INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians–
Universität
München

A Compositional Semantics for Logic Programs and Deductive Databases

François Bry

Short version in: Proc. Joint International Conference and Symposium on Logic Programming, MIT Press, 1996
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1996-4, August 1996

A Compositional Semantics for Logic Programs and Deductive Databases

François Bry

Abstract

Considering integrity constraints and program composition, it is first argued that a semantics for logic programs and deductive databases should not accommodate inconsistencies *globally* like in classical logic, but *locally.* It is then shown that minimal logic, a natural deduction style weakening of classical logic, is sufficient to provide for a proof theory for a generalization of logic programs corresponding to deductive databases with integrity constraints and disjunctive logic programs. Minimal logic differs from classical logic inasmuch as it precludes refutation proofs. Finally, a (nonclassical) model theory is proposed for generalized programs, which is based on a weakening of the usual notion of model inspired from (but not corresponding to) minimal logic. This model theory allows *local inconsistencies.* The proposed model theory naturally extends the minimal model, stable, and completion semantics of positive programs. The proposed semantics is shown to be "compositional" in the sense that it fulfils a natural program composition requirement. Arguably, it appropriately conveys a practician's intuition.

Keywords: logic programs, deductive databases, semantics, negation, nonmonotonic reasoning, paraconsistent logic.

1 Introduction

The semantics of positive and definite logic programs is inherited from classical first-order logic, the syntax of positive programs accounting for their semantical specificities. The procedural semantics of positive definite programs is defined in terms of SLD-resolution, a specialization to positive program clauses of the linear resolution proof procedure for first-order logic. Similarly, the declarative semantics of positive definite programs is defined in three equivalent manners, as a minimal Herbrand model, as the model of the program completion, or, inductively, as the least fixpoint of the immediate consequence operator, all three referring to the model theory of first-order logic.

The semantics of positive and definite logic programs does not easily extend to nonpositive programs, i.e. programs containing clauses with negative body literals. Although each of the above mentioned semantics can be quite naturally extended to general programs, the extensions are not equivalent. Negation as failure, for example, gives rise to an extension of SLD-resolution, but has no satisfying model theoretic counterparts. The stable model semantics [4] elegantly extends the nonconstructive notion of minimal model to general programs and the well-founded semantics [15] provides with a nice extension of the constructive fixpoint semantics, but the two semantics do not coincide on all programs. Applied to general programs, the completion semantics sometimes yields inconsistent theories. Several other semantics have been proposed, that suffer from similar drawbacks. An additional problem is what is called *the PhD effect* in [6]: A PhD in logic seems to be needed for understanding a semantics which adequately conveys the intuitive meaning of negation in logic programs! Thus, semantics have been proposed that, in spite of their theoretical interest, do not always convey a practician's common intuition.

In this article, the issue is reconsidered with a strong bias towards applications. Considering integrity constraints in relational and deductive databases on the one hand, and the composition of logic programs on the other hand, it is first argued that a convenient semantics for logic programs and deductive databases should not accommodate inconsistencies *globally,* like classical logic does, but should provide with a *local* notion of inconsistency.

It is then shown that a weakening of classical logic's proof theory, namely minimal logic, is sufficient to account for local inconsistencies in logic programs and deductive databases. Minimal logic is a natural deduction style weakening of classical logic proof theory, which does not allow for indirect – or refutation – proofs. Minimal logic is shown to provide with a sufficient proof theory for a generalization of positive logic programs corresponding to deductive databases with integrity constraints and disjunctive logic programs.

Finally, a weakening inspired from minimal logic of the usual notions of interpretation and model is introduced. In the proposed *weak interpretations,* a formula is true or false, like in classical interpretations, but can also be both, true and false, i.e. inconsistent. Weak models give rise to a simple and intuitive extension to general programs of the semantics of positive programs. The proposed semantics is formalized as well in terms of minimal (weak) models, and related to the program completion. First investigations indicate that an inductive, fixpoint-like definition of the proposed intended models is possible. Moreover, it is shown to be *compositional,* in the sense that it fulfils a natural program composition requirement. Thus, the semantics described in this article seems to avoid some of the drawbacks of other proposals.

The paper consists of 6 sections, the first of which is this introduction. In Section 2, examples are discussed and the approach is motivated. In Section 3 the terminology and notations are introduced. Section 4 is devoted to proof theory, while Section 5 introduces the nonclassical model theory.

Perspectives for further research are given in Section 6.

This paper is a revised, complete version of [2].

2 Motivation

By definition of interpretations and models, inconsistent theories have no models in classical logic. Therefore, every formula follows from an inconsistent theory. Entailment in classical logic is defined in model theoretic terms, indeed: A formula F follows from a theory $\mathcal{T}$ if and only if every model of $\mathcal{T}$ is also a model of F. This condition is trivially fulfilled if $\mathcal{T}$ is inconsistent, since it then has no models. This treatment of inconsistency gives rise to indirect – or refutation – proofs: A formula F can be proved to follow from a theory $\mathcal{T}$ by deriving an inconsistency from $\mathcal{T} \cup \{\neg F\}$.

Refutation proofs are often referred to in logic programming. SLD-resolution, for example, is usually defined as a refutation procedure: an answer substitution is computed by deriving the empty clause from a positive program and the negation of a query. However, refutation proofs sometimes contradict a database designer's and programmer's intuition, as the following examples show. In Section 3 it is shown that the procedural semantics of positive logic programs can be formalized without referring to refutation proofs.

2.1 Inconsistent Databases

While standard logic programs cannot be inconsistent, a database is inconsistent as soon as some of its integrity constraints are violated. Integrity constraints are closed formulas expressing properties that the database should always satisfy after some updates. They are conveniently represented relying on clauses whose heads are $\bot$. For example, the following closed formula

$$\forall x \quad (emp(x) \rightarrow \exists y \quad dpt(y) \wedge member(x, y))$$

("every employee works in a department") can be represented by the following clause, where $\neg$ means negation as failure and *in-a-dpt* is an auxiliary predicate symbol:

$$\begin{array}{c} \bot \leftarrow emp(x) \wedge \neg \ \textit{in-a-dpt}(x) \\ \textit{in-a-dpt}(x) \leftarrow dpt(y) \wedge member(x, y) \end{array}$$

It is a common practice to query an inconsistent database. For the lack of a convenient semantics, this practice could not be formalized. For example, one could query the inconsistent database $D = \{p(a), q(a), q(b), (\bot \leftarrow p(x) \wedge q(x))\}$ for determining which atoms could be discarded so as to restore consistency. In doing so, one would expect the query $p(a)$ to be positively answered, but the query $p(b)$ to be negatively answered, although a refutation proof gives rise to derive $p(b)$ from D.

Intuitively, database inconsistencies are considered to be local and not to affect query answering. Refutation proofs are, as far as querying inconsistent databases is concerned, undesirable, because they make inconsistencies global. Intuitively, an integrity constraint does not contribute to define tuples. A constraint such as $C := \perp \leftarrow \neg p(a)$ requires $p(a)$ to be true, but does not *define* it, although $p(a)$ follows from the constraint in classical logic.

2.2 Program Composition

As mentioned in the previous section, it is a common intuition that a program P defines an atom A only if there exists an instance $A \leftarrow B$ of a clause in P the body B of which is true. This intuition underlies the notion of supported definitions and of the completion semantics [3]. Also, it leads to define the semantics of disjunctive [8] and of call-consistent [11]) programs in terms of alternative models and not of logical entailment. The following program, e.g.,

$$P_1 = \{a \leftarrow \neg b \ ; \ b \leftarrow \neg a\}$$

is in general considered to specify the two intended models $\{a\}$ and $\{b\}$. With some (non-call-consistent) programs like

$$P_2 = \{a \leftarrow \neg a\}$$
$$P_3 = \{a \leftarrow \neg b \ ; \ b \leftarrow \neg c \ ; \ c \leftarrow \neg a\}$$

this intuition leads to difficulties, for their completions [3] are inconsistent.

While arguably the inconsistency of programs such as P_2 and P_3 is not counterintuitive, a semantics not fulfilling a "composition requirement" does not reflect a practician's intuition. This requirement can be informally specified as follows:

> *Composition requirement:* The meaning of a program P is not modified if P is extended with a program Q such that (1) none of the atoms defined in P are also defined in Q, and (2) no definitions in P depend on atoms defined in Q.

The completion semantics and the composition requirement are incompatible in classical logic. According to a "compositional" semantics, the atom b should be true in $P_4 = \{b\} \cup P_2$, while the atom c should not. If P_2 is considered an inconsistent specification, in classical logic both, b and c follow from P_4.

Although programs such as P_4 can be seen as inconsistent and therefore undesirable, a semantics under which inconsistencies are *local* is, arguably, desirable. While it is possible to investigate a mathematical theory only after it has been correctly axiomatized, partly verified programs need to be run for the very purpose of their verification. Like for every programming language, syntactical conditions in general are not sufficient to detect "incorrect" logic programs. Therefore, incorrectly axiomatized logic programs have to be given a semantics. Because of its *gobal* notion of inconsistency, classical logic is inadequate to convey the intuition of both, supported definitions and program composition.

3 Preliminaries

As usual, we shall consider first-order languages with denumerably many variables (so as not to bound the size of proofs) and with at least one constant (so as to ensure the nonemptiness of the Herbrand universe). Although not necessary, it is convenient to assume first-order languages to have denumerably many function and predicate symbols of every arity, so as to account for queries containing symbols not occurring in the program. In this article, first-order languages are assumed to be without equality, except when otherwise stated, and to have the following logical symbols: $\wedge$, $\vee$, $\rightarrow$, $\forall$, $\exists$, and $\perp$. The symbols $\top$, $\neg$, and $\leftrightarrow$ will be used for shorthand notations and are defined by $\top := \neg\perp$, $\neg F := (F \rightarrow \perp)$, and $F \leftrightarrow G := (F \rightarrow G) \wedge (G \rightarrow F)$. Complying with a logic programming usage $G \rightarrow F$ will also be written $F \leftarrow G$.

