Theoretical Computer Science 86 (1991) 93-106
Elsevier

Semantics of Horn and disjunctive logic programs

Jorge Lobo and Arcot Rajasekar

Department of Computer Science, University of Maryland, College Park, MD 20742, USA

Jack Minker

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA

Abstract

Lobo, J., A. Rajasekar and J. Minker, Semantics of Horn and disjunctive logic programs, Theoretical Computer Science 86 (1991) 93-106.

Van Emden and Kowalski proposed a fixpoint semantics based on model-theory and an operational semantics based on proof-theory for Horn logic programs. They prove the equivalence of these semantics using fixpoint techniques. The main goal of this paper is to present a unified theory for the semantics of Horn and disjunctive logic programs. For this, we extend the fixpoint semantics and the operational or procedural semantics to the class of disjunctive logic programs and prove their equivalence using techniques similar to the ones used for Horn programs.

1. Introduction

The main goal of this paper is to present a unified theory for the semantics of Horn and disjunctive logic programs. We present a declarative and a procedural semantics that embed the semantics of Horn programs as presented in [6] and [2]. In [6, 2], two approaches to the semantics of Horn programs were studied. A fixpoint semantics based on the model theory of first-order logic and an operational semantics based on proof-theory form the core of these papers. Proof of equivalence between model-theory and proof-theory using fixpoint techniques instead of Gödel's Completeness Theorem is among the important contributions presented in [6] and [2]. In this paper we extend the fixpoint semantics and the operational or procedural semantics to a broader class of logic programs which include disjunctive logic programs. We prove the equivalence of the two semantics using techniques similar to the ones used in [6, 2].

Fixpoint semantics is based on operators that transform elements of a given lattice to elements in the same lattice. Van Emden and Kowalski [6] define an operator that applies to a lattice formed by sets of atoms using set inclusion as partial order and maps a set of atoms to a set of atoms. In our paper we use sets of positive clauses instead of atoms to apply the concepts in [6] to the extended theory. The results on fixpoint semantics are taken from [15]. Procedural semantics in logic programming uses implementation-independent proof procedures and describes the semantics of the programs as the theorems provable through the given procedures. SLD-resolution (SL-resolution for Definite clauses) [8, 6] is used as a basis for the procedural semantics of Horn theories. One of the underlying characteristics of SLD-resolution is that it has a very simple operational interpretation. Horn programs are formed of clauses that consist of two parts, an antecedent that consists of a conjunction of atoms and a consequent that consists of an atomic formula. SLD-resolution considers the consequent of a clause to be a problem that can be solved by reducing it to the subproblems given in the antecedent. In this paper, we present a procedural semantics, SLO-resolution, for the extended class of programs that keeps the problem–subproblem operational flavor of SLD-resolution. The paper is organized as follows. In the remainder of this section we present some preliminary definitions about logic programming and fixpoint theory. Section 2 contains the fixpoint semantics for disjunctive programs. In Section 3 we present the procedural semantics and its equivalence with the fixpoint semantics.

1.1. Preliminaries: Logic programs

A *logic program P* is a finite set of clauses of the form

$$A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

where $n \geqslant 1$, $m \geqslant 0$, and the A's and B's are atomic formulas. The disjunction of atoms $A_1 \vee \ldots \vee A_n$ is called the *head* of the clause. The conjunction of atoms $B_1 \wedge \ldots \wedge B_m$ is called the *body* of the clause. We assume that all variables that occur in a clause are universally quantified in front of the clause. A *definite Horn* clause is a clause where $n = 1$. A *Horn* program consists of only definite Horn clauses. An *indefinite* or *disjunctive* clause is one where $n \geqslant 2$. A logic program is a *disjunctive program* if it contains a disjunctive clause. A *positive* clause or *assertion* is a clause with an empty body. The *Herbrand Universe U_P* of a logic program P, is the set of all ground terms that can be formed from the constants and function symbols in P (if there are no constants in P an arbitrary constant is placed in U_P). The *Herbrand Base* of a logic program P, $HB(P)$, is defined as the set of all ground atoms that can be formed by using predicates from P with terms from the Herbrand Universe U_P as arguments [9]. An *Herbrand interpretation I* for P is a subset of the Herbrand Base of P, in which all atoms in I are assumed to be *true* while those not in I are assumed to be *false*. A *Herbrand model* of P is a Herbrand interpretation of P that makes all clauses in P true. A substitution is a finite set of pairs $\{(x_1, t_1), \ldots, (x_n, t_n)\}$ where the x_s are distinct variables, the t_s terms and each x_i is different from t_i. Given a disjunction or a conjunction of disjunctions of atoms F and a substitution $\theta = \{(x_1, t_1), \ldots, (x_n, t_n)\}$, $F\theta$ is the formula obtained by simultaneously replacing all the occurrences of x_i in F by t_i, for $1 \leq i \leq n$. A clause C subsumes a clause D if there is a substitution θ and a subclause C' of C such that $C'\theta = D$.

