# Signed Logic Programs

Hudson Turner
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78703
hudson@cs.utexas.edu

May 1, 1995

## 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] 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.

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.)

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

[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] 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.

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 of 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 page constraints.

## 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 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 $head(r)$, $pos(r)$ and $neg(r)$. 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 $r \in P$, $neg(r)$ 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 $pos(r) \subset X$, $head(r) \cap X$ 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 $neg(r) \cap X$ is empty, consider the rule $r'$ defined by $head(r') = head(r)$, $pos(r') = pos(r)$, $neg(r') = \emptyset$. 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 $Cn(P)$ we denote the set of literals entailed by program $P$. Finally, $P$ is *consistent* if $Cn(P)$ 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 $r \in P$, $head(r)$ is nonempty. A program $P$ is *nondisjunctive* if for every rule $r \in P$, $head(r)$ is a singleton.[5]

[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].

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

Observe that if a set $X$ of literals is closed under a positive program $P$, then so is $X \cap Head(P)$. 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.

## 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

[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 $P$.

signed programs and the class of locally stratified programs overlap, and neither contains the other.

Consider the following program $P_1$.

$$\begin{aligned} a &\leftarrow \mathit{not}\ b \\ b &\leftarrow \mathit{not}\ a \\ \neg a &\leftarrow \end{aligned}$$

Program $P_1$ has a signing $S = \{b\}$. Observe that neither $\{a, \neg a\}$ nor $\{a\}$ is a signing for $P$, since we have $a, \neg a \in \mathit{Head}(P)$. 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 $\overline{S} \cap \mathit{Head}(P)$ that appears complemented in $\mathit{Head}(P)$.

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

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

Let $r$ be a rule in a program $P$ with signing $S$. Since signed programs are constraint-free, the set $\mathit{head}(r)$ is nonempty. Thus, either $\mathit{head}(r) \subset S$ or $\mathit{head}(r) \subset \overline{S}$, 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 $P_2$.

$$\begin{aligned} a \mid b &\leftarrow \\ b &\leftarrow \mathit{not}\ c \\ c &\leftarrow \mathit{not}\ a \end{aligned}$$

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 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 $\alpha(P) \subset \mathit{Head}(P)$.

**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 $\Gamma_P$. It is easy to verify that $\Gamma_P$ is anti-monotone. Consequently, $\Gamma_P^2$ is monotone. Because $\Gamma_P^2$ is monotone, we know by the Knaster–Tarski theorem [Tarski, 1955] that $\Gamma_P^2$ has a least and a greatest fixpoint.

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

If $P$ is a normal program, then these two sets — $WF_\bot(P)$ 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 $L \in WF_\bot(P)$; "unknown" when $L \in WF_\top(P) \setminus WF_\bot(P)$; and "no" when $L \notin WF_\top(P)$.

Even when a nondisjunctive program $P$ includes classical negation, the sets $WF_\bot(P)$ 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

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

because each fixpoint of $\Gamma_P$ is also a fixpoint of $\Gamma_P^2$.[7]

[7] The reader may notice that the sets $WF_\bot(P)$ 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].

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

*(i)* *$P$ is a consistent program.*

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

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

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 $WF_\bot(P)$, $WF_\top(P)$ and $S$.

We also have the following corollary.

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

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

Consider, for example, program $P_1$ from the previous section. Since $P_1$ is finite, and very small, we can conveniently calculate $WF_\bot(P_1)$ 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 $WF_\bot(P_1) = \{\neg a\}$. Recall that $S = \{b\}$ is a signing for $P_1$, so $WF_\bot(P_1) \cap \overline{S} = \{\neg a\}$. Since $WF_\bot(P_1) \cap \overline{S}$ 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 $\Gamma_P$ that $WF_\top(P) = \Gamma_P(WF_\bot(P))$, 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

[8] Note that the set $WF_\bot(P_1) \cup (WF_\top(P_1) \cap \overline{S}) = \{\neg a\} \cup \{\neg a, a\} = \{\neg a, a\}$ 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.

program $P_1$ is consistent and $WF_\bot(P_1) \cap \overline{S} = \{\neg a\}$, 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 $\alpha(P) \subset \mathit{Head}(P)$. 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)* *$WF_\bot(P) \cup (WF_\top(P) \cap S)$ and $WF_\bot(P) \cup (WF_\top(P) \cap \overline{S})$ are consistent answer sets for $P$;*

*(iii)* *$Cn(P) = WF_\bot(P)$.*

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

# 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 $\mathit{head}(r')$ is a singleton, $\mathit{head}(r') \subset \mathit{head}(r)$, $\mathit{pos}(r') = \mathit{pos}(r)$, and $\mathit{neg}(r') = \mathit{neg}(r)$.

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 $good\text{-}covers(P)$ 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*

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

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 $WF_\bot(P_3) = \{a, b\}$ and $WF_\bot(P_4) = \{b, c\}$. So by Theorem 2 we can conclude that $Cn(P_2) \cap \overline{S} = WF_\bot(P_3) \cap WF_\bot(P_4) \cap \overline{S} = \{b\}$.[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

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

[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 $Cn(P_2) = \{b, c\}$.

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 of $P$ is consistent, and thus that $\mathit{covers}(P) = \mathit{good\text{-}covers}(P)$. 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*

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

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

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

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

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

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 the partial order $\langle candidates(P), \leq_S \rangle$ 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

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

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.

[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.

**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 $Cn(P) \cap \overline{S} \subset Cn(Q) \cap \overline{S}$.*

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 answer set $A'$ for program $Q$, there is an 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 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 remainder of this section, 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.

### 7.1 Example: Two-guns Domain

Before embarking on the definition of $\mathcal{A}_d$ and the translation from $\mathcal{A}_d$ into logic programming, we informally describe an action domain and present a correct program for it. We also indicate in part how properties related to signings can be of use in showing the program correct.

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.

The following program $P_5$ correctly formalizes the two-guns domain.

1. $Holds(Alive, S_0) \leftarrow$
2. $Holds(Loaded_1, S_0) \mid Holds(Loaded_2, S_0) \leftarrow$
3. $\neg Holds(Alive, Result(Shoot_1, s)) \leftarrow Holds(Loaded_1, s)$
4. $Noninertial(Alive, Shoot_1, s) \leftarrow not\ \neg Holds(Loaded_1, s)$
5. $\neg Holds(Alive, Result(Shoot_2, s)) \leftarrow Holds(Loaded_2, s)$
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. $Holds(f, S_0) \mid \neg Holds(f, S_0) \leftarrow$

Before we describe the intended meaning of each of these rules, we point out that rules 3–9 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.[13]

Rule 1 says that the turkey is initially alive, and rule 2 says that at least one of the pilgrim's guns is initially loaded. Rule 3 says that the turkey is not alive after the pilgrim fires gun one, if gun one is loaded. Rule 4 indicates that if gun one is fired when it might be loaded, it is not safe to assume that the turkey remains alive. Rules 5 and 6 tell us the same things with respect to the second gun. Rules 7 and 8 are "inertia" rules; they express the commonsense notion that things generally don't change unless they are made to. Finally, rule 9 represents the fact that it must be the case initially that each fluent is either true or false.[14]

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 $P_5$. The following program $P_6$ is one of these covers.

1. $Holds(Alive, S_0) \leftarrow$
2.1. $Holds(Loaded_1, S_0) \leftarrow$
3. $\neg Holds(Alive, Result(Shoot_1, s)) \leftarrow Holds(Loaded_1, s)$
4. $Noninertial(Alive, Shoot_1, s) \leftarrow not\ \neg Holds(Loaded_1, s)$
5. $\neg Holds(Alive, Result(Shoot_2, s)) \leftarrow Holds(Loaded_2, s)$
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. $\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), not\ Noninertial(f, a, s)$
9.1.1. $Holds(Alive, S_0) \leftarrow$
9.1.2. $Holds(Loaded_1, S_0) \leftarrow$
9.1.3. $\neg Holds(Loaded_2, S_0) \leftarrow$

[13] For convenience in the exposition below, we will speak of the family of rules specified by a given schematic rule as if it were in fact a single rule.

[14] This obvious fact about the initial state must be made explicit in our program because the law of excluded middle is not implied by the semantics of logic programs (unless we impose a closed world assumption). In effect, rule 9 enforces the law of excluded middle for fluents in the initial situation, and the correctness of rules 3–9 preserves this property in all subsequent situations.

The reader may notice that program $P_6$ is very much like the programs for reasoning about action proposed by Apt and Bezem [1990], and thus it is relatively easy to reason about. For instance, we can show that program $P_6$ correctly formalizes a "specialization" of the two-guns domain in which we know in addition that the first gun is loaded initially and the second isn't.

In fact, each consistent cover of program $P_5$ correctly formalizes one of the consistent specializations of the two-guns domain in which we know the initial values of all fluents. Conversely, each such consistent specialization of the two-guns domain is correctly formalized by one of the consistent covers of $P_5$. We will make our claims precise in a more general setting below, but the key is that the two-guns domain entails just what is entailed by all such consistent specializations. Meanwhile, our observations above indicate that whatever is entailed by all such consistent specializations of the two-guns domain is exactly what is entailed about the values of fluents by all consistent covers of program $P_5$. Finally, we know by Theorem 2 that whatever is entailed about the values of fluents by all consistent covers of program $P_5$ is exactly what is entailed about the values of fluents by program $P_5$ itself. Therefore, the two-guns domain entails just what is entailed about the values of fluents by program $P_5$.

## 7.2 The Language $\mathcal{A}_d$

In the language $\mathcal{A}_d$, a description of an action domain is a set of propositions of two kinds: value propositions, which restrict the values of fluents in situations resulting from sequences of actions, and effect propositions, which describe the effect of an action on a fluent.

First we describe the syntax of $\mathcal{A}_d$. We begin with two disjoint nonempty sets of symbols, called *fluent names* and *action names*. A *fluent expression* is a fluent name possibly preceded by $\neg$. An *atomic value proposition* is an expression of the form $F$ **after** $A_1; \ldots; A_m$, where $F$ is a fluent expression, and $A_1, \ldots, A_m$ $(m \geq 0)$ are action names. If $m = 0$, we write instead **initially** $F$. A *value proposition* is an expression of the form $V_1$ **or** $\ldots$ **or** $V_k$, $(k \geq 1)$ where all of $V_1, \ldots, V_k$ are atomic value propositions.[15] An *effect proposition* is an expression of the form $A$ **causes** $F$ **if** $G_1, \ldots, G_n$, where $A$ is an action name, and each of $F, G_1, \ldots, G_n$ $(n \geq 0)$ is a fluent expression.

[15] In $\mathcal{A}$, all value propositions are atomic. $\mathcal{A}$ and $\mathcal{A}_d$ coincide on domains that include only atomic value propositions.

About this proposition we say that it describes the *effect* of $A$ on $F$, and that $G_1, \ldots, G_n$ are its *preconditions*. If $n = 0$, we drop the **if** and write simply $A$ **causes** $F$.

To describe the semantics of $\mathcal{A}_d$, we must define what the "models" of a domain description are, and when a value proposition is "entailed" by a domain description.

A *state* is a set of fluent names. Given a fluent name $F$ and a state $\sigma$, we say that $F$ holds in $\sigma$ if $F \in \sigma$; and that $\neg F$ holds in $\sigma$ if $F \notin \sigma$. A *transition function* is a mapping $\Phi$ from the set of pairs $(A, \sigma)$, where $A$ is an action name and $\sigma$ is a state, into the set of states. A *structure* is a pair $(\sigma_0, \Phi)$, where $\sigma_0$ is a state (the *initial state* of the structure), and $\Phi$ is a transition function.

For any structure $M$ and any action names $A_1, \ldots, A_m$, by $M^{A_1,\ldots,A_m}$ we denote the state

$$\Phi(A_m, \Phi(A_{m-1}, \ldots, \Phi(A_1, \sigma_0) \ldots)) \ ,$$

where $\Phi$ is the transition function of $M$, and $\sigma_0$ is the initial state of $M$. We say that an atomic value proposition $F$ **after** $A_1; \ldots; A_m$ is *true* in a structure $M$ if $F$ holds in the state $M^{A_1,\ldots,A_m}$, and that it is *false* otherwise. In particular, a proposition of the form **initially** $F$ is true in $M$ if and only if $F$ holds in the initial state of $M$. Furthermore, we say that a value proposition $V_1$ **or** $\ldots$ **or** $V_k$ (where each of $V_1, \ldots, V_k$ is an atomic value proposition) is *true* in $M$ if at least one of $V_1, \ldots, V_k$ is true in $M$, and that it is *false* otherwise.