Atoms, literals, formulas, etc. are defined as usual. $\perp$ (read "false") denotes a special formula which is satisfied in no interpretations. $\perp$ is considered not to be an atom (so as not to modify the usual definitions of the Herbrand base). An atom A *occurs positively* (*negatively,* resp.) in $B_1 \vee ... \vee B_n$ ($B_1 \wedge ... \wedge B_n$, resp.) ($n \geq 1$) in case $A = B_i$ ($\neg A = B_i$, resp.) for some $i \in \{1, ..., n\}$. An atom A *occurs* in $B_1 \vee ... \vee B_n$ ($B_1 \wedge ... \wedge B_n$, resp.) if it occurs positively or negatively in $B_1 \vee ... \vee B_n$ ($B_1 \wedge ... \wedge B_n$, resp.).

The Herbrand universe $HU(\mathcal{L})$ and the Herbrand base $HB(\mathcal{L})$ of a first-order language $\mathcal{L}$ are defined as usual [7]. The *extended Herbrand base* $EHB(\mathcal{L})$ of a first-order language $\mathcal{L}$ is the set consisting of $\perp$ and all ground *literals* of $\mathcal{L}$.

A *clause* is an expression of the form $B_1 \wedge ... \wedge B_n \rightarrow H_1 \vee ... \vee H_m$ with $n \geq 0$, $m \geq 0$, $m \neq 0$ if $n = 0$, and where the B_i are literals and the H_j are atoms. $B_1 \wedge ... \wedge B_n$ is the *body* of the clause, $H_1 \vee ... \vee H_m$ its *head.* For $n = 0$ the body of the clause is, by convention, defined as $\top$. For $m = 0$, the head of the clause is, by convention, defined as $\perp$. It is assumed that that $\perp$ and $\top \rightarrow \perp$ are not clauses. A clause $B_1 \wedge ... \wedge B_n \rightarrow H_1 \vee ... \vee H_m$ denotes the formula $\forall^*(B_1 \wedge ... \wedge B_n \rightarrow H_1 \vee ... \vee H_m)$ where $\forall^*(F)$ denotes the universal closure of F. A clause whose head is $\perp$ is also called a *denial.* A clause $B_1 \wedge ... \wedge B_n \rightarrow H_1 \vee ... \vee H_m$ is called *positive,* if all B_i ($1 \leq i \leq n$) are positive literals, *definite,* if $m = 1$, and *disjunctive,* if $m \geq 2$.

A *program* is a finite set of clauses. An *unconstrained program* is a program containing no denials. A *constrained program* is a program containing at least one denial. A *positive program* (*definite program,* resp.) is a program containing only positive (definite, resp.) clauses. A *disjunctive* program is a program containing at least one disjunctive clause.

4 Proof Theory

In this section, we briefly recall a system of natural deduction for classical logic as introduced in [5]. For comprehensive introductions, see also e.g. [10] or Chapter 2, Sections 1 and 3, of [14]. A first-order language $\mathcal{L}$ is assumed.

4.1 Natural Deduction

A *deduction* $\mathcal{D}^F$ of a formula F in a theory $\mathcal{T}$ is a tree, the nodes of which are occurrences of formulas. The root of $\mathcal{D}^F$ is an occurrence of F. The leaves of $\mathcal{D}^F$ are occurrences of formulas in $\mathcal{T}$, or are *open assumptions.* Open assumptions are formula occurrences that can, later in the proof, be *discharged* by applications of certain inference rules. Formula occurrences are distinguished from formulas, because an application of an inference rule discharges one formula occurrence, but not all open assumptions in the already expanded deduction that are occurrences of the same formula.[1] Deductions $\mathcal{D}^F$ of formulas F in a theory $\mathcal{T}$ are inductively defined together with the set $O(\mathcal{D}^F)$ of their open assumptions as follows:

1. If $F \notin \mathcal{T}$, a one node tree $\mathcal{D}^F$ consisting of an occurrence of F is a deduction of F in $\mathcal{T}$, the only open assumption of which is the occurrence $\mathcal{D}^F$ of F itself.

2. If $F \in \mathcal{T}$, a one node tree $\mathcal{D}^F$ consisting of an occurrence of F is a deduction of F in $\mathcal{T}$ with no open assumptions.

3. For $k = 1, ..., n$ let $\mathcal{D}^{F_k}$ be a deduction of F_k in $\mathcal{T}$ with set of open assumptions $O(\mathcal{D}^{F_k})$. $\mathcal{D}^F := \dfrac{\mathcal{D}^{F_1} \; ... \; \mathcal{D}^{F_n}}{F}$ is a deduction of F in $\mathcal{T}$ if

   $\dfrac{F_1 \; ... \; F_n}{F}$ is an application of one of the inference rules below, the open assumptions of which are the formula occurrences in the sets $O(\mathcal{D}^{F_k})$ $(k = 1, ..., n)$ except, possibly, for discharged formula occurrences. A discharged occurrence of a formula F is indicated in the inference rules below by (F).

[1] In technical terms, the "crude discharge convention" is not assumed.

Introduction rules: *Elimination rules:*

$$\wedge\mathrm{I} \quad \frac{A \qquad B}{A \wedge B} \qquad\qquad \vee\mathrm{E} \quad \frac{A \vee B \qquad \begin{matrix}(A)\\ C\end{matrix} \qquad \begin{matrix}(B)\\ C\end{matrix}}{C}$$

$$\vee\mathrm{I}_r \quad \frac{A}{A \vee B} \qquad\qquad \wedge\mathrm{E}_r \quad \frac{A \wedge B}{A}$$

$$\vee\mathrm{I}_l \quad \frac{B}{A \vee B} \qquad\qquad \wedge\mathrm{E}_l \quad \frac{A \wedge B}{B}$$

$$\rightarrow\mathrm{I} \quad \frac{\begin{matrix}(A)\\ B\end{matrix}}{A \rightarrow B} \qquad\qquad \rightarrow\mathrm{E} \quad \frac{A \qquad A \rightarrow B}{B}$$

$$\forall\mathrm{I} \quad \frac{A}{\forall y A[y/x]} \qquad\qquad \forall\mathrm{E} \quad \frac{\forall x A}{A[t/x]}$$

$$\exists\mathrm{I} \quad \frac{A[t/x]}{\exists x A} \qquad\qquad \exists\mathrm{E} \quad \frac{\exists y A[y/x] \qquad \begin{matrix}(A)\\ B\end{matrix}}{B}$$

Absurdity rule:

$$\perp_c \quad \frac{\begin{matrix}(\neg A)\\ \perp\end{matrix}}{A}$$

Condition on the rule $\forall$I*:* y must not occur free in any open assumption on which A depends, i.e. y must not occur free in any formula occurrence in $\mathcal{O}^A \cup O(\mathcal{D}^B)$.

Condition on the rule $\exists$E*:* y must not occur free in $\exists x A$, or in B, or in any assumption on which the upper occurrence of B depends other than A, i,e. y must not occur free in any formula occurrence in $\{\exists x A, B\} \cup O(\mathcal{D}^B)$.

It is easy to show that applications of the rule $\perp_c$ can be restricted to those cases where $A \neq \perp$. This restriction is assumed in the sequel. The correctness of the inference rules given above can easily be shown to follow from the definition of a model.

If $\mathcal{D}^F$ is a deduction of a formula F in a theory $\mathcal{T}$ with no open assumptions, i.e. if $O(\mathcal{D}^F) = \emptyset$, then it is a *proof* of F in $\mathcal{T}$, the existence of which is, as usual, denoted by $\mathcal{T} \vdash F$.

4.2 Classical and Minimal Logic

Classical and minimal logic are simple to compare: For deductions in classical logic, all the inference rules can be applied. For deductions in minimal logic, only the introduction and elimination rules can be applied, applications of the absurdity rule $\bot_c$ are precluded. Let $\vdash_c$ and $\vdash_m$ denote provability in classical and minimal logic, respectively.

Classical logic provability can also be defined by replacing the rule $\bot_c$ with the elimination of double negation rule $\dfrac{\neg\neg A}{A}$ or, alternatively, as minimal logic provability from a (infinite) theory consisting of stability axioms $\neg\neg A \rightarrow A$ for every formula A, as shown by the following classical logic deduction (the numbers indicates the rule application by which a formula occurrence is discharged, recall that $\neg A$ stands for $A \rightarrow \bot$):

$$\dfrac{\dfrac{\dfrac{(2)\ \neg A \qquad\qquad (1)\ \neg\neg A}{\bot}\ \rightarrow \mathrm{E}}{A}\ \bot_c\ (2)}{\neg\neg A \rightarrow A}\ \rightarrow \mathrm{I}\ (1)$$

The absurdity rule $\bot_c$ "globalizes" inconsistencies, in the sense that it makes it possible to derive every formula in an inconsistent theory ("ex falso quodlibet" principle), i.e. $\vdash_c \bot \rightarrow A$, as follows from the following deduction of $\{\neg\neg A \rightarrow A\} \vdash_m \bot \rightarrow A$:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}(2)\ \neg A \\ (1)\ \bot\end{matrix}}{\neg\neg A}\ \rightarrow \mathrm{I}\ (2) \qquad\qquad \neg\neg A \rightarrow A}{A}\ \rightarrow \mathrm{E}}{\bot \rightarrow A}\ \rightarrow \mathrm{I}\ (1)$$

Since minimal logic does not treat the formula $\bot$ differently from an atom, $\nvdash_m \bot \rightarrow A$. [*Proof:* Otherwise, for every atom B, $\vdash_m B \rightarrow A$ would hold. Let A and B be two atoms. Since there are interpretations that do not satisfy $B \rightarrow A$, this would contradict the correctness of the inference rules.] Thus, inconsistencies are "local" in minimal logic because it does not satisfy the "ex falso quodlibet" principle.[2]

Since no absurdity rule is allowed for deduction in minimal logic, minimal logic is not complete for first-order logic. [*Proof:* If $\mathcal{I}$ is an interpretation and A an atom, then $\mathcal{I} \models \neg\neg A \rightarrow A$. Since minimal logic treats $\bot$ like

[2]Like classical logic intuitionistic logic [10] "globalizes" inconsistencies, for it has the "ex falso quodlibet" principle as a deduction rule.

any atom B, if $\vdash_m \neg\neg A \rightarrow A$, then $\vdash_m ((A \rightarrow B) \rightarrow B) \rightarrow A$. This is incorrect, because there are interpretations $\mathcal{I}$ such that $\mathcal{I} \not\models A$ and $\mathcal{I} \models B$, i.e. $\mathcal{I} \not\models ((A \rightarrow B) \rightarrow B) \rightarrow A)$. The result follows from the correctness of the inference rules.]