1.2. Preliminaries: Fixpoint theory

Let S be a set and the relation $\leq$ be a binary relation on S and assume $\leq$ forms a partial order on the elements of S (i.e. $\leq$ is reflexive, transitive and antisymmetric). If X is a subset of S, then $a \in S$ is an *upper bound* of X if $\forall x \in X$, $x \leq a$. $a \in S$ is the *least upper bound* (*lub*) of X of S if a is an upper bound of X and for all upper bounds a' of X, we have $a \leq a'$. We can define a *greatest lower bound* (*glb*) of X in a similar manner. S is a *complete lattice* if $lub(X)$ and $glb(X)$ exists for every subset X of S. Given a complete lattice S, an operator $T: S \rightarrow S$ is said to be *continuous* if for every chain $x_1 \leq x_2 \leq \cdots$ of elements of S, $T(lub\{x_1, x_2, \ldots\}) = lub\{T(x_1), T(x_2), \ldots\}$. For a lattice S, given $x \in S$, x is a *fixpoint* (*fp*) of T if $T(x) = x$. We say x is the *least fixpoint* (*lfp*) of T if $x \leq x'$ for all fixpoints x' of T. For an operator T, we define the ordinal powers of T as follows:

$$T\uparrow 0 = glb(S)$$
$$T\uparrow \alpha = T(T\uparrow(\alpha - 1)), \quad \text{if } \alpha \text{ is successor ordinal}$$
$$T\uparrow \alpha = lub\{T\uparrow\beta : \beta < \alpha\}, \quad \text{if } \alpha \text{ is a limit ordinal.}$$

The next theorem contains a well-known property of continuous functions.

Theorem 1.1 (Lloyd [9]). *For a continuous operator $T: S \rightarrow S$, $lfp(T) = T\uparrow\omega$, where ω is the first limit ordinal.*

2. Declarative semantics

2.1. Model-state semantics

Among all the models of a program P we are interested in the Herbrand models. In particular, we are interested in the minimal Herbrand models since they have a close relation with fixpoint semantics. A model M of P is *minimal* if there is no proper subset M' of M such that M' is a model of P. Every Horn program P has a unique minimal Herbrand model M_P. The intended meaning of P could be characterized by any of its models but there is a strong reason that makes M_P its intended interpretation. That is, the atoms in M_P are precisely those that are logical consequences of P [6]. We can generalize this statement and say that every positive ground clause that is a logical consequence of P is subsumed by an atom in M_P. Hence, all logical consequences of a Horn program P are fully characterized by its unique minimal model M_P. A different situation occurs when we extend Horn programs to disjunctive programs. A disjunctive program can have more than one minimal model all of which characterize its logical consequences.

Theorem 2.1 (Minker [13]). *Let P be a disjunctive logic program. A positive ground clause C is a logical consequence of P iff C is true in every minimal model of P.*

Proof. We have that C is a logical consequence of P
iff $P \cup \{\neg C\}$ is unsatisfiable
iff $P \cup \{\neg C\}$ has no Herbrand models, by Proposition 3.3 in [9]
iff $\neg C$ is false w.r.t. all Herbrand models of P
iff C is true w.r.t. all Herbrand models of P
iff C is true w.r.t. all minimal Herbrand models of P $\square$

A characteristic that distinguishes Horn and disjunctive programs is that in disjunctive programs we can have a clause that is a logical consequence of P but none of its subclauses are. For example, take the simple disjunctive program $P = \{A \vee B\}$ where $A \vee B$ is a logical consequence of the program P. But neither A nor B are consequences of P. We refer to clauses such as $A \vee B$ in program P as *minimal* clauses of the program P since no subclause is a logical consequence of the program. We are interested in capturing such logical consequences in our semantics. In the case of Horn programs, logical consequences are characterized by atomic formulas and Herbrand models and Herbrand interpretations provide the proper structure for capturing them.

