Splitting a Logic Program

Vladimir Lifschitz
Department of Computer Sciences and Department of Philosophy
University of Texas at Austin
Austin, TX 78712
vl@cs.utexas.edu

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

Abstract

In many cases, a logic program can be divided into two parts, so that one of them, the "bottom" part, does not refer to the predicates defined in the "top" part. The "bottom" rules can be used then for the evaluation of the predicates that they define, and the computed values can be used to simplify the "top" definitions. We discuss this idea of splitting a program in the context of the answer set semantics. The main theorem shows how computing the answer sets for a program can be simplified when the program is split into parts. The programs covered by the theorem may use both negation as failure and classical negation, and their rules may have disjunctive heads. The usefulness of the concept of splitting for the investigation of answer sets is illustrated by several applications. First, we show that a conservative extension theorem by Gelfond and Przymusinska and a theorem on the closed world assumption by Gelfond and Lifschitz are easy consequences of the splitting theorem. Second, (locally) stratified programs are shown to have a simple characterization in terms of splitting. The existence and uniqueness of an answer set for such a program can be easily derived from this characterization. Third, we relate the idea of splitting to the notion of order-consistency.

1 Introduction

In many cases, a logic program can be divided into two parts, so that one of them, the "bottom" part, does not refer to the predicates defined in the "top" part. The "bottom" rules can be used then for the evaluation of the predicates that they define, and the computed values can be used to simplify the "top" definitions.

This idea of *splitting a logic program into parts* has a rather long history. Although it is applicable even to positive programs, it turned out to be

particularly useful when negation as failure is involved. The best known application of splitting is found in the notion of a stratification [Apt *et al.*, 1988]. In a stratified program P, the first stratum is a bottom part that does not contain negation as failure. Having substituted the values of the bottom predicates in the bodies of the remaining rules, we reduce P to a program with fewer strata. By applying the splitting step several times, and computing every time the "minimal model" of a positive bottom, we will arrive at the "intended model" of P. In fact, this step-by-step reduction to a series of positive programs is sometimes applicable even when P is not stratified. This observation leads to the notion of a weakly stratified program [Przymusinska and Przymusinski, 1988].

The notion of a stratification has been also extended in two other directions. First, it may be possible to split a program into an infinite—or even transfinite—sequence of parts, instead of a finite number. This is what happens in locally stratified programs [Przymusinski, 1988]. (As observed above, splitting a program into finitely many parts can be achieved by repeatedly splitting into two; introducing infinite splittings is a nontrivial generalization.) Second, the bottom does not need to be a positive program. This idea has led Schlipf [1992] to the definition of a "stratified pair," and Dix [1992] to the definitions of "relevance" and "modularity."

In this paper, we discuss splitting in the context of the "answer set semantics" of [Gelfond and Lifschitz, 1991]. The main theorem shows how computing the answer sets for a program can be simplified when the program is split into parts. The programs covered by the theorem may use both negation as failure and classical negation, and their rules may have disjunctive heads.

The usefulness of the concept of splitting for the investigation of answer sets is illustrated by several applications. First, we generalize a conservative extension theorem from [Gelfond and Przymusinska, 1991] and the theorem on the closed world assumption from [Gelfond and Lifschitz, 1991], and show that these generalizations can be easily proved as consequences of the splitting theorem. Second, (locally) stratified programs are shown to have a simple characterization in terms of splitting. This characterization leads to a new proof of the existence and uniqueness of an answer set for a stratified program. Third, we relate the idea of splitting to the syntactic property of programs called "order-consistency"; that property is important in view of the fact that it implies the existence of at least one answer set [Fages, 1994]. Order-consistent programs can be characterized in terms of splitting also.

After a brief review of the syntax and semantics of (disjunctive) programs (Section 2), we state the special case of the main theorem that deals with splitting a program into two parts (Section 3) and show how it can be applied to the study of conservative extensions (Section 4) and of the closed world assumption (Section 5). Then the main theorem is stated in full generality (Section 6); it allows us to split a program into a transfinite sequence of parts. The theorem is applied to stratified programs in Section 7 and to

order-consistent programs in Section 8. We conclude with Section 9.