Although incomplete for (full) first-order logic, minimal logic is proven below to be complete a proof theory for positive unconstrained programs.

4.3 Completeness of Minimal Logic for Positive Unconstrained Programs

Some deductions might contain useless so-called detours, e.g.:

$$\dfrac{\dfrac{\mathcal{D}^A \qquad\qquad \mathcal{D}^B}{A \wedge B}\ \wedge\mathrm{I}}{A}\ \wedge\mathrm{E}_r$$

Deduction without detours, called *normal deductions,* have been formalized [10]. For the purpose of this article, their definition is not needed. The proofs will only refer to the fact that a deduction occurring within a normal deduction is normal, and to the following two theorems.

Theorem 1 (Normal Deduction Theorem) *If $\mathcal{T} \vdash_c F$, then there is a (classical logic) normal deduction of F in $\mathcal{T}$.*

A proof of the Normal Deduction Theorem is given in [10] for restricted languages in which $\vee$ and $\exists$ are expressed in terms of $\wedge$, $\forall$, and $\neg$. Although this proof does not extend to the general syntax, the validity of the Normal Deduction Theorem in the general case, i.e. of Theorem 1, is mentioned in [10].

Normal deduction have an interesting structure. In particular they satisfy a subformula property referring to the following inductive definition:

1. F is a subformula of F.
2. If $A \wedge B$, $A \vee B$, or $A \rightarrow B$ is a subformula of F, then so are also A and B.
3. If $\forall x A$ or $\exists x A$ is a subformula of F, then so is also $A[t/x]$.

Theorem 2 (Subformula Property) *Every formula occurring in a (classical logic) normal deduction of F in $\mathcal{T}$ is a subformula of F or of some formula of $\mathcal{T}$, except for assumptions discharged by applications of the rule $\perp_c$ and for occurrences of $\perp$ that stand immediately below such assumptions.*

A simpler form of the Subformula Property is proved in [10] for (unrestricted syntax) intuitionistic and minimal logic. This proof directly extends to classical logic, provided the proviso "except for assumptions ..." is made.

Given a program P we shall abuse the notation and also denote by P the set of formulas consisting of the universal closures of the clauses of P. Moreover, it will always be assumed that a same variable does not occur in distinct clauses of a program.

Lemma 3 *Let P be a positive program. If $P \vdash_c \bot$, then $P \vdash_m \bot$.*

Proof: If $P \vdash_c \bot$, then by the Normal Deduction Theorem there is a normal deduction of $\bot$ in P. Let $\mathcal{D}^\bot$ be a normal deduction of $\bot$ in P containing an application of the rule $\bot_c$. Let $\mathcal{A}^G := \dfrac{\begin{matrix}(\neg G)\\ \bot\end{matrix}}{G}\,\bot_c$ be a subdeduction of $\mathcal{D}^\bot$ containing no other application of $\bot_c$. We show how to transform $\mathcal{A}^G$ into a derivation of G in P containing no applications of $\bot_c$. The lemma then follows by induction.

Case 1: If G is an instance of a clause in P, then $\mathcal{A}^G$ can be replaced by a sequence of applications of the rule $\forall$E to an occurrence of this clause.

Case 2: Else. By the Subformula Property, G is an atom, or a conjunction of atoms, or a disjunction of atoms. (G cannot be $\bot$, since the rule $\bot_c$ has been assumed in Section 4.1 never to discharge $\neg\bot$.) Since $\bot \notin P$ and since $\mathcal{A}^G$ contains no other applications of $\bot_c$, there are a formula B and a normal derivation $\mathcal{D}^B$ of B in P such that $\mathcal{A}^G$ has the shape:

$$\dfrac{\dfrac{\mathcal{D}^B \qquad \begin{matrix}(\neg G)\\ \mathcal{D}^{B\to\bot}\end{matrix}}{\bot}\,\to\mathrm{E}}{G}\,\bot_c$$

By the Subformula Property, B is a subformula of a clause in P or of $\neg G = G \to \bot$.

Case 2.1: If B is an instance of a clause in P, then $\mathcal{D}^{B\to\bot}$ is a deduction of $\bot$ in P which contains by hypothesis no applications of $\bot_c$, i.e. $P \vdash_m \bot$.

Case 2.2: Else. Since $B \to \bot$ is not a subformula of an instance of a clause in P, by the Subformula Property it is a subformula of $G \to \bot$. Therefore, either $B = G$, or $B \to \bot$ is a subformula of G. Since G is an atom, or a conjunction of atoms, or a disjunction of atoms, $B = G$. $\mathcal{A}^G$ therefore has the shape:

$$\dfrac{\dfrac{\mathcal{D}^G \qquad \begin{array}{c}(\neg G)\\ \mathcal{D}^{G\to\perp}\end{array}}{\perp} \to \mathrm{E}}{G} \perp_c$$

$\mathcal{A}^G$ can be replaced in $\mathcal{D}^\perp$ by the derivation $\mathcal{D}^G$ which, by hypothesis, contains no application of $\perp_c$. ∎

Theorem 4 (Completeness of Minimal Logic for Positive Unconstrained Programs) *Let P be a positive unconstrained program and F be an atom, or a conjunction of atoms, or a disjunction of atoms. If $P \vdash_c F$, then $P \vdash_m F$.*

Proof: Let $\mathcal{D}^F$ be a normal derivation of F in P containing an application of $\perp_c$. Let $\mathcal{A}^G := \dfrac{\begin{array}{c}(\neg G)\\ \mathcal{D}^\perp\end{array}}{G} \perp_c$ be a subdeduction of $\mathcal{D}^F$ containing exactly one application of $\perp_c$. Since P is an unconstrained program, it contains no denials. Hence, $\mathcal{A}^G = \dfrac{\dfrac{\mathcal{E}^G \qquad (G \to \perp)}{\perp} \perp_c}{G} \to \mathrm{E}$. By hypothesis, $\mathcal{E}^G$ contains no applications of $\perp_c$. $\mathcal{A}^G$ can therefore be replaced in $\mathcal{D}^F$ by $\mathcal{E}^G$. The result follows by induction. ∎

Theorem 4 is established in [13] for definite – i.e. non-disjunctive – positive unconstrained programs. Related results are given in [12, 9].

4.4 Procedural Semantics

The following Theorem, the proof of which is easy [1], shows that minimal logic derivations and SLD-resolution proofs of atoms and conjunctions of atoms from positive, definite, and unconstrained programs are, up to the shape, identical. Note that a successful *branch* of an SLD-resolution tree corresponds to a minimal logic derivation, i.e. a *tree.*

Theorem 5 (Isomorphy Theorem) *Let P be a positive, definite, and unconstrained program, and F an atom or a conjunction of atoms. There exists two transformations ϕ and ψ such that:*

1. *ϕ maps a minimal logic derivation of F in P into a SLD-resolution proof of F from P. ψ maps a SLD-resolution proof of F from P into a minimal logic derivation of F in P.*

2. *ϕ and ψ preserve subdeductions.*

3. *If $\mathcal{D}$ is a minimal logic deduction of F in P, then $\psi(\phi(\mathcal{D})) = \mathcal{D}$. If $\mathcal{P}$ is a SLD-resolution proof of F from P, then $\phi(\psi(\mathcal{P})) = \mathcal{P}$.*

By Theorems 4 and 5, minimal logic and SLD-resolution are two equivalent definitions of the procedural semantics of positive, definite, and unconstrained programs. By Theorem 5, any search strategy defined in terms of the one proof method is applicable to the other. Minimal logic is however more general than SLD-resolution since by Theorem 4 it provides with a procedural semantics for disjunctive and constrained programs as well.

Like SLD-resolution, minimal logic can be extended with various forms of "negation as failure" [3] depending on the termination properties resulting from the adopted search strategy. Since the "ex falso quodlibet" principle does not hold in minimal logic, this calculus, possibly extended with a form of negation as failure, is arguably more convenient a basis than classical logic linear resolution for defining the procedural semantics of logic programs and deductive databases.

5 Model Theory

Though convenient a proof theory for logic programs and deductive databases, minimal logic has no model theory. Indeed, as shown in Section 4.2, minimal logic is an incomplete proof calculus. In this section, we propose a notion of *weak interpretation* as a counterpart to classical logic's notion of interpretation. It departs from the classical notion inasmuch as it makes *local* inconsistencies possible. Although not faithful to minimal logic, it is appropriate for logic programs and databases.

Except when otherwise stated, we consider in the sequel a fixed first-order language $\mathcal{L}$ in which the programs are defined.

5.1 Weak Models

In this Section, weak interpretations and weak models are defined through slight changes in the classical definitions (see, e.g. [7]): since in weak interpretations the truth of a formula does not exclude that of its negation, assignments of both truth values **t** (true) and **f** (false) are specified in the following definition.