For disjunctive programs, the logical consequences are characterized by positive clauses but a single Herbrand interpretation or model does not capture this concept. Our first step is to extend the definition of the Herbrand Base to cover the disjunctive cases.

Definition 2.2 (Minker, Rajasekar [15]). Given a disjunctive logic program P, the *Disjunctive Herbrand Base of P*, $DHB(P)$, is the set of all positive clauses that can be formed with distinct atoms from $HB(P)$.

The need of the Disjunctive Herbrand Base is also reflected in the minimal models of a program. In contrast to Horn programs, disjunctive programs may have more than one minimal model. For the program P in the previous example, the minimal models are $\{A\}$ and $\{B\}$. We want to condense this information in a unique simple structure. For this, we extend the definitions of interpretations and models to *states* and *model-states*.

Definition 2.3. For a disjunctive logic program P
(1) a state of P is a subset of the Disjunctive Herbrand Base of P, $DHB(P)$;
(2) a model-state of P is a state S of P, such that
  (a) Every minimal model of P is a model of S.
  (b) Every minimal model of S is a model of P.

Lemma 2.4. *Every disjunctive program P has a model-state MS.*

Proof. Let MS be the set $\{C \in DHB(P)$: C is a logical consequence of $P\}$. We prove that MS is a model-state of P. Part (a) of the definition of model-states follows directly from Theorem 2.1. Let M be a minimal model of MS and by contradiction, assume M is not a model of P. Then, there is a ground instance $A_1 \vee \cdots \vee A_m \leftarrow B_1, \ldots, B_n$ of a clause in P such that $B_1 \wedge \cdots \wedge B_n$ is true in M but $A_1 \vee \cdots \vee A_m$ is false in M, i.e. $B_1 \in M, \ldots, B_n \in M$ and $A_1 \notin M, \ldots, A_m \notin M$. $B_1 \in M, \ldots, B_n \in M$ implies there are (possibly empty) positive clauses $C_1, \ldots, C_n$ such that $B_1 \vee C_1, \ldots, B_n \vee C_n \in MS$ and $C_1, \ldots, C_n$ are false in M, otherwise M is not minimal. Therefore, $C \vee C_1 \vee \cdots \vee C_n$ is a ground logical consequence of P. Therefore, $C \vee C_1 \vee \cdots \vee C_n$ belongs to MS. Then, M is a model of $C \vee C_1 \vee \cdots \vee C_n$. Hence, M is a model of C since $C_1, \ldots, C_n$ are false in M. Therefore, M is a model of $A_1 \vee \cdots \vee A_m \leftarrow B_1, \ldots, B_n$ contradicting our assumption. □

Now, we can collapse the information contained in the minimal models of a disjunctive program to its minimal model-states.

Definition 2.5. A model-state S of a program P is minimal iff there is no model-state of P which is a proper subset of S.

The following two theorems justify the choice of minimal model-states as the intended meaning of logic programs.

Theorem 2.6. *Every logic program P has a unique minimal model-state MS_P (the least model-state.)*

Proof. By Definition 2.3 a model M is a minimal model of a model-state of program P iff M is a minimal model of P. Assume M_1 and M_2 are minimal model-states of a program P. We have to prove $M_1 = M_2$. Let C be a clause in M_1. Hence, $M_2 \vdash C$ since every minimal model of M_1 is a minimal model of M_2. Since M_2 is a set of positive clauses then there is a clause $C' \in M_2$ such that $C' \subseteq C$. If $C' = C$ then $C \in M_2$. If $C' \subset C$ and $C' \in M_2$, using a similar argument we know that there is $C'' \in M_1$ such that $C'' \subseteq C' \subset C$. Therefore, $M_1 - \{C\}$ is a model-state contradicting that M_1 is minimal. We can use a similar argument to prove that every clause in M_2 is also in M_1. □

A consequence of this theorem is a corollary similar to the intersection model property for Horn programs [6].

Corollary 2.7. *For a logic program P the intersection of all model-states is the least model-state MS_P.*

Theorem 2.8. *For every positive ground clause C which is a logical consequence of a logic program P, there is a clause in MS_P that subsumes C.*

Proof. Follows from the definition of model-states. □

The set MS_P has been identified by [7] following a different approach. They define for each predicate Q in a program P, the set $PIGC[Q]$ which contains the minimal clauses where the predicate Q occurs in clauses which are derivable from P. Taking the union of the $PIGC$ sets over all the predicate symbols in P we obtain MS_P.

