\neg a, a\} = \{\neg a, a\}$ is not an answer set for P_1; so the asymmetry in the definition of a signing for a nondisjunctive program is needed for Theorem 1.

5 Signed Disjunctive Programs

Our subsequent results rely on the close relationship between a signed disjunctive program and the set of its (signed nondisjunctive) covers, defined as follows.

Definition. Let P be a constraint-free program. A nondisjunctive program P' (in the language of P) is a *cover* of P if P' can be obtained from P by replacing each rule $r \in P$ with a rule r' such that $head(r')$ is a singleton, $head(r') \subset head(r)$, $pos(r') = pos(r)$, and $neg(r') = neg(r)$.

Of course we're particularly interested in the covers of signed programs, so notice that if P is a program with signing S, then each cover of P is a nondisjunctive program with signing S.

For example, the following two programs are the only covers of program P_2 from Section 3.

Program P_3:	Program P_4:
$a \leftarrow$	$b \leftarrow$
$b \leftarrow \; not\ c$	$b \leftarrow \; not\ c$
$c \leftarrow \; not\ a$	$c \leftarrow \; not\ a$

Definition. Let P be a constraint-free program. By *covers*(P) we denote the set of all covers of P, and by *good-covers*(P) we denote the consistent programs in *covers*(P).

Thus, $covers(P_2) = \{P_3, P_4\} = good\text{-}covers(P_2)$, since programs P_3 and P_4 are both consistent.[9]

Now we state our main theorem relating a signed program to its covers.

Theorem 2 *Let P be a program. If S is a signing for P, then*

$$\begin{aligned} Cn(P) \cap \overline{S} &= \left(\bigcap_{P' \in covers(P)} Cn(P') \right) \cap \overline{S} \\ &= \left(\bigcap_{P' \in good\text{-}covers(P)} WF_{\perp}(P') \right) \cap \overline{S} \,. \end{aligned}$$

For example, we've already noted that $S = \{c\}$ is a signing for program P_2, and that $covers(P_2) = \{P_3, P_4\} = good\text{-}covers(P_2)$. It's not hard to check that $WF_{\perp}(P_3) = \{a, b\}$ and $WF_{\perp}(P_4) = \{b, c\}$. So by Theorem 2 we can conclude that $Cn(P_2) \cap \overline{S} = WF_{\perp}(P_3) \cap WF_{\perp}(P_4) \cap \overline{S} = \{b\}$.[10]

Recall that if S is a signing for a program P, then $h_S(P)$ is a nondisjunctive program. Without this asymmetry, Theorem 2 can fail to hold even in very simple cases. For example, program $h_{\overline{S}}(P_2)$ is not nondisjunctive, so $\overline{S}$ is not a signing for P_2. And we can see that

$$\bigcap_{P' \in good\text{-}covers(P_2)} WF_{\perp}(P') \cap S = WF_{\perp}(P_3) \cap WF_{\perp}(P_4) \cap S = \emptyset$$

and yet $Cn(P_2) \cap S = \{c\}$.

If a program P is signed and head-consistent, each of its covers is a signed head-consistent nondisjunctive program. It follows by Corollary 2(i) that every cover

[9] Program P_3 has the unique answer set $\{a, b\}$, and program P_4 has the unique answer set $\{b, c\}$.
[10] The unique answer set for program P_2 is $\{b, c\}$, so $Cn(P_2) = \{b, c\}$.

of P is consistent, and thus that $covers(P) = good\text{-}covers(P)$. This gives us the following corollary to Theorem 2.

Corollary 3 *Let P be a program with signing S. If P is head-consistent, then*

$$Cn(P) \cap \overline{S} = \left(\bigcap_{P' \in covers(P)} WF_{\perp}(P') \right) \cap \overline{S} \, .$$

The following theorem can be derived from Theorem 2 and the definitions.

Theorem 3 *Let P be a program with signing S. The following three conditions are equivalent.*

(i) P is a consistent program.

(ii) P has a consistent cover.

(iii) $Cn(P) \cap \overline{S}$ is a consistent set.

Again, since the covers of a signed program are signed nondisjunctive programs, and since such programs are consistent whenever they are head-consistent (Corollary 2(i)), we have the following corollary to Theorem 3.