Definition 6 (Weak Interpretation) *A* weak interpretation *$\mathcal{I}$ of the language $\mathcal{L}$ consists of a non-empty set D, called the domain of $\mathcal{I}$, and of a mapping defined by the following assignments:*

1. *For each constant c in $\mathcal{L}$, the assignment of an element $c^{\mathcal{I}}$ in D.*

2. *For each n-ary ($n \geq 1$) function symbol f in $\mathcal{L}$, the assignment of a mapping $f^{\mathcal{I}} : D^n \rightarrow D$.*

3. *The assignment of two truth values $\perp^{\mathcal{I}}_{pos} \in \{\mathbf{t}, \mathbf{f}\}$ and $\perp^{\mathcal{I}}_{neg} = \mathbf{t}$ to $\perp$ (possibly $\perp^{\mathcal{I}}_{pos} = \mathbf{t}$).*

4. *For each predicate symbol p of arity 0 in $\mathcal{L}$, the assignment of two truth values $p^{\mathcal{I}}_{pos} \in \{\mathbf{t}, \mathbf{f}\}$ and $p^{\mathcal{I}}_{neg} \in \{\mathbf{t}, \mathbf{f}\}$ such that $\mathbf{t} \in \{p^{\mathcal{I}}_{pos}, p^{\mathcal{I}}_{neg}\}$ (possibly $p^{\mathcal{I}}_{pos} = p^{\mathcal{I}}_{neg} = \mathbf{t}$).*

5. *For each n-ary ($n \geq 1$) predicate symbol p in $\mathcal{L}$, the assignment of two n-ary relations $p^{\mathcal{I}}_{pos}$ and $p^{\mathcal{I}}_{neg}$ over D such that $p^{\mathcal{I}}_{pos} \cup p^{\mathcal{I}}_{neg} = D^n$ (possibly $p^{\mathcal{I}}_{pos} \cap p^{\mathcal{I}}_{neg} \neq \emptyset$).*

A weak interpretation $\mathcal{I}$ of $\mathcal{L}$ is consistent *if and only if:*

1. *$\perp^{\mathcal{I}}_{pos} = \mathbf{f}$, and*

2. *for every predicate symbol p of arity 0 in $\mathcal{L}$, $p^{\mathcal{I}}_{pos} \neq p^{\mathcal{I}}_{neg}$, and*

3. *for every n-ary ($n \geq 1$) predicate symbol p in $\mathcal{L}$, $p^{\mathcal{I}}_{pos} \cap p^{\mathcal{I}}_{neg} = \emptyset$.*

It is inconsistent *otherwise.*

Clearly, consistent weak interpretations specify classical interpretations. We recall that variable assignments into a domain D map variables of $\mathcal{L}$ to elements of D. A term assignment with respect to a weak interpretation $\mathcal{I}$ and variable assignment $\mathcal{V}$ is inductively defined by $f(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} := f^{\mathcal{I}}(t_1{}^{\mathcal{I}}_{\mathcal{V}}, ..., t_n{}^{\mathcal{I}}_{\mathcal{V}})$, for every n-ary function symbol f and terms $t_1, ..., t_n$. The truth of a formula F in a weak interpretation $\mathcal{I}$ with respect to a variable assignment $\mathcal{V}$ will be denoted $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. Since the truth of a formula F does not preclude the truth of its negation, both, the truth of F and of $\neg F$ have to be specified.

Definition 7 (Satisfaction in Weak Interpretations) *Let $\mathcal{I}$ be a weak interpretation, $\mathcal{V}$ a variable assignment, F and G formulas, H a formula different from $\perp$, p a predicate symbol of arity 0, q a predicate symbol of arity $n \geq 1$, and $t_1, ..., t_n$, n terms.*

1. *$\perp^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ if and only if $\perp^{\mathcal{I}}_{pos} = \mathbf{t}$.*
   $\neg\perp^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

2. *$p^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ if and only if $p^{\mathcal{I}}_{pos} = \mathbf{t}$.*
   $\neg p^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ if and only if $p^{\mathcal{I}}_{neg} = \mathbf{t}$.

3. $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} \in q^{\mathcal{I}}_{pos}$.
   $\neg q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} \in p^{\mathcal{I}}_{neg}$.

4. $(F \wedge G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
   $\neg(F \wedge G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *or* $\neg G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

5. $(F \vee G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *or* $G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
   $\neg(F \vee G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $\neg G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

6. $(F \rightarrow H)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$, *then* $H^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
   $\neg(F \rightarrow H)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $\neg H^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

7. $(\forall x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if for all variable assignments* $\mathcal{W}$ *that coincide with* $\mathcal{V}$ *on every variable except possibly on* x, $F^{\mathcal{I}}_{\mathcal{W}} = \mathbf{t}$.
   $\neg(\forall x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $(\exists x \neg F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

8. $(\exists x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if for some variable assignment* $\mathcal{W}$ *that coincide with* $\mathcal{V}$ *on every variable except possibly on* x, $F^{\mathcal{I}}_{\mathcal{W}} = \mathbf{t}$.
   $\neg(\exists x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *if and only if* $(\forall x \neg F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

9. $\neg\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

Note that Definition 7 directly defines the truth of negated formulas $\neg F$ without referring to the definition $\neg F := F \rightarrow \perp$. Note also that for all weak interpretations $\mathcal{I}$ and all variable assignments $\mathcal{V}$, $\neg\perp^{\mathcal{I}}_{\mathcal{V}} = \top^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ and for every formula F, $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ or $\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ and possibly $F^{\mathcal{I}}_{\mathcal{V}} = \neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. Since $\neg\neg F = \neg(F \rightarrow \perp)$, it follows from Definition 7 that, for all weak interpretations $\mathcal{I}$ and variable assignments $\mathcal{V}$, $\neg\neg F^{\mathcal{I}}_{\mathcal{V}} = \neg(F \rightarrow \perp)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ if and only if $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ and $\neg\perp^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$, i.e. if and only if $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. Note also that, for all *inconsistent* weak interpretations $\mathcal{I}$ and all variable assignments $\mathcal{V}$, $\perp^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. Indeed, if $\mathcal{I}$ is inconsistent, then there exists a formula F such that $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ and $\neg F^{\mathcal{I}}_{\mathcal{V}} = (F \rightarrow \perp)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. The truth of $\perp$ in a weak interpretation can therefore be seen as a witness of inconsistency.

Definition 8 *Let* F *be a closed formula and* $\mathcal{S}$ *a set of closed formulas.*

- F *is* true in $\mathcal{I}$, *denoted* $F^{\mathcal{I}}$:$\mathbf{t}$, *if and only if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *for some variable assignment* $\mathcal{V}$.
- F *is* false in $\mathcal{I}$, *denoted* $F^{\mathcal{I}}$:$\mathbf{f}$, *if and only if* $\neg F^{\mathcal{I}}$:$\mathbf{t}$.
- F *is* inconsistent in $\mathcal{I}$ *if and only if* F *is both, true and false in* $\mathcal{I}$. *Otherwise,* F *is* consistent in $\mathcal{I}$.

- F *is* consistently true *(*consistently false*, resp.)* in $\mathcal{I}$*, if and only if* F *is true (false, resp.) and consistent in* $\mathcal{I}$*.*

- *A weak interpretation* $\mathcal{I}$ satisfies F *(*$\mathcal{S}$*, resp.) if and only if* F *(every formula in* $\mathcal{S}$*, resp.) is true in* $\mathcal{I}$*. In this case,* $\mathcal{I}$ *is called a* weak model *of* F *(*$\mathcal{S}$*, resp.).*

- *A weak interpretation* $\mathcal{I}$ consistently satisfies F *(*$\mathcal{S}$*, resp.) if and only if* F *(every formula in* $\mathcal{S}$*, resp.) is consistently true in* $\mathcal{I}$*.*

Note that $\top = \neg\bot$ ($\bot$, resp.) is true in every weak interpretation. (inconsistent weak interpretation, resp.). The following Proposition shows that clauses are interpreted in weak interpretations according to a logic programmer's intuition.

Proposition 9 *Let* $\mathcal{I}$ *be a weak interpretation,* A*,* B *and* C *closed formulas, and* $H \leftarrow D$ *a ground clause.*

1. $(A \rightarrow B \vee C)^{\mathcal{I}}$**:t** *if and only if* $(\ (A \rightarrow B) \vee (A \rightarrow C)\ )^{\mathcal{I}}$**:t***.*

2. $A \vee B$ *is consistently true in* $\mathcal{I}$ *if and only if at least one of* A *and* B *is consistently true in* $\mathcal{I}$*.*

3. $A \vee B$ *is consistently false in* $\mathcal{I}$ *if and only if both,* A *and* B *are consistently false in* $\mathcal{I}$*.*

4. $A \wedge B$ *is consistently true in* $\mathcal{I}$ *if and only if both,* A *and* B *are consistently true in* $\mathcal{I}$*.*

5. $A \wedge B$ *is consistently false in* $\mathcal{I}$ *if and only if at least one of* A *and* B *is consistently false in* $\mathcal{I}$*.*

6. *If* $(H \leftarrow D)^{\mathcal{I}}$**:t** *and if* H *is consistently false in* $\mathcal{I}$*, then* D *is consistently false in* $\mathcal{I}$*.*

Proof: 1. By Definition 7, $(A \rightarrow B \vee C)^{\mathcal{I}}$**:t** if and only if if $A^{\mathcal{I}}$**:t**, then $(B \vee C)^{\mathcal{I}}$**:t** if and only if if $A^{\mathcal{I}}$**:t**, then $B^{\mathcal{I}}$**:t** or $C^{\mathcal{I}}$**:t** if and only if $(A \rightarrow B)^{\mathcal{I}}$**:t** or $(A \rightarrow C)^{\mathcal{I}}$**:t**. $2-5$. By the truth table for disjunctions and conjunctions. 6. By Definition 7 $(H \leftarrow D)^{\mathcal{I}}$**:t** if and only if if $D^{\mathcal{I}}$**:t**, then $H^{\mathcal{I}}$**:t** if and only if if not $H^{\mathcal{I}}$**:t**, then not $D^{\mathcal{I}}$**:t** if and only if if H is consistently false in $\mathcal{I}$, then D is consistently false in $\mathcal{I}$. ∎

Two equivalent representations of the truth tables for disjunctions and conjunctions are given in Figure 1. The third truth table is a more compact version of the first two. In this truth table in compact form, **i** stands for "inconsistent", **ct** for "consistently true", and **cf** for "consistently false".

