Semantics of Disjunctive Programs with Monotone Aggregates — an Operator-based Approach

Nikolay Pelov
Department of Computer Science
K.U. Leuven
Celestijnenlaan 200A
B-3001 Heverlee, Belgium

Mirosław Truszczyński
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046, USA

Abstract

All major semantics of normal logic programs and normal logic programs with aggregates can be described as fixpoints of the one-step provability operator or of operators that can be derived from it. No such systematic operator-based approach to semantics of disjunctive logic programs has been developed so far. This paper is the first step in this direction. We formalize the concept of one-step-provability for disjunctive logic programs by means of non-deterministic operators on the lattice of interpretations. We establish characterizations of models, minimal models, supported models and stable models of disjunctive logic programs in terms of pre-fixpoints and fixpoints of non-deterministic immediate-consequence operators and their extensions to the four-valued setting. We develop our results for programs in propositional language extended with monotone aggregate atoms. For the most part, our concepts, results and proof techniques are algebraic, which opens a possibility for further generalizations to the abstract algebraic setting of non-deterministic operators on complete lattices.

Introduction

All major semantics of normal logic programs can be described as fixpoints of the *one-step provability* operator (van Emden & Kowalski 1976) or operators that can be derived from it (Fitting 2002). Generalizing these characterizations, researchers demonstrated that constructions leading to the fixpoints and the corresponding semantics are of purely algebraic nature and can be stated in abstract terms of operators on complete lattices (Denecker, Marek, & Truszczyński 2000). That algebraic approach is an effective tool in logic programming and was recently exploited to develop and study semantics of logic programs with cardinality constraints (Marek, Niemelä, & Truszczyński 2004) and aggregate atoms (Denecker, Pelov, & Bruynooghe 2001; Pelov, Denecker, & Bruynooghe 2004).

One of the motivations for this work is to extend all these semantics of programs with aggregates to *disjunctive* logic programs. However, no systematic operator-based approach of the semantics of disjunctive logic programs has been developed so far. This paper is the first step in this direction. Our contributions are as follows. We formalize the concept of one-step-provability for disjunctive logic programs by means of *non-deterministic* operators on the lattice of interpretations. We develop characterizations of models, minimal models, supported models and stable models of disjunctive logic programs in terms of pre-fixpoints and fixpoints of non-deterministic immediate-consequence operators and their extensions to the setting of four-valued interpretations.

Since defining a semantics of programs with aggregates is not the main focus of the paper we consider a language with a limited form of aggregate atoms, namely *monotone aggregate atoms*. Such atoms extend the notion of monotone cardinality atoms (Marek, Niemelä, & Truszczyński 2004) and correspond to positive aggregate atoms of (Pelov, Denecker, & Bruynooghe 2004). Logic programs with this form of aggregation exhibit properties that do not hold for programs without aggregates. For example a normal logic program with monotone aggregate atoms may have models but no minimal models, which is not the case for programs without aggregates (Seipel, Minker, & Ruiz 1997). So, we develop our results in a way that addresses that issue.

We point out that, as in the case of normal logic programs, some of our concepts and results have direct generalizations to the abstract algebraic setting of non-deterministic operators on complete lattices. We believe that the results of our paper will prove useful in abstracting properties of non-deterministic operators corresponding to disjunctive logic programs with monotone aggregates to more general (and algebraically defined) classes of non-deterministic operators on complete lattices and in constructing for non-deterministic operators on lattices a counterpart of the theory of approximating operators from (Denecker, Marek, & Truszczyński 2000).

Preliminaries

For a set X we denote the set of all subsets of X with $\mathcal{P}(X)$. A collection $\mathcal{C} \subseteq \mathcal{P}(X)$ is *upward closed* if $A \in \mathcal{C}$ and $A \subseteq B$ implies $B \in \mathcal{C}$.

In the paper, we study a class of disjunctive logic programs with monotone aggregates. We focus on the propositional case as the first-order case can be handled by lifting concepts and results from the propositional one in a standard way by *grounding*. We assume a fixed countable set At of atoms. A *monotone aggregate atom*, or *ma-atom* for short, is an expression of the form $\mathcal{C}(X)$ where $\mathcal{C} \subseteq \mathcal{P}(At)$ is an upward closed set and $X \subseteq At$. We call X the *scope* of $\mathcal{C}(X)$. In our discussion, we allow aggregate atoms with infinite scopes. We denote with $\mathcal{C}_k$ the set of all subsets of

At that contain at least k elements. Clearly, $\mathcal{C}_k$ is upward closed, so $\mathcal{C}_k(X)$ is a monotone aggregate atom.

A *propositional interpretation* (or, simply, an *interpretation*) is a set of atoms. We denote the set of all interpretations with $\mathcal{I} = \mathcal{P}(At)$. An interpretation I *satisfies* an atom $A \in At$, $I \models A$, if $A \in I$. An interpretation I *satisfies* an ma-atom $\mathcal{C}(X)$, denoted with $I \models \mathcal{C}(X)$, if $I \cap X \in \mathcal{C}$. Otherwise, $I \not\models \mathcal{C}(X)$. The concept of satisfaction, as well as the notation, extends in the standard way to literals built of atoms and ma-atoms, and their conjunctions and disjunctions.

Satisfiability of aggregate atoms is monotone, which justifies their name.

Proposition 1 *For every ma-atom $\mathcal{C}(X)$ and every two interpretations $I, I' \subseteq At$, if $I \subseteq I'$ and $I \models \mathcal{C}(X)$ then $I' \models \mathcal{C}(X)$.*

Due to their monotonicity property, ma-atoms can be regarded as direct generalization of "regular" propositional atoms. With that in mind, we define a *disjunctive rule with monotone aggregates* as an expression of the form

$$A_1 \vee \cdots \vee A_k \leftarrow L_1 \wedge \ldots \wedge L_m,$$

where $k \geq 1$, A_i are atoms from At, and L_i are literals built of atoms from At and of ma-atoms. The disjunction $A_1 \vee \cdots \vee A_m$ is the *head* of r. The conjunction $L_1 \wedge \ldots \wedge L_n$ is the *body* of r. We use the notation $head(r)$ and $body(r)$ for the head and the body of r, respectively.