Corollary 4 *Every signed program with at least one head-consistent cover is consistent.*

Notice that Corollary 4 may apply even to programs that are not themselves head-consistent. Notice also that, as a special case of Corollary 4, we get the result that every signed program without classical negation has at least one consistent answer set.

Proof sketch (Theorem 2) : The chief task is to show that

$$Cn(P) \cap \overline{S} = \left(\bigcap_{P' \in good\text{-}covers(P)} WF_{\perp}(P') \right) \cap \overline{S} \, .$$

(Right-to-left): Show that every consistent answer set for P is an answer set for a consistent cover of P. By Corollary 1, if A is a consistent answer set for a cover P' of P, then $WF_{\perp}(P') \cap \overline{S} \subset A \cap \overline{S}$.
(Left-to-right): Show that for every consistent cover P' of P there is a consistent cover P'' of P such that $WF_{\perp}(P'') \cap \overline{S} \subset WF_{\perp}(P') \cap \overline{S}$ and $WF_{\perp}(P'') \cup (WF_{\top}(P'') \cap S)$ is an answer set for P. There are two key lemmas. First, define $candidates(P)$ as follows.

$$candidates(P) = \{ WF_{\perp}(P') \cup (WF_{\top}(P') \cap S) : P' \in good\text{-}covers(P) \}$$

Define a partial order $\leq_S$ on $candidates(P)$ such that $X \leq_S Y$ if the following two conditions hold: (i) $X \cap \overline{S} \subset Y \cap \overline{S}$; (ii) if $X \cap \overline{S} = Y \cap \overline{S}$, then $X \subset Y$. Show that each chain in this partial order has a lower bound in $candidates(P)$. Second, show that each minimal element in this partial order is an answer set for P. Notice that this proof sketch suggests the following additional result.

Signing Lemma. *Let P be a program with signing S. The partial order $\langle candidates(P), \leq_S \rangle$ has minimal elements, each of which is a consistent answer set for P. Moreover, if A is a consistent answer set for P or any cover of P, then there is a minimal element A' in $\langle candidates(P), \leq_S \rangle$ such that $A' \cap \overline{S} \subset A \cap \overline{S}$.*

6 Restricted Monotonicity

Here we generalize and extend the restricted monotonicity theorem from [Turner, 1993].[11] A signing S for a program P is a subset of the literals in the language of P. The restricted monotonicity theorem shows, as a special case, that we can augment program P with any subset of the set of rules $\{L \leftarrow\ : L \in \overline{S}\}$ without losing any consequences of P in $\overline{S}$. Of course even this simple monotonicity property does not hold for arbitrary programs P and arbitrary subsets S of $\mathcal{L}_P$.

In order to formulate the restricted monotonicity theorem, we first introduce an ordering on the rules of programs, which will be used in turn to define an ordering on programs themselves. Before defining the ordering on rules, we describe the intuition underlying the definition. We have in mind three simple ways to strengthen a rule. First, one can strengthen a rule r by removing literals from $pos(r)$ and $neg(r)$. Second, one can strengthen a rule r by removing literals from $head(r)$. Based on these two methods for strengthening a rule, there would be a transparent formal definition; it is the third method for strengthening a rule that makes the definition less obvious, although still intuitively straightforward. So, third, one can strengthen a rule r by removing a literal from $pos(r)$ and adding the complementary literal to $head(r)$.

Definition. Given rules r and r', we say that r *is subsumed by* r', and we write $r \preceq r'$, if the following three conditions hold:

(i) $neg(r') \subset neg(r)$,

(ii) $pos(r') \subset pos(r)$,

(iii) every literal in $head(r') \setminus head(r)$ appears complemented in $pos(r)$.[12]

Following are some additional observations about the $\preceq$ ordering for rules. First, note that if $r \preceq r'$ and no literal in $head(r')$ has its complement in $pos(r)$, then $head(r') \subset head(r)$. On the other hand, if $pos(r) \cup head(r')$ is inconsistent, then we may have $r \preceq r'$ and yet $head(r') \not\subset head(r)$. For example, let r be the rule $b \leftarrow \neg a$ and let r' be the rule $a \mid b \leftarrow$. In this case we have $r \preceq r'$, and yet $head(r') \not\subset head(r)$.