A structure $(\sigma_0, \Phi)$ is a *model* of a domain description $D$ if every value proposition from $D$ is true in $(\sigma_0, \Phi)$, and, for every action name $A$, every fluent name $F$, and every state $\sigma$, the following three conditions are satisfied:

- if $D$ includes an effect proposition describing the effect of $A$ on $F$ whose preconditions hold in $\sigma$, then $F \in \Phi(A, \sigma)$;
- if $D$ includes an effect proposition describing the effect of $A$ on $\neg F$ whose preconditions hold in $\sigma$, then $F \notin \Phi(A, \sigma)$;
- if $D$ does not include such effect propositions, then $F \in \Phi(A, \sigma)$ if and only if $F \in \sigma$.

A domain description is *consistent* if it has a model. A value proposition is *entailed* by a domain description $D$ if it is true in every model of $D$.

For example, we represent the two-guns domain $D_2$ in $\mathcal{A}_d$ as follows, with actions $\{Shoot_1, Shoot_2, Wait\}$ and fluents $\{Loaded_1, Loaded_2, Alive\}$.

**initially** $Alive$
**initially** $Loaded_1$ **or initially** $Loaded_2$
$Shoot_1$ **causes** $\neg Alive$ **if** $Loaded_1$
$Shoot_2$ **causes** $\neg Alive$ **if** $Loaded_2$

Domain $D_2$ 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 are among the value propositions entailed by domain $D_2$.

$\neg Alive$ **after** $Wait; Shoot_1; Shoot_2$
$\neg Alive$ **after** $Wait; Shoot_2; Shoot_1$
$\neg Alive$ **after** $Shoot_1$ **or** $\neg Alive$ **after** $Shoot_2$

## 7.3 The Translation from $\mathcal{A}_d$ into Logic Programming

In specifying the translation from a domain description $D$ in $\mathcal{A}_d$ into a logic program $\pi D$, we borrow a number of conventions from Gelfond and Lifschitz [1993]. We utilize schematic rules containing sorted variables $f, a, s$ for fluents, actions, and situations, as discussed previously. Also, for any fluent expression $F$, $|F|$ denotes the fluent name contained in $F$, and for any fluent name $F$ and situation term $t$, $Holds(\neg F, t)$ stands for $\neg Holds(F, t)$.

Now we specify the translation. To begin, if $V$ is an atomic value proposition $F$ **after** $A_1; \ldots; A_m$, then $\pi V$ stands for the literal

$$Holds(F, Result(A_m, Result(A_{m-1}, \ldots, Result(A_1, S_0) \ldots))). \qquad (1)$$

In particular, $\pi($**initially** $F)$ is $Holds(F, S_0)$.

For any domain description $D$, the program $\pi D$ consists of the following four kinds of rules.

*1. Rules for the value propositions.*
Each value proposition $V_1$ **or** ... **or** $V_k$ in $D$ (where $k > 0$ and each of $V_1, \ldots, V_k$ is atomic) is translated into the rule

$$\pi V_1 \mid \ \ldots \mid \pi V_k \quad \leftarrow \quad . \tag{2}$$

*2. Rules for the effect propositions.*
Each effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ is translated into a pair of rules. The first of them is the "effect" rule

$$Holds(F, Result(A, s)) \leftarrow Holds(G_1, s), \ldots, Holds(G_n, s). \tag{3}$$

The second is the "noninertial" rule

$$Noninertial(|F|, A, s) \leftarrow not \ \ \overline{Holds(G_1, s)}, \ldots, not \ \ \overline{Holds(G_n, s)}, \tag{4}$$

where for any literal $L$, $\overline{L}$ denotes its complement.

*3. Inertia rules.*

$$\begin{array}{lcl} Holds(f, Result(a, s)) & \leftarrow & Holds(f, s), not \ \ Noninertial(f, a, s) \\ \neg Holds(f, Result(a, s)) & \leftarrow & \neg Holds(f, s), not \ \ Noninertial(f, a, s) \end{array} \tag{5}$$

*4. Complete initial situations rule.*

$$Holds(f, S_0) \mid \neg Holds(f, S_0) \quad \leftarrow \tag{6}$$

For example, the logic program $\pi D_2$ obtained from the two-guns domain $D_2$ is precisely the program $P_5$ considered previously.

Of course the *Noninertial* literals play an auxiliary role in programs such as these. We are primarily interested in the *Holds* literals. Thus, if $S$ is defined as before, value propositions contribute only rules with their heads in $\overline{S}$, so we can conclude by the restricted monotonicity theorem (Theorem 4) that adding value propositions to a domain $D$ can only increase the *Holds* literals entailed by the corresponding program $\pi D$. Furthermore, we can see by Theorem 2 that the *Holds* literals entailed by a program $\pi D$ are exactly those *Holds* literals entailed by every cover of $\pi D$. We use properties such as these to establish the following general theorem showing that for any consistent domain $D$, the program $\pi D$ is sound and complete with respect to the semantics of $\mathcal{A}_d$.

**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 $\pi D$ entails $\pi V$.*

Although Theorem 6 applies only to atomic value propositions, we also have the following stronger result which applies also to disjunctive value propositions.

**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** *$V_k$, domain $D$ entails $V$ if and only if each consistent cover of $\pi D$ entails at least one of $\pi V_1, \ldots, \pi V_k$.*

In point of fact, Theorem 6 follows easily from Theorem 7 by Theorem 2, which relates the behavior of a signed program $\pi D$ to the behavior of its covers.

We remark again that the covers of such a program $\pi D$ resemble the acyclic (normal) programs for reasoning about action proposed by Apt and Bezem [1990]. Thus, these covers are relatively easy to reason about. Nonetheless, more than half of the proof of Theorem 7 is devoted essentially to proving the correctness of such nondisjunctive programs for "complete temporal projection" domains, in which value propositions are atomic, refer only to the initial situation, and completely specify the values of fluents in the initial situation. This intermediate result is not particularly difficult, but it brings together a number of complex definitions: the definition of the translation $\pi$, the definition of a model in $\mathcal{A}_d$, and the definitions related to the Splitting Sequence Theorem from [Lifschitz and Turner, 1994]. Once we've established the correctness of such nondisjunctive programs for complete temporal projection domains, we use the restricted monotonicity of signed programs (Theorem 4) to show that for any consistent domain $D$, each consistent cover of program $\pi D$ corresponds to a model of $D$, and vice versa.

# 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 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.[16]

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 result 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

[16] 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.

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.

[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.

## Appendix : Proofs

**Lemma 1** *Let $P$ be a nondisjunctive program with signing $S$, and with $X \subset \mathcal{L}_P$.*

*(i)* $\alpha(P^X) = \alpha(h_S(P)^{X \cap \overline{S}}) \cup \alpha(h_{\overline{S}}(P)^{X \cap S})$

*(ii)* $\alpha(P^X) \cap S = \alpha(h_S(P)^{X \cap \overline{S}})$

*(iii)* $\alpha(P^X) \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{X \cap S})$

**Proof.** Observe that $P = h_S(P) \cup h_{\overline{S}}(P)$. Thus, by the definition of the reduct operator, we have $P^X = (h_S(P) \cup h_{\overline{S}}(P))^X = h_S(P)^X \cup h_{\overline{S}}(P)^X$. By the definition of a signing, we know that for every rule $r \in h_S(P)$, $head(r) \cup pos(r) \subset S$. So $Lit(h_S(P)^X) \subset S$. Likewise, $Lit(h_{\overline{S}}(P)^X) \subset \overline{S}$. So we've shown that $Lit(h_S(P)^X) \cap Lit(h_{\overline{S}}(P)^X) = \emptyset$, from which we can conclude that $\alpha(P^X) = \alpha(h_S(P)^X) \cup \alpha(h_{\overline{S}}(P)^X)$. Again by the definition of a signing, we know that for every rule $r \in h_S(P)$, $neg(r) \subset \overline{S}$. So $h_S(P)^X = h_S(P)^{X \cap \overline{S}}$. Likewise, $h_{\overline{S}}(P)^X = h_S(P)^{X \cap S}$. Therefore we have $\alpha(P^X) = \alpha(h_S(P)^{X \cap \overline{S}}) \cup \alpha(h_{\overline{S}}(P)^{X \cap S})$, with $\alpha(P^X) \cap S = \alpha(h_S(P)^{X \cap \overline{S}})$ and $\alpha(P^X) \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{X \cap S})$. □

**Lemma 2** *Let $P$ be a nondisjunctive program with signing $S$. The set of literals $WF_\bot(P) \cup (WF_\top(P) \cap S)$ is a fixpoint of $\Gamma_P$.*

**Proof.** Let $A = WF_\bot(P) \cup (WF_\top(P) \cap S)$. Observe that $A \cap \overline{S} = WF_\bot(P) \cap \overline{S}$ and $A \cap S = WF_\top(P) \cap S$. Now, because $WF_\top(P)$ a fixpoint of $\Gamma_P^2$, we know that $\Gamma_P(WF_\top(P))$ is also a fixpoint of $\Gamma_P^2$. Therefore, by the anti-monotonicity of $\Gamma_P$, along with the fact that $WF_\bot(P)$ and $WF_\top(P)$ are the least and greatest fixpoints of $\Gamma_P^2$, we can conclude that $WF_\bot(P) = \Gamma_P(WF_\top(P))$. By the definition of $\Gamma_P$, $\Gamma_P(WF_\top(P)) = \alpha(P^{\mathrm{WF}_\top(P)})$. Thus, by Lemma 1(iii), we have $\Gamma_P(WF_\top(P)) \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{\mathrm{WF}_\top(P) \cap S})$. Combining these observations, we have the following.

$$\begin{aligned} A \cap \overline{S} &= WF_\bot(P) \cap \overline{S} \\ &= \Gamma_P(WF_\top(P)) \cap \overline{S} \\ &= \alpha(h_{\overline{S}}(P)^{\mathrm{WF}_\top(P) \cap S}) \\ &= \alpha(h_{\overline{S}}(P)^{A \cap S}) \end{aligned}$$

By symmetric reasoning, we can conclude that $A \cap S = \alpha(h_S(P)^{A \cap \overline{S}})$. Finally, by the definition of $\Gamma_P$ and Lemma 1(i), along with the foregoing observations, we have shown the following.

$$\begin{aligned} \Gamma_P A &= \alpha(h_S(P)^{A \cap \overline{S}}) \cup \alpha(h_{\overline{S}}(P)^{A \cap S}) \\ &= (A \cap S) \cup (A \cap \overline{S}) \\ &= A \end{aligned}$$

□

**Lemma 3** *Let $P$ be a nondisjunctive program with signing $S$, with $B$ a fixpoint of $\Gamma_P$. $B$ is a consistent set if and only if $B \cap \overline{S}$ is a consistent set.*

**Proof**. By the definition of a signing, no literal in $Head(P) \cap S$ appears complimented in $Head(P)$. Since $B$ is a fixpoint of $\Gamma_P$, we know that $B \subset Head(P)$. So no literal in $B \cap S$ appears complemented in $B$. Thus any literal in $B$ that appears complemented in $B$ must appear in $B \cap \overline{S}$. That is, any complementary pair of literals in $B$ is contained in $B \cap \overline{S}$. Therefore, $B$ is consistent if and only if $B \cap \overline{S}$ is consistent. □

**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 $P$.*

**Proof**. By Lemma 2, $WF_\perp(P) \cup (WF_\top(P) \cap S)$ is a fixpoint of $\Gamma_P$. So if $WF_\perp(P) \cup (WF_\top(P) \cap S)$ is consistent, it is an answer set for $P$. By Lemma 3, $WF_\perp(P) \cup (WF_\top(P) \cap S)$ is consistent if and only if $WF_\perp(P) \cap \overline{S}$ is consistent. So we've established the equivalence of parts (ii) and (iii). Furthermore, it's clear that (iii) implies (i). We complete our proof by showing that (i) implies (ii). Assume that $P$ is consistent, so $P$ has a consistent answer set $B$. Since $B$ is a fixpoint of $\Gamma_P$, we can conclude by Lemma 3 that $B \cap \overline{S}$ is consistent. We know that $WF_\perp(P) \subset B$, so $WF_\perp(P) \cap \overline{S}$ is consistent. □

**Lemma 4** *Let $P$ be a program with signing $S$:*

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

**Proof.** Every cover of $P$ is a signed nondisjunctive program, so by Corollary 1 we know that for each cover $P'$ of $P$, if $P'$ is consistent, then $Cn(P') \cap \overline{S} = WF_{\bot}(P') \cap \overline{S}$. So clearly

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

Since every program with a signing is constraint-free, we know that $covers(P)$ is nonempty. And since each inconsistent cover of $P$ entails $\mathcal{L}_P$, we're done. □

**Definition.** Let $P$ be a nondisjunctive program with signing $S$, with $X \subset \mathcal{L}_P$:

$$\begin{aligned} \Delta_S^P X &= \alpha(h_S(P)^{\alpha(h_{\overline{S}}(P)^X)}) \; , \\ \Delta_{\overline{S}}^P X &= \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^X)}) \; . \end{aligned}$$

Observe that $\Delta_S^P$ and $\Delta_{\overline{S}}^P$ are monotone operators (by the monotonicity of $\alpha$ and the anti-monotonicity of the reduct operators).