2.2. Fixpoint semantics

The power set of the Herbrand Base of a program P, $2^{HB(P)}$, is a complete lattice under the set inclusion relation. Van Emden and Kowalski [6] define a closure operator that maps a Herbrand interpretation to a Herbrand interpretation of a program P. They have shown that the operator is continuous for Horn programs and hence has a least fixpoint. The least fixpoint is also shown to define the intended meaning of a Horn program P in the sense that the least fixpoint of the program is the least model M_P of P. Here, we use the power set of $DHB(P)$, $2^{DHB(P)}$, (i.e. the set of all states of a program P) with the partial order set inclusion $\subseteq$ as the complete lattice underlying the fixpoint semantics of disjunctive programs. The closure operator that maps states to states of a program P is defined as follows:

Definition 2.9 (Minker, Rajasekar [15]). For a program P, a mapping $T_P: 2^{DHB(P)} \to 2^{DHB(P)}$ is defined as follows. Let S be a state of a program P, (i.e., S is a subset of $DHB(P)$), then

$$T_P(S) = \{C \in DHB(P) \mid C' \leftarrow B_1, B_2, \ldots, B_n \text{ is a ground instance of a program clause in } P,\ \{B_1 \vee C_1, \ldots, B_n \vee C_n\} \subseteq S \text{ where } \forall i,\ 1 \leq i \leq n,\ C_i \text{ can be null, } C'' = C' \vee C_1 \vee \cdots \vee C_n \text{ and } C \text{ is the smallest factor of } C''\}.$$

The smallest factor of a ground clause C' is defined as the clause C such that C contains only distinct atoms and C logically implies C' and C' logically implies C.

Example 2.10. Consider the program

$$P = \{p(X) \vee q(f(X)) \leftarrow r(X);\ t(X) \leftarrow q(X);\ p(b) \vee q(b);\ r(a) \vee s(a)\}$$

and the state

$$S = \{p(b) \vee q(b),\ r(a) \vee s(a)\}$$

then

$$T_P(S) = \{p(b) \vee q(b), r(a) \vee s(a), p(a) \vee q(f(a)) \vee s(a), p(b) \vee t(b)\}.$$

Minker and Rajasekar [15] prove that for a program P, the mapping T_P is continuous. Hence, $T_P \uparrow \omega$ is its least fixpoint. The next theorem shows that for a program P the least fixpoint of T_P contains all positive clauses that are derivable from the program P. First, we have to distinguish between the terms *derivability* and *provability*. We say a disjunctive program P derives a clause C if there is a finite sequence $C_1, C_2, \ldots, C_k$ of clauses such that C_i is either a clause in P, an instance of a clause preceding C_i, or a (binary) resolvent of clauses preceding C_i, and $C_k = C$. A clause is provable from a program when it is a logical consequence of the program. In the case of Horn programs the notions of provability and derivability of atoms coincide. For disjunctive programs this is not valid. With respect to the semantics we are developing, we are only interested in the intended meaning of a program in the *derivable* sense. That is, our intended semantics will achieve a state that contains all (and only) the clauses which are derivable from a logic program. Since any provable clause also has a subclause that is derivable, we feel justified in restricting our intended meaning of a logic program to derivable clauses without loss of generality.

Theorem 2.11 (Minker, Rajasekar [15]). *Given a program P,*

$$lfp(T_P) = \{C \in DHB(P) \mid C \text{ derivable from } P\}.$$

Next, we establish the equivalence between the fixpoint and model semantics for logic programs. For a program P, we denote by $MM(P)$ the set of minimal models of P. Using Theorems 2.1 and 2.11 we have the following result:

Lemma 2.12 (Minker, Rajasekar [15]). *Given a program P and a ground clause C,*

$$\forall M \in MM(P), \quad M \models C \text{ iff } lfp(T_P) \vdash C.$$

The next theorem follows directly from the lemma.

Theorem 2.13. *Let P be a logic program and S be a state of P.*

(i) *S is a model-state for P iff for all clauses $C \in T_P(S)$ there is a clause C' such that C' implies C.*

(ii) *S is the minimal model-state for P iff $S = can(lfp(T_P))$,*

where for a given set of positive ground clauses S, the canonical set of S, $can(S)$, is defined as $can(S) = \{C \mid C \in S$ and $\neg\exists C'$ such that $C' \in S$ and C' is a proper subclause of $C\}$.