A *disjunctive program with monotone aggregates* is a (possibly infinite) set of disjunctive rules with monotone aggregates. To simplify the notation, we write *disjunctive program* for *disjunctive programs with monotone aggregates*. We explicitly emphasize departures from that convention. A disjunctive program is *definite* if it contains only atoms and ma-atoms in the bodies of its rules.

A *positive clause* is a finite disjunction of atoms from At. The *disjunctive base* of a program P, $B^{\vee}(P)$, is the set of all positive clauses consisting of atoms in P. For a set C of positive clauses, $At(C)$ denotes the set of all atoms in C. We note that heads of disjunctive rules are positive clauses.

An interpretation I *satisfies*, or is a *model of* a disjunctive rule r, written as $I \models r$, if $I \models body(r)$ implies that $I \models head(r)$. An interpretation I *satisfies*, or is a *model of* a disjunctive program P, $I \models P$, if I satisfies every rule of P.

A model I of a disjunctive program P is *minimal* if for every model I' of P, $I' \subseteq I$ implies that $I' = I$. We denote the set of all models of a program P by $Mod(P)$ and the set of all minimal models by $MM(P)$.

The fact that we allow ma-atoms with infinite scope is important and makes the setting essentially different from the standard one. In particular, it is known (Seipel, Minker, & Ruiz 1997) that every model of a disjunctive logic program (without aggregates) contains a minimal model. That property does not hold in our setting (even without disjunction in the heads).

Example 1 *Let P be the program consisting of all clauses of the form*

$$p \leftarrow \mathcal{C}_k(At), \neg(\mathcal{C}_{k+1}(At))$$

where $p \in At$, $k = 1, 2, \ldots$ and $\mathcal{C}_k$ is the collection of sets we introduced earlier. Every infinite interpretation I, $I \subseteq At$, is a model of P (the body of every rule is false in I). However, no finite interpretation I, $I \subseteq At$, is a model of P (for every $p \in At$, there is a rule in P with the head p and with the body true in I).

Non-deterministic Operators on the Lattice of Interpretations

Our goal is to generalize supported-model and stable-model semantics to the case of disjunctive programs with monotone aggregates and show that they can be described and studied by means of non-deterministic operators on the lattice of interpretations. In this section, we recall some basic terminology and results.

Definition 1 *A* non-deterministic operator *on $\mathcal{I}$ is any function $N \colon \mathcal{I} \to \mathcal{P}(\mathcal{I})$ such that for every $I \in \mathcal{I}$, $N(I) \neq \emptyset$.*

Informally, $N(I)$ describes all possible outcomes of applying an operator N to I. There is at least one outcome to choose from and each of them can be chosen.

An important class of (deterministic) operators that appear in the studies of normal logic programs (also in the case of normal logic programs with aggregates) is the class of *monotone* operators on the lattice of interpretations. There are several ways to generalize that class in the non-deterministic setting. The one that is relevant to us is based on the *Smyth pre-order*, $\preceq^S$. Given subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{I}$, we write $\mathcal{A} \preceq^S \mathcal{B}$ if

for every $B \in \mathcal{B}$ there is $A \in \mathcal{A}$ such that $A \subseteq B$.

The relation $\preceq^S$ is reflexive and transitive but, in general, not antisymmetric. It is therefore a pre-order but, in general, not an order on the set $\mathcal{P}(\mathcal{I})$.

Let N be a non-deterministic operator on $\mathcal{I}$. The operator N is *Smyth-monotone* (or, simply, monotone — as we do not consider here any other pre-orders on $\mathcal{P}(\mathcal{I})$) if for every interpretations $I, J \in \mathcal{I}$,

$I \subseteq J$ implies $N(I) \preceq^S N(J)$.

An interpretation I is a *fixpoint* of N if $I \in N(I)$. We denote the set of all fixpoints of N with $fp(N)$ and the set of all minimal fixpoints of N with $mfp(N)$. A *pre-fixpoint* of N is an interpretation I such that $N(I) \preceq^S \{I\}$, that is, there exists $J \in N(I)$ such that $J \subseteq I$. We denote the set of all pre-fixpoints of N with $pre(N)$. We have the following basic result concerning pre-fixpoints and fixpoints of monotone non-deterministic operators.

Lemma 1 *Let $N : \mathcal{I} \to \mathcal{P}(\mathcal{I})$ be a monotone non-deterministic operator on $\mathcal{I}$. If I is a minimal pre-fixpoint of N then I is a minimal fixpoint of N.*

Proof: Since I is a pre-fixpoint of N, there exists $J \in N(I)$ such that $J \subseteq I$. By the monotonicity of N, $N(J) \preceq^S N(I)$ and as $J \in N(I)$, it follows that $N(J) \preceq^S \{J\}$. Consequently, J is a pre-fixpoint of N. Since I is a minimal pre-fixpoint of N and $J \subseteq I$, $J = I$. Thus, $I \in N(I)$ and I is a fixpoint of N. Since fixpoints of N are pre-fixpoints of N, I is a minimal fixpoint of N. □

Operators, for which also the converse holds, play a particularly important role in our considerations.