2 Programs

We begin with a brief review of the syntax and semantics of disjunctive logic programs. Consider a nonempty set of symbols called *atoms*. A *literal* is an atom possibly preceded by the classical negation symbol $\neg$. A *rule* is determined by three finite sets of literals—the set of *head literals*, the set of *positive subgoals* and the set of *negated subgoals*. The rule with the head literals $L_1, \ldots, L_l$, the positive subgoals $L_{l+1}, \ldots, L_m$ and the negated subgoals $L_{m+1}, \ldots, L_n$ is written as

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

We will denote the three parts of a rule r by $\mathit{head}(r)$, $\mathit{pos}(r)$ and $\mathit{neg}(r)$; $\mathit{lit}(r)$ stands for $\mathit{head}(r) \cup \mathit{pos}(r) \cup \mathit{neg}(r)$.

A *program* is a set of rules. For any program P, by $\mathit{lit}(P)$ we denote the union of the sets $\mathit{lit}(r)$ for all $r \in P$; the literals in this set are said to *occur* in P.

Note that this description of the syntax of programs is in some ways different from what is usually found in the literature. First, a program is a *set* of rules, rather than a list; similarly, the literals in the head and in the body of a rule are not supposed to be ordered. The order of rules and subgoals is essential for query evaluation, but it is irrelevant as long as we are interested in the declarative semantics of the program. Second, we accept an abstract view of what atoms are and say nothing about their internal structure. 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. Here again, the difference between a "schematic rule" and the set of its "ground instances," fundamental for the procedural view, is irrelevant for the study of declarative properties.

A program P is *positive* if, for every rule $r \in P$, $\mathit{neg}(r) = \emptyset$. The notion of an answer set is first defined for positive programs, as follows. A set X of literals is *closed* under a positive program P if, for every rule $r \in P$ such that $\mathit{pos}(r) \subset X$, $\mathit{head}(r) \cap X \neq \emptyset$. (We write $X \subset Y$ when X is a subset of Y, not necessarily proper.) A set of literals is *logically closed* if it is consistent or contains all literals. An *answer set* for a positive program P is a minimal set of literals that is both closed under P and logically closed.

Now let P be an arbitrary program. Take a set X of literals. For each rule $r \in P$ such that $\mathit{neg}(r) \cap X = \emptyset$, consider the rule r' defined by

$$\mathit{head}(r') = \mathit{head}(r),\ \mathit{pos}(r') = \mathit{pos}(r),\ \mathit{neg}(r') = \emptyset.$$

The positive program consisting of all rules r' obtained in this way is the *reduct* of P relative to X, denoted by P^X. We say that X is an *answer set* for P if X is an answer set for P^X.

For example, $\{a\}$ is an answer set for the program

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

because the reduct of this program relative to $\{a\}$ is $\{\ a \leftarrow\ \}$, and $\{a\}$ is an answer set for this reduct. The only other answer set for this program is $\{b\}$.

A literal L is a *consequence* of a program P if L belongs to all answer sets for P.

For future reference, we will summarize here some simple facts about answer sets.

Fact 1. *If X is a consistent answer set for a program P, then every literal in X belongs to the head of one of the rules of P.*

Fact 2. *If a program P has a consistent answer set, then all answer sets for P are consistent.*

Fact 3. *A literal L is a consequence of a program P if and only if L belongs to all consistent answer sets for P.*

3 Splitting Sets

A *splitting set* for a program P is any set U of literals such that, for every rule $r \in P$, if $\mathit{head}(r) \cap U \neq \emptyset$ then $\mathit{lit}(r) \subset U$. If U is a splitting set for P, we also say that U *splits* P. The set of rules $r \in P$ such that $\mathit{lit}(r) \subset U$ 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. It is clear that the head literals of all rules in $P \setminus b_U(P)$ belong to $\mathit{lit}(P) \setminus U$.

Every program P is split, trivially, by the empty set and by $\mathit{lit}(P)$. For an example of a nontrivial splitting, consider the following program P_1:

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

The set $U = \{c\}$ splits P_1; the last rule of P_1 belongs to the bottom, and the first two rules form the top.

A splitting set for a program P can be used to break the task of computing the answer sets for P into several tasks of the same kind for smaller programs. This process involves the "partial evaluation" of the top of P with respect to each of the answer sets for the bottom of P.