**Lemma 5** *Let $P$ be a nondisjunctive program with signing $S$, and with $X \subset \mathcal{L}_P$. $X$ is a fixpoint of $\Gamma_P^2$ if and only if $X \cap S$ is a fixpoint of $\Delta_S^P$ and $X \cap \overline{S}$ is a fixpoint of $\Delta_{\overline{S}}^P$.*

**Proof.** By the definition of $\Gamma_P$, we have $\Gamma_P X = \alpha(P^X)$. By Lemma 1(ii), $\Gamma_P X \cap S = \alpha(h_S(P)^{X \cap \overline{S}})$, and by Lemma 1(iii), $\Gamma_P X \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{X \cap S})$. Again by the definition of $\Gamma_P$, $\Gamma_P^2 X = \alpha(P^{\Gamma_P X})$. Again by Lemma 1(ii&iii), we have $\Gamma_P^2 X \cap S = \alpha(h_S(P)^{\Gamma_P X \cap \overline{S}})$ and $\Gamma_P^2 X \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{\Gamma_P X \cap S})$. In sum, $\Gamma_P^2 X \cap S = \alpha(h_S(P)^{\Gamma_P X \cap \overline{S}}) = \alpha(h_S(P)^{\alpha(h_{\overline{S}}(P)^{X \cap S})}) = \Delta_S^P(X \cap S)$. Likewise, $\Gamma_P^2 X \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{\Gamma_P X \cap S}) = \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{X \cap \overline{S}})}) = \Delta_{\overline{S}}^P(X \cap \overline{S})$. Therefore we can conclude the following.

$$\begin{aligned} \Gamma_P^2 X = X \quad & \textit{iff} \quad \Gamma_P^2 X \cap S = X \cap S \textit{ and } \Gamma_P^2 X \cap \overline{S} = X \cap \overline{S} \\ & \textit{iff} \quad \Delta_S^P(X \cap S) = X \cap S \textit{ and } \Delta_{\overline{S}}^P(X \cap \overline{S}) = X \cap \overline{S} \end{aligned}$$

□

**Lemma 6** *Let $P$ be a nondisjunctive program with signing $S$:*

$$lfp(\Delta^P_{\overline{S}}) = WF_\perp(P) \cap \overline{S} \ .$$

**Proof.** Because the operators $\Delta^P_S$ and $\Delta^P_{\overline{S}}$ are monotone, we know by the Knaster-Tarski theorem that each has a least and greatest fixpoint. Since $WF_\perp(P)$ is a fixpoint of $\Gamma^2_P$, it follows by the previous lemma that $WF_\perp(P) \cap \overline{S}$ is a fixpoint of $\Delta^P_{\overline{S}}$. It remains to show that for any fixpoint $X$ of $\Delta^P_{\overline{S}}$, $WF_\perp(P) \cap \overline{S} \subset X$. So let $X \subset \overline{S}$ be a fixpoint of $\Delta^P_{\overline{S}}$, and let $Y \subset S$ be a fixpoint of $\Delta^P_S$. By the previous lemma, $X \cup Y$ is a fixpoint of $\Gamma^2_P$. But because $WF_\perp(P)$ is the least fixpoint of $\Gamma^2_P$, we have $WF_\perp(P) \subset X \cup Y$, and therefore $WF_\perp(P) \cap \overline{S} \subset (X \cup Y) \cap \overline{S} = X$. □

**Lemma 7** *Let $T : 2^S \to 2^S$ be a monotone operator, with $B \subset S$. If $B$ is a pre-fixpoint of $T$ (that is, if $T(B) \subset B$), then $lfp(T) \subset B$.*

**Proof.** By the Knaster-Tarski theorem [Tarski, 1955], $lfp(T) = \bigcap_{T(A) \subset A} A$. That is, $lfp(T)$ coincides with the intersection of all pre-fixpoints of $T$. So every pre-fixpoint of $T$ is a superset of $lfp(T)$. □

In subsequent proofs we will sometimes find it convenient to represent a rule $r$ by the ordered triple $\langle\ X, Y, Z\ \rangle$, where $X = head(r)$, $Y = pos(r)$ and $Z = neg(r)$.

**Lemma 8** *Let $P$ be a program. A nondisjunctive program $P'$ in the language $\mathcal{L}_P$ is a cover of $P$ if and only if there is a total surjective function $\phi : P \to P'$ such that for every rule $r \in P$, $\phi r = \langle\ \{L\}, pos(r), neg(r)\ \rangle$, with $L \in head(r)$.*

**Proof.** Immediate from the definitions. □

**Lemma 9** *Let $P$ be a program with cover $Q$, and with $A, B \subset \mathcal{L}_P$. If $B$ is closed under $Q^A$, then $B$ is closed under $P^A$.*

**Proof.** Assume that $B$ is closed under $Q^A$. Let $r$ be a rule in $P^A$, with $pos(r) \subset B$. We must show that $head(r) \cap B \neq \emptyset$. By the definition of reduct, there is a rule $r' \in P$ s.t. $\{r\}^A = r'$. Of course $neg(r) \cap A = \emptyset$.

By Lemma 8 there is a total surjective function $\phi : P \rightarrow Q$ s.t. $\phi r' = \langle \{L\}, pos(r'), neg(r') \rangle$, with $L \in head(r')$. Let $r'' = \langle head(\phi r'), pos(r'), \emptyset \rangle$. It's clear that $r'' \in Q^A$, and we know that $pos(r'') \subset B$. Since $B$ is closed under $Q^A$, we can conclude that $head(r'') \cap B \neq \emptyset$. And since $head(r'') = head(\phi r') \subset head(r') = head(r)$, we have shown that $head(r) \cap B \neq \emptyset$. $\Box$

**Lemma 10** *Let $P$ be a program, with $A \subset \mathcal{L}_P$. For each cover $P'$ of $P^A$, there is a cover $Q$ of $P$ such that $Q^A = P'$.*

**Proof.** Let $P'$ be a cover of $P^A$. By Lemma 8, we know there is a total surjective function $\phi : P^A \rightarrow P'$ s.t. for every rule $r \in P^A$, $\phi r = \langle \{L\}, pos(r), neg(r) \rangle$, with $L \in head(r)$. By the definition of reduct, we can also specify a partial surjective function $\psi : P \rightarrow P^A$ s.t. for every rule $r \in P$, if $neg(r) \cap A = \emptyset$, then $\psi r = \langle head(r), pos(r), \emptyset \rangle$. Now we construct a cover $Q$ of $P$, with $Q^A = P'$. For each $r \in P$, do the following: (i) if $neg(r) \cap A \neq \emptyset$, replace $r$ by a rule $r' = \langle \{L\}, pos(r), neg(r) \rangle$, with $L \in head(r)$; (ii) if $neg(r) \cap A = \emptyset$, replace $r$ by the rule $r' = \phi\psi r$. $\Box$

**Lemma 11** *Let $P$ be a program with cover $Q$, and with $A, B \subset \mathcal{L}_P$. If $B$ is a consistent answer set for $P^A$, then there is a cover $P'$ of $P^A$ such that $B$ is closed under $P'$.*

**Proof.** Assume that $B$ is a consistent answer set for $P^A$. In step (ii) below we use the fact that $B$ is closed under $P^A$. To construct a suitable cover $P'$ for $P^A$, do the following for each $r \in P^A$: (i) if $neg(r) \cap A \neq \emptyset$ or if $pos(r) \not\subset B$, replace $r$ by a rule $r' = \langle \{L\}, pos(r), neg(r) \rangle$, with $L \in head(r)$; (ii) if $neg(r) \cap A = \emptyset$ and $pos(r) \subset B$, replace $r$ by a rule $r' = \langle \{L\}, pos(r), neg(r) \rangle$, with $L \in head(r) \cap B$. $\Box$

**Lemma 12** *Let $P$ be a program, with $A, B \subset \mathcal{L}_P$. If $B$ is a consistent answer set for $P^A$, then there is a cover $Q$ of $P$ such that $B$ is an answer set for $Q^A$.*

**Proof.** By Lemma 11, there is a cover $P'$ of $P^A$ such that $B$ is closed under $P'$. Let $Q$ be a cover of $P$ such that $Q^A = P'$. The existence of such a $Q$ is guaranteed by Lemma 10. Observe that $B$ is closed under $Q^A$ and consistent. It remains only to show that $B$ is the least subset of $\mathcal{L}_P$ that

is closed under $Q^A$. Let $D$ be a subset of $B$ that is closed under $Q^A$. We will show that $D = B$. By Lemma 9, we can conclude that $D$ is also closed under $P^A$. But $B$ is a consistent answer set for $P^A$, so $D = B$. Thus, $B$ is an answer set for $Q^A$. □

**Lemma 13** *Every consistent answer set for a constraint-free program $P$ is an answer set for some cover of $P$.*

**Proof.** Immediate from Lemma 12 and the definition of an answer set. □

**Lemma 14** *Let $P$ be a constraint-free program:*

$$\bigcap_{P' \in covers} Cn(P') \subset Cn(P) \ .$$

**Proof.** If $P$ is inconsistent, we're done; so assume that $P$ is consistent. By Lemma 13, every consistent answer set for $P$ is also an answer set for some cover $P'$ of $P$. □

**Proposition 1** *Let $P$ be a positive program. If $B \subset \mathcal{L}_P$ is closed under $P$ and logically closed, then some subset of $B$ is an answer set for $P$.*

**Proof.** Let $S$ be the set of subsets of $B$ that are closed under $P$. Consider the partial order $\langle S, \subset \rangle$. We will show that every chain in $\langle S, \subset \rangle$ has a lower bound (in $S$), from which it follows that $\langle S, \subset \rangle$ has minimal elements. Thus, some subset of $B$ is an answer set for $P$, since any minimal subset of $B$ that is closed under $P$ is also a minimal subset of $\mathcal{L}_P$ that is both closed under $P$ and logically closed (that is, an answer set for $P$). Let $\mathcal{C}$ be a chain in $\langle S, \subset \rangle$, with index set $I$. What we show, more precisely, is that the set of literals $\bigcap_{\alpha \in I} C_\alpha$, denoted by $\bigcap \mathcal{C}$, is closed under $P$, and thus must be included in $S$. Suppose $\bigcap \mathcal{C}$ is not closed under $P$. We will derive a contradiction. Since $\bigcap \mathcal{C}$ is not closed under $P$, we know there is a rule $r \in P$ s.t. $pos(r) \subset \bigcap \mathcal{C}$ and $head(r) \cap \bigcap \mathcal{C} = \emptyset$. Since $\bigcap \mathcal{C} \subset B$, we have $pos(r) \subset B$. And since $B$ is closed under $P$, we know that $head(r) \cap B \neq \emptyset$. For each literal $L \in head(r) \cap B$, let $X_L$ denote the intersection of all members of $\mathcal{C}$ that include the literal $L$. Observe that for each such $X_L$ we have $\bigcap \mathcal{C} \subset X_L$, and

of course $head(r) \cap X_L \neq \emptyset$. Also we can see that for all $L, L' \in head(r) \cap B$, it is the case that $X_L \subset X_{L'}$ or $X_{L'} \subset X_L$. Let $X$ denote the set

$$\bigcap_{L \in head(r) \cap B} X_L \ .$$

Note that $\bigcap \mathcal{C} \subset X$. Furthermore, since $head(r)$ is finite, so is $head(r) \cap B$. It follows that there is an $L \in head(r) \cap B$ such that $X = X_L$, from which we can conclude that $X \cap head(r) \neq \emptyset$. Since $\bigcap \mathcal{C} \cap head(r) = \emptyset$, we know that $X \neq \bigcap \mathcal{C}$. We can conclude there is a proper subset $Y$ of $X$ such that $Y$ belongs to $\mathcal{C}$. And by the construction of $X$, we know that $Y \cap head(r) = \emptyset$. Since $\bigcap \mathcal{C} \subset Y$, we have $pos(r) \subset Y$. So we've shown that $\mathcal{C}$ includes a set $Y$ that is not closed under $P$. Contradiction. So $\bigcap \mathcal{C}$ is closed under $P$, and thus belongs to $S$. That is, $\bigcap \mathcal{C}$ is a lower bound of $\mathcal{C}$. □

**Definition**. Let $P$ be a program with signing $S$:

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

**Definition**. Let $\mathcal{L}$ be a language, with $A, B, S \subset \mathcal{L}$, where $\overline{S} = S \setminus \mathcal{L}$. We say that $A \leq_S B$ if the following two conditions hold:

(i) $A \cap \overline{S} \subset B \cap \overline{S}$ ,

(ii) if $A \cap \overline{S} = B \cap \overline{S}$, then $A \subset B$ .

Observe that $\leq_S$ partially orders the subsets of $\mathcal{L}$.