A	$\neg A$	B	$\neg B$	$A \vee B$	$\neg(A \vee B)$ $\neg A \wedge \neg B$	$A \wedge B$	$\neg(A \wedge B)$ $\neg A \vee \neg B$
t	**t**	**t**	**t**	**t**	**t**	**t**	**t**
t	**t**	**t**	–	**t**	–	**t**	**t**
t	**t**	–	**t**	**t**	**t**	–	**t**
t	–	**t**	**t**	**t**	–	**t**	**t**
t	–	**t**	–	**t**	–	**t**	–
t	–	–	**t**	**t**	–	–	**t**
–	**t**	**t**	**t**	**t**	**t**	–	**t**
–	**t**	**t**	–	**t**	–	–	**t**
–	**t**	–	**t**	–	**t**	–	**t**

A	$\neg A$	B	$\neg B$	$A \rightarrow B$	$\neg(A \rightarrow B)$ $A \wedge \neg B$
t	**t**	**t**	**t**	**t**	**t**
t	**t**	**t**	–	**t**	–
t	**t**	–	**t**	–	**t**
t	–	**t**	**t**	**t**	**t**
t	–	**t**	–	**t**	–
t	–	–	**t**	–	**t**
–	**t**	**t**	**t**	**t**	–
–	**t**	**t**	–	**t**	–
–	**t**	–	**t**	**t**	–

A	B	$A \vee B$	$A \wedge B$	$A \rightarrow B$
i	**i**	**i**	**i**	**i**
i	**ct**	**ct**	**i**	**ct**
i	**cf**	**i**	**cf**	**cf**
ct	**i**	**ct**	**i**	**i**
ct	**ct**	**ct**	**ct**	**ct**
ct	**cf**	**ct**	**cf**	**cf**
cf	**i**	**i**	**cf**	**ct**
cf	**ct**	**ct**	**cf**	**ct**
cf	**cf**	**cf**	**cf**	**ct**

Figure 1: Two representations of the truth table for disjunctions, conjunctions and implications.

The truth tables have 9 lines only, since by Definition 6, at least one of a formula and its negation is true.

Note that $A \rightarrow B$ und $\neg A \vee B$ do not have the same truth values: If A is inconsistent and B consistently false, then $\neg A \vee B$ is true while $A \rightarrow B$ is consistently false.

Example 1 In the following examples, a weak interpretation $\mathcal{I}$ is specified by the elements of $EHB(\mathcal{L})$ that are consistently true in $\mathcal{I}$ or inconsistent in $\mathcal{I}$, without explicitly listing the elements of $EHB(\mathcal{L})$ that are consistently false in $\mathcal{I}$.

1. $\neg\neg p = \perp \leftarrow \neg p$ is satisfied by a weak interpretation specified by $\{p, \neg p, \perp\}$ or $\{p\}$.

2. $\{a \leftarrow \neg b \ ; \ b \leftarrow \neg a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \neg b \perp\}$, or $\{a, \neg b\}$, or $\{\neg a, b\}$, or $\{a, b\}$.

3. $\{a \leftarrow \neg b \ ; \ b \leftarrow \neg c \ ; \ c \leftarrow \neg a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \neg b, c, \neg c, \perp\}$, or $\{a, b, \neg b, c, \neg c, \perp\}$, or $\{a, b, c, \neg c, \perp\}$, or $\{a, b, c\}$, or $\{a, b\}$.

4. $\{\perp \leftarrow b \ ; \ b \leftarrow a \ ; \ a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \neg b, \perp\}$ or $\{a, b, \neg b, \perp\}$.

5. $\{b \leftarrow a \ ; \ \perp \leftarrow a \ ; \ a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \perp\}$ or $\{a, \neg a, b, \neg b, \perp\}$.

6. $\{a \leftarrow b \ ; \ b \leftarrow a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \neg b, \perp\}$, or $\{a, b, \neg b, \perp\}$, or $\{a, b\}$, or $\emptyset$.

7. $\{a \vee b \ ; \ \perp \leftarrow a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, \perp\}$ or $\{b\}$.

8. $\{a \leftarrow b \ ; \ b \leftarrow a \ ; \ c \leftarrow \neg a\}$ is satisfied by a weak interpretation specified by $\{a, \neg a, b, \neg b, c, \neg c, \perp\}$, or $\{a, \neg a, b, \neg b, c, \perp\}$, or $\{a, \neg a, b, c, \perp\}$, or $\{a, b, c\}$, or $\{a, b\}$, or $\{c\}$.

9. $\{p(x) \leftarrow p(f(x)) \ ; \ q(a) \leftarrow \neg p(x)\}$ is satisfied by a weak interpretation specified by $\{p(a), p(f(a), ..., p(f^n(a)), ...\}$ or $\{q(a)\}$.

Note that a weak interpretation satisfying all ground literals as well as the formula $\perp$, i.e. a weak interpretation satisfying $EHB(\mathcal{L})$, satisfies every formula.

5.2 Weak Herbrand Models

A weak interpretation (weak model, resp.) the domain of which is the Herbrand universe of $\mathcal{L}$ and which interprets ground terms by themselves will be called a *weak Herbrand interpretation* (*weak Herbrand model*, resp.). Like a (classical) Herbrand interpretation is characterized by the set of ground atoms it satisfies, a weak Herbrand interpretation $\mathcal{I}$ can be characterized by both, the set of ground atoms that are consistently true in $\mathcal{I}$, and the set of ground atoms that are inconsistent in $\mathcal{I}$.

Definition 10 *Let $S \subseteq EHB(\mathcal{L})$. Let A be a ground atom or $\perp$, i.e. $A \in HB(\mathcal{L}) \cup \{\perp\}$.*

- *$\mathcal{H}_w(S)$ denotes the weak Herbrand interpretation in which A is consistently false if $A \notin S$, consistently true if $A \in S$ and $\neg A \notin S$, and inconsistent otherwise, i.e. if $A \in S$ and $\neg A \in S$.*
- *$\overline{S} := \{A \mid A \in S \cap (HB(\mathcal{L}) \cup \{\perp\})\} \cup \{\neg A \mid \neg A \in S \wedge A \in S\}$.*

Note that in case $A \notin S$, then A is consistently false in $\mathcal{H}_w(S)$, no matter whether $\neg A \in S$ or $\neg A \notin S$. Thus, $\mathcal{H}_w(S) = \mathcal{H}_w(\overline{S})$ and classical Herbrand interpretations can be represented as usual [7].

Notation 11 *Let $\mathcal{I}$ be a weak interpretation.*

$$S_{\mathcal{I}} := \overline{\{L \mid L \in EHB(\mathcal{L}) \wedge L^{\mathcal{I}}: \mathbf{t}\}}.$$

Note that, if $\mathcal{I}$ is a weak interpretation, then $\mathcal{H}_w(S_{\mathcal{I}})$ is a weak Herbrand interpretation.

Proposition 12 *Let P be a set of clauses and $\mathcal{I}$ a weak interpretation. $\mathcal{I}$ is a weak model of P if and only if $\mathcal{H}_w(S_{\mathcal{I}})$ is a weak Herbrand model of P.*

Proof: Let P be a set of clauses, $\mathcal{I}$ a weak interpretation, and $L \in EHB(\mathcal{L})$. By definition of $S_{\mathcal{I}}$, $L^{\mathcal{I}}$:**t** if and only if $L^{\mathcal{H}_w(S_{\mathcal{I}})}$:**t**. The result follows. ∎

$\mathcal{H}_w(EHB(\mathcal{L}))$ is a weak Herbrand model of every program. It is "globally inconsistent" in the sense that every ground atom is inconsistent in $\mathcal{H}_w(EHB(\mathcal{L}))$. By restricting the set of satisfied ground literals, weak Herbrand interpretations can be defined that more accurately satisfy a program in the sense that they contain less inconsistencies. The rest of this section is devoted to formalize a notion of "intended model" based on this idea. To this aim, an order on weak Herbrand interpretations is induced from the subset relation, like for classical Herbrand interpretations.

Definition 13 *Let $\mathcal{H}_w(S_1)$ and $\mathcal{H}_w(S_1)$ be two weak Herbrand interpretations.*

- *$\mathcal{H}_w(S_1) \preceq \mathcal{H}_w(S_2)$ if and only if $\overline{S_1} \subseteq \overline{S_2}$.*
- *$\mathcal{H}_w(S_1) \cap \mathcal{H}_w(S_2) := \mathcal{H}_w(S_1 \cap S_2)$.*

5.3 Supported Weak Models

Clearly, $\preceq$ extends to weak Herbrand interpretations the usual order relation defined on classical Herbrand interpretations. A (classical) Herbrand model of program P which is minimal for the order $\preceq$ of Definition 13 clearly is also minimal in the classical sense. Thus, the intended model of positive, unconstrained and definite program (as defined in [7]) is its (unique) minimal weak model. However, not all minimal weak models of a non-positive program convey its intuitive meaning. Consider for example the program $P = \{p \leftarrow \neg p\}$. The minimal model $\mathcal{H}_w(\{p\})$ of P clearly does not convey the intuition, because it is not *supported.*

Definition 14 (Supported Weak Model) *Let P be a set of clauses, $\mathcal{M}$ a weak model of P, A a ground atom, and $L_1, ..., L_n$ ($n \geq 1$) ground literals. Literals and conjunction of literals* supported in $\mathcal{M}$ with respect to P *are inductively defined as follows:*

- $\top$ *is* supported in $\mathcal{M}$ with respect to P.
- A *is* supported in $\mathcal{M}$ with respect to P *if and only if A is consistently true in $\mathcal{M}$ and there exists a ground instance $H \leftarrow B$ of a clause in P such that A occurs in H and B is supported in $\mathcal{H}_w(S_{\mathcal{M}} \cup \{\neg A\})$.*
- $L_1 \wedge ... \wedge L_n$ *is* supported in $\mathcal{M}$ with respect to P *if and only if each positive L_i is supported in $\mathcal{M}$ with respect to P and each negative L_i is consistently false in $\mathcal{M}$.*

$\mathcal{M}$ *is a* supported weak model of P *if and only if every ground atom which is consistently true in $\mathcal{M}$ is supported in $\mathcal{M}$ with respect to P.*

Consider a set of clauses P and a weak model $\mathcal{M}$ of P. Since $\top$ is supported in $\mathcal{M}$ with respect to P, every ground instance of a fact in P, i.e. of a definite clause with body $\top$, is supported in $\mathcal{M}$ with respect to P. A negative ground literal $\neg A$ is supported in $\mathcal{M}$ with respect to P if and only if A is consistently false in $\mathcal{M}$. A stengthening of Definition 14 is possible, so as to account for the intuitive notion of *finite failure.*

Since $\mathcal{H}_w(EHB(\mathcal{L}))$ is a model of every set of clauses and since no atoms are consistently true in $\mathcal{H}_w(EHB(\mathcal{L}))$, $\mathcal{H}_w(EHB(\mathcal{L}))$ is a supported weak Herbrand model of every set of clauses.