Proof. Directly from Lemma 2.12 and definitions of model-state and minimal model-state. □

3. Procedural semantics

In this section we are concerned with the procedural semantics of logic programs. Procedural interpretations provide implementation-independent proof procedures for deriving inferences from logic programs. In the case of a Horn program the derivable consequences consist of atoms. Hence, a query consists of an atom or a conjunction of atoms of the form $\exists(A_1 \wedge \cdots \wedge A_n)$; $n \geqslant 0$. The "successful" answer to such a query is simple and consists of substitutions for the variables in the query. A substitution θ is a correct answer substitution for a query if $\forall((A_1 \wedge \cdots \wedge A_n)\theta)$ is a logical consequence of P. This provides the declarative meaning to the answer for such a query.

In the case of disjunctive programs the derivable consequences consist of disjunctions of atoms. Hence, a natural extension of a Horn query to the disjunctive domain is a query consisting of a disjunction of atoms or a conjunction of such disjunctions. A disjunctive query is of the form $\exists(C_1 \wedge \cdots \wedge C_n)$ where the C_i's are positive clauses and $n \geqslant 0$. But the answer to a disjunctive query is not a simple substitution as in the case of Horn programs as we can see in the following example. Consider, the disjunctive program $P = \{p(a) \vee p(b)\}$ and the query $Q = \exists X(p(X))$. We want to know if the query is a logical consequence of the program P. There is no single substitution which makes an appropriate answer for the query Q. However, in some cases a disjunctive query can also have an answer given as a single substitution. We call such answers simple answers. Consider the query $Q' = \exists X, Y(p(X) \vee p(Y))$ for the same program, then there exists a substitution $\{X = a,\ Y = b\}$ which provides a correct answer for the query Q'. As in Horn programs, a simple answer substitution θ is a correct answer substitution if $\forall(C_1 \wedge \cdots \wedge C_n)\theta$ is a logical consequence of the program. In this section we describe a procedure to answer simple queries, SLO-resolution. We refer to these queries as goals to distinguish them from queries with disjunctive answers. This procedure is similar to SLD-resolution [8]. Complete proof procedures for indefinite theories that use resolution based on model elimination [2] are highly expensive due to ancestry resolution and factoring. However, the similarities between SLD and SLO might lead to a good implementation for SLO-resolution.

Definition 3.1. A goal is of the form: $\leftarrow C_1, \ldots, C_n$, $n \geqslant 0$, where the C's are positive clauses.

Definition 3.2. Given a positive clause $C = A_1 \vee \cdots \vee A_p$, we say that C θ-subsumes a clause D if θ is the most general unifier for $\{A_1 = D_1, \ldots, A_p = D_p\}$ where $D_1 \vee \cdots \vee D_p$ is a subclause of D.

Definition 3.3. Let P be a disjunctive logic program and G be a goal. An SLO-derivation from P with top-goal G consists of a (possibly infinite) sequence of goals $G_0 = G$, $G_1, \ldots,$ such that for all $i \geqslant 0$, the goal G_{i+1} is obtained from $G_i = \leftarrow C_1, \ldots, C_m, \ldots, C_r$ (where the C's are positive clauses) as follows:

(1) C_m is a clause in G_i (C_m is called the selected clause),

(2) $C \leftarrow B_1, \ldots, B_q$ is a standardized variant of a program clause in P,

(3) C θ-subsumes C_m,

(4) G_{i+1} is the goal

$$\leftarrow (C_1, \ldots, C_{m-1}, B_1 \vee C_m, \ldots, B_q \vee C_m, C_{m+1}, \ldots, C_r)\theta.$$

The standardized variant is a renaming of all the variables in the original clause (in P) by variables that do not appear in the derivation up to G_i. Notice that when the body of the program clause is empty, G_{i+1} is equal to $\leftarrow (C_1, \ldots, C_{m-1}, C_{m+1}, \ldots, C_k)\theta$.

Definition 3.4. An SLO-refutation from P with top-goal G is a finite SLO-derivation of the null clause $\square$ from P with top-goal G. If $G_n = \square$, we say the SLO-refutation has length n.

Example 3.5. Let P be the following program:

$$P = \{t(X) \leftarrow p(f(X));\ p(X) \leftarrow m(X);\ p(f(X)) \leftarrow q(X);$$
$$q(X) \leftarrow m(f(f(X)));\ q(X) \leftarrow p(X);\ m(0) \vee m(f(f(X))) \leftarrow \}.$$

An SLO-refutation for the goal $\leftarrow t(0)$ is given below.