Consider, for instance, the unique answer set for the bottom of P_1, which is $\{c\}$. The "partial evaluation" of the top part of P_1 consists in dropping its

first rule, because the negated subgoal c makes it "useless," and in dropping the "trivial" positive subgoal c in the second rule. The result of simplification is the program consisting of one rule:

$$b \leftarrow not\ \ a. \tag{1}$$

The only answer set for P_1 can be obtained by adding the only answer set for (1), which is $\{b\}$, to the answer set for the bottom used in the evaluation process, $\{c\}$.

To define how this procedure works in general, we need the following notation. Consider two sets of literals U, X and a program P. For each rule $r \in P$ such that $pos(r) \cap U$ is a part of X and $neg(r) \cap U$ is disjoint from X, take the rule r' defined by

$$head(r') = head(r),\ pos(r') = pos(r) \setminus U,\ neg(r') = neg(r) \setminus U.$$

The program consisting of all rules r' obtained in this way will be denoted by $e_U(P, X)$. For example,

$$e_U(P_1 \setminus b_U(P_1), \{c\}) = \{\ b \leftarrow not\ a\ \}.$$

Let U be a splitting set for a program P. A *solution* to P (with respect to U) is a pair $\langle X, Y\rangle$ of sets of literals such that

- X is an answer set for $b_U(P)$,
- Y is an answer set for $e_U(P \setminus b_U(P), X)$,
- $X \cup Y$ is consistent.

For example, $\langle\{c\}, \{b\}\rangle$ is the only solution to P_1 (with respect to $\{c\}$).

Every literal occurring in $b_U(P)$ belongs to $lit(P) \cap U$, and every literal occurring in $e_U(P \setminus b_U(P), X)$ belongs to $lit(P) \setminus U$. In view of Fact 1 (Section 2), it follows that, for any solution $\langle X, Y\rangle$ to P,

$$X \subset lit(P) \cap U,$$
$$Y \subset lit(P) \setminus U,$$

and consequently $X \cap Y = \emptyset$.

Splitting Set Theorem. *Let U be a splitting set for a program P. A set A of literals is a consistent answer set for P if and only if $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to P with respect to U.*

In Section 6, this theorem is extended to sequences of splitting sets.

In view of Fact 2 (Section 2), we conclude:

Corollary 1. *Let U be a splitting set for a program P, such that there exists at least one solution to P with respect to U. Program P is consistent, and a set A of literals is an answer set for P if and only if $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to P with respect to U.*

As another example, take the following program P_2:

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

Let $U = \{a, b\}$. The bottom consists of the last two rules and has two answer sets, $\{a\}$ and $\{b\}$. Program $e_U(P_2 \setminus b_U(P_2), \{a\})$ consists of one rule, $c \leftarrow$, and has one answer set, $\{c\}$. Thus the first solution to P_2 is $\langle\{a\}, \{c\}\rangle$. Similarly, the second solution is $\langle\{b\}, \{c\}\rangle$. By Corollary 1, the answer sets for P_2 are $\{a, c\}$ and $\{b, c\}$.

In view of Fact 3 (Section 2), the Splitting Set Theorem implies:

Corollary 2. *Let U be a splitting set for a program P. A literal L is a consequence of P if and only if, for every solution $\langle X, Y\rangle$ to P with respect to U, $L \in X \cup Y$.*

The next example illustrates the role of the consistency condition in the definition of a solution. Let P_3 be the program

$$\begin{aligned} \neg b &\leftarrow\ , \\ a \mid b &\leftarrow\ . \end{aligned}$$

The only solution to P_3 with respect to $\{a, b\}$ is $\langle\{a\}, \{\neg b\}\rangle$. The pair $\langle\{b\}, \{\neg b\}\rangle$ is not a solution, because the set $\{b, \neg b\}$ is inconsistent. This set is not an answer set for P_3.

The proof of the Splitting Set Theorem is based on the following observations. The statement of the theorem can be reformulated as follows: If U is a splitting set for P, then a consistent set X of literals is an answer set for P^X if and only if

- $X \cap U$ is an answer set for $b_U(P)^{X \cap U}$,
- $X \setminus U$ is an answer set for $e_U(P \setminus b_U(P), X \cap U)^{X \setminus U}$.

Since these reducts have no common literals, the last two conditions can be combined into one: X is an answer set for

$$b_U(P)^{X \cap U} \cup e_U(P \setminus b_U(P), X \cap U)^{X \setminus U}. \tag{2}$$

On the other hand, it is easy to see that P^X is the same as

$$b_U(P)^{X \cap U} \cup (P \setminus b_U(P))^X. \tag{3}$$

In the proof, we verify that X is an answer set for (2) if and only if it is an answer set for (3).