Requiring the body B of a ground instance $H \leftarrow B$ of a clause in P to be supported in $\mathcal{H}_w(S_{\mathcal{M}} \cup \{\neg A\})$ and not in $\mathcal{H}_w(S_{\mathcal{M}})$, for an atom A occurring in H to be supported in $\mathcal{M}$ with respect to P precludes "cyclic supports". Indeed, if A is supported in $\mathcal{M}$ with respect to P, then A is inconsistent in $\mathcal{H}_w(S_{\mathcal{M}} \cup \{\neg A\})$ and cannot serve in a "support" of B. For example, $\mathcal{H}_w(\{a, b\})$ is not a supported weak model of $\{a \leftarrow b\ ;\ b \leftarrow a\}$. (In the preliminary version [2] of this paper, a simpler but weaker form of Definition 14 is given, which uncorrectly does not preclude "cyclic supports".)

Proposition 15 *Let P be a set of positive clauses and $\mathcal{I}$ a Herbrand interpretation. $\mathcal{I}$ is a minimal (classical) model of P if and only if it is a supported model of P.*

Proof: Immediate. ■

If $\mathcal{M}$ is a supported weak model of a program P, then by Proposition 9 (6) for every ground instance $H \leftarrow B$ of a clause in P, if H is consistently false, then so is also B. However, H migth be inconsistent in $\mathcal{M}$ and B consistently true in $\mathcal{M}$, as for $H = b$ and $B = a$ in the supported weak model $\mathcal{H}_w(\{a, b, \neg b, \perp\})$ of $P = \{\perp \leftarrow b \ ; \ b \leftarrow a \ ; \ a\}$.

5.4 Intended Models

A semantics for logic programs and disjunctive databases is defined by specifying for each program P, which models P specifies, i.e. which models should be considered to have been intended by the programmer of P. The "Compositional Semantics" is defined by specifying the "intended models" of a program as follows.

Definition 16 (Intended Model of the Compositional Semantics) *An* intended model *of a set P of clauses is a supported weak Herbrand model of P which is minimal for $\preceq$ among the supported weak Herbrand models of P.*

Proposition 17 *Every set of clauses, in particular every program has an intended model.*

Proof: Let P be a set of clauses and $SM(P)$ denote the set of supported weak Herbrand models of P. Since $\mathcal{H}_w(EHB(\mathcal{L})) \in SM(P)$, $SM(P) \neq \emptyset$. Let $\mathcal{C}$ be a chain in $SM(P)$, i.e. a subset of $SM(P)$ such that for every $\mathcal{M}_1$ and $\mathcal{M}_2$ in $SM(P)$, $\mathcal{M}_1 \preceq \mathcal{M}_2$ or $\mathcal{M}_2 \preceq \mathcal{M}_2$. Let $\mathcal{M} = \bigcap\{\mathcal{N} \mid \mathcal{N} \in \mathcal{C}\}$. Like every $\mathcal{N} \in \mathcal{C}$, $\mathcal{M}$ is a supported weak Herbrand model of P and it is a lower bound of $\mathcal{C}$. By Zorn's lemma, $SM(P)$ has a minimal element. ■

Disjunctive and non-positive programs might have several intended models, as the following example shows.

Example 2 Consider the programs of Example 1.

1. The only intended model of $\{\perp \leftarrow \neg p\}$ is $\mathcal{H}_w(\{p, \neg p, \perp\})$.

2. The intended models of $\{a \leftarrow \neg b \ ; \ b \leftarrow \neg a\}$ are $\mathcal{H}_w(\{a, \neg b\})$ and $\mathcal{H}_w(\{\neg a, b\})$.

3. The only intended model of $\{a \leftarrow \neg b \ ; \ b \leftarrow \neg c \ ; \ c \leftarrow \neg a\}$ is $\mathcal{H}_w(\{a, \neg a, b, \neg b, c, \neg c, \bot\})$.

4. The only intended model of $\{\bot \leftarrow b \ ; \ b \leftarrow a \ ; \ a\}$ is $\mathcal{H}_w(\{a, b, \neg b, \bot\})$.

5. The only intended model of $\{b \leftarrow a \ ; \ \bot \leftarrow a \ ; \ a\}$ is $\mathcal{H}_w(\{a, \neg a, b, \neg b, \bot\})$.

6. The only intended model of $\{a \leftarrow b \ ; \ b \leftarrow a\}$ is $\mathcal{H}_w(\emptyset)$.

7. The intended models of $\{a \vee b \ ; \ \bot \leftarrow a\}$ are $\mathcal{H}_w(\{a, \neg a, \bot\})$ and $\mathcal{H}_w(\{b\})$.

8. The only intended model of $\{a \leftarrow b \ ; \ b \leftarrow a \ ; \ c \leftarrow \neg a\}$ is $\mathcal{H}_w(\{c\})$.

9. The only intended model of $\{p(x) \leftarrow p(f(x)) \ ; \ q(a) \leftarrow \neg p(x)\}$ is $\mathcal{H}_w(\{q(a)\})$.

The following proposition shows that, according to a programmer's intuition, Proposition 9 (6) has a converse in intended models.

Proposition 18 *Let P be a program, $\mathcal{I}$ an intended model of P, and A a ground atom or the formula $\bot$. If for all ground instances $H \leftarrow B$ of clauses in P such that A occurs in H, B is consistently false in $\mathcal{I}$, then A is consistently false in $\mathcal{I}$.*

Proof: Let $\mathcal{I} = \mathcal{H}_w(S)$. Assume that

$$(*) \text{ for all ground instances } H \leftarrow B \text{ of clauses in } P \text{ such that } A \text{ occurs in } H,\ B \text{ is consistently false in } \mathcal{I}.$$

a) A is not consistently true in $\mathcal{I}$. Otherwise, since $\mathcal{I}$ is an intended model of P, A would be supported in $\mathcal{I}$ with respect to P, i.e., for some ground instance $H \leftarrow B$ of a clause in P, B would be true in $\mathcal{I}$, contradicting $(*)$.

b) A is not inconsistent in $\mathcal{I}$. Assume the contrary. Let $S' = S \setminus \{A\}$ and $\mathcal{I}' = \mathcal{H}_w(S')$. Since $\mathcal{I}$ is an intended model of P, $\mathcal{I}'$ is not a supported weak model of P. Since A is inconsistent in $\mathcal{I}$, by Proposition 9, it does not contribute to make ground atoms supported in $\mathcal{I}$ with respect to P.

Hence, if $\mathcal{I}'$ is a weak model of P, it is like $\mathcal{I}$ a supported model of P, a contradiction. Therefore, $\mathcal{I}'$ is not a weak model of P. Hence there exists a ground instance $F \leftarrow C$ of a clause in P which is not true in $\mathcal{I}'$, i.e. C is true in $\mathcal{I}'$ and F is not true in $\mathcal{I}'$. We now show that C is not true in $\mathcal{I}$. Assume the contrary. Since $\mathcal{I}$ is a weak model of P, if C is true in $\mathcal{I}$, then F is true in $\mathcal{I}$. Since F is not true in $\mathcal{I}'$, by definition of $\mathcal{I}'$, necessarily $F = A$. By hypothesis $(*)$, C is consistently false in $\mathcal{I}$, a contradiction. Since C is true in $\mathcal{I}'$ and not true in $\mathcal{I}$, by definition of $\mathcal{I}'$, $\neg A$ occurs in C and $\neg A$ is not true in $\mathcal{I}$. This contradict the assumption that A is inconsistent in $\mathcal{I}$. ∎

The following proposition shows that intended models are conform to a common programmer's intuition.

Proposition 19 *Let P be a program, $\mathcal{I}$ an intended model of P, and A a ground atom or the formula $\perp$. A is true in $\mathcal{I}$ if and only if there exists a ground instance $H \leftarrow B$ of a clause in P such that A occurs in H and B is true in $\mathcal{I}$.*

Proof: By Definition 7, it sufficies to show that the necessary condition holds. If A is consistently true in $\mathcal{I}$, then since $\mathcal{I}$ is a supported model, the result follows. Assume that A is inconsistent in $\mathcal{I}$. Let $\mathcal{I} = \mathcal{H}_w(S)$ and $\mathcal{I}' = \mathcal{H}_w(S \setminus \{A\})$. Since $\mathcal{I}$ is an intended model of P, $\mathcal{I}'$ is not a supported weak model of P. If $\mathcal{I}'$ is a weak model of P, then it is like $\mathcal{I}$ a supported weak model of P, for S and $S \setminus \{A\}$ differ only on the inconsistent formula A and, by Definition 14, inconsistent atoms do not contribute to the supportedness of weak models. Hence, $\mathcal{I}'$ is not a weak model of P, i.e. there exists a ground instance $H \leftarrow B$ of a clause in P such that B is true in $\mathcal{I}'$ and H is not true in $\mathcal{I}'$. Since $\mathcal{I}$ is a weak model of $H \leftarrow B$, necessarily A occurs in H. The result follows. ∎

Since for (disjunctive as well as definite) positive and unconstrained programs classical intended models coincide with the intended models according to Definition 16, Theorem 4 establishes the completeness of the minimal logic proof calculus for such programs with respect to the "Compositional Semantics" as defined by Definition 16.

A completeness proof for programs referring to negation as finite failure cannot be given here, for it would have to refer not only to the proof calculus, but also to the search strategy. Which proofs can be constructed by a deduction system computing negation as finite failure depends on the search strategy under which the system is run, indeed. For example, a system extending SLDAL-resolution [16] with negation as failure would be complete for Datalog$^\neg$ programs [16]. In contrast, SLDNF-resolution is not complete for Datalog$^\neg$ programs.