**Lemma 15** *Let $P$ be a program with signing $S$. If $P$ has a consistent cover, then the partial order $\langle candidates(P), \leq_S \rangle$ has minimal elements.*

**Proof**. Since $P$ has a consistent cover, we know by the definition of $candidates(P)$ that $candidates(P) \neq \emptyset$. We will show that each maximal chain in $\langle candidates(P), \leq_S \rangle$ has a least element, from which it follows that $\langle candidates(P), \leq_S \rangle$ has minimal elements.

Let $\mathcal{C}$ be a maximal chain in $\langle candidates(P), \leq_S \rangle$, with index set $I$. If $\mathcal{C}$ is not infinite decreasing, then $\mathcal{C}$ has a least element and we're done. So we'll suppose that $\mathcal{C}$ is infinite decreasing, and from this assumption we'll derive a contradiction.

For each $i \in I$, let $P_i$ be a cover of $P$ s.t. $C_i = WF_\bot(P_i) \cup (WF_\top(P_i) \cap S)$. The existence of such a family of programs is guaranteed by the definition of $candidates(P)$. For each $i \in I$, let $\phi_i : P \to P_i$ be a total surjective function such that for every rule $r \in P$, $\phi_i r = \langle\ \{L\},\ pos(r),\ neg(r)\ \rangle$, with $L \in head(r)$. The existence of such a family of functions is guaranteed by Lemma 8.

For convenience, assume a well-founded total ordering on $\mathcal{L}_P$. Given a rule $r \in P$ and a chain $\mathcal{C}' \subset \mathcal{C}$ (with index set $I' \subset I$), let $choice(r,\mathcal{C}')$ denote the least literal $L \in head(r)$ which satisfies the following condition: for every $j \in I'$ there is a $i \in I'$ s.t. $i \leq j$ and $\phi_i r = \{L\}$. We can be sure of the existence of such an $L \in head(r)$ because $head(r)$ is finite.

Given a rule $r \in P$ and a chain $\mathcal{C}' \subset \mathcal{C}$ (with index $I' \subset I$), let $fix(r,\mathcal{C}') = \{C_i : i \in I' \wedge head(\phi_i r) = choice(r,\mathcal{C}')\}$. It's clear that $fix(r,\mathcal{C}')$ is always a subchain of $\mathcal{C}'$. Furthermore, by the definition of $choice(r,\mathcal{C}')$, we know that if $\mathcal{C}'$ is infinite decreasing, then so is $fix(r,C')$. Finally, observe that for all $i,j \in I$, if $C_i \in fix(r,C')$ and $C_j \in fix(r,C')$, then $\phi_i r = \phi_j r$.

Now, let $J$ be a well-founded, totally-ordered index set for the rules $r$ in $P$. For convenience we also stipulate that $J$ should have a greatest element, which we will denote by $\top$. Let $\bot$ denote the least element of $J$. For all $j \in J$,

$$fix_j = \begin{cases} fix(r_j, \mathcal{C}) & , \quad if\ j = \bot\ , \\ fix(r_j, \bigcap_{i<j} fix_i) & , \quad otherwise\ . \end{cases}$$

Since we've assumed that $\mathcal{C}$ is infinite decreasing, and because the *fix* function preserves the infinite decreasing property, it is clear (by induction) that for every $j \in J$, $fix_j$ is an infinite decreasing subchain of $\mathcal{C}$. But consider the chain $\mathcal{C}_J = fix_\top$. On the one hand, we know that $\mathcal{C}_J$ should be infinite decreasing. On the other hand, we will show that $\mathcal{C}_J$ must be a singleton.

Let $I_J = \{i \in I : C_i \in \mathcal{C}_J\}$. By the definition of $\mathcal{C}_J$ in terms of the *fix* function, we know that for every rule $r \in P$, and for all $i,j \in I_J$, it must be the case that $\phi_i r = \phi_j r$ Therefore we can conclude that for all $i,j \in I_J$, $P_i = P_j$, and thus that $C_i = C_j$. That is to say, $\mathcal{C}_J$ is a singleton.

So by supposing that there is a maximal chain in $\langle candidates(P), \leq_S \rangle$ that has no least element, we derive a contradiction, thereby establishing that each maximal chain in $\langle candidates(P), \leq_S \rangle$ has a least element.

□

**Definition**. Let $P$ be a program with signing $S$. We denote the set of members of *candidates*$(P)$ that are minimal in $\langle$*candidates*$(P), \leq_S\rangle$ by *minimal-candidates*$(P)$.

Note that the use of minimality in this definition is justified by the previous lemma.

**Lemma 16** *Let $P_d$ be a program with signing $S$. If $P_d$ has a consistent cover, then each member of minimal-candidates$(P_d)$ is a consistent answer set for $P_d$.*

**Proof**. Since $P_d$ has a consistent cover, we know by the definition of *candidates*$(P_d)$ that *minimal-candidates*$(P_d)$ is nonempty. Furthermore, by the previous lemma, $\langle$*candidates*$(P_d), \leq_S\rangle$ has minimal elements. Let $A$ be an element in *minimal-candidates*$(P_d)$. By the definition of *candidates*$(P_d)$, we know that there is a cover $Q$ of $P_d$ such that $A = WF_\bot(Q) \cup (WF_\top(Q) \cap S)$. Also, since $Q$ is a consistent program, we know by Theorem 1 that $A$ is a consistent answer set for $Q$. So $A$ is consistent and thus logically closed. By Lemma 9, we know that $A$ is closed under $P_d^A$. By Proposition 1 it follows that some subset of $A$ is an answer set for $P_d^A$. So, let us assume that $B \subset A$ is a consistent answer set for $P_d^A$; we'll be able to establish that $B$ must coincide with $A$, which will suffice to prove the lemma.

Let $P$ be a cover of $P_d$ such that $B = \alpha(P^A)$; the existence of such a program $P$ is guaranteed by Lemma 12. Note that since $B$ is a consistent set, we know that $P^A$ is a consistent program. Since $S$ is a signing for $P_d$, $S$ is also a signing for $P$. And since $P$ is a nondisjunctive program, we can conclude by Lemma 1(ii) that $B \cap S = \alpha(h_S(P)^{A \cap \overline{S}})$. By the asymmetry in the definition of a signing, we know that $h_S(P) = h_S(P_d)$; so $B \cap S = \alpha(h_S(P_d)^{A \cap \overline{S}})$. Recall that $Q$ is a cover of $P_d$ such that $A = \alpha(Q^A)$. Again by Lemma 1(ii), $A \cap S = \alpha(h_S(Q)^{A \cap \overline{S}})$; and again by the asymmetry of signings, $h_S(Q) = h_S(P_d)$. Therefore, $A \cap S = \alpha(h_S(P_d)^{A \cap \overline{S}})$, and thus we've established that $B \cap S = A \cap S$.

Now we will show that $A \cap S \subset WF_\top(P) \cap S$. We do this by supposing otherwise and deriving a contradiction. So suppose that $A \cap S \not\subset WF_\top(P) \cap S$. We have already established that $A \cap S = \alpha(h_S(P_d)^{A \cap \overline{S}})$; and since $h_S(P_d) = h_S(P)$, we have $A \cap S = \alpha(h_S(P)^{A \cap \overline{S}})$. Because $WF_\bot(P)$ is a fixpoint of $\Gamma_P^2$, we know that $\Gamma_P(WF_\bot(P))$ is also a fixpoint of $\Gamma_P^2$. Therefore, by the anti-monotonicity of $\Gamma_P$, along with the fact that $WF_\bot(P)$

and $WF_{\top}(P)$ are the least and greatest fixpoints of $\Gamma_P^2$, we can conclude that $WF_{\top}(P) = \Gamma_P(WF_{\bot}(P))$. Thus, by Lemma 1(ii) and the definition of $\Gamma_P$, we have $WF_{\top}(P) \cap S = \alpha(h_S(P)^{\mathrm{WF}_{\bot}(P) \cap \overline{S}})$. So we have shown that $\alpha(h_S(P)^{A \cap \overline{S}}) = A \cap S \not\subset WF_{\top}(P) \cap S = \alpha(h_S(P)^{\mathrm{WF}_{\bot}(P) \cap \overline{S}})$. By the monotonicity of $\alpha$ and the anti-monotonicity of the reduct operators, we can conclude that $WF_{\bot}(P) \cap \overline{S} \not\subset A \cap \overline{S}$. And since $B \cap \overline{S} \subset A \cap \overline{S}$, we have shown that $WF_{\bot}(P) \cap \overline{S} \not\subset B \cap \overline{S}$. By Lemma 6, $lfp(\Delta_S^P) = WF_{\bot}(P) \cap \overline{S}$. Thus, $lfp(\Delta_S^P) \not\subset B \cap \overline{S}$. Because $lfp(\Delta_S^P) \not\subset B \cap \overline{S}$, we know by Lemma 7 that $B \cap \overline{S}$ is not a pre-fixpoint of $\Delta_S^P$. That is, we know that $\Delta_S^P(B \cap \overline{S}) \not\subset B \cap \overline{S}$. Yet we can show that $\Delta_S^P(B \cap \overline{S}) \subset B \cap \overline{S}$, as follows. To begin, by definition, $\Delta_S^P(B \cap \overline{S}) = \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{B \cap \overline{S}})})$. Since $B \cap \overline{S} \subset A \cap \overline{S}$, we have $\alpha(h_S(P)^{A \cap \overline{S}}) \subset \alpha(h_S(P)^{B \cap \overline{S}})$, and therefore $\alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{B \cap \overline{S}})}) \subset \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{A \cap \overline{S}})})$ (by the anti-monotonicity of reduct operators and the monotonicity of $\alpha$). We've already established that $\alpha(h_S(P)^{A \cap \overline{S}}) = A \cap S$, and since $B = \alpha P^A$, we know by Lemma 1(iii) that $\alpha(h_{\overline{S}}(P)^{A \cap S}) = B \cap \overline{S}$. So, to sum up, we have shown that $\Delta_S^P(B \cap \overline{S}) = \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{B \cap \overline{S}})}) \subset \alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^{A \cap \overline{S}})}) = \alpha(h_{\overline{S}}(P)^{A \cap S}) = B \cap \overline{S}$. Thus, both $\Delta_S^P(B \cap \overline{S}) \subset B \cap \overline{S}$ and $\Delta_S^P(B \cap \overline{S}) \not\subset B \cap \overline{S}$, which is a contradiction. So we've established that $A \cap S \subset WF_{\top}(P) \cap S$.

We will now show that $B \cap \overline{S} = A \cap \overline{S}$. By the anti-monotonicity of the reduct operator and the monotonicity of $\alpha$, we can show that $\alpha(h_{\overline{S}}(P)^{\mathrm{WF}_{\top}(P) \cap S}) \subset \alpha(h_{\overline{S}}(P)^{A \cap S})$. Since $WF_{\top}(P)$ is a fixpoint of $\Gamma_P^2$, we know that $\Gamma_P(WF_{\top}(P))$ is also a fixpoint of $\Gamma_P^2$. Therefore, by the anti-monotonicity of $\Gamma_P$, along with the fact that $WF_{\bot}(P)$ and $WF_{\top}(P)$ are the least and greatest fixpoints of $\Gamma_P^2$, we can conclude that $WF_{\bot}(P) = \Gamma_P(WF_{\top}(P))$. Thus, by Lemma 1(iii) and the definition of $\Gamma_P$, we have $\alpha(h_{\overline{S}}(P)^{\mathrm{WF}_{\top}(P) \cap S}) = WF_{\bot}(P) \cap \overline{S}$. We have already shown that $\alpha(h_{\overline{S}}(P)^{A \cap S}) = B \cap \overline{S}$. To sum up so far, we have $WF_{\bot}(P) \cap \overline{S} = \alpha(h_{\overline{S}}(P)^{\mathrm{WF}_{\top}(P) \cap S}) \subset \alpha(h_{\overline{S}}(P)^{A \cap S}) = B \cap \overline{S}$. Yet $B \cap \overline{S} \subset A \cap \overline{S}$. So $WF_{\bot}(P) \cap \overline{S} \subset B \cap \overline{S} \subset A \cap \overline{S}$. And since $A$ is minimal in $candidates(P_d)$, we know that $WF_{\bot}(P) \cap \overline{S}$ cannot be a proper subset of $A \cap \overline{S}$. Therefore, $WF_{\bot}(P) \cap \overline{S} = B \cap \overline{S} = A \cap \overline{S}$. And since we've already shown that $B \cap S = A \cap S$, we have $B = A$. So we've shown that $A$ is a consistent answer set for $P_d$, and since $A$ was an arbitrarily chosen member of $minimal\text{-}candidates(P_d)$, it follows that each member of $minimal\text{-}candidates(P_d)$ is a consistent answer set for $P_d$.