4 Application: Conservative Extensions

If we extend a program P by rules whose heads do not occur in P, then, typically, we do not expect to see any new consequences of the program among the literals occurring in P. This conservative extension property (called "the weak principle of stratification" by Schlipf [1992] and "relevance" by Dix [1992]) is not valid without additional restrictions, however. For instance, after we extend a program by the contradictory rules $a \leftarrow$ and $\neg a \leftarrow$, every literal in the language will become its consequence.

One case when the conservative extension property does hold is described in [Gelfond and Przymusinska, 1991], Proposition 2.1. In this section, we state a slightly more general fact and prove it as a corollary to the Splitting Set Theorem.

A program is *nondisjunctive* if the head of each of its rules is a singleton.

Proposition 1. *Let P be a program, and let C be a consistent set of literals that do not occur in P and whose complements also do not occur in P. Let Q be a nondisjunctive program such that, for every rule $r \in Q$, $head(r) \subset C$ and $neg(r) \subset lit(P)$. For any literal $L \notin C$, L is a consequence of $P \cup Q$ if and only if L is a consequence of P.*

For instance, after adding the first two rules of program P_2 to its last two rules, a and b cannot turn into its consequences (take $C = \{c\}$).

The theorem by Gelfond and Przymusinska mentioned above is the special case when, additionally, P is assumed to be nondisjunctive, neither P nor Q uses classical negation, and $pos(r) \subset lit(P)$ for every $r \in Q$.

Proof of Proposition 1. Let $U = lit(P)$. From the fact that no literal in C occurs in P, we conclude that U splits $P \cup Q$, with the bottom P and the top Q. Take any consistent answer set X for P. The program $e_U(Q, X)$ is nondisjunctive and positive, and the heads of all its rules are contained in C. Since C is consistent, it follows that this program has a unique answer set Y, and $Y \subset C$. Furthermore, since no literal in C has its complement in $lit(P)$, and since $X \subset lit(P)$, the set $X \cup Y$ is consistent. Thus, $\langle X, Y \rangle$ is a solution to $P \cup Q$ with respect to U. Consequently, for every consistent answer set X for P there exists a set $Y \subset C$ such that $\langle X, Y \rangle$ is a solution to $P \cup Q$; moreover, if $\langle X, Y \rangle$ is a solution to $P \cup Q$, then $Y \subset C$. By Corollary 2 to the Splitting Set Theorem, it follows that a literal $L \notin C$ is a consequence of $P \cup Q$ if and only if it is a consequence of P.

5 Application: Closed World Assumption

The *closed world assumption rule* for a literal L is the rule

$$L \leftarrow not\ \overline{L},$$

where $\overline{L}$ stands for the literal complementary to L. Rules of this kind play an important part in knowledge representation ([Gelfond and Lifschitz, 1991], Section 3).

The following theorem describes the effect of adding a set of closed world assumption rules to a program:

Proposition 2. *Let P be a program, let C be a consistent set of literals that do not occur in P, and let P' be the program obtained from P by adding the closed world assumption rules for all literals in C. If X is a consistent answer set for P, then*

$$X \cup \{L \in C : \overline{L} \notin X\} \tag{4}$$

is a consistent answer set for P'. Moreover, every consistent answer set for P' can be represented in form (4) for some consistent answer set X for P.

This theorem can be illustrated by the following example. Let the set of atoms be $\{p(1), p(2), q(1), q(2)\}$, let P_4 be the program

$$\begin{array}{rcl} p(1) & \leftarrow & , \\ \neg q(2) & \leftarrow & , \end{array}$$

and let P_4' be obtained from P_4 by adding the closed world assumption rules

$$\begin{array}{rcl} \neg p(x) & \leftarrow & not\ p(x), \\ q(x) & \leftarrow & not\ \neg q(x), \end{array}$$

($x \in \{1, 2\}$). Since the only answer set for P_4 is $\{p(1), \neg q(2)\}$, Proposition 2 shows that the only answer set for P_4' is

$$\{p(1), \neg q(2)\} \cup \{\neg p(2), q(1)\}.$$

Proposition 4 from [Gelfond and Lifschitz, 1991] is the special case of Proposition 2 in which P is a nondisjunctive program without classical negation, and C is the set of all negative literals.