Clearly the $\preceq$ ordering for rules is reflexive. As it happens, it is not anti-symmetric. For instance, let r be the rule $a \leftarrow a, \neg a$ and let r' be the rule $\neg a \leftarrow a, \neg a$. We have $r \preceq r'$ and $r' \preceq r$, yet $r \neq r'$. It is not difficult to establish also that the $\preceq$ ordering for rules is transitive.

Definition. Given programs P and Q, we say that P *is subsumed by* Q, and we write $P \preceq Q$, if for each rule $r \in P$ there is a rule $r' \in Q$ such that $r \preceq r'$.

Because the $\preceq$ ordering for rules is reflexive and transitive, we can conclude that the $\preceq$ ordering for programs is reflexive and transitive as well. On the other hand, because the $\preceq$ ordering for rules is not anti-symmetric, neither is the $\preceq$ ordering for programs.

Now we can state the restricted monotonicity theorem.

Theorem 4 *Let P, Q be programs in the same language, both with signing S. If $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$ and $h_S(Q) \preceq h_S(P)$, then $Cn(P) \cap \overline{S} \subset Cn(Q) \cap \overline{S}$.*

[11] The notion of restricted monotonicity is given a general definition in [Lifschitz, 1993].

[12] As subsequent discussion will illustrate, this third condition makes the definition of the $\preceq$ ordering for rules more general than that used in [Turner, 1993], even when we consider only nondisjunctive rules.

In the interest of simplicity, Theorem 4 is stated in terms of the consequences of signed programs; we also have the following stronger result in terms of answer sets.

Theorem 5 *Let P, Q be programs in the same language, both with signing S. If $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$ and $h_S(Q) \preceq h_S(P)$, then for every consistent answer set A' for program Q, there is a consistent answer set A for program P such that $A \cap \overline{S} \subset A' \cap \overline{S}$.*

Consider for example the program P_2 discussed previously, along with the program P_2' obtained from P_2 by removing the rule $b \leftarrow not\ c$.

Program P_2:	Program P_2':
$a \mid b \leftarrow$	$a \mid b \leftarrow$
$b \leftarrow not\ c$	$c \leftarrow not\ a$
$c \leftarrow not\ a$	

The two programs share a signing $S = \{c\}$, and we can see that $h_S(P_2) = h_S(P_2')$, while $h_{\overline{S}}(P_2') \preceq h_{\overline{S}}(P_2)$. Theorem 4 tells us that program P_2 is stronger in $\overline{S}$ than program P_2'. Recall that P_2 has a unique answer set — $\{b, c\}$ — and so entails the literal b from $\overline{S}$. Program P_2' has an additional answer set — $\{a\}$ — and so entails no literals from $\overline{S}$.

Next we demonstrate that the two asymmetries in the definition of a signing are necessary for the restricted monotonicity theorem. First, we might suppose that for a nondisjunctive program P, a set S of literals should be a signing if for every rule $r \in P$, either $head(r) \cup pos(r) \subset S$ and $neg(r) \subset \overline{S}$, or $head(r) \cup pos(r) \subset \overline{S}$ and $neg(r) \subset S$. But under this symmetric definition, the restricted monotonicity property is lost. For instance, consider again the program P_1, along with the program P_1' obtained from P_1 by removing the rule $\neg a \leftarrow$.

Program P_1:	Program P_1':
$a \leftarrow not\ b$	$a \leftarrow not\ b$
$b \leftarrow not\ a$	$b \leftarrow not\ a$
$\neg a \leftarrow$	

Suppose that $S = \{a, \neg a\}$ was in fact a signing for programs P_1 and P_1'. Thus we would have $Cn(P_1) \cap \overline{S} = \{b\}$ and $Cn(P_1') \cap \overline{S} = \emptyset$. Yet we would also have $h_{\overline{S}}(P_1) = h_{\overline{S}}(P_1')$ and $h_S(P_1') \preceq h_S(P_1)$, and by restricted monotonicity we could mistakenly conclude that $Cn(P_1) \cap \overline{S} \subset Cn(P_1') \cap \overline{S}$.

We might also suppose that a disjunctive program P should be signed whenever all its covers are. Under such an alternative definition, we would say that $S = \{a, b\}$ should be a common signing for programs P_2 and P_2' above, and from this we could mistakenly conclude by restricted monotonicity that program P_2' should be stronger in $\overline{S}$ than program P_2.