Definition 2 *We call a non-deterministic operator $N : \mathcal{I} \rightarrow \mathcal{P}(\mathcal{I})$* downward closed *if the set of pre-fixpoints of N is closed under greatest lower bounds of descending chains. That is, N is downward closed if for every sequence of $\mathcal{A} = \{A_\xi\}_{\xi<\alpha}$ of interpretations such that*

1. *for every $\xi < \alpha$, A_ξ is a pre-fixpoint of N, and*
2. *for every $\xi < \xi' < \alpha$, $A_{\xi'} \subseteq A_\xi$*

the set $A = \bigcap \mathcal{A}$ is a pre-fixpoint of N.

The following property of downward closed operators follows directly from Zorn Lemma.

Lemma 2 *Let $N : \mathcal{I} \rightarrow \mathcal{P}(\mathcal{I})$ be a non-deterministic operator on $\mathcal{I}$ that is downward closed. Then, for every pre-fixpoint I of N there is a minimal pre-fixpoint I' of N such that $I' \subseteq I$.*

Corollary 1 *Let $N : \mathcal{I} \rightarrow \mathcal{P}(\mathcal{I})$ be a monotone non-deterministic operator on $\mathcal{I}$ that is downward closed. Then, an interpretation I is a minimal pre-fixpoint of N if and only if I is a minimal fixpoint of N.*

Proof: In view of Lemma 1, only the "if" part needs a proof. Let I be a minimal fixpoint of N. Then I is a pre-fixpoint of N. By Lemma 2, there is $I' \subseteq I$ such that I' is a minimal pre-fixpoint of N. By Lemma 1, I' is a fixpoint of N. By minimality of I, $I' = I$ and I is a minimal pre-fixpoint of N. □

Immediate-consequence Operators for Disjunctive Programs

To develop an operator-based framework for disjunctive logic programs, the key step is to introduce for such programs the concept of the immediate-consequence operator.

Since the heads of rules are disjunctions of atoms, unlike in the case of normal logic programs (with aggregates), when the body of a rule is satisfied by an interpretation, there is no single atom that is forced. Instead, there are several ways to satisfy the disjunctive constraint described by the head of the rule. This suggests that, given an input interpretation, the process of drawing immediate consequences from a disjunctive program is inherently *non-deterministic*. Consequently, to formalize it we utilize *non-deterministic* operators on the lattice of interpretations. The crucial feature of our approach is that we do not propose a single specific immediate-consequence operator but define the concept by imposing some general desiderata. Interestingly, it turns out that all operators that satisfy our requirements have the same basic properties and properly capture key semantic properties of the program.

We proceed in three steps. In the first step we collect the heads of all clauses whose bodies are satisfied by an input interpretation I.

Definition 3 *The* head reduct *of a disjunctive program P is a function $HR_P : \mathcal{I} \rightarrow \mathcal{P}(B^\vee(P))$ defined as:*

$$HR_P(I) = \{head(r) \mid r \in P \text{ and } I \models body(r)\}.$$

We have the following simple property of the head reduct of a disjunctive program.

Lemma 3 *Let P be a disjunctive program. For every interpretation I, $I \models P$ if and only if $I \models HR_P(I)$.*

Proof: Let C be a positive clause. By the definition, $C \in HR_P(I)$ if and only if there exists a clause $r \in P$ such that $head(r) = C$ and $I \models body(r)$. Thus, if $I \models P$, then $I \models HR_P(I)$. Conversely, let $r \in P$ be such that $I \models body(r)$. Then, $head(r) \in HR_P(I)$ and $I \models head(r)$. Consequently, $I \models r$. □

In the second step, we introduce the concept of a *selection function*. Let P be a disjunctive program. Given a collection of positive clauses $C \subseteq B^\vee(P)$, a selection function returns some set of models of C. We will denote the set of models of C with $Mod(C)$ (since a collection of positive clauses can be viewed as a disjunctive program whose rules have empty bodies, this notation is consistent with the notation $Mod(P)$, we introduced earlier for the set of models of a program P).

Definition 4 (Selection Function) *Let P be a disjunctive program.* A selection function *is a function $Sel\colon \mathcal{P}(B^\vee(P)) \rightarrow \mathcal{P}(\mathcal{I})$ such that for every set of positive clauses $C \subseteq B^\vee(P)$, $Sel(C)$ satisfies the following conditions:*

$$Sel(C) \subseteq Mod(C) \quad \text{(P1)}$$

$$Sel(C) \preceq^S Mod(C) \quad \text{(P2)}$$

$$I \subseteq At(C) \text{ for every } I \in Sel(C) \quad \text{(P3)}$$

The first condition ensures that $Sel(C)$ contains only models of C. The second condition ensures that every model of C is *covered* by some interpretation in $Sel(C)$. Finally, the third condition guarantees that interpretations in $Sel(C)$ consist only of relevant atoms, that is, atoms that appear in clauses in C. This condition allows us later to properly generalize the concept of a supported model.

We note that Definition 4 implies directly that for every selection function Sel, $Sel(\emptyset) = \{\emptyset\}$. Furthermore, it is easy to show that for every selection function Sel, $Sel(C)$ always includes the set of minimal models of C, which we denote with $MM(C)$.

Proposition 2 *For every selection function Sel, $MM(C) \subseteq Sel(C)$.*

Proof: Let $I \in MM(C)$. Then I is a model of C and, by (P2), there exists $J \in Sel(C)$ such that $J \subseteq I$. By (P1), $J \in Mod(C)$. Since I is a minimal model of C, $J = I$ and $I \in Sel(C)$. □

In fact, assigning to a collection of positive clauses C the set $MM(C)$ of minimal models of C yields a selection function.

