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

Ludwig
Maximilians
Universität
München
LMU

A Compositional Semantics for Logic Programs and Deductive Databases

François Bry

appeared in Proc. Joint Int. Conf. and Symp. on Logic Programming (JICSLP), Bonn, Germany, MIT Press, 453-467, Sept. 1996
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1996-3, June 1996

A Compositional Semantics for Logic Programs and Deductive Databases

François Bry
Institut für Informatik, Ludwig-Maximilians-Universität München
Oettingenstraße 67, D – 80538 München, Germany
bry@informatik.uni-muenchen.de

Abstract

Considering integrity constraints and program composition, it is argued that a semantics for logic programs and deductive databases should not accommodate inconsistencies *globally* like in classical logic, but *locally.* It is shown that minimal logic, a weakening of classical logic which precludes refutation proofs, is sufficient to provide for a proof theory for generalized programs corresponding to deductive databases and disjunctive logic programs. A (nonclassical) model theory is proposed for these programs, which allows *local inconsistencies.* The proposed semantics naturally extends the minimal model and completion semantics of positive programs and is *compositional.* Arguably, it appropriately conveys a practician's intuition.

1 Introduction

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 approach to specifying the semantics of positive programs can be quite naturally extended to general programs, the extensions are not equivalent. Negation as failure [2], for example, gives rise to an extension of SLD-resolution, but has no satisfying model theoretic counterparts. The stable model semantics [5] elegantly extends the nonconstructive notion of minimal model to general programs and the well-founded semantics [16] 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 [2] 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 [7]: A PhD in logic seems to be needed for understanding a semantics which adequately conveys the intuitive meaning of negation in logic programs!

In this article, the issue is reconsidered with a strong bias towards applications. Considering database integrity constraints 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 for which inconsistencies are local,

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, and therefore does not "globalize" inconsistencies. 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 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 interpretations give rise to a simple and intuitive extension to general programs of the minimal model and of the completion semantics of positive programs. First investigations indicate that an inductive, fixpoint-like definition of the proposed intended models should be possible. Moreover, the proposed semantics is shown to be *compositional,* in the sense that it fulfills a natural program composition requirement.

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. Due to space limitations, proofs are not given in this article. They can be found in its full version [1] available at: http://www.pms.informatik.uni-muenchen.de/publikationen/

2 Motivation

By definition of interpretations and models, inconsistent theories have no models in classical logic. Therefore, every formula follows from an inconsistent theory. This treatment of inconsistency gives rise to indirect, or refutation, proofs. Refutation proofs are often referred to in logic programming. SLD-resolution, for example, is usually defined as a refutation procedure [8]. However, refutation proofs sometimes contradict a database designer's and programmer's intuition.

2.1 Inconsistent Databases

While standard logic programs cannot be inconsistent, a database is inconsistent if 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 clauses, where $\neg$ means negation as failure:

$\bot \leftarrow emp(x) \wedge \neg\ in\text{-}a\text{-}dpt(x) \qquad in\text{-}a\text{-}dpt(x) \leftarrow dpt(y) \wedge member(x,y)$

It is a common (although unformalized) practice to query an inconsistent database. 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 any atom. A constraint such as $C := \bot \leftarrow \neg p(a)$ requires $p(a)$ to be true, but does not *define* it.

2.2 Program Composition

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 [2]. Also, it leads to define the semantics of disjunctive [9] and of call-consistent [12] programs in terms of alternative models and not of