$\leftarrow \underline{t(0)}$

using $t(X) \leftarrow p(f(X))$

$\leftarrow \underline{p(f(0))} \vee t(0)$

using $p(f(X)) \leftarrow q(X)$

$\leftarrow \underline{q(0)} \vee p(f(0)) \vee t(0)$

using $q(X) \leftarrow m(f(f(X)))$

$\leftarrow m(f(f(0))) \vee \underline{q(0)} \vee p(f(0)) \vee t(0)$

using $q(X) \leftarrow p(X)$

$\leftarrow \underline{p(0)} \vee m(f(f(0))) \vee q(0) \vee p(f(0)) \vee t(0)$

using $p(X) \leftarrow m(X)$

$\leftarrow \underline{m(0)} \vee p(0) \vee \underline{m(f(f(0)))} \vee q(0) \vee p(f(0)) \vee t(0)$

using $m(0) \vee m(f(f(X))) \leftarrow$

$\square$

The following two theorems establish the soundness and completeness of SLO-resolution with respect to derivability, i.e. for a positive clause C and a disjunctive program P, there is an SLO-refutation for $\leftarrow C$ if and only if C is derivable from P. The proofs are similar to the soundness and completeness proofs of SLD-resolution [9].

Theorem 3.6 (Soundness). *Let P be a disjunctive program, $G = \leftarrow C_1, \ldots, C_k$ be a goal and $\theta_1, \ldots, \theta_n$ be substitutions, obtained from an SLO-refutation from P with top goal G, then $\forall((C_1 \wedge \cdots \wedge C_k)\theta_1, \ldots, \theta_n)$ is a logical consequence of P.*

Proof. We prove the theorem by induction on the length of the SLO-refutation.

(*Base case*) One step refutation ($n = 1$). Since G is a goal of the form $\leftarrow C_1$ there exists a program clause of the form $C\leftarrow$, such that $C\theta_1$ subsumes $C_1\theta_1$. Since $C\theta_1$ is an instance of an assertion clause in P, $C\theta_1$ is a logical consequence of P. Also $C_1\theta_1$ is a logical consequence of P, since $C\theta_1$ subsumes $C_1\theta_1$.

(*Induction hypothesis*) The theorem is valid for all SLO-refutations which are of size less than n.

(*Induction case*) *SLO-refutation of length n.* Let $C \leftarrow B_1, \ldots, B_q$ be the program clause used in the first step of the derivation, i.e. in the derivation of the goal G_1 from the starting goal $G_0 = G$ with C_m as the selected clause in G and θ_1 as the substitution used in the subsumption. Then

$$G_1 = \leftarrow(C_1, \ldots, C_{m-1}, B_1 \vee C_m, \ldots, B_q \vee C_m, C_{m+1}, \ldots, C_k)\theta_1.$$

Now, from the induction hypothesis, there is a refutation of length $n-1$ from P with top-clause G_1, using $\theta_2, \ldots, \theta_n$ as substitutions,

$\Rightarrow \forall((C_1 \wedge \cdots \wedge C_{m-1} \wedge B_1 \vee C_m \wedge \cdots \wedge B_q \vee C_m \wedge C_{m+1} \wedge \cdots \wedge C_k)\theta_1, \ldots, \theta_n)$ is a logical consequence of P

$\Rightarrow \forall((B_1 \vee C_m \wedge \cdots \wedge B_q \vee C_m)\theta_1, \ldots, \theta_n)$ is a logical consequence of P

$\Rightarrow \forall((C \vee C_m)\theta_1, \ldots, \theta_n)$ is a logical consequence of P since $C \leftarrow B_1, \ldots, B_q$ is a program clause

$\Rightarrow \forall((C_m)\theta_1, \ldots, \theta_n)$ is a logical consequence of P since $C\theta_1$ subsumes $C_m\theta_1$ from the definition of SLO-derivation

$\Rightarrow \forall((C_1 \wedge \cdots \wedge C_{m-1} \wedge C_m \wedge C_{m+1} \wedge \cdots \wedge C_k)\theta_1, \ldots, \theta_n)$ is a logical consequence of P. □

Theorem 3.7 (Completeness). *Let P be a disjunctive program and C be a ground clause which is derivable from P. Then there is an SLO-refutation from P with top goal C.*

Proof. C is derivable from P

$\Rightarrow C \in T_P \uparrow n$, for some $n \in \omega$.

We prove that $C \in T_P \uparrow n$ implies there is an SLO-refutation from P with top goal C. We show this by induction on n.