Proposition 3 *The function $MM(C)$ is a selection function.*

Proof: Let $C \subseteq B^\vee(P)$. Clearly, by the definition of minimal models, $MM(C) \subseteq Mod(C)$. Thus, the condition (P1) follows. Next, $MM(C) \preceq^S Mod(C)$, as every model of C contains a minimal model of C (this fact is well known and follows, in particular, from the result by (Seipel, Minker, & Ruiz 1997), which we mentioned earlier). Consequently, the condition (P2) follows. Finally, the condition (P3) is evident. □

Another possible selection function assigns to every set C of positive clauses the set of all those models of C that consist only of atoms from C. We denote this selection function with Mod^s and the corresponding set of interpretations with $Mod^s(C)$ (the subscript s indicates that we focus here on models built of atoms appearing in C and *not* on all models of C built of atoms in At). The selection functions MM and Mod^s are the two extreme cases of selection functions in the sense that $MM(C) \subseteq Sel(C) \subseteq Mod^s(C)$ for every selection function Sel. We now show that every selection function Sel is $\preceq^S$-monotone.

Lemma 4 *Let C_1 and C_2 be two sets of positive clauses. If $C_1 \subseteq C_2$ then $Sel(C_1) \preceq^S Sel(C_2)$.*

Proof: Let $J_2 \in Sel(C_2)$. By property (P1) $J_2 \models C_2$. Since $C_1 \subseteq C_2$ then also $J_2|_{At(C_1)} \models C_1$. By (P2) there exists $J_1 \in Sel(C_1)$ such that $J_1 \subseteq J_2|_{At(C_1)} \subseteq J_2$. □

We are ready to complete the definition of non-deterministic immediate-consequence operators of a disjunctive logic program P. The basic intuition is to view a disjunctive program P as a device to revise interpretations. To revise an interpretation I, we first derive the heads of all rules in P that are applicable with respect to I, that is, all positive clauses in $HR_P(I)$. Given a fixed selection function, this set of clauses determines some collection of models of $HR_P(I)$. Any of these models can be viewed as a plausible revision of I and its immediate consequence with respect to P (and under this fixed selection function). This intuition is formalized below.

Definition 5 (Non-deterministic Operator) *Given a selection function Sel, a* non-deterministic immediate-consequence operator $N_P^{Sel}\colon \mathcal{I} \rightarrow \mathcal{P}(\mathcal{I})$*, based on Sel is defined as:*

$$N_P^{Sel}(I) = Sel(HR_P(I)).$$

We simply write N_P instead of N_P^{Sel}, whenever the selection function Sel is clear from the context or the choice of any particular selection function is immaterial.

The next two results hold for every choice of the selection function. First, we show that non-deterministic immediate-consequence operators collapse to the van Emden-Kowalski immediate-consequence operator (no matter what selection function is used) if we restrict attention to the class of normal logic programs (no disjunctions in the heads of the rules and no aggregates) or, more generally, to the van Emden-Kowalski immediate-consequence operator for normal logic programs with aggregates (Pelov, Denecker, & Bruynooghe 2004).

Proposition 4 *If P is normal logic program with monotone aggregates, then $N_P(I) = \{T_P(I)\}$ (P is regarded as a disjunctive program, when computing $N_P(I)$, and as a normal logic program, when computing $T_P(I)$).*

Proof: Since P is a normal logic program with aggregates, $HR_P(I)$ consists of atoms and $HR_P(I) = T_P(I)$. Moreover, for every selection function Sel, $Sel(HR_P(I)) = \{HR_P(I)\}$ and the assertion follows. □

Next, we show that models of a disjunctive program coincide with pre-fixpoints of an immediate-consequence operator (no matter what selection function is used). That result generalizes a similar characterization of models of normal programs (Apt, Blair, & Walker 1988).

Theorem 1 *Let P be a disjunctive program. Then $Mod(P) = pre(N_P)$.*

Proof: Let $I \in pre(N_P)$. For every rule $r \in P$, if $I \not\models body(r)$ then $I \models r$. Thus, let us consider a rule $r \in P$ such that $I \models body(r)$. By the definition of $HR_P(I)$, $head(r) \in HR_P(I)$. Since, $N_P(I) \preceq^S \{I\}$, there is an interpretation $J \in N_P(I)$ such that $J \subseteq I$. Clearly, $J \models HR_P(I)$ and, consequently, $J \models head(r)$. Since $J \subseteq I$ and since $head(r)$ is a positive clause, $I \models r$.

Conversely, let $I \in Mod(P)$. By Lemma 3, $I \models HR_P(I)$. By (P2) there exists $J \in Sel(HR_P(I)) = N_P(I)$ such that $J \subseteq I$. Thus, $J \in N_P(I)$ and, consequently, $N_P(I) \preceq^S \{I\}$. □

For specific selection functions we can show a correspondence between fixpoints of the non-deterministic operator N_P and supported models. There are two different notions of supported models for disjunctive logic programs: weakly supported (Brass & Dix 1997) and supported (Brass & Dix 1997; Inoue & Sakama 1998). The difference between the two is whether a single rule can be used as a support of more than one atom in the head.

Definition 6 *An interpretation I is a* weakly supported model *of a program P if $I \models P$ and for every atom $A \in I$ there exists a rule $r \in P$ such that $I \models body(r)$ and $A \in head(r)$. An interpretation I is a* supported model *of a program P if $I \models P$ and for every atom $A \in I$ there exists a rule $r \in P$ such that $I \models body(r)$, $A \in head(r)$, and $head(r) \cap I = \{A\}$. The sets of weakly supported and supported models of a program P are denoted with W-SUPP(P) and SUPP(P) respectively.*

Our first result on supported models holds for any selection function.

Proposition 5 *Let P be a disjunctive program and Sel a selection function. Then SUPP$(P) \subseteq fp(N_P^{Sel}) \subseteq$ W-SUPP(P).*

Proof: Let I be a supported model. Because $I \models P$ then, by Theorem 1, there exists $J \in N_P^{Sel}(I)$ such that $J \subseteq I$. Suppose that $J \subset I$, i.e., there exists an atom $A \in I - J$. Because I is supported and $A \in I$ then there exists a rule $r \in P$ such that $I \models body(r)$ and $I \cap head(r) = \{A\}$. Consequently, $head(r) \in HR_P(I)$ but $J \not\models head(r)$, which is a contradiction with $J \in N_P^{Sel}(I)$ and (P1).