□

**Theorem 2** *Let $P$ be a program with signing $S$:*

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

**Proof**. By Lemma 4, we have

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

By Lemma 14, we know that

$$\bigcap_{P'\in covers(P)} Cn(P')\cap\overline{S} \subset Cn(P)\cap\overline{S}\ .$$

We complete the proof by showing that

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

If $P$ has no consistent covers, we're done; so assume otherwise. It follows that

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

By the definition of $candidates(P)$,

$$\bigcap_{P'\in good\text{-}covers(P)} WF_\bot(P')\cap\overline{S} = \bigcap_{A\in candidates(P)} A\cap\overline{S}\ .$$

Furthermore, by the definition of $minimal\text{-}candidates(P)$, we can conclude that

$$\bigcap_{A\in candidates(P)} A\cap\overline{S} = \bigcap_{A\in minimal\text{-}candidates(P)} A\cap\overline{S}\ .$$

By Lemma 16, every element in $minimal\text{-}candidates(P)$ is a consistent answer set for $P$. Therefore,

$$Cn(P)\cap\overline{S} \subset \bigcap_{A\in minimal\text{-}candidates(P)} A\cap\overline{S}\ .$$

□

**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 the partial order $\langle candidates(P), \leq_S \rangle$ such that $A' \cap \overline{S} \subset A \cap \overline{S}$.*

**Proof.** To prove the first part, we first show that $\langle candidates(P), \leq_S \rangle$ has minimal elements. If $P$ has no consistent covers, then $candidates(P)$ is empty and the assertion is trivial. On the other hand, if $P$ has a consistent cover, then the assertion follows by Lemma 15. Next, assume that $A$ is a minimal element in $\langle candidates(P), \leq_S \rangle$. Thus, by the definition of $candidates(P)$, $P$ has a consistent cover, and it follows by Lemma 16 that $A$ is a consistent answer set for $P$. To prove the second part, observe first that by Lemma 13, every consistent answer set for $P$ is an answer set for a cover of $P$. Next, assume that $A$ is a consistent answer set for a cover $P'$ of $P$. Let $A'' = WF_{\bot}(P') \cup (WF_{\top}(P') \cap S)$. By the definition of $candidates(P)$, $A'' \in candidates(P)$. Furthermore, it's easy to show that $A'' \cap \overline{S} \subset A \cap \overline{S}$. We already know that $\langle candidates(P), \leq_S \rangle$ has minimal elements, so there must be a minimal element $A'$ in $\langle candidates(P), \leq_S \rangle$ such that $A' \cap \overline{S} \subset A'' \cap \overline{S}$. □

**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.*

**Proof.** ( (i) ⇒ (ii) ) : By Lemma 13, each consistent answer set for $P$ is an answer set for some cover of $P$.

( (ii) ⇒ (i) ) : If $P$ has a consistent cover, then *minimal-candidates*$(P)$ is nonempty. By Lemma 16, each member of *minimal-candidates*$(P)$ is a consistent answer set for $P$.

( (i) ⇒ (iii) ) : If $P$ is consistent, then $Cn(P)$ is consistent; so $Cn(P) \cap \overline{S}$ is too.

( (iii) ⇒ (i) ) : Assume that $Cn(P) \cap \overline{S}$ is consistent. Consider two cases.

*Case 1 :* $P$ is head-consistent.

In this case, each cover of $P$ is a signed head-consistent nondisjunctive program, so by Corollary 2, each cover of $P$ is consistent. We've already established that if $P$ has a consistent cover, then $P$ is consistent.

*Case 2 :* $P$ is not head-consistent.

It follows that $\overline{S}$ is an inconsistent set. And since $Cn(P) \cap \overline{S}$ is consistent, we know that $Cn(P) \cap \overline{S} \neq \overline{S}$. So $Cn(P) \neq \mathcal{L}_P$. From this we can conclude that $P$ is consistent.

□

**Lemma 17** *The ordering relation $\preceq$ for rules is transitive.*

**Proof**. Assume $r_3 \preceq r_2$ and $r_2 \preceq r_1$. Thus, $neg(r_1) \subset neg(r_2)$ and $\{\overline{L} : L \in head(r_1) \setminus head(r_2)\} \cup pos(r_1) \subset pos(r_2)$. And also $neg(r_2) \subset neg(r_3)$ and $\{\overline{L} : L \in head(r_2) \setminus head(r_3)\} \cup pos(r_2) \subset pos(r_3)$. So clearly we have $neg(r_1) \subset neg(r_3)$ and $pos(r_1) \subset pos(r_3)$. It is easy to verify that $X \setminus Z \subset X \setminus Y \cup Y \setminus Z$, for arbitrary sets $X, Y, Z$. Thus, we can conclude that $\{\overline{L} : L \in head(r_1) \setminus head(r_3)\} \subset \{\overline{L} : L \in (head(r_1) \setminus head(r_2)) \cup (head(r_2) \setminus head(r_3))\}$. And since $\{\overline{L} : L \in head(r_1) \setminus head(r_2)\} \subset pos(r_3)$, and $\{\overline{L} : L \in head(r_2) \setminus head(r_3)\} \subset pos(r_3)$, we have $\{\overline{L} : L \in head(r_1) \setminus head(r_3)\} \subset pos(r_3)$. Therefore, $neg(r_1) \subset neg(r_3)$ and $\{\overline{L} : L \in head(r_1) \setminus head(r_3)\} \cup pos(r_1) \subset pos(r_3)$; that is, $r_3 \preceq r_1$. So $\preceq$ for rules is transitive. □

**Lemma 18** *The ordering relation $\preceq$ for programs is transitive.*

**Proof**. Immediate by the definition of $\preceq$ for programs and Lemma 17. □

**Lemma 19** *Let $P, Q$ be nondisjunctive programs, with $\mathcal{L}_P = \mathcal{L}_Q$, and with $X, Y \subset \mathcal{L}_P$. If $P \preceq Q$ and $Y \subset X$, then $P^X \preceq Q^Y$.*

**Proof**. By the definition of $\preceq$ for programs, we know that for every rule $r \in P$ there is a rule $r' \in Q$ s.t. $r \preceq r'$. By the definition of $\preceq$ for rules, we can conclude that for every rule $r \in P$ there is a rule $r' \in Q$ s.t. $neg(r') \subset neg(r)$ and $\{\overline{L} : L \in head(r') \setminus head(r)\} \cup pos(r') \subset pos(r)$. In particular, for every rule $r \in P$, if $neg(r) \cap X = \emptyset$, then there is a rule $r' \in Q$

s.t. $neg(r') \cap X = \emptyset$ and $\{\overline{L} : L \in head(r') \setminus head(r)\} \cup pos(r') \subset pos(r)$. Therefore, we can conclude that $P^X \preceq Q^X$. Since $Y \subset X$, we know by the monotonicity of reduct that $Q^X \subset Q^Y$, and thus that $Q^X \preceq Q^Y$. So by the transitivity of $\preceq$ (Lemma 18), we have $P^X \preceq Q^Y$. $\Box$

**Lemma 20** *Let $P, Q$ be basic programs. If $P \preceq Q$ and $Q$ is consistent, then $\alpha P \subset \alpha Q$.*

**Proof**. It seems that in order to prove this lemma we must resort to a version of the standard "immediate consequence" operator $T_P$, where $P$ is a basic program. Relying on the familiarity of $T_P$, we make several claims about it without proof. Given a set of literals $X \subset \mathcal{L}_P$, $T_P X$ is, intuitively, the set of literals derivable by the rules of $P$ in one step from the literals in $X$. Formally,

$$T_P X = \{L \in \mathcal{L}_P : \exists r \in P \,.\, pos(r) \subset X \wedge head(r) = \{L\}\} \ .$$

$T_P$ is monotone and continuous, with

$$\bigcup_{i \in \{0,1,2,\ldots\}} T_P^i \emptyset = \alpha(P) \ .$$

We give an inductive proof for the following assertion, of which this lemma is an immediate consequence: for basic programs $P$ and $Q$,

$$P \preceq Q \Rightarrow \left(\forall n \in \{0,1,2,\ldots\} \,.\, T_Q^n \emptyset \textit{ is consistent } \Rightarrow T_P^n \emptyset \subset T_Q^n \emptyset\right) \ .$$

*Base Case :* $n = 0$. Trivial.

*Inductive Step :* Assume that $T_Q^{n+1}\emptyset$ is consistent. Since $T_Q$ is monotone, we know that $T_Q^n \emptyset \subset T_Q^{n+1}\emptyset$, so $T_Q^n \emptyset$ is also consistent. Therefore, by the inductive hypothesis, we have $T_P^n \emptyset \subset T_Q^n \emptyset$. We need to show that $T_P^{n+1}\emptyset \subset T_Q^{n+1}\emptyset$.

Now, suppose that there is a rule $r \in P$ s.t. $pos(r) \subset T_P^n \emptyset$ and $head(r) \neq head(\phi r)$. We'll show that this supposition leads to a contradiction of our assumption that that $T_Q^{n+1}\emptyset$ is consistent. Since $P \preceq Q$, we know by the definition of $\preceq$ that there is a total function $\psi : P \to Q$ s.t. for every rule $r \in P$, $\{\overline{a} : a \in head(\psi r) \setminus head(r)\} \cup pos(\psi r) \subset pos(r)$. By the definition of $\psi$, we know that $\{\overline{a} : a \in head(\phi r) \setminus head(r)\} \cup pos(\phi r) \subset$

$pos(r)$. Since $head(\phi r) \neq head(r)$, we know that there is a literal $a$ such that $head(\phi r) = \{a\}$ and $\overline{a} \in pos(r)$. Since $pos(r) \subset T_P^n\emptyset$, we have $\overline{a} \in T_P^n\emptyset$. Since $T_P^n\emptyset \subset T_Q^n\emptyset$, we have $\overline{a} \in T_Q^n\emptyset$. And since $T_Q$ is monotone, we have $\overline{a} \in T_Q^{n+1}\emptyset$. But since rule $\phi r$ of program $Q$ contributes the literal $a$ to $T_Q^{n+1}\emptyset$, we also have $a \in T_Q^{n+1}\emptyset$. Therefore $a, \overline{a} \in T_Q^{n+1}\emptyset$, and thus $T_Q^{n+1}\emptyset$ is not consistent. So we've shown by contradiction that for every rule $\in P$, if $pos(r) \subset T_P^n\emptyset$, then $head(\phi r) = head(r)$. We complete the proof by considering the corresponding cases on rules $r \in P$.

*Case 1 :* $pos(r) \not\subset T_P^n\emptyset$.

In this case, rule $r$ does not contribute the literal in $head(r)$ to $T_P^n\emptyset$, so such a rule $r \in P$ cannot cause the falsification of $T_P^{n+1}\emptyset \subset T_Q^{n+1}\emptyset$.

*Case 2 :* $pos(r) \subset T_P^n\emptyset$.

In this case, rule $r$ does contribute the literal in $head(r)$ to $T_P^{n+1}\emptyset$. But because $T_P^n\emptyset \subset T_Q^n\emptyset$ and $pos(\phi r) \subset pos(r)$ and $head(\phi r) = head(r)$, we know that rule $\phi r$ of program $Q$ contributes the same literal to $T_Q^{n+1}\emptyset$. So such a rule $r \in P$ cannot cause the falsification of $T_P^{n+1}\emptyset \subset T_Q^{n+1}\emptyset$.

□

**Lemma 21** *Let $P, Q$ be nondisjunctive programs with $\mathcal{L}_P = \mathcal{L}_Q$, and with $S$ a signing for both $P$ and $Q$, and with $X \subset \mathcal{L}_P$. If $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$ and $h_S(Q) \preceq h_S(P)$ and $\Delta_S^Q X$ is consistent, then $\Delta_S^P X \subset \Delta_S^Q X$.*

**Proof**. By Lemma 19, since $h_S(Q) \preceq h_S(P)$, we know that $h_S(Q)^X \preceq h_S(P)^X$. Since $S$ is a signing for $P$, we know that $h_S(P)^X$ is a head-consistent positive program, and thus that it is consistent. So by Lemma 20 $\alpha(h_S(Q)^X) \subset \alpha(h_S(P)^X)$. Once more by Lemma 19, since $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$ and $\alpha(h_S(Q)^X) \subset \alpha(h_S(P)^X)$ we have $h_{\overline{S}}(P)^{\alpha(h_S(P)^X)} \preceq h_{\overline{S}}(Q)^{\alpha(h_S(Q)^X)}$. Because $\Delta_S^Q X$ is consistent, we know that program $h_{\overline{S}}(Q)^{\alpha(h_S(Q)^X)}$ is consistent; so by Lemma 20 we have $\alpha(h_{\overline{S}}(P)^{\alpha(h_S(P)^X)}) \subset \alpha(h_{\overline{S}}(Q)^{\alpha(h_S(Q)^X)})$. That is to say, $\Delta_S^P X \subset \Delta_S^Q X$. □

Now we prove a slightly more general, and stronger, version of the restricted monotonicity theorem from [Turner, 1993] for nondisjunctive programs.