Proof sketch (Theorem 5) : First, prove the theorem for the nondisjunctive case. (Roughly, follow the restricted monotonicity proof in [Turner, 1993], using the new definition of $\preceq$.) Second, show that for appropriate P and Q with signing S, for every cover Q' of Q there is a cover P' of P such that $h_{\overline{S}}(P') \preceq h_{\overline{S}}(Q')$ and $h_S(Q') \preceq h_S(P')$. Given these results, assume that A' is a consistent answer set for Q. By the Signing Lemma (Section 5), there is a consistent answer set A'' for Q such that $A'' \cap \overline{S} \subset A' \cap \overline{S}$ and A'' is a minimal element in $\langle candidates(Q), \leq_S \rangle$. By the definition of $candidates(Q)$, there is a cover Q' of Q such that A'' is an answer set for Q'. It follows that there is a cover P' of P with consistent answer set A''' such that $A''' \cap \overline{S} \subset A'' \cap \overline{S}$. Again by the Signing Lemma we can conclude that there is a consistent answer set A for P such that $A \cap \overline{S} \subset A''' \cap \overline{S}$.

7 Application : Reasoning about Action

Recently Gelfond and Lifschitz [1993] defined a simple, elegant language, called $\mathcal{A}$, in which many benchmark problems of commonsense reasoning about action can be formalized simply and (intuitively) correctly. They also specified a translation from $\mathcal{A}$ into nondisjunctive logic programming, and used properties of signed normal programs to prove the translation sound. Subsequently, several sound and complete translations from $\mathcal{A}$ into variants of logic programming have been proposed. Denecker [1993] and Dung [1993] each define a version of abductive logic programming into which they specify a translation from $\mathcal{A}$. A translation into "equational logic programming" has also been proposed [Hölldobler and Thielscher, 1993]. In the full version of this paper, we define a slight extension of $\mathcal{A}$, called $\mathcal{A}_d$, and specify a sound and complete translation from $\mathcal{A}_d$ into disjunctive logic programming with classical negation. This translation produces signed programs, and our soundness and completeness proof exploits many of the results reported in this paper, including the restricted monotonicity property of signed programs and the close relationship between a signed disjunctive program and its (signed) nondisjunctive covers.

Due to lack of space, here we will present informally an example of an action domain, its representation in the syntax of $\mathcal{A}_d$, and its translation into a logic program. We also indicate in part how properties related to signings can be of use in showing the translation correct. Full details are available in the longer version of the paper.

We will consider yet another variant of the Yale Shooting domain [Hanks and McDermott, 1987], which can be called the "two-guns" domain. There is a pilgrim and a turkey. The pilgrim has two guns. Initially, the turkey is alive, but if the pilgrim fires a loaded gun, the turkey dies. Furthermore, at least one of the pilgrim's guns is loaded initially. We can conclude that the turkey will be dead if the pilgrim performs either of the following sequences of actions: (i) wait, shoot gun one, shoot gun two; or (ii) wait, shoot gun two, shoot gun one.

In $\mathcal{A}_d$ we represent the two-guns domain as follows.

$$\begin{array}{l}\textbf{initially } \mathit{Alive}\\ \textbf{initially } \mathit{Loaded}_1 \textbf{ or initially } \mathit{Loaded}_2\\ \mathit{Shoot}_1 \textbf{ causes } \neg \mathit{Alive} \textbf{ if } \mathit{Loaded}_1\\ \mathit{Shoot}_2 \textbf{ causes } \neg \mathit{Alive} \textbf{ if } \mathit{Loaded}_2\end{array}$$

This domain has three models: one in which Loaded_1 holds initially and Loaded_2 doesn't, one in which Loaded_2 holds initially and Loaded_1 doesn't, and one in which both Loaded_1 and Loaded_2 hold initially.