Let $I \in fp(N_P^{Sel})$, i.e., $I \in N_P^{Sel}(I) = Sel(HR_P(I))$. Then for every atom $A \in I$ there exists a clause $C \in HR_P(I)$ such that $A \in C$ (by (P3)). Moreover, for this clause there exists a rule $r \in P$ such that $I \models body(r)$ and $C = head(r)$. This implies that I is a weakly supported model. □

We can obtain a precise characterization of the weakly supported and supported models as fixpoints of N_P by taking specific selection functions.

Theorem 2 *Let P be a disjunctive program. Then, SUPP$(P) = fp(N_P^{MM})$.*

Proof: The inclusion *SUPP*$(P) \subseteq fp(N_P^{MM})$ follows from Proposition 5. Let $I \in N_P^{MM}(I) = MM(HR_P(I))$. Then for every $A \in I$ there exists at least one clause $C \in HR_P(I)$ such that $I \cap C = \{A\}$ (otherwise, $I - \{A\}$ would be a model of $HR_P(I)$). But then there also exists a rule $C \leftarrow B \in P$ such that $I \models B$ and $I \cap C = \{A\}$. Hence, I is a supported model. □

Theorem 3 *Let P be a disjunctive program. Then, W-SUPP$(P) = fp(N_P^{Mod^s})$.*

Proof: Let $I \in$ *W-SUPP*(P). Since $I \models P$, it follows by Lemma 3 that $I \models HR_P(I)$. Moreover, since I is a weakly supported model, for every atom $A \in I$ there exists a rule $r \in P$ such that $I \models body(r)$ and $A \in head(r)$. The first condition implies that $head(r) \in HR_P(I)$ and together with the second condition we have $I \subseteq At(HR_P(I))$. Thus $I \in Mod^s(HR_P(I)) = N_P^{Mod^s}(I)$.

The opposite direction follows from Proposition 5. □

Definite Disjunctive Programs

Definite programs without disjunctions have a single minimal (in fact, least) model. Moreover, the T_P operator is monotone and its least fixpoint is equal to the least model of P. On the other hand, definite *disjunctive* programs can have several minimal models. However, we can still establish a correspondence between minimal models of P and minimal fixpoints of N_P, which is the goal of this subsection.

Proposition 6 *If P is a definite disjunctive program, then N_P is monotone, that is, $I \subseteq J$ implies $N_P(I) \preceq^S N_P(J)$.*

Proof: Since P is a definite program, if $I \subseteq J$ then $HR_P(I) \subseteq HR_P(J)$. Thus, by Lemma 4,

$$Sel(HR_P(I)) \preceq^S Sel(HR_P(J)),$$

which implies $N_P(I) \preceq^S N_P(J)$. □

The main result of this section is that for a definite disjunctive program the set of minimal models is equal to the set of minimal fixpoints of N_P. That result holds no matter what selection function is used. We start by showing that we can apply Corollary 1.

Lemma 5 *Let P be a definite disjunctive program. Then N_P is downward closed.*

Proof: Let $\mathcal{A} = \{A_\xi \mid \xi < \alpha\}$ be a descending chain of pre-fixpoints models of N_P. Let $B = \bigcap \mathcal{A}$. We will show that B is a pre-fixpoint of P. It is so if $\mathcal{A}$ is finite. So, we will assume that $\mathcal{A}$ is infinite.

Let $r \in P$ and $B \models body(r)$. For every $A \in \mathcal{A}$, $B \subseteq A$. Since r is a definite clause, $A \models body(r)$. Moreover, A is a model of r. Thus, $A \models head(r)$ and there is a disjunct of $head(r)$ that belongs to A. The number of disjuncts in $head(r)$ is finite. Thus, at least one of them belongs to infinitely many interpretations A in $\mathcal{A}$. Since $\mathcal{A}$ is a descending chain, that particular disjunct belongs to all interpretations in $\mathcal{A}$ and, so, to B as well. It follows that $B \models head(r)$, $B \models r$ and, finally, that $B \models P$. By Theorem 1, B is a pre-fixpoint of N_P. □

Theorem 4 *Let P be a definite disjunctive program. Then $MM(P) = mfp(N_P)$.*

Proof: ($\Rightarrow$) By Theorem 1 there is a one to one correspondence between models of P and pre-fixpoints of N_P. So, minimal models of P are minimal pre-fixpoints of N_P and, by Lemma 1, they are minimal fixpoints.
($\Leftarrow$) Let M be a minimal fixpoint of N_P. Since N_P is downward closed (Lemma 5), by Corollary 1, M is a minimal pre-fixpoint of N_P and, consequently, a minimal model of P (by Theorem 1). □

The following example shows that Theorem 4 does not hold for general disjunctive programs no matter what selection function is used.

Example 2 *Consider the following disjunctive logic program P:*

$$a \vee b.$$
$$c \leftarrow \neg d.$$

The interpretation $I = \{a, d\}$ is a minimal model of P. However, I is not a fixpoint of N_P for any selection function. Indeed, using the selection function Mod^s we have $N_P^{Mod^s}(I) = \{\{a\}, \{b\}, \{a, b\}\}$ and $I \notin N_P(I)$. The claim holds for every selection function Sel since $Sel(C) \subseteq Mod^s(C)$.

One result from the theory of monotone operators, which we did not generalize is the computation of the least fixpoint by iterating the immediate-consequence operator. There are two issues here. First, one needs to generalize the notion of iteration of an operator to the setting of non-deterministic operators. Second, one needs to choose an appropriate selection function to specify a particular immediate-consequence operator. The question is whether one can address both issues so that the results of computations with respect to the selected immediate-consequence operator N_P are precisely the minimal models of P (minimal pre-fixpoints of N_P).