Proof of Proposition 2. Let $U = lit(P)$. From the fact that no literal in C occurs in P we conclude that U splits P' and $b_U(P') = P$. Take any consistent answer set X for P. The program $e_U(P' \setminus b_U(P'), X)$ consists of the rules $L \leftarrow$ for all literals $L \in C$ such that $\overline{L} \notin X$. Obviously, the only answer set Y for this program is $\{L \in C : \overline{L} \notin X\}$. Since X and C are consistent, $X \cup Y$ is consistent also. It follows that the solutions to P' are the pairs $\langle X, \{L \in C : \overline{L} \notin X\}\rangle$, where X is an answer set for P. Now the assertion of Proposition 2 follows from the Splitting Set Theorem.

6 Splitting Sequences

A *(transfinite) sequence* is a family whose index set is an initial segment of ordinals, $\{\alpha : \alpha < \mu\}$. The ordinal μ 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 $\bigcup_{\alpha<\mu} U_\alpha = lit(P)$.

For instance, if U_0 is a splitting set for P, then $\langle U_0, lit(P)\rangle$ is a splitting sequence for P of length 2. Consider the well-known "even number" program P_5:

$$\begin{array}{rcl} p(0) & \leftarrow & , \\ p(S(x)) & \leftarrow & not\ \ p(x) \end{array}$$

$(x = 0, S(0), S(S(0)), \ldots)$. The following sequence of length ω is a splitting sequence for P_5:

$$\langle \{p(0)\}, \{p(0), p(S(0))\}, \{p(0), p(S(0)), p(S(S(0)))\}, \ldots \rangle. \tag{5}$$

The definition of a solution with respect to a splitting set is extended to splitting sequences as follows. 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 α 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.

It is easy to see that the solutions to P with respect to a splitting sequence $\langle U_0, lit(P)\rangle$ are the same as the solutions to P with respect to the splitting set U_0. The only solution $\langle X_0, X_1, \ldots\rangle$ to P_5 with respect to (5) is defined by the equations:

$$X_n = \begin{cases} \{p(S^n(0))\}, & \text{if } n \text{ is even,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

This is easy to check by induction on n.

Let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a program P, and let $\langle X_\alpha \rangle_{\alpha<\mu}$ be a sequence of sets of literals. Every literal occurring in $b_{U_0}(P)$ belongs to $lit(P) \cap U_0$, and every literal occurring in

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

$(\alpha + 1 < \mu)$ belongs to $lit(P) \cap (U_{\alpha+1} \setminus U_\alpha)$. In view of Fact 1 (Section 2), it follows that, if $\langle X_\alpha \rangle_{\alpha<\mu}$ is a solution, then

$$X_0 \subset lit(P) \cap U_0,$$
$$X_{\alpha+1} \subset lit(P) \cap (U_{\alpha+1} \setminus U_\alpha).$$

It follows that the members of any solution are pairwise disjoint.

The following propositions generalize the Splitting Set Theorem and its corollaries.

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.*

Corollary 1. *Let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a program P, such that there exists at least one solution to P with respect to U. Program P is consistent, and a set A of literals is an 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.*

Corollary 2. *Let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a program P. A literal L is a consequence of P if and only if, for every solution $\langle X_\alpha \rangle_{\alpha<\mu}$ to P with respect to U, $L \in \bigcup_{\alpha<\mu} X_\alpha$.*

By Corollary 1, it follows that the only answer set for P_5 is

$$\{p(S^n(0)) \,:\, n \text{ is even}\}.$$

The proof of the Splitting Sequence Theorem is based on the Splitting Set Theorem.

7 Components

Let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a program P, and let $\langle X_\alpha \rangle_{\alpha<\mu}$ be a sequence of sets of literals. Some applications of the Splitting Sequence Theorem depend on the syntactic form of the programs whose answer sets can be members of a solution:

$$\begin{array}{l} b_{U_0}(P), \\ e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), \bigcup_{\nu \le \alpha} X_\nu) \qquad (\alpha + 1 < \mu). \end{array} \tag{6}$$

It is clear that each rule of each of these programs is obtained from a rule of P by removing some of its subgoals. A more specific claim regarding the structure of programs (6) can be made, using the following terminology.