5.5 Stability

In this section, it is shown that the "compositional semantics" is "stable" under the evaluation of consistently true literals and generalizes the stable model semantics [4].

Notation 20 *Let P be a set of ground clauses, $\mathcal{I}$ a weak Herbrand interpretation, and S a set of ground literals that are consistently true in $\mathcal{I}$. $Simp(P, S)$ denotes the set of clauses obtained by discarding every occurrence of a literal in S from the body of a clause in P.*

Theorem 21 *Let P be a set of ground clauses, $\mathcal{I}$ a weak Herbrand interpretation, and S a set of ground literals that are consistently true in $\mathcal{I}$. If $\mathcal{I}$ is an intended model of P, then it is also an intended model of $Simp(P, S)$.*

Proof: Assume that $\mathcal{I}$ is an intended model of the set P of ground clauses. Clearly, $\mathcal{I}$ is a supported model of $Simp(P, S)$. Let A be an atom which is consistently true in $\mathcal{I}$. Since $\mathcal{I}$ is an intended model of P there exists a clause $H \leftarrow L_1 \wedge ... \wedge L_n$ in P such that A occurs in H and $L_1 \wedge ... \wedge L_n$ is supported in $\mathcal{H}_w(S_{\mathcal{I}} \cup \{\neg A\})$. By definition, there are $j_1, ..., j_k$ $(k \leq n)$ in $\{1, ..., n\}$ such that $H \leftarrow L_{j_1} \wedge ... \wedge L_{j_k}$ is a clause in $Simp(P, S)$. Like $L_1 \wedge ... \wedge L_n$, $L_{j_1} \wedge ... \wedge L_{j_k}$ is supported in $\mathcal{H}_w(S_{\mathcal{I}} \cup \{\neg A\})$. ∎

Note that Theorem 21 does not extend to simplifications with respect to inconsistent literals.

Corollary 22 *Let P be a set of definite clauses and $\mathcal{I}$ a weak Herbrand interpretation. If $\mathcal{I}$ is a consistent intended model of P, then $\mathcal{I}$ is a stable model of P.*

Proof: Let P be a set of definite clauses, Q the set of ground instances of clauses in P, $\mathcal{I}$ a weak Herbrand interpretation, and $Neg(\mathcal{I})$ the set of negative ground literals that are true in $\mathcal{I}$. If $\mathcal{I}$ is an intended model of P, then it is an intended model of Q. By Theorem 21, $\mathcal{I}$ is an intended model of $Simp(Q, Neg(\mathcal{I}))$, i.e., since all clauses in Q are positive, a minimal (classical) model of Q. By definition [4], $\mathcal{I}$ is a stable model of Q. It follows that $\mathcal{I}$ is a stable model of P. ∎

Theorem 23 *Let P be a set of definite clauses and $\mathcal{I}$ a weak Herbrand interpretation. If $\mathcal{I}$ is a stable model of P, then $\mathcal{I}$ is an intended model of P.*

Proof: Let P be a set of definite clauses, Q the set of ground instances of clauses in P, $\mathcal{I}$ a weak Herbrand interpretation, and $Neg(\mathcal{I})$ the set of negative ground literals that are true in $\mathcal{I}$. If $\mathcal{I}$ is a stable model of P, it is a stable model of Q, i.e. it is a minimal model of $Simp(Q, Neg(\mathcal{I}))$. By Proposition 15, $\mathcal{I}$ is a supported model of $Simp(Q, Neg(\mathcal{I}))$. It follows that $\mathcal{I}$ is also a supported model of Q and P. By Corollary 22, if $\mathcal{H}_w(S)$ is an intended model of P such that $S \subseteq S_{\mathcal{I}}$, then then $\mathcal{H}_w(S)$ is a stable model of P. By minimality of a stable model, $S = S_{\mathcal{I}}$ and $\mathcal{I}$ is an intended model of P. ■

5.6 Compositionality

In this section, the "composition principle" mentioned in Section 2.2 is formalized and the semantics defined in Section 5.4 is shown to be "compositional".

Definition 24 *Let A be an atom or the formula $\perp$, and let P and Q be sets of clauses.*

- *P* defines *A if and only if A occurs in an instance of the head of a clause in P.*
- *P* depends *on Q if and only if an atom occurring (positively or negatively) in the body of a clause of P is defined by Q.*

A semantics for logic programs and disjunctive databases is defined by specifying for each program P, which models should be considered to have been intended by the programmer of P. Depending on the semantics, these "intended" models are given different names, such as "perfect", well-founded", or "stable" models. Here, for a semantics $\mathcal{S}$, they will be refered to as *$\mathcal{S}$-intended models.*

Notation 25 *Let P be a set of clauses. $EHB(\mathcal{L})_P$ denotes the subset of $EHB(\mathcal{L})$ containing $\perp$ if it is defined in P and the literals whose atoms are defined in P.*

Definition 26 (Compositional Semantics) *Let $\mathcal{S}$ be a semantics. The semantics $\mathcal{S}$ is* compositional *if for every programs P and Q such that*

1. *P does not depend on Q, and*
2. *no atoms are defined by both P and Q,*

and for every $\mathcal{S}$-intended model $\mathcal{M}_P$ of P, there exists an $\mathcal{S}$-intended model $\mathcal{M}_{P \cup Q}$ of $P \cup Q$ such that every ground literal in $EHB(\mathcal{L})_P$ which is true (false, inconsistent, resp.) in $\mathcal{M}_P$ is also true (false, inconsistent, resp.) in $\mathcal{M}_{P \cup Q}$.

Simple examples show that the conditions of Definition 26 cannot be relaxed, even for definite, positive and unconstrained programs, without contradicting a common programmer's intuition.

Lemma 27 *Let $S \subseteq EHB(\mathcal{L})$ and let P_1 and P_2 be sets of clauses such that*

1. *P_1 does not depend on P_2, and*
2. *no atoms are defined by both P_1 and P_2.*

If $\mathcal{H}_w(S)$ is a supported model of $P_1 \cup P_2$, then $\mathcal{H}_w(S \cap EHB(\mathcal{L})_{P_1})$ is a supported model of P_1.

Proof: Assume P_1 and P_2 fulfill the hypotheses of the lemma. Let $\mathcal{H}_w(S)$ be a supported model of $P_1 \cup P_2$ and $H \leftarrow B$ a ground instance of a clause in P_1. Assume that B is true in $\mathcal{H}_w(S \cap EHB(\mathcal{L})_{P_1})$. Since $\mathcal{H}_w(S)$ is a model of $P_1 \cup P_2$, H is true in $\mathcal{H}_w(S)$. By definition of $EHB(\mathcal{L})_{P_1}$ the atoms (or possibly $\perp$) occurring in H are in $EHB(\mathcal{L})_{P_1}$. It follows that H is true in $\mathcal{H}_w(S \cap EHB(\mathcal{L})_{P_1})$. $\mathcal{H}_w(S \cap EHB(\mathcal{L})_{P_1})$ therefore is a weak model of P_1. Let A be an atom in $EHB(\mathcal{L})_{P_1}$. If A is consistently true in $\mathcal{H}_w(S)$, then since $\mathcal{H}_w(S)$ is a supported model of $P_1 \cup P_2$, there exists a ground instance $H \leftarrow B$ of a clause in $P_1 \cup P_2$ such that A occurs in H and B is supported in $\mathcal{H}_w(S \cup \{\neg A\})$. By hypothesis on P_1 and P_2, $H \leftarrow B$ is a ground instance of a clause in P_1 and B is supported in $\mathcal{H}_w((S \cup \{\neg A\}) \cap EHB(\mathcal{L})_{P_1})$. It follows that $\mathcal{H}_w(S \cap EHB(\mathcal{L})_{P_1})$ is a supported model of P_1. ∎

Theorem 28 *The semantics specified by Definition 16 is compositional.*

Proof: Let P and Q be sets of clauses such that P does not depend on Q, and no atoms are defined by both P and Q. Let $\mathcal{I}$ be an intended model of P. Let P' be the set of ground clauses obtained from the ground instances of clauses in P by discarding the head literals that are not true in $\mathcal{I}$, i.e. that are not in $S_{\mathcal{I}}$. $\mathcal{I}$ is an intended model of P' and, by definition of P', if $\mathcal{J}$ is a supported weak Herbrand model of P', then $S_{\mathcal{J}} \subseteq S_{\mathcal{I}}$, i.e. $S_{\mathcal{J}} = S_{\mathcal{I}}$. $\mathcal{I}$ is therefore the only intended model of P'. By Proposition 17, there exists an intended model $\mathcal{M}$ of $P' \cup Q \cup S_{\mathcal{I}}$. By definition of $S_{\mathcal{I}}$ and P', $S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'} \subseteq S_{\mathcal{I}}$. By Lemma 27 $\mathcal{H}_w(S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'})$ is a supported model of $P' \cup Q$. Let $\mathcal{N}$ be an intended model of $P' \cup Q$ such that $S_{\mathcal{N}} \cap EHB(\mathcal{L})_{P'} \subseteq S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'}$. By definition of P' and hypotheses on P and Q, $\mathcal{N}$ is an intended model of $P \cup Q$. By Lemma 27 $\mathcal{H}_w(S_{\mathcal{N}} \cap EHB(\mathcal{L})_{P'})$ is a supported model P'. Since $\mathcal{I}$ is the only intended model of P', $S_{\mathcal{I}} = S_{\mathcal{N}} \cap EHB(\mathcal{L})_{P'}$. Hence:

$$S_{\mathcal{I}} = S_{\mathcal{N}} \cap EHB(\mathcal{L})_{P'} \subseteq S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'} \subseteq S_{\mathcal{I}}$$

It follows that $S_{\mathcal{N}} \cap EHB(\mathcal{L})_{P'} = S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'}$, i.e. $\mathcal{H}_w(S_{\mathcal{M}} \cap EHB(\mathcal{L})_{P'})$ is an intended model of P', hence of P. ∎

Theorem 28 cannot be strengthend in the sense that, P and Q might satisfy the conditions of the theorem, and an atom might be consistent in P and inconsistent in $P \cup Q$, as the following example shows.