A possible way to approach the first issue is to generalize the concept of an iteration of an operator by means of *computations* (Marek, Niemelä, & Truszczyński 2004). Given a non-deterministic operator N_P on $\mathcal{P}(At)$, we define a computation of N_P as a sequence $\{X_n\}_{n=0,1,\ldots}$, where $X_0 = \emptyset$, $X_{n+1} \in N_P(X_n)$, and $X_n \subseteq X_{n+1}$, for every $n \geq 0$. We call $\bigcup_{n=0}^{\infty} X_n$ the *result* of the computation $\{X_n\}_{n=0,1,\ldots}$.

As concerns the second issue, the most natural candidate seems to be the selection function MM. Indeed, other selection function would allow for a possibility of selecting non-minimal models of the set of rule heads computed in intermediate steps, and would not guarantee minimality of the results of computations. However, the selection function MM does not work. There are programs for which some minimal models cannot be obtained as a result of any computation of the operator N_P^{MM}, and there are programs for which the results of some computations are not minimal models. The following two examples illustrate these cases.

Example 3 *Let P consists of the clauses*

$a \vee b \vee c$
$a \leftarrow b$
$b \leftarrow c$
$c \leftarrow a.$

Applying the operator N_P^{MM} to the empty interpretation we obtain $N_P^{MM}(\emptyset) = \{\{a\},\{b\},\{c\}\}$. However, $N_P^{MM}(\{a\}) = \{\{c\}\}$, $N_P^{MM}(\{c\}) = \{\{b\}\}$, and $N_P^{MM}(\{b\}) = \{\{a\}\}$. Thus the only model of the program $\{a,b,c\}$ (which is also minimal) cannot be reached by any computation.

Example 4 *Let $At = \{a_1, a_2, \ldots\}$ and let P consist of the clauses*

$$a_1 \vee a_2$$
$$a_i \vee a_j \leftarrow a_{j-2},\ j \geq 3, 1 \leq i < j.$$

Let $X_n = \{a_1, \ldots, a_n\}$. Clearly the sequence $X_0, X_1, \ldots$ is a computation and At is its result. However, At is not a minimal model of this program. For instance, $\{a_2, a_3, \ldots\}$ is a model, of P, too.

These examples indicate that either (1) minimal models of definite programs do not have a characterization as results of computations for an immediate-consequence operator, or that (2) the notion of computation is not the correct generalization of the idea of iterating an operator or, finally (and least likely) that (3) a selection function other than MM needs to be used.

Approximating Operators and Stable Models

The main goal of this section is to develop a fixpoint characterization of the stable semantics of disjunctive logic programs (Gelfond & Lifschitz 1991; Przymusinski 1991). We exploit ideas from approximation theory (Denecker, Marek, & Truszczyński 2000) and define the notion of an approximating operator A_P of the non-deterministic operator N_P. Unlike in (Denecker, Marek, & Truszczyński 2000), where both the domain and the range of the T_P operator are approximated, here we approximate only the domain of the N_P operator. Consequently, we define only a two-valued stable semantics and not the entire class of three and four valued stable models (which include the well-founded model). The presentation is developed in the language with ma-atoms. As a result, we obtain a novel definition of a stable semantics for disjunctive programs with monotone aggregates. This semantics extends the stable semantics of disjunctive logic programs without aggregates (Gelfond & Lifschitz 1991; Przymusinski 1991) and the stable semantics of normal logic programs with positive aggregate atoms (Pelov, Denecker, & Bruynooghe 2004).

The definition of A_P is similar to that of N_P but to compute the head reduct we evaluate rule bodies with respect to *4-valued interpretations*. These are functions assigning to propositional atoms, logical values **t** (true), **f** (false), **u** (unknown) and **i** (inconsistent). We represent 4-valued interpretations as pairs $(I_1, I_2) \in \mathcal{I} \times \mathcal{I}$.

Definition 7 *Satisfiability of atoms, ma-atoms, literals, and conjunctions of literals in 4-valued interpretations is defined as follows:*

$(I_1, I_2) \models A$ if $I_1 \models A$, where A is an atom or ma-atom

$(I_1, I_2) \models \neg A$ if $(I_2, I_1) \not\models A$, where A is an atom or ma-atom

$(I_1, I_2) \models L_1 \wedge L_2$ if $(I_1, I_2) \models L_1$ and $(I_1, I_2) \models L_2$, where L_1 and L_2 are literals (atoms, ma-atoms or their negations).

Definition 8 *The* extended head reduct *for a disjunctive program P is a function $E_P : \mathcal{I} \times \mathcal{I} \to \mathcal{P}(B^{\vee}(P))$ defined as:*

$$E_P(I_1, I_2) = \{head(r) \mid r \in P \text{ and } (I_1, I_2) \models body(r)\}$$

The second step in the definition of A_P is exactly the same as for N_P.

Definition 9 *Let P be a disjunctive program, Sel a selection function and $N_P^{Sel} : \mathcal{I} \to \mathcal{P}(\mathcal{I})$ a non-deterministic operator based on Sel. The* approximating operator *$A_P^{Sel} : \mathcal{I} \times \mathcal{I} \to \mathcal{P}(\mathcal{I})$ of P is defined as:*

$$A_P^{Sel}(I_1, I_2) = Sel(E_P(I_1, I_2)).$$

Again, whenever the selection function Sel is clear from the context or its choice is irrelevant, we simply write A_P instead of A_P^{Sel}. The operator A_P has similar properties and relationship with N_P as an approximating operator on the product lattice to the corresponding operator on the lattice, according to approximation theory (Denecker, Marek, & Truszczyński 2000). To state the appropriate result, we note that elements of $\mathcal{I} \times \mathcal{I}$ can be ordered by the *precision* ordering $\leq_p$. Namely, we write $(I_1, I_2) \leq_p (J_1, J_2)$ if

$$I_1 \subseteq I_2 \text{ and } J_2 \subseteq J_1.$$

We refer the reader to (Denecker, Marek, & Truszczyński 2000) for a detailed discussion of this ordering and its role in algebraic approaches to knowledge representation.