For any program P and any set X of literals, let $rm(P, X)$ be the part of P obtained by removing all subgoals that belong to X, both positive and negated, from each of the rules of P. For any program P and any splitting sequence $U = \langle U_\alpha \rangle_{\alpha<\mu}$ for P, the programs

$$\begin{array}{l} b_{U_0}(P), \\ rm(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), U_\alpha) \qquad (\alpha + 1 < \mu) \end{array}$$

will be called the *U-components* of P.

For example, the U-components of P_5 are the programs $\{\ p(S^n(0)) \leftarrow\ \}$ for all n.

It is easy to see that for any set X of literals, $e_{U_\alpha}(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), X)$ is a subset of $rm(b_{U_{\alpha+1}}(P) \setminus b_{U_\alpha}(P), U_\alpha)$. Consequently, each program in (6) is a subset of a U-component of P.

To demonstrate the usefulness of the notion of a U-component, we will show now that it leads to a simple characterization of the class of stratified programs.

A *level mapping* is a function from literals to ordinals. A program P is *stratified* if there exists a level mapping f such that, for every rule $r \in P$ and any literals L_1, L_2,

- if $L_1, L_2 \in head(r)$ then $f(L_1) = f(L_2)$,
- if $L_1 \in head(r)$ and $L_2 \in pos(r)$ then $f(L_1) \geq f(L_2)$,
- if $L_1 \in head(r)$ and $L_2 \in neg(r)$ then $f(L_1) > f(L_2)$.

For instance, every positive program is stratified: take $f(L) = 0$ for every literal L. Program P_5 is a stratified program: take $f(p(S^n(0))) = n$. Programs P_1 and P_2 are not stratified.

The definition given above is equivalent to the usual definition of a "locally stratified" program [Przymusinski, 1988] when the set of atoms is defined as the set of ground atoms of a first-order language, and there is no classical negation. (A "nonlocal stratification" does not make sense in the context of the abstract view of atoms accepted here.)

A rule r is a *constraint* if $head(r) = \emptyset$. Clearly, if a program is stratified, this property will not be affected by adding or deleting constraints.

Proposition 3. *A program P that does not contain constraints is stratified if and only if it has a splitting sequence U such that all U-components of P are positive.*

Proof. Assume that P is stratified, and let f be the corresponding level mapping. Take μ to be the smallest ordinal that is greater than all values of f, and define, for every $\alpha < \mu$,

$$U_\alpha = \{L\ :\ f(L) < \alpha\}.$$

It is easy to check that $U = \langle U_\alpha \rangle_{\alpha<\mu}$ is a splitting sequence for P, and that all U-components of P are positive. Conversely, if $U = \langle U_\alpha \rangle_{\alpha<\mu}$ is a splitting sequence for P such that all U-components of P are positive, then we can define $f(L)$ as the smallest α such that $L \in U_\alpha$.

Proposition 3 leads to a new proof of a familiar property of stratified nondisjunctive programs without classical negation ([Gelfond and Lifschitz, 1988], Corollary 1):

Proposition 4. *Every stratified, nondisjunctive program without classical negation has a unique answer set.*

Proof. Let P be a stratified, nondisjunctive program without classical negation, and let $U = \langle U_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for P such that all U-components of P are positive. Then, for any sequence $\langle X_\alpha \rangle_{\alpha<\mu}$ of sets of literals, every program in (6) is a positive nondisjunctive program without classical negation. Consequently, each of these programs has a unique answer set. It follows that the definition of a solution can be reformulated in this case as follows: $\langle X_\alpha \rangle_{\alpha<\mu}$ is a solution to P if

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

The first three conditions provide a recursive definition of $\langle X_\alpha \rangle_{\alpha<\mu}$. Consequently, there is exactly one sequence satisfying these conditions. Every element of every member of this sequence is an atom, so that the last condition is satisfied also.

8 Splitting and Order-Consistency

The notions of a "signed" program (program with a signing) [Kunen, 1989] and an "order-consistent" program [Sato, 1990] can be defined as follows. In the definitions, P is assumed to be a nondisjunctive program without classical negation.