**Proposition 2** *Let $P, Q$ be nondisjunctive 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 if $A'$ is*

*consistent answer set for program $Q$, then there is a consistent answer set $A$ for program $P$ such that $A \cap \overline{S} \subset A' \cap \overline{S}$.*

**Proof**. If $Q$ is inconsistent, we're done; so assume that $Q$ is consistent. Assume that $A'$ is a consistent answer set for $Q$. By Proposition 1 we know that $WF_\perp(Q) \cap \overline{S}$ is consistent, and by Corollary 1, $Cn(Q) \cap \overline{S} = WF_\perp(Q) \cap \overline{S}$. It follows that $WF_\perp(Q) \cap \overline{S} \subset A' \cap \overline{S}$. By Lemma 6, we know that $WF_\perp(P) \cap \overline{S} = lfp(\Delta^P_S)$ and $WF_\perp(Q) \cap \overline{S} = lfp(\Delta^Q_S)$. Since $WF_\perp(Q) \cap \overline{S}$ is a fixpoint of $\Delta^Q_S$, we have $\Delta^Q_S(WF_\perp(Q) \cap \overline{S}) = WF_\perp(Q) \cap \overline{S}$. So $\Delta^Q_S(WF_\perp(Q) \cap \overline{S})$ is a consistent set, and thus by Lemma 21 we know that $\Delta^P_S(WF_\perp(Q) \cap \overline{S}) \subset \Delta^Q_S(WF_\perp(Q) \cap \overline{S})$. So $\Delta^P_S(WF_\perp(Q) \cap \overline{S}) \subset WF_\perp(Q) \cap \overline{S}$. That is, $WF_\perp(Q) \cap \overline{S}$ is a pre-fixpoint of $\Delta^P_S$. By the Knaster-Tarski theorem [Tarski, 1955], the least fixpoint of a monotone operator coincides with the intersection of all its pre-fixpoints. Thus we've shown that $WF_\perp(P) \cap \overline{S} \subset WF_\perp(Q) \cap \overline{S}$. So $WF_\perp(P) \cap \overline{S}$ is consistent. Let $A = WF_\perp(P) \cup (WF_\top(P) \cap \overline{S})$. By Theorem 1, $A$ is a consistent answer set for $P$; and we can see that $A \cap \overline{S} \subset A' \cap \overline{S}$. □

**Lemma 22** *, Let $P, Q$ be programs. If $P \preceq Q$, then for every cover $Q'$ of $Q$ there is a cover $P'$ of $P$ such that $P' \preceq Q'$.*

**Proof**. Let $\psi : P \to Q$ be a total function such that for every rule $r \in P$, $r \preceq \psi r$. We know that such a function exists since $P \preceq Q$. Let $Q'$ be an arbitrary cover of program $Q$. By Lemma 8, we know there is a total surjective function $\phi : Q \to Q'$ such that for every rule $r \in Q$, $\phi r = \langle\ \{L\}, pos(r), neg(r)\ \rangle$, with $L \in head(r)$. Since $P \preceq Q$, we know that for every rule $r \in P$ $neg(\psi r) \subset neg(r)$ and $\{\overline{L} : L \in head(\psi r) \setminus head(r)\} \cup pos(\psi r) \subset pos(r)$. Thus we know that for every rule $r \in P$, $neg(\phi\psi r) \subset neg(r)$ and $pos(\phi\psi r) \subset pos(r)$.

Now we specify a construction for a cover $P'$ of $P$ s.t. $P' \preceq Q'$. Let $\phi'$ be a surjective function with domain $P$ s.t. for every rule $r \in P$: (i) if $head(\phi\psi r) \subset head(r)$, then $\phi' r = \langle\ head(\phi\psi r), pos(r), neg(r)\ \rangle$; (ii) if $head(\phi\psi r) \not\subset head(r)$, $\phi' r = \langle\ \{L\}, pos(r), neg(r)\ \rangle$, with $L \in head(r)$. By Lemma 8, $P' = \{\phi' r : r \in P\}$ is a cover of $P$. It remains to show that $P' \preceq Q'$. Consider cases on the rules $\phi' r \in P'$.

*Case 1 :* $head(\phi\psi r) \subset head(r)$.

In this case, we have $head(\phi' r) = head(\phi\psi r)$, with $neg(\phi\psi r) \subset neg(\phi' r)$ and $pos(\phi\psi r) \subset pos(\phi' r)$. So $\phi' r \preceq \phi\psi r$.

*Case 2 :* $head(\phi\psi r) \not\subset head(r)$.

Since $r \preceq \psi r$, we know that $\{\overline{L} : L \in head(\psi r) \setminus head(r)\} \cup pos(\psi r) \subset pos(r)$. Let $L \in \mathcal{L}_P$ be such that $head(\phi\psi r) = \{L\}$. We see that $L \in head(\psi r) \setminus head(r)$, so $\overline{L} \in pos(r)$. And since $pos(\phi' r) = pos(r)$. We have $\{\overline{L} : L \in head(\phi\psi r) \setminus head(\phi' r)\} \subset pos(\phi' r)$. We also know that $neg(\phi\psi r) \subset neg(\phi' r)$ and $pos(\phi\psi r) \subset pos(\phi' r)$. So $\phi' r \preceq \phi\psi r$.

□

**Lemma 23** , *Let $P, Q$ be programs with $\mathcal{L}_P = \mathcal{L}_Q$, and with $S$ a signing for both $P$ and $Q$. If $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$ and $h_S(Q) \preceq h_S(P)$, then 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 $h_S(Q') \preceq h_S(P')$.*

**Proof**. Recall that if $S$ is a signing for a program, then $S$ is also a signing for each cover of that program. We know by the asymmetry of the definition of a signing that for every cover $P'$ of $P$ we have $h_S(P') = h_S(P)$. Likewise for every cover $Q'$ of $Q$. So we already know that if $h_S(Q) \preceq h_S(P)$, then for every cover $Q'$ of $Q$ there is a cover $P'$ of $P$ such that $h_S(Q') \preceq h_S(P')$. It remains only to show that if $h_{\overline{S}}(P) \preceq h_{\overline{S}}(Q)$, then for every cover $Q'$ of $Q$ there is a cover $P'$ of $P$ such that $h_{\overline{S}}(P') \preceq h_{\overline{S}}(Q')$. Notice that if $P'$ is a cover of $h_S(P)$ and $P''$ is a cover of $h_{\overline{S}}(P)$, then $P' \cup P''$ is a cover of $P$, with $P' \cap P'' = \emptyset$, and with $h_S(P' \cup P'') = P'$ and $h_{\overline{S}}(P' \cup P'') = P''$. Thus, it is enough to show that for any cover $Q'$ of $h_{\overline{S}}(Q)$ there is a cover $P'$ of $h_{\overline{S}}(P)$ such that $P' \preceq Q'$, which follows from the previous lemma. □

**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 $Cn(P) \cap \overline{S} \subset Cn(Q) \cap \overline{S}$.*

**Proof**. This theorem will follow easily from the next theorem. □

**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 if $A'$ is a consistent answer set for $Q$, then there is a consistent answer set $A$ for $P$ such that $A \cap \overline{S} \subset A' \cap \overline{S}$.*

**Proof**. 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 the partial order $\angle candidates(Q), \leq_S\rangle$. By the definition of $candidates(Q)$, there is a cover $Q'$ of $Q$ such that $A''$ is an answer set for $Q'$. By Lemma 23, there is a cover $P'$ of $P$ s.t. $h_{\overline{S}}(P') \preceq h_{\overline{S}}(Q')$ and $h_S(Q') \preceq h_S(P')$. So by Proposition 2, there is a consistent answer set $A'''$ for $P'$ such that $A''' \cap \overline{S} \subset A'' \cap \overline{S}$. Again by the Signing Lemma there is a consistent answer set $A$ for $P$ such that $A \cap \overline{S} \subset A''' \cap \overline{S}$. □

The proof of Theorem 6 will use Theorem 7, so we present the proof of Theorem 7 first. We begin the proof of Theorem 7 by reproducing the Splitting Sequence Theorem from [Lifschitz and Turner, 1994], which is needed in the proofs of two of the following lemmas.

**Definition**. Given a rule $r$, $literals(r)$ stands for $head(r) \cup pos(r) \cup neg(r)$. A *splitting set* for a program $P$ is any set $U$ of literals such that, for every rule $r \in P$, if $head(r) \cap U$ is nonempty then $literals(r) \subset U$. The set of rules $r \in P$ such that $literals(r) \subset U$ is called the *bottom* of $P$ relative to the splitting set $U$ and denoted by $b_U(P)$. The set $P \setminus b_U(P)$ is the *top* of $P$ relative to $U$. A *(transfinite) sequence* is a family whose index set is an initial segment of ordinals, $\{\alpha : \alpha < \mu\}$. The ordinal $\mu$ is the *length* of the sequence. A sequence $\langle U_\alpha\rangle_{\alpha<\mu}$ of sets is *monotone* if $U_\alpha \subset U_\beta$ whenever $\alpha < \beta$, and *continuous* if, for each limit ordinal $\alpha < \mu$, $U_\alpha = \bigcup_{\eta<\alpha} U_\eta$. A *splitting sequence* for a program $P$ is a monotone, continuous sequence $\langle U_\alpha\rangle_{\alpha<\mu}$ of splitting sets for $P$ such that $\bigcup_{\alpha<\mu} U_\alpha = literals(P)$. Let $U = \langle U_\alpha\rangle_{\alpha<\mu}$ be a splitting sequence for a program $P$. A *solution* to $P$ (with respect to $U$) is a sequence $\langle X_\alpha\rangle_{\alpha<\mu}$ of sets of literals such that

- $X_0$ is an answer set for $b_{U_0}(P)$,
- for any $\alpha$ such that $\alpha + 1 < \mu$, $X_{\alpha+1}$ is an answer set for

$$e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu\leq\alpha} X_\nu),$$

- for any limit ordinal $\alpha < \mu$, $X_\alpha = \emptyset$,
- $\bigcup_{\alpha<\mu} X_\alpha$ is consistent.

**Splitting Sequence Theorem.** *Let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a program $P$. A set $A$ of literals is a consistent answer set for $P$ if and only if $A = \bigcup_{\alpha<\mu} X_\alpha$ for some solution $\langle X_\alpha \rangle_{\alpha<\mu}$ to $P$ with respect to $U$.*

Next we extend the standard definition of stratification to apply as well to programs with classical negation. Under this definition, it is easy to see that the programs obtained by the translation $\pi$ are stratified.

**Definition**. A constraint-free program $P$ is *stratified* if there is a level mapping $f$ from $\mathcal{L}_P$ to the set of ordinals such that for every rule $r \in P$ the following conditions hold:

- for all $L, L' \in head(r)$, $f(L) = f(L')$,
- for all $L \in pos(r)$ there is an $L' \in head(r)$ such that $f(L') \geq f(L)$,
- for all $L \in neg(r)$ there is an $L' \in head(r)$ such that $f(L') > f(L)$.

**Lemma 24** *If a consistent nondisjunctive program is stratified, it has a unique answer set.*

This lemma is an easy consequence of the Splitting Sequence Theorem. We omit the proof, which is almost identical to the proof of Proposition 4 in [Lifschitz and Turner, 1994].

**Lemma 25** *Let $D$ be a domain description without value propositions. Each consistent cover of $\pi D$ has a unique answer set.*

**Proof**. Follows immediately from the previous lemma, since each consistent cover of $\pi D$ is a stratified nondisjunctive program. □

**Lemma 26** *Let $D$ be a consistent domain description without value propositions. Let $M$ be a model of $D$. There is a consistent cover $P$ of $\pi D$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ belongs to the unique answer set for $P$.*

**Proof**. The plan of the proof is as follows. First we specify a suitable cover $P$ of $\pi D$. Second we define a consistent set $X$ of literals such that an atomic value proposition $V$ is true in $M$ if and only if $\pi V \in X$. We then show that $X$ is an answer set for $P$. From this it follows by the previous lemma that $X$ is the unique answer set for $P$, which suffices to establish the lemma.

We define a suitable $P$ as follows. For each fluent expression $F$, $P$ includes the rule $\mathit{Holds}(F, S_0) \leftarrow$ if the atomic value proposition **initially** $F$ is true in $M$. $P$ also includes the effect rule (3) and the noninertial rule (4) for each effect proposition in $D$, as well as the inertia rules (5). Clearly $P$ is a cover of $\pi D$.

We define a suitable set $X$ of literals as follows. For each atomic value proposition $V$, the literal $\pi V$ belongs to $X$ if $V$ is true in $M$. Furthermore, for each fluent $F$, action $A$, and sequence of actions $A_1, \ldots, A_k$ $(k \geq 0)$, the literal $\mathit{Noninertial}(F, A, \mathit{Result}(A_k, \ldots, \mathit{Result}(A_1, S_0) \ldots))$ belongs to $X$ if either: (i) there is an effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$, or (ii) there is an effect proposition $A$ **causes** $\neg F$ **if** $G_1, \ldots, G_n$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Clearly $X$ is consistent.