Proposition 7 *Let P be a disjunctive program.*

1. $E_P(I, I) = HR_P(I)$
2. A_P extends N_P, *i.e.*, $A_P(I, I) = N_P(I)$.
3. $(I_1, I_2) \leq_p (J_1, J_2)$ *implies* $E_P(I_1, I_2) \subseteq E_P(J_1, J_2)$
4. A_P *is monotone, that is,* $(I_1, I_2) \leq_p (J_1, J_2)$ *implies* $A_P(I_1, I_2) \preceq^S A_P(J_1, J_2)$.

For every interpretation $I \in \mathcal{I}$, the operator A_P induces a non-deterministic operator on $\mathcal{I}$, namely: $A_P(\cdot, I)$. This operator has the following properties.

Proposition 8 *Let P be a disjunctive program.*

1. *For every interpretation $I \in \mathcal{I}$, the operator $A_P(\cdot, I)$ is monotone, that is, $J \subseteq J'$ implies $A_P(J, I) \preceq^S A_P(J', I)$*
2. *For every interpretation $I \in \mathcal{I}$, the operator $A_P(\cdot, I)$ is downward closed*
3. *If $J \in mfp(A_P(\cdot, I))$, then $J \in MM(E_P(J, I))$.*

Proof: (Sketch) The first assertion follows directly from Proposition 7(4). The second assertion can be proved in exactly the same way as Lemma 5. To prove the third assertion, let us assume that $J \in mfp(A_P(\cdot, I))$. By the definition, J is a model of $E_P(J, I)$. Since $E_P(J, I)$ is a collection of standard propositional clauses, there is a minimal model J' of $E_P(J, I)$ such that $J' \subseteq J$. Since $E_P(J', I) \subseteq E_P(J, I)$ (Proposition 7(3)), J' is a model of $E_P(J', I)$. By the definition of a selection function, it follows that $A_P(J', I) \preceq^S \{J'\}$. In other words, J' is a pre-fixpoint of $A_P(\cdot, I)$. Since J is a minimal fixpoint of $A_P(\cdot, I)$, it is a minimal pre-fixpoint of $A_P(\cdot, I)$ (by Corollary 1, as $A_P(\cdot, I)$ is monotone and downward closed). Consequently, $J' = J$ and $J \in MM(E_P(J, I))$. □

We are now ready to extend the notion of stable models to the class of disjunctive programs with monotone aggregates.

Definition 10 *Let P be a disjunctive program. An interpretation I is a* stable model *of P if $I \in mfp(A_P(\cdot, I))$.*

The notion of a stable model is well defined as it does not depend on the selection function used to define the operator N_P. That property follows directly from our next result.

Proposition 9 *Let P be a disjunctive program and let Sel be a selection function. Then $mfp(A_P^{Sel}(\cdot, I)) = mfp(A_P^{MM}(\cdot, I))$.*

Proof: Let $I \in mfp(A_P^{Sel}(\cdot, I))$. Then $I \in MM(E_P(I, I))$ (Proposition 8). Consequently, $I \in A_P^{MM}(I, I)$. If $J \in A_P^{MM}(J, I)$, for some $J \subseteq I$, then $J \in A_P^{Sel}(J, I))$ (as, for every interpretation J, $A_P^{MM}(J, I) \subseteq A_P^{Sel}(J, I)$.). Since $I \in mfp(A_P^{Sel}(\cdot, I))$, $I = J$ and, consequently, $I \in mfp(A_P^{MM}(\cdot, I))$.

Conversely, if $I \in mfp(A_P^{MM}(\cdot, I))$, then $I \in A_P^{MM}(I, I)$. Thus, $I \in A_P^{Sel}(I, I)$. Let $J \subseteq I$ be such that $J \in mfp(A_P^{Sel}(\cdot, I))$. Such J exists as $A_P^{Sel}(\cdot, I)$ is downward closed and Lemma 2 applies. By Proposition 8(3), $J \in MM(E_P(J, I))$ and, consequently, $J \in A_P^{MM}(J, I)$. Since $I \in mfp(A_P^{MM}(\cdot, I))$, $J = I$ and $I \in mfp(A_P^{Sel}(J, I))$. □

The next theorem states that stable models are models of a disjunctive logic program P (the definition does not make that explicit). In fact, stable models are both minimal and supported models of P.

Theorem 5 *Let P be a disjunctive program. If I is a stable model of P then I is both supported and minimal model of P.*

Proof: Let $I \in mfp(A_P(\cdot, I))$. Then I is a fixpoint of $A_P^{MM}(\cdot, I)$ (as the definition of stable models does not depend on the choice of the selection function), that is, $I \in A_P^{MM}(I, I)$. By Proposition 7, $I \in N_P^{MM}(I)$. Thus, I is a fixpoint of N_P^{MM} and, by Theorem 2, a supported model of P.

Let us assume that J is a model of P such that $J \subseteq I$. One can show that J is a pre-fixpoint of $A_P(\cdot, I)$. By Corollary 1 (we recall that $A_P(\cdot, I)$ is monotone and downward closed), J contains a minimal pre-fixpoint J' of $A_P(\cdot, I)$. By Lemma 1, J' is also a fixpoint of $A_P(\cdot, I)$. Since $I \in mfp(A_P(\cdot, I))$, $J' = I$ and, consequently, $J = I$. Thus, I is a minimal model of P. □

When restricted to the appropriate classes of programs, our semantics coincides with the stable-model semantics of disjunctive logic programs (Gelfond & Lifschitz 1991; Przymusinski 1991) and exact stable models of aggregate programs without disjunction (Pelov, Denecker, & Bruynooghe 2004). Indeed, we have the following two results.