(*Base case*) $n = 0$. $T_P \uparrow 0 = \emptyset$ and there is nothing to prove.

(*Induction hypothesis*) The theorem is valid for values less than n.

(*Induction case*) $C \in T_P \uparrow n$ *and* $C \notin T_P \uparrow n-1$.

$C \in T_P \uparrow n$

$\Rightarrow$ There exists a program clause in P, $C' \leftarrow B_1, B_2, \ldots, B_q$ such that $C = (C' \vee C_1 \vee \cdots \vee C_q)\theta$, where θ is a substitution, where $(C' \vee C_1 \vee \cdots \vee C_q)\theta$ is ground and $(B_1 \vee C_1)\theta, \ldots, (B_q \vee C_q)\theta$ are in $T_P \uparrow n-1$ where C_i, $1 \leq i \leq q$ is a positive clause, possibly null (by definition of T_P),

$\Rightarrow (B_i \vee C_i)\theta$, $1 \leq i \leq q$ have an SLO-refutation from P (by the induction hypothesis),

$\Rightarrow$ There exists an SLO-refutation from P with $G = \leftarrow(B_1 \vee C_1, \ldots, B_q \vee C_q)\theta$ as the top goal. Since, each of the $(B_i \vee C_i)\theta$ is ground and has an SLO-refutation, these refutations can be combined into a refutation with G,

$\Rightarrow$ There exists an SLO-refutation from P with $G' = \leftarrow(B_1 \vee C, \ldots, B_q \vee C)\theta$ as the top goal. Since each $C_i\theta$ is a subclause of C and $(B_i \vee C_i)\theta$ has an SLO-refutation, $(B_i \vee C)\theta$ also has an SLO-refutation,

$\Rightarrow$ There exists an SLO-refutation from P with $G_0 = (C \vee C')\theta$ as the top. With G_0 as top goal we have $G_1 = (B_1 \vee C \vee C', \ldots, B_q \vee C \vee C')\theta$. G_1 has an SLO-refutation hence G_0 also has an SLO-refutation,

$\Rightarrow$ There exists an SLO-refutation from P with $G = C$ (C is ground) as the top goal, since $C'\theta$ is a subclause of C. □

In general, θ-subsumption between clauses is not unique. This introduces a new nondeterministic step (Step 3) not present in SLD-resolution. There are some heuristics that can be used to guide the subsumption. We currently have an implementation of SLO-resolution in Prolog which gives priority to the most recently added atoms of a goal clause while doing θ-subsumption. We also include a mechanism which checks for repetition of goals to detect some of the infinite derivations. Although the similarities between SLO and SLD might suggest efficient implementations of SLO-resolution, the restriction on the type of queries requires further investigation in the area. In [16, 15] Minker and Rajasekar present SLI-resolution as an alternative proof procedure for disjunctive logic programs. SLI-resolution is a full theorem prover developed by Minker and Zanon [17] (SLI-resolution was first named LUST-resolution by the authors). However, it might be possible to define a simpler system for disjunctive programs where explicit representation of negative information is not present.

4. Summary

We have presented three different characterizations for the semantics of (disjunctive) logic programs: a fixpoint characterization, a model theoretic one, based on model-states, and a proof-procedure characterization. We have also shown the equivalence between the three characterizations. The results can be summarized in the following theorem.

Theorem 4.1 (Disjunction characterization). *Let P be a logic program and $C \in DHB(P)$. Then the following are equivalent:*

(a) *C is true in every minimal Herbrand model of P.*
(b) *C is logically implied by a clause in the least model-state MS_P of P.*
(c) *C is logically implied by a clause in the least fixpoint of T_P.*
(d) *$\leftarrow C$ has an SLO-refutation using P.*
(e) *C is a logical consequence of P.*

A similar theorem in [6] describes the semantics for Horn programs. Moreover, all the results presented in this paper reduce to previous results obtained for Horn programs as indicated in Table 1. The fixpoint semantics extends the theory based on the operator T_P of van Emden and Kowalski [6] for Horn programs.

Table 1. Semantics for logic programs
(*Positive consequences*)

	Horn		Disjunctive	
Semantics	Theory	Reference	Theory	Reference
Fixpoint semantics	$T_P \uparrow \omega$	[6]	$T_P \uparrow \omega$	[15]
Model theory	Least model	[6]	Minimal model	[13]
			Model-state	Sect. 2.1
Procedure	SLD	[6, 8]	SLO	Sect. 3

The model-state semantics extends the least model semantics described in [6] and is equivalent to the minimal model semantics [13] developed for disjunctive logic programs. SLO-resolution is an extension of SLD-resolution of Horn programs [8].