We say that P is *signed* if there exists a set S of atoms such that, for every rule r in P,
$$head(r) \cup pos(r) \subset S,\ neg(r) \cap S = \emptyset$$
or
$$(head(r) \cup pos(r)) \cap S = \emptyset,\ neg(r) \subset S.$$

To define the much wider class of order-consistent programs, we need the following notation. For any atom A, P_A^+ and P_A^- are the smallest sets of atoms such that $A \in P_A^+$ and, for every rule $r \in P$,

- if $head(r) \subset P_A^+$ then $pos(r) \subset P_A^+$ and $neg(r) \subset P_A^-$,
- if $head(r) \subset P_A^-$ then $pos(r) \subset P_A^-$ and $neg(r) \subset P_A^+$.

Program P is called *order-consistent* if there exists a level mapping f such that $f(B) < f(A)$ whenever $B \in P_A^+ \cap P_A^-$.

The following theorem describes the relationship between these classes of programs:

Proposition 5. *Let P be a nondisjunctive program without classical negation. Program P is order-consistent if and only if it has a splitting sequence U such that all U-components of P are signed.*

For instance, the following program P_6 is order-consistent, but not signed:

$$\begin{array}{lcl} a & \leftarrow & b, \\ a & \leftarrow & not\ b. \end{array}$$

Let U be the sequence $\langle \{b\}, \{a, b\} \rangle$. This sequence splits P_6, and the U-components of P_6 are the signed programs $\{\ a \leftarrow\ \}$ and $\emptyset$.

Using Proposition 5 and the Splitting Sequence Theorem, we can derive Fages's theorem [Fages, 1994] on the existence of an answer set for an order-consistent program from a similar—and easier—theorem for signed programs.

Proof of Proposition 5 (sketch). Let P be a program with a splitting sequence $U = \langle U_\alpha \rangle_{\alpha < \mu}$ such that all U-components of P are signed. A level mapping f required in the definition of order-consistency can be defined as follows: For each atom A, $f(A)$ is the least ordinal α such that $A \in U_\alpha$. Assume, on the other hand, that P is order-consistent, and let f be the corresponding level mapping. Arrange all atoms in a transfinite sequence $\langle A_\alpha \rangle_{\alpha < \mu}$ so that $f(A_\alpha) < f(A_\beta)$ whenever $\alpha < \beta$. A splitting sequence for P can be defined by $U_\alpha = \bigcup_{\beta < \alpha} (P^+_{A_\beta} \cup P^-_{A_\beta})$. For this sequence U, all U-components of P are signed.

9 Conclusion

The usefulness of splitting is illustrated in this paper by several applications. The Splitting Set Theorem is also employed in the paper "Language Independence and Language Tolerance in Logic Programs" [McCain and Turner, 1993], which appears in this volume. It is used there to prove one of the central results—Theorem 6.1, which shows that, under some conditions, one can ignore the fact that the language of a logic program is many-sorted. We expect that, in the future, the idea of splitting will find many other uses.

Acknowledgements

The authors would like to thank Norman McCain for useful discussions on the subject of this paper. We are also grateful to Enrico Giunchiglia and G. N. Kartha for their comments. This work was partially supported by National Science Foundation under grants IRI-9101078 and IRI-9306751.

References

[Apt *et al.*, 1988] Krzysztof Apt, Howard Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, San Mateo, CA, 1988.

[Dix, 1992] Jürgen Dix. Classifying semantics of disjunctive logic programs (extended abstract). In Krzysztof Apt, editor, *Proc. Joint Int'l Conf. and Symp. on Logic Programming*, pages 798–812, 1992.

[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, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.

[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 Przymusinska, 1991] Michael Gelfond and Halina Przymusinska. Definitions in epistemic specifications. In Anil Nerode, Wiktor Marek, and V. S. Subramanian, editors, *Logic Programming and Non-monotonic Reasoning: Proceedings of the First International Workshop*, pages 245–259, 1991.

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

[McCain and Turner, 1993] Norman McCain and Hudson Turner. Language independence and language tolerance in logic programs. Submitted for publication, 1993.

[Przymusinska and Przymusinski, 1988] Halina Przymusinska and Teodor Przymusinski. Weakly perfect model semantics for logic programs. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1106–1120, 1988.

[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.

[Sato, 1990] Taisuke Sato. Completed logic programs and their consistency. *Journal of Logic Programming*, 9:33–44, 1990.

[Schlipf, 1992] John Schlipf. Formalizing a logic for logic programming. *Annals of Mathematics and Artificial Intelligence*, 5:279–302, 1992.