Theorem 6 *Let P be a disjunctive logic program without aggregates. Then M is a stable model of P (according to (Gelfond & Lifschitz 1991)) if and only if M is a stable model according to the definition of stable models of programs with aggregates outlined above.*

Theorem 7 *If P is an aggregate program without disjunction then M is a stable model of P (according to Definition 10) if and only if P is a stable model of P according to (Pelov, Denecker, & Bruynooghe 2004).*

We close this section by noting that our proposal for the stable-model semantics of disjunctive programs with aggregates differs from other approaches such as (Dell'Armi *et al.* 2003; Gelfond 2002) in that in our semantics stable models are always minimal (general version of Theorem 5).

Conclusions and Future Work

In the paper we proposed an operator-based approach to semantics of disjunctive logic programs with monotone aggregates. We described models, supported models, weakly supported models and stable models of disjunctive programs in terms of fixpoints of appropriate non-deterministic operators on the lattice of interpretations. For the most part, our proofs are algebraic, which opens a possibility for further generalizations.

We think that it is of considerable interest to extend the full framework of approximation theory of operators on lattices (Denecker, Marek, & Truszczyński 2000) to non-deterministic operators on lattices, and to provide a purely abstract algebraic characterization of the stable semantics of disjunctive logic programs. We believe that the results in this section are an important step in this direction. We identified several concepts and properties, which may prove useful.

First, we identified the *Smyth pre-order* as the basis for introducing the class of *monotone* non-deterministic operators. This does not come as a surprise considering that this order was originally introduced to model non-deterministic computations (Smyth 1978). Next, we showed that models of programs correspond to pre-fixpoints and supported models correspond to fixpoints of non-deterministic immediate-consequence operators, extending the corresponding property in normal logic programming (Apt, Blair, & Walker 1988). Finally, we identified a crucial class of downward closed operators and showed that all monotone non-deterministic operators that arise in our considerations are downward closed. That class of operators is important as every pre-fixpoint of a downward closed operator contains a minimal pre-fixpoint, a property we used several times in the paper. It is an interesting question to identify other simple conditions that imply that property.

Our definition of the approximating operator A_P differs from the abstract notion of approximating operator of (Denecker, Marek, & Truszczyński 2000) in two ways. First, we did not introduce an abstract algebraic definition of a non-deterministic approximating operator in the same way as (Denecker, Marek, & Truszczyński 2000). Instead, we gave a concrete definition of an approximating operator and then showed that it is monotone and extends the N_P operator. A more important difference with (Denecker, Marek, &

Truszczyński 2000) is that we approximate only the domain of the N_P operator and not its result. Consequently, we are able to characterize only the set of two-valued stable models of disjunctive logic programs and not the entire set of four and three-valued stable models (Przymusinski 1991). This is an important direction for future research.

Although our language is restricted only to monotone aggregate atoms, we believe that our results can be extended to disjunctive programs with arbitrary aggregate atoms, for example using the techniques and constructions of (Pelov, Denecker, & Bruynooghe 2004). This is another direction for future work.

Acknowledgment

The first author was supported by the GOA project LP+, the FWO project "Theory of Nonmonotone Inductive Definitions and its use for building Knowledge-based systems", the "Inductive Knowledge Bases" project and the Working group on Answer Set Programming. The second author was supported by the NSF grants IIS-0097278 and IIS-0325063.

References

Apt, K.; Blair, H.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of deductive databases and logic programming. Papers from the workshop held in Washington, D.C., August 18–22, 1986*, 89–142. Palo Alto, CA. Morgan Kaufmann.

Brass, S., and Dix, J. 1997. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming* 32(3):207–228.

Dell'Armi, T.; Faber, W.; Ielpa, G.; Leone, N.; and Pfeifer, G. 2003. Aggregate functions in DLV. In De Vos, M., and Provetti, A., eds., *Answer Set Programming: Advances in Theory and Implementation*, volume 78 of *CEUR Workshop proceedings*, 274–288. Online: CEUR-WS.org/Vol-78/.

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Minker, J., ed., *Logic-Based Artificial Intelligence*, 127–144. Kluwer Academic Publishers.

Denecker, M.; Pelov, N.; and Bruynooghe, M. 2001. Ultimate well-founded and stable semantics for logic programs with aggregates. In Codognet, P., ed., *Logic programming, Proceedings of the 2001 International Conference on Logic Programming*, volume 2237, 212–226. Springer.

Fitting, M. C. 2002. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science* 278:25–51.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, M. 2002. Representing knowledge in A-Prolog. In Kakas, A. C., and Sadri, F., eds., *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Computer Science*, 413–451. Springer.

Inoue, K., and Sakama, C. 1998. Negation as failure in the head. *Journal of Logic Programming* 35:39–78.

Marek, V.; Niemelä, I.; and Truszczyński, M. 2004. Characterizing stable models of logic programs with cardinality constraints. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 2923 of *Lecture Notes in Artificial Intelligence*, 154–166. Springer.

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2004. Partial stable semantics for logic programs with aggregates. In Lifschitz, V., and Niemelä, I., eds., *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 2923 of *Lecture Notes in Artificial Intelligence*, 207–219. Springer.

Przymusinski, T. 1991. Stable semantics for disjunctive programs. *New Generation Computing* 9:401–424.

Seipel, D.; Minker, J.; and Ruiz, C. 1997. Model generation and state generation for disjunctive logic programs. *Journal of Logic Programming* 32:49–69.

Smyth, M. 1978. Power domains. *Journal of Computer and System Sciences* 16:23–36.

van Emden, M., and Kowalski, R. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23(4):733–742.