The following program P_5 correctly formalizes the two-guns domain.[13]

1. $\mathit{Holds}(\mathit{Alive}, S_0) \leftarrow$
2. $\mathit{Holds}(\mathit{Loaded}_1, S_0) \mid \mathit{Holds}(\mathit{Loaded}_2, S_0) \leftarrow$
3. $\neg \mathit{Holds}(\mathit{Alive}, \mathit{Result}(\mathit{Shoot}_1, s)) \leftarrow \mathit{Holds}(\mathit{Loaded}_1, s)$
4. $\mathit{Noninertial}(\mathit{Alive}, \mathit{Shoot}_1, s) \leftarrow \mathit{not}\ \neg \mathit{Holds}(\mathit{Loaded}_1, s)$
5. $\neg \mathit{Holds}(\mathit{Alive}, \mathit{Result}(\mathit{Shoot}_2, s)) \leftarrow \mathit{Holds}(\mathit{Loaded}_2, s)$

[13] We point out that rules 3–9 listed in the program are "schematic rules." In our description of languages and programs in Section 2, we adopted an abstract view of what atoms are and said nothing about their internal structure. But the most important case is when the set of atoms is defined as the set of ground atoms of a first-order language; then a large (even infinite) set of rules can be specified by a single schematic rule with variables. Thus, the atoms of $\mathcal{L}_{P_5}$ correspond to the ground atoms of the appropriate many-sorted first-order language, with variables f, a, s for sorts *fluents*, *actions*, and *situations*. The rules of program P_5 correspond to the ground instances in this language of the schematic rules given above.

6. $Noninertial(Alive, Shoot_2, s) \leftarrow not\ \neg Holds(Loaded_2, s)$
7. $Holds(f, Result(a, s)) \leftarrow Holds(f, s), not\ Noninertial(f, a, s)$
8. $\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), not\ Noninertial(f, a, s)$
9. $Holds(f, S_0) \mid \neg Holds(f, S_0) \leftarrow$

Notice that the set S consisting of all the *Noninertial* literals is a signing for program P_5, with $h_S(P_5)$ consisting of rules 4 and 6, and with $h_{\overline{S}}(P_5)$ consisting of the remaining rules. Of course we are primarily interested in the *Holds* literals entailed by this program; that is, we are interested in the literals belonging to $\overline{S}$. Theorem 2 allows us to determine the consequences of P_5 in $\overline{S}$ by considering each of the covers of P_5, which are very much like the acyclic programs for reasoning about action proposed by Apt and Bezem [1990], and thus relatively easy to reason about.

Below we state (without definitions) two correctness theorems for the general translation from a domain description D in $\mathcal{A}_d$ to a logic program πD. Interested readers are referred to the full version of the paper for details.

Theorem 6 *Let D be a consistent domain description in $\mathcal{A}_d$. For every atomic value proposition V, D entails V if and only if πD entails πV.*

Theorem 7 *Let D be a consistent domain description in $\mathcal{A}_d$. For all value propositions V and all atomic value propositions $V_1, \ldots, V_k$ $(k \geq 1)$ such that $V = V_1$ **or** $\ldots$ **or** V_k, D entails V if and only if each consistent cover of πD entails at least one of $\pi V_1, \ldots, \pi V_k$.*

8 Conclusion

The existence of a signing for a logic program is a simple syntactic criterion that guarantees many convenient declarative properties for the program under the answer set semantics.

One such property is the existence of a consistent answer set for signed disjunctive programs with at least one head-consistent cover (Corollary 4). As a special case of this, we have the consistency of signed disjunctive programs without classical negation. It seems likely though that consistency may hold also for larger classes of programs: for instance, disjunctive programs for which all covers are signed and one cover is also head-consistent. For now though, the consistency of programs in this larger class remains an open question.

On the other hand, without going into details, we know that the consistency result for signed disjunctive programs with a head-consistent cover can be extended in another direction by use of the notion of "U-components" from [Lifschitz and Turner, 1994]. More specifically, we can show that a disjunctive program P with a head-consistent cover is consistent whenever there is a "splitting sequence" U for P such that every U-component of P is a signed program. Again we have as a special case the fact that a disjunctive program without classical negation is consistent if it has a splitting sequence U with all U-components signed. This result is strictly more general than the corresponding result for (locally) stratified programs, since a disjunctive program P without classical negation is (locally) stratified if and only if it has a splitting sequence U such that every U-component of P is a positive program.

Of course this generalization of the consistency result for signed programs holds as well in the nondisjunctive case. Recall that one of the most general results on the consistency of nondisjunctive programs belongs to Fages [1994] who showed that "order-consistent" normal programs have answer sets. In Section 5 of [Lifschitz and Turner, 1994], we show that a normal program is order-consistent if and only if it has a splitting sequence U such that all U-components are signed. Thus Fages' consistency theorem is a special case of the more general theorem alluded to above.[14]

In contrast to the consistency results for signed programs, the examples considered in this paper suggest that it should be difficult or impossible to extend very significantly the restricted monotonicity results for signed programs.