It remains only to show that $X$ is an answer set for $P$. We will use the Splitting Sequence Theorem for this purpose, so we begin by defining a suitable splitting sequence for $P$.

First, we define a sequence $T = \langle T_\alpha \rangle_{\alpha < \omega}$ of sets of situation terms, as follows. Let $T_0 = \{S_0\}$, and for all $\alpha < \omega$, $T_{\alpha+1}$ is the set of all terms of the form $\mathit{Result}(A, S)$, where $A$ is an action name and $S \in T_\alpha$.

Given $T$, we define a splitting sequence $U = \langle U_\alpha \rangle_{\alpha < \omega}$ as follows. For all $\alpha < \omega$, $U_{2\alpha}$ consists of the literals in $\bigcup_{\gamma < 2\alpha} U_\gamma$ along with all literals of the form $\mathit{Holds}(F, S)$, where $F$ is a fluent expression and $S \in T_\alpha$. For all $\alpha < \omega$, $U_{2\alpha+1}$ consists of the literals in $\bigcup_{\gamma < 2\alpha+1} U_\gamma$ along with all literals of the form $\mathit{Noninertial}(F, A, S)$, where $F$ is a fluent name, $A$ is an action and $S \in T_\alpha$.

Finally, we define the sequence $\langle X_\alpha \rangle_{\alpha < \omega}$ such that for all $\alpha < \omega$, $X_\alpha = X \cap (U_\alpha \setminus \bigcup_{\gamma < \alpha} U_\gamma)$. We will show that this sequence is a solution to $P$ with respect to $U$. Since $\bigcup_{\alpha < \omega} X_\alpha = X$, it will follow by the Splitting Sequence Theorem that $X$ is an answer set for $P$.

Program $b_{U_0}(P)$ consists entirely of rules of the form

$$\mathit{Holds}(F, S_0) \leftarrow ,$$

where $F$ is a fluent expression. In fact, for each fluent name $F$, the rule $Holds(F, S_0) \leftarrow$ belongs to program $b_{U_0}(P)$ if and only if the value proposition **initially** $F$ is true in $M$, and likewise, the rule $\neg Holds(F, S_0) \leftarrow$ belongs to program $b_{U_0}(P)$ if and only if the value proposition **initially** $\neg F$ is true in $M$. Thus, by the definition of $X$, we know that for each fluent expression $F$, the rule $Holds(F, S_0) \leftarrow$ belongs to program $b_{U_0}(P)$ if and only if the literal $Holds(F, S_0)$ belongs to $X_0$. We can conclude that $X_0$ is an answer set for program $b_{U_0}(P)$.

There are no limit ordinals less than $\omega$, so for any limit ordinal $\alpha < \omega$, $X_\alpha = \emptyset$. Furthermore, we know that $\bigcup_{\alpha<\omega} X_\alpha$ is consistent, since $X$ is. It remains only to show that for all $\alpha + 1 < \omega$, $X_{\alpha+1}$ is an answer set for the program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu\leq\alpha} X_\nu)$. This last obligation requires consideration of many details, but is otherwise straightforward. We consider two cases, beginning with the simpler case.

*Case 1 :* $\alpha$ is even.

In this case, the program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu\leq\alpha} X_\nu)$ consists entirely of rules of the form

$$Noninertial(F, A, S) \leftarrow ,$$

where the literal $Noninertial(F, A, S)$ belongs to $U_{\alpha+1} \setminus U_\alpha$. So what we need to show is that a literal $Noninertial(F, A, S)$ from $U_{\alpha+1} \setminus U_\alpha$ is a member of $X_{\alpha+1}$ if and only if the rule $Noninertial(F, A, S) \leftarrow$ is a member of $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu\leq\alpha} X_\nu)$.

We can begin by observing that the program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ consists entirely of "noninertial" rules of the form

$$Noninertial(|F|, A, S) \leftarrow not\ \ \overline{Holds(G_1, S)}, \ldots, not\ \ \overline{Holds(G_n, S)} ,$$

where $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ is an effect proposition in $D$ and the literal $Noninertial(|F|, A, S)$ is a member of $U_{\alpha+1} \setminus U_\alpha$. For each such rule $r$, the corresponding rule $Noninertial(|F|, A, S) \leftarrow$ belongs to program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu\leq\alpha} X_\nu)$ if $neg(r) \cap X_\alpha = \emptyset$.

(Left-to-right): Assume $Noninertial(F, A, S) \in X_{\alpha+1}$. Let $A_1; \ldots; A_k$ be the sequence of actions such that

$$S = Result(A_k, Result(A_{k-1}, \ldots, Result(A_1, S_0) \ldots)) .$$

By the definition of $X$, there exists an action $A$ along with fluent expressions $G_1, \ldots, G_n$ $(n \geq 0)$ such that either: (i) there is an effect proposition

$A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$, or (ii) there is an effect proposition $A$ **causes** $\neg F$ **if** $G_1, \ldots, G_k$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Since either (i) or (ii) holds, we know that the rule

$$\mathit{Noninertial}(F, A, S) \leftarrow \mathit{not}\ \ \overline{\mathit{Holds}(G_1, S)}, \ldots, \mathit{not}\ \ \overline{\mathit{Holds}(G_n, S)}\ ,$$

belongs to program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$. Again by the definition of $X$, the literals $\mathit{Holds}(G_1, S), \ldots, \mathit{Holds}(G_n, S)$ are members of $X_\alpha$, and since $X$ is consistent, it follows that the literals $\overline{\mathit{Holds}(G_1, S)}, \ldots, \overline{\mathit{Holds}(G_n, S)}$ are not members of $X_\alpha$. Thus we can conclude by the definition of $e_{U_\alpha}$ that the rule $\mathit{Noninertial}(F, A, S) \leftarrow$ belongs to program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$.

(Right-to-left): Assume that the rule $\mathit{Noninertial}(F, A, S) \leftarrow$ belongs to program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$. We can conclude that program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ includes a rule

$$\mathit{Noninertial}(F, A, S) \leftarrow \mathit{not}\ \ \overline{\mathit{Holds}(G_1, S)}, \ldots, \mathit{not}\ \ \overline{\mathit{Holds}(G_n, S)}\ ,$$

satisfying the following three conditions.

- The literal $\mathit{Noninertial}(F, A, S)$ is a member of $U_{\alpha+1} \setminus U_\alpha$.
- Either: (i) $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ is an effect proposition in $D$, or (ii) $A$ **causes** $\neg F$ **if** $G_1, \ldots, G_n$ is an effect proposition in $D$.
- For all $i \in \{1, \ldots, n\}$, $\overline{\mathit{Holds}(G_i, S)} \notin X_\alpha$.

Let $A_1; \ldots; A_k$ be the sequence of actions such that

$$S = \mathit{Result}(A_k, \mathit{Result}(A_{k-1}, \ldots, \mathit{Result}(A_1, S_0) \ldots))\ .$$

We can conclude by the definition of $X$ that for all $i \in \{1, \ldots, n\}$, the value proposition $\overline{G_i}$ **after** $A_1; \ldots; A_k$ is false in $M$, where $\overline{G_i}$ denotes the fluent expression complementary to $G_i$. It follows that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. And since $D$ includes either: (i) an effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$, or (ii) an effect proposition $A$ **causes** $\neg F$ **if** $G_1, \ldots, G_n$, we can conclude by the definition of $X$ that the literal $\mathit{Noninertial}(F, A, S)$ belongs to $X_{\alpha+1}$.

*Case 2 :* $\alpha$ is odd.

In this case, the program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$ consists entirely of rules of the form

$$Holds(F, Result(A, S)) \leftarrow ,$$

where $F$ is a fluent expression and the literal $Holds(F, Result(A, S))$ belongs to $U_{\alpha+1} \setminus U_\alpha$. What we must show is that a literal $Holds(F, Result(A, S))$ from $U_{\alpha+1} \setminus U_\alpha$ belongs to $X_{\alpha+1}$ if and only if the rule $Holds(F, Result(A, S)) \leftarrow$ belongs to program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$.

The program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ includes one rule of the form

$$Holds(F, Result(A, S)) \leftarrow Holds(G_1, S), \ldots, Holds(G_n, S)$$

for each effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$, where the literal $Holds(F, Result(A, S))$ belongs to $U_{\alpha+1} \setminus U_\alpha$. Also, for each fluent expression $F$, program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ includes the inertia rule

$$Holds(F, Result(A, S)) \leftarrow Holds(F, S), \mathit{not}\ Noninertial(|F|, A, S) ,$$

where the literal $Holds(F, Result(A, S))$ belongs to $U_{\alpha+1} \setminus U_\alpha$.

(Left-to-right) : Assume $Holds(F, Result(A, S)) \in X_{\alpha+1}$, where $F$ is a fluent expression. Let $A_1; \ldots; A_k$ be the sequence of actions such that

$$S = Result(A_k, Result(A_{k-1}, \ldots, Result(A_1, S_0) \ldots)) .$$

By the definition of $X$, the value proposition $F$ **after** $A_1; \ldots; A_k; A$ is true in $M$. Now consider two subcases.

*Subcase 1 :* $Noninertial(|F|, A, S) \in X_\alpha$.

By the definition of $X$, there exists an action $A$ along with fluent expressions $G_1, \ldots, G_n$ $(n \geq 0)$ such that either: (i) there is an effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$, or (ii) there is an effect proposition $A$ **causes** $\overline{F}$ **if** $G_1, \ldots, G_n$ in $D$, where $\overline{F}$ is the fluent expression complementary to $F$, such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Since we know that the value proposition $F$ **after** $A_1; \ldots; A_k; A$ is true in $M$, we can conclude that (ii) cannot be the case, so (i) holds. So program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ includes the rule

$$Holds(F, Result(A, S)) \leftarrow Holds(G_1, S), \ldots, Holds(G_n, S) .$$

Let $\beta$ be such that $\beta + 1 = \alpha$. Again by the definition of $X$, we know that the literals $Holds(G_1, S), \ldots, Holds(G_n, S)$ are members of $X_\beta$. Thus these literals belong to $\bigcup_{\nu \leq \alpha} X_\nu$, so it follows by the definition of $e_{U_\alpha}$ that the rule $Holds(F, Result(A, S)) \leftarrow$ belongs to the program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$.

*Subcase 2 :* $Noninertial(|F|, A, S) \notin X_\alpha$.

By the definition of $X$, we can conclude that there is no effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Nor is there an effect proposition $A$ **causes** $\overline{F}$ **if** $G_1, \ldots, G_n$ in $D$, where $\overline{F}$ is the fluent expression complementary to $F$, such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Since the value proposition $F$ **after** $A_1; \ldots; A_k; A$ is true in $M$, we can conclude by the definition of $\Phi$ that the value proposition $F$ **after** $A_1; \ldots; A_k$ is true in $M$. Let $\beta$ be such that $\beta + 1 = \alpha$. By the definition of $X$, we have $Holds(F, S) \in X_\beta$. So we've shown that the literal $Holds(F, S)$ belongs to $\bigcup_{\nu \leq \alpha} X_\nu$, while the literal $Noninertial(|F|, A, S)$ does not. Program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ includes the inertia rule

$$Holds(F, Result(A, S)) \leftarrow Holds(F, S), \mathit{not}\ Noninertial(|F|, A, S)\ ,$$

and it follows by the definition of $e_{U_\alpha}$ that the rule $Holds(F, Result(A, S)) \leftarrow$ belongs to $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$.

(Right-to-left) : Assume that the rule $Holds(F, Result(A, S)) \leftarrow$ belongs to program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$. Let $A_1; \ldots; A_k$ be the sequence of actions such that

$$S = Result(A_k, Result(A_{k-1}, \ldots, Result(A_1, S_0) \ldots))\ .$$

Again consider two subcases.

*Subcase 1 :* $Noninertial(|F|, A, S) \in X_\alpha$.

It is clear at this point that the rule

$$Holds(F, Result(A, S)) \leftarrow$$

in program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$ must correspond to an effect rule

$$Holds(F, Result(A, S)) \leftarrow Holds(G_1, S), \ldots, Holds(G_n, S)$$

in program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$ such that all of $Holds(G_1, S), \ldots, Holds(G_n, S)$ belong to $\bigcup_{\nu \leq \alpha} X_\nu$. It follows by the definitions of $X$ and $\pi$ that there is an effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Thus by the definition of $\Phi$ we know that the value proposition $F$ **after** $A_1; \ldots; A_k; A$ is true in $M$, and by the definition of $X$, the literal $Holds(F, Result(A, S))$ belongs to $X_{\alpha+1}$.

*Subcase 2 :* $Noninertial(|F|, A, S) \notin X_\alpha$.