Based on the results presented here and the correspondence between these results and the results in the Horn domain, a large spectrum of new developments have been achieved and reported upon elsewhere [5, 15, 18, 11, 14] by us and others. Using the Generalized Closed World Assumption (GCWA), developed by Minker [13], as a consistent rule of negation for disjunctive theories, it was possible to extend the semantics of disjunctive programs to general programs (where negated atoms are allowed in the body of program clauses). Minker and Rajasekar extend the concept of stratified programs of Apt, Blair and Walker [1] to disjunctive programs and describe an iterative definition for negation using the GCWA. A weaker definition of negation called the Weak Generalized Closed World Assumption [11] was used in [10] to describe a completion theory for disjunctive programs. Dung extended the completion theory to capture the Generalized Closed World Assumption [5]. Results extending the well-founded semantics for general Horn programs to disjunctive programs have been also reported [4, 3, 19]. Finally, the strong connections between negation in general Horn programs and nonmonotonic reasoning mechanisms like circumscription and default logic suggest that similar results might be obtained in the case of disjunctive programs with negation.

Acknowledgment

We wish to express our appreciation to the National Science Foundation for their support of our work under grant number IRI-86-09170 and the Army Research Office under grant number DAAG-29-85-K-0-177.

References

[1] K.R. Apt, H.A. Blair and A. Walker, Towards a theory of declarative knowledge, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, 1988) 89-148.

[2] K.R. Apt and M.H. van Emden, Contributions to the theory of logic programming, *J. ACM* **29**(3) (1982) 841-862.

[3] C. Baral, J. Lobo and J. Minker, Generalized disjunctive well-founded semantics for logic programs: declarative semantics, Submitted to *ICLP 90.*

[4] C. Baral, J. Lobo and J. Minker, Generalized well-founded semantics for logic programs, in: *Proc. 5th Internat. Symp. Methodologies for Intelligent Systems* (1990) 465-473.

[5] P.M. Dung and K. Kanchanasut, On the generalized predicate completion of non-Horn programs, in: E.L. Lusk and R.A. Overbeek, eds., *Proc. North Amer. Conf. of Logic Programming*, Cleveland, OH (1988) 587-603.

[6] M.H. van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. ACM* **23**(4) (1976) 733-742.

[7] L.J. Henschen and A. Yahya, Deduction in non-Horn databases, *J. Automat. Reason.* **1**(2) (1985) 141-160.

[8] R. Hill, Lush resolution and its completeness, Technical Report DCL Memo 78, Department of Artificial Intelligence, University of Edinburgh, August 1974.

[9] J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1984).

[10] J. Lobo, J. Minker and A. Rajasekar, Weak completion theory for non-Horn programs, in: *Proc. 5th Internat. Conf. Symp. on Logic Programming*, Seattle, Washington (1988) 828-842.

[11] J. Lobo, J. Minker and A. Rajasekar, Weak generalized closed world assumption, *J. Automat. Reason.* **5** (1989) 293-307.

[12] D.W. Loveland, *Automated Theorem Proving: A Logical Basis* (North-Holland, Amsterdam, 1978).

[13] J. Minker, On indefinite databases and the closed world assumption, in: Lecture Notes in Computer Science, Vol 138 (Springer, Berlin, 1982) 292-308.

[14] J. Minker and A. Rajasekar, On stratified disjunctive programs, *Annals of Mathematics and Artificial Intelligence* **1** (1990) 339-357.

[15] J. Minker and A. Rajasekar, A fixpoint semantics for disjunctive logic programs, *J. Logic Programming* **5**(1) (1990) 45-74.

[16] J. Minker and A. Rajasekar, Procedural interpretation of non-Horn logic programs, in: *Proc. 9th Internat. Conf. on Automated Deduction*, Argonne, IL (1988) 278-293.

[17] J. Minker and G. Zanon, An extension to linear resolution with selection function, *Inform. Process. Lett.* **14**(3) (1982) 191-194.

[18] A. Rajasekar, Semantics for disjunctive logic programs, Ph.D. Thesis, Department of Computer Science, University of Maryland, 1989.

[19] K. Ross, Well-founded semantics for disjunctive logic programs, in: *Proc. 1st Internat. Conf. on Deductive and Object Oriented Databases*, Kyoto, Japan (1989).