In general, the asymmetric nature of these results may limit their utility. But we imagine that in many cases it may be possible to write programs for which the literals belonging to a signing S represent auxiliary concepts, as in fact they do in our programs for reasoning about action. In such cases, we are interested primarily in the literals that belong to $\overline{S}$, about which our theorems have much to say.

Acknowledgements

Special thanks to Vladimir Lifschitz. Thanks also to Michael Gelfond and Norman McCain. This work was partially supported by the National Science Foundation under grant IRI-9306751.

References

[Apt and Bezem, 1990] Krzysztof Apt and Marc Bezem. Acyclic programs. In David Warren and Peter Szeredi, editors, *Logic Programming: Proceedings of the Seventh International Conference*, pages 617–633, 1990.

[Chen and Warren, 1993] Weidong Chen and David S. Warren. Towards effective evaluation of general logic programs. Technical Report 93-CSE-11, Southern Methodist University, 1993.

[Denecker and DeSchreye, 1993] Marc Denecker and Danny DeSchreye. Representing incomplete knowledge in abductive logic programming. In *Logic Programming: Proceedings of the 1993 International Symposium*, pages 147–163, 1993.

[Dung, 1992] Phan Minh Dung. On the relations between stable and well-founded semantics of logic programs. *Theoretical Computer Science*, 105:222–238, 1992.

[Dung, 1993] Phan Minh Dung. Representing actions in logic programming and its applications in database updates. In David S. Warren, editor, *Logic Programming: Proceedings of the Tenth International Conference*, pages 7–25. MIT Press, 1993.

[Fages, 1994] François Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1(1):51–60, 1994. To appear.

[Gelfond and Lifschitz, 1990] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David Warren and Peter Szeredi, editors, *Logic Programming: Proceedings of the Seventh International Conference*, pages 579–597, 1990.

[14]The underlying idea of this redefinition of order-consistency is already apparent in Kunen's [1989] use of signings in relation to "call-consistent" programs. He noticed, roughly speaking, that every call-consistent program (and thus every "strict" program) has signed parts, and he explained certain behaviors of those programs in terms of the behavior of their signed parts.

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

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17:301–322, 1993.

[Gelfond *et al.*, 1991] Michael Gelfond, Vladimir Lifschitz, Halina Przymusińska, and Miroslaw Truszczyński. Disjunctive defaults. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 230–237, 1991.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Hölldobler and Thielscher, 1993] Steffen Hölldobler and Michael Thielscher. Actions and specificity. In *Logic Programming: Proceedings of the 1993 International Symposium*, pages 164–180, 1993.

[Kunen, 1989] Kenneth Kunen. Signed data dependencies in logic programs. *Journal of Logic Programming*, 7(3):231–245, 1989.

[Lifschitz and Turner, 1994] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *Logic Programming: Proceedings of the Eleventh International Conference*, 1994. To appear.

[Lifschitz *et al.*, 1993] Vladimir Lifschitz, Norman McCain, and Hudson Turner. Reasoning about actions with SLG. Manuscript, 1993.

[Lifschitz, 1993] Vladimir Lifschitz. Restricted monotonicity. In *Proc. AAAI-93*, pages 432–437, 1993.

[Pereira and Alferes, 1992] Luis Pereira and Jose Alferes. Well-founded semantics for logic programs with explicit negation. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pages 102–106, 1992.

[Przymusinski, 1988] Teodor Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, San Mateo, CA, 1988.

[Przymusinski, 1990] Teodor Przymusinski. Extended stable semantics for normal and disjunctive programs. In David Warren and Peter Szeredi, editors, *Logic Programming: Proceedings of the Seventh International Conference*, pages 459–477, 1990.

[Tarski, 1955] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Turner, 1993] Hudson Turner. A monotonicity theorem for extended logic programs. In David S. Warren, editor, *Logic Programming: Proceedings of the Tenth International Conference*, pages 567–585. MIT Press, 1993.

[Van Gelder *et al.*, 1990] Allen Van Gelder, Kenneth Ross, and John Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, pages 221–230, 1990.