By the definition of $X$, we can conclude that there is no effect proposition $A$ **causes** $F$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$. Nor is there an effect proposition $A$ **causes** $\overline{F}$ **if** $G_1, \ldots, G_n$ in $D$ such that for all $i \in \{1, \ldots, n\}$, the value proposition $G_i$ **after** $A_1; \ldots; A_k$ is true in $M$, where $\overline{F}$ denotes the fluent expression complementary to $F$. We can conclude that the rule

$$Holds(F, Result(A, S)) \leftarrow$$

in program $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \leq \alpha} X_\nu)$ must correspond to the inertia rule

$$Holds(F, Result(A, S)) \leftarrow Holds(F, S), \mathit{not}\ Noninertial(|F|, A, S)$$

in program $b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P)$. It follows by the definition of $e_{U_\alpha}$ that the literal $Holds(F, S)$ belongs to $X_\beta$, where $\beta$ is such that $\beta + 1 = \alpha$. Thus, by the definition of $X$, the value proposition $F$ **after** $A_1; \ldots; A_k$ is true in $M$. By the definition of $\Phi$, since there are no applicable effect propositions whose preconditions are all true in $M^{A_1; \ldots; A_k}$, we can conclude that $F$ **after** $A_1; \ldots; A_k; A$ is true in $M$, from which it follows by the definition of $X$ that the literal $Holds(F, Result(A, S))$ belongs to $X_{\alpha+1}$.

□

**Lemma 27** *Let $P, Q$ be programs. If $Q$ is positive, then $Cn(Q) \subset Cn(P \cup Q)$.*

**Proof.** If $P \cup Q$ is inconsistent, we're done; so assume that $B$ is a consistent answer set for $P \cup Q$. Since $B$ is an answer set for $P \cup Q$, we know that $B$ is closed under $(P \cup Q)^B = P^B \cup Q$, from which it follows that $B$ is closed under $Q$. Let $A$ be the unique answer set for $Q$. Since $A$ is the least set closed under $Q$, we have $A \subset B$. □

**Lemma 28** *Let $D$ be a consistent domain description without value propositions. Let $P$ be a consistent cover of $\pi D$. There is a model $M$ of $D$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ is a member of the unique answer set for $P$.*

**Proof**. Given $P$, we define a suitable $M$ as follows. Let $\sigma_0$ be the set of fluent names $F$ such that $\mathit{Holds}(F, S_0) \in Cn(P)$. Let $\Phi$ be the transition function for $D$. Since $D$ is without value propositions, we know that $M = (\sigma_0, \Phi)$ is indeed a model of $D$. We must show that for each sequence of actions $A_1, \ldots, A_k$ $(k > 0)$, and for each fluent expression $F$, $F$ **after** $A_1; \ldots; A_k$ is true in $M$ if and only if $\pi(F$ **after** $A_1; \ldots; A_k)$ is entailed by $P$. We know by Lemma 26 that there is a consistent cover $P'$ of $\pi D$ with this property. We will show that in fact $P = P'$.

We know that for each fluent name $F$, $\pi D$ includes a disjunctive rule

$$\mathit{Holds}(F, S_0) \mid \neg \mathit{Holds}(F, S_0) \leftarrow \ .$$

Since $D$ is without value propositions, there are no other disjunctive rules in $\pi D$, and no nondisjunctive rule in $\pi D$ has a head containing a literal of the form $\mathit{Holds}(F, S_0)$, where $F$ is a fluent expression. Let $Q$ be the program containing exactly those rules from $P$ of the form $\mathit{Holds}(F, S_0) \leftarrow$ , where $F$ is a fluent expression. Note that $Q \subset P$, and that $P \setminus Q$ is the set of nondisjunctive rules in $\pi D$. Program $Q$ is positive, so by the previous lemma we know that $Cn(Q) \subset Cn(P)$. And since $P$ is consistent, it follows that a literal of the form $\mathit{Holds}(F, S_0)$, where $F$ is a fluent expression, belongs to $Cn(P)$ if and only if the rule $\mathit{Holds}(F, S_0) \leftarrow$ belongs to program $Q$. Thus we can conclude by the definition of $M$ that for each fluent expression $F$, the program $Q$ includes the rule $\mathit{Holds}(F, S_0) \leftarrow$ if and only if the atomic value proposition **initially** $F$ is true in $M$. By the definition of program $P'$ in Lemma 26, we know that for each fluent expression $F$, $P'$ includes the rule $\mathit{Holds}(F, S_0) \leftarrow$ if and only if the atomic value proposition **initially** $F$ is true in $M$. Thus $Q \subset P'$, and $P' \setminus Q$ is the set of nondisjunctive rules in $\pi D$, from which we can conclude that $P' = P$.

□

**Lemma 29** *Let $P$ be a program with a consistent answer set $A$. Let $B$ be a subset of $A$, and let $Q$ be the program $\{L \leftarrow : L \in B\}$. $A$ is an answer set for program $P \cup Q$.*

**Proof**. We know that $(P \cup Q)^A = P^A \cup Q$. Of course $A$ is closed under $Q$, and under $P^A$, so $A$ is closed under $P^A \cup Q$. Suppose there is a set $C$ such that $A \not\subset C$ and $C$ is closed under $P^A \cup Q$. Then $C$ is also closed under $P^A$, contradicting the fact that $A$ is an answer set for $P^A$. So $A$ is a minimal set closed under $P^A \cup Q = (P \cup Q)^A$. Thus $A$ is an answer set for $P \cup Q$. □

**Lemma 30** *Let $D$ be a consistent domain description, and let $M$ be a model of $D$. There is a consistent cover $P$ of $\pi D$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ is entailed by $P$.*

**Proof**. Let $D_e$ be the domain consisting of only the effect propositions of $D$, and let $\pi_v D$ be the program consisting of the translations of only the value propositions of $D$. Of course $\pi D_e \cup \pi_v D = \pi D$. Observe that $M$ is a model of $D_e$, so $D_e$ is consistent. By Lemma 26, there is a consistent cover $P'$ of $\pi D_e$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ is a member of the unique answer set for program $P'$.

Since $M$ is a model of domain $D$, every value proposition in $D$ has at least one atomic part $V$ that is true in $M$. Let $Q$ be a cover of $\pi_v D$ constructed as follows: from each value proposition in $D$ select one atomic part $V$ that is true in $M$ and add to program $Q$ the rule $\pi V \leftarrow$ . For each such rule we know already that the literal $\pi V$ belongs to the unique answer set for program $P'$, so by the preceding lemma we can conclude that the unique answer set for $P'$ is also an answer set for program $P' \cup Q$. So program $P' \cup Q$ is consistent.

If $P' \cap Q$ is empty, it's clear that $P$ is a cover for program $\pi D$. On the other hand, any rule in $\pi D_e \cap \pi_v D$ must have the form $\mathit{Holds}(F, S_0) \mid \neg \mathit{Holds}(F, S_0) \leftarrow$ , where $F$ is a fluent name. And since program $P' \cup Q$ is consistent, we can conclude that either: (i) the rule $\mathit{Holds}(F, S_0) \leftarrow$ belongs to both $P'$ and $Q$; or (ii) $\neg \mathit{Holds}(F, S_0) \leftarrow$ belongs to both $P'$ and $Q$. It follows that program $P' \cup Q$ is a cover for program $\pi D$.

Let $P = P' \cup Q$. By monotonicity (Theorem 2), $Cn(P') \cap \overline{S} \subset Cn(P) \cap \overline{S}$. Furthermore, for every literal $L \in \overline{S}$, either $L \in Cn(P') \cap \overline{S}$ or $\overline{L} \in Cn(P') \cap \overline{S}$. It follows that $Cn(P) \cap \overline{S} = Cn(P') \cap \overline{S}$, and thus $P$ is a consistent cover of $\pi D$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ is entailed by $P$.

□

**Lemma 31** *Let $D$ be a consistent domain description, and let $P$ be a consistent cover of $\pi D$. There is a model $M$ of $D$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V$ is entailed by $P$.*

**Proof.** Let $D_e$ be the domain consisting of only the effect propositions of $D$, and let $\pi_v D$ be the program consisting of the translations of only the value propositions of $D$. Since $\pi D = \pi D_e \cup \pi_v D$, it is clear that program $P$ can be represented in the form $P' \cup Q$, where program $P'$ is a cover of program $\pi D_e$ and program $Q$ is a cover of $\pi_v D$. Let $S$ be the set of all *Noninertial* literals. It's clear that $h_S(P) = h_S(P')$ and $h_{\overline{S}}(P') \subset h_{\overline{S}}(P)$, so we can conclude by monotonicity (Theorem 4) that $Cn(P') \cap \overline{S} \subset Cn(P) \cap \overline{S}$. Since $P$ is consistent, we can conclude by Proposition 2 that program $P'$ is consistent. By Lemma 28, there is a model $M$ of $D_e$ such that for every atomic value proposition $V$, $V$ is true in $M$ if and only if $\pi V \in Cn(P') \cap \overline{S}$. Thus, for every *Holds* literal $L$, either $L \in Cn(P') \cap \overline{S}$ or $\overline{L} \in Cn(P') \cap \overline{S}$. Again since $P$ is a consistent program, we can conclude that $Cn(P') \cap \overline{S} = Cn(P) \cap \overline{S}$, and it follows that an atomic value proposition $V$ is true in $M$ if and only if $\pi V \in Cn(P) \cap \overline{S}$.

We need to show that every value proposition in $D$ is true in $M$. So let $V$ be a value proposition in $D$, with $V = V_1$ **or** $\ldots$ **or** $V_k$, where each of $V_1, \ldots, V_k$ is an atomic value proposition. By definition, $V$ is true in $M$ if at least one of $V_1, \ldots, V_k$ is true in $M$. We know that the rule

$$\pi V_1 \mid \ldots \mid \pi V_k \leftarrow$$

belongs to $\pi_v D$. Since $Q$ is a cover of $\pi_v D$, we know there is an $i$ $(1 \leq i \leq k)$ such that the rule

$$\pi V_i \leftarrow$$

belongs to program $Q$. Thus we have $\pi V_i \in Cn(Q)$. By Lemma 27 we know that $Cn(Q) \subset Cn(P)$. It follows that $V_i$ is true in $M$, so $V$ is also.

□

**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** $V_k$, domain $D$ entails $V$ if and only if each consistent cover of $\pi D$ entails at least one of $\pi V_1, \ldots, \pi V_k$.*

**Proof.** Let $V$ be a value proposition and let $V_1, \ldots, V_k$ ($k \geq 1$) be atomic value propositions such that $V = V_1$ **or** $\ldots$ **or** $V_k$.

(Left-to-right) : Assume that $D$ entails $V$. We must show that each consistent cover of $\pi D$ entails at least one of $\pi V_1, \ldots, \pi V_k$. Assume that $P$ is a consistent cover of $\pi D$. By Lemma 31 there is a model $M$ of $D$ such that for every atomic value proposition $V'$ that is true in $M$, $P$ entails $\pi V'$. Since $D$ entails $V$, we know by the definition of entailment in $\mathcal{A}_d$ that at least one of $V_1, \ldots, V_k$ is true in $M$. Therefore $P$ entails at least one of $\pi V_1, \ldots, \pi V_k$.

(Right-to-left) : Assume that each consistent cover of $\pi D$ entails at least one of $\pi V_1, \ldots, \pi V_k$. We must show that $D$ entails $V$. That is, we must show that at least one of $V_1, \ldots, V_k$ is true in each model of $D$. Let $M$ be a model of $D$. By Lemma 30 there is a consistent cover $P$ of $\pi D$ such that for every atomic value proposition $V'$, $V'$ is true in $M$ whenever $P$ entails $\pi V'$. Since $P$ entails at least one of $\pi V_1, \ldots, \pi V_k$, at least one of $V_1, \ldots, V_k$ is true in $M$.

□

**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 $\pi D$ entails $\pi V$.*

**Proof.** The set $S$ of all *Noninertial* literals is a signing for $\pi D$, and we know that for every atomic value proposition $V$, program $\pi D$ entails $\pi V$ if and only if $\pi V \in Cn(\pi D) \cap \overline{S}$. By Theorem 2,

$$Cn(\pi D) \cap \overline{S} = \bigcap_{P \in covers(\pi D)} Cn(P) \cap \overline{S} \, .$$

Since domain $D$ is consistent, we know by Lemma 30 that program $\pi D$ has a consistent cover, so

$$\bigcap_{P \in covers(\pi D)} Cn(P) \cap \overline{S} = \bigcap_{P \in good\text{-}covers(\pi D)} Cn(P) \cap \overline{S} \, .$$

Again by the definition of $S$, we know that for every consistent cover $P$ of $\pi D$, and for every atomic value proposition $V$, $\pi V \in Cn(P) \cap \overline{S}$ if and only if $P$ entails $\pi V$. So we've shown that for every atomic value proposition $V$, $\pi D$ entails $\pi V$ if and only if every consistent cover of $\pi D$ entails $\pi V$. The theorem now follows immediately from Theorem 7.

□