Example 3 Let $P = \{p\}$ and $Q = \{\bot \leftarrow p\}$. P does not depend on Q and no atoms are defined by both, P and Q. p is consistently true in the only intended model of P and inconsistent in the only intended model $P \cup Q$.

Note that Theorem 28 implies that the usual semantics of definite, positive and unconstrained programs is compositional. Note also that the stable model semantics is not compositional. For example, the stable model of $P = \{a\}$ does not extend into a stable model of $P \cup \{b \leftarrow \neg b\}$.

5.7 Program Completion

In this section, a language $\mathcal{L}$ with equality is considered and weak interpretations are assumed to interpret the equality predicate of $\mathcal{L}$ as the equality relation. The completion $Comp(P)$ of a definite and unconstrained program [7] – i.e. the set of completed clauses of P augmented with Clark's equality theory CET – extends to disjunctive (possibly constrained) programs P as follows [8]:

Let p be an k-ary ($k \geq 0$) predicate symbol or $\bot$. If p does not occur in the head of a clause in P, then the *completed definition of p (with respect to P)* is the formula:

$$\forall x_1 ... \forall x_k \ \neg p(x_1, ..., x_k)$$

Otherwise, let $A_1^i \vee ... \vee p(t_1^i, ..., t_k^i) \vee ... \vee A_{n_i}^i \longleftarrow B^i[y_1, ..., y_{k_i}]$ $(1 \leq i \leq l)$ denote the $l \geq 1$ (distinct) occurrences of p in heads of clauses in P. The *completed definition of p (with respect to P)* is the formula:

$$\forall x_1 ... \forall x_k \ \left(p(x_1, ..., x_k) \rightarrow \bigvee_{i=1}^{l} \exists y_1 ... \exists y_{k_i} \left(\bigwedge_{j=1}^{k} x_j = t_j^i \ \wedge \ B^i[y_1, ..., y_{k_i}] \right) \right)$$

The *completion* $Comp(P)$ of a (possibly disjunctive) program P consists in the union of the following sets of formulas:

- The set of the completed definitions of the predicates occurring in P,
- CET,
- $\{\forall^* \ C \mid C \in P\}$.

Not every weak model of the completion of a program is a supported model of this program, as the following example shows. However, for *consistent* weak interpretation, the expected result holds, as established by the theorem below.

Example 4 Let $P = \{a \leftarrow b\ ;\ \bot \leftarrow b\ ;\ b\}$ and $S = \{a, b, \neg b, \bot\}$. $\mathcal{H}_w(S)$ is a (minimal) weak model of $Comp(P) = \{a \leftrightarrow b\ ;\ \bot \leftrightarrow b\ ;\ b \leftrightarrow \top\}$. $\mathcal{H}_w(S)$ is not an intended model of P, for a is not supported with respect to P in $\mathcal{H}_w(S)$.

Theorem 29 (Completion Semantics) *Let P be a program and $\mathcal{I}$ a weak Herbrand interpretation.*

1. *If $\mathcal{I}$ is a consistent weak model of $Comp(P)$, then $\mathcal{I}$ is a supported model of P.*
2. *If $\mathcal{I}$ is a minimal weak model of $Comp(P)$ and if $\mathcal{I}$ is consistent, then $\mathcal{I}$ is an intended model of P.*
3. *If $\mathcal{I}$ is an intended model of P and if $\mathcal{I}$ is consistent, then $\mathcal{I}$ is a minimal weak model of $Comp(P)$.*

Proof: Let $p(t_1, ..., t_k)$ be a ground atom or $\bot$ and let C be the completed definition $\forall x_1 ... \forall x_k\ (p(x_1, ..., x_k) \rightarrow D)$ (with respect to P) of $p(t_1, ..., t_k)$.
1. Assume that $\mathcal{I}$ is a consistent weak model of $Comp(P)$ and that $p(t_1^i, ..., t_k^i)$ is consistently true in $\mathcal{I}$. Since $C \in Comp(P)$, $D[t_1/x_1, ..., t_k/x_k]$ is true in $\mathcal{I}$. Since $\mathcal{I}$ is consistent, $D[t_1/x_1, ..., t_k/x_k]$ is consistently true in $\mathcal{I}$, i.e. by Proposition 9 (2) $p(t_1, ..., t_k)$ is supported in $\mathcal{I}$.
2. By 1 and Definition 16.
3. Assume that $\mathcal{I}$ is an intended model of P and that $\mathcal{I}$ is consistent. Let p be a predicate symbol of arity $n \geq 0$, A a ground atom with predicate symbol p, σ a substitution such that $A = p(x_1, ..., x_n)\sigma$, and $\forall x_1 ... \forall x_n\ (p(x_1, ..., x_n) \rightarrow F)$ the completed definition of p. If A is true in $\mathcal{I}$, then, by Proposition 19 (2), $F\sigma$ is true in $\mathcal{I}$. It follows that $\mathcal{I}$ satisfies the completed definition of every predicate symbol. Since $\mathcal{I}$ satisfies Clark's equality theory CET, $\mathcal{I}$ is a weak model of $Comp(P)$. Since $\mathcal{I}$ is consistent, by Definition 10, there exists a set S of ground atoms such that $\mathcal{I} = \mathcal{H}_w(S)$. If $\mathcal{I}$ is not a minimal weak model of $Comp(P)$, then there exists $A \in S$ such that $\mathcal{I}' = \mathcal{H}_w(S \setminus \{A\})$ is a weak model of $Comp(P)$. By definition of $Comp(P)$, $\mathcal{I}'$ is a supported weak model of P, a contradiction. ∎

6 Conclusion

In this article, a semantics has been proposed according to which, some logic programs or deductive databases might contain *local inconsistencies.* This semantics departs from classical logic which *globalizes* inconsistencies.

The proposed semantics has been motivated first by practical considerations on database query answering and program composition. It has also been motivated by considering the proof theory underlying SLD-resolution.

It has been shown that minimal logic – a weakening of classical logic proof theory which precludes refutation proofs – is complete for positive (disjunctive as well as definite) programs. Moreover, minimal logic derivations from definite and unconstrained programs have been shown to be equivalent to SLD-resolution proofs. Thus, the intuition that refutation proofs are undesirable for querying databases and evaluating logic programs, which was suggested by practical considerations, is confirmed from a proof theoretic viewpoint.

In order to provide logic programs and deductive databases with a model theory conforming to the rejection of refutation proofs, a nonclassical model theory has been proposed. This model theory is based on a generalization of the classical notion of interpretation. Relying on this model theory, a notion of intended model for a generalization of logic programs has been proposed. This generalization accounts for databases integrity constraints and disjunctive logic programs.

The proposed intended models are minimal models specified by sets of ground literals, thus naturally extending the notion of minimal models of positive and definite logic programs. The proposed notion of intended model has also been characterized within classical logic in terms of program completions, thus extending the completion semantics of positive and definite logic programs.

Finally, the proposed semantics has been shown to be generalize the stable model semantics [4] and, in contrast to this semantics, to be "compositional" in the sense that it fulfils a natural program composition requirement.

The compositional semantics proposed in this paper seems to be natural and simple enough for being as easily applied as the usual semantics of positive logic programs. Moreover, since it is defined only in terms of notions that are well established in logic programming and databases, it is not unreasonable to hope that it could be used not only by logicians, but also by practicians.

Further research on the subject is needed. First, an inductive definition in terms of fixpoint of the proposed intended models remains to be given. Since the general programs considered in this article might have several intended models, a fixpoint operator could be specified in terms of disjunctions of ground atoms instead of ground atoms, like the operator defined in [8] for disjunctive databases. Second, the usefulness of the proposed semantics for practical applications of logic programming and deductive database should be experimentally investigated. Finally, if the proposed notion of local inconsistencies appears to be useful in practice, techniques would have to be developed for detecting whether an answer relies on inconsistent parts of the logic program or database, or not. For specifying such methods, techniques applied for integrity verification in databases are likely to be applicable.

Acknowledgements

I thank Norbert Eisinger and Heribert Schütz and the anonymous referes of [2] for useful comments, Gopalan Nadathur for fruitful discussions on natural deduction and logic programming, and Helmut Schwichtenberg for drawing my attention to minimal logic.

References

[1] F. Bry. *Theorie der Logikprogrammierung.* Lecture Notes. Unpublished, 1995.

[2] F. Bry. A Compositional Semantics for Logic Programs and Deductive Databases. *Proc. of the Joint International Conference and Symposium on Logic Programming.* MIT Press, 1996

[3] K. Clark. Negation as Failure. *Logic and Databases.* H. Gallaire and J. Minker, eds. 293-322, Plenum Press, New York, 1978.

[4] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. *Proc. of the Fifth International Conference and Symposium on Logic Programming.* 1070-1080, Seattle, MIT Press, 1988.

[5] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift.* Vol. 39, 176-210, 1934.

[6] K. Kunen. Declarative Semantics of Logic Programming. *Bulletin of the European Association for Theoretical Computer Science.* Vol. 44, 147-167, 1991.

[7] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd Ed., 1987.

[8] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, 1992.

[9] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic.* Vol. 51, 125-157, 1991.

[10] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almqvist & Wiksell, Stockholm, 1965.

[11] T. Sato. Completed Logic Programs and Their Consistency. *Journal of Logic Programming.* Vol. 9, No. 1, 33-44, 1990.

[12] H. Schwichtenberg. Minimal From Classical Proofs. *Computer Science Logic.* H. Kleine-Büning and M.M. Richter, eds. 326-328, Springer Verlag LNCS 626, 1992.

[13] R. F. Stärk. A Direct Proof for the Completeness of SLD-Resolution. *Computer Science Logic.* E. Börger, H. Kleine-Büning and M.M. Richter, eds. 382-383, Springer-Verlag LNCS 440, 1990.

[14] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction.* Vol 1. North-Holland, Amsterdam, 1988.

[15] A. van Gelder, K. Ross, and J. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM.* Vol. 38, No. 3, 620-650, 1991.

[16] L. Vieille. Recursive Query Processing: The Power of Logic. *Theoretical Computer Science.* Vol. 68, No. 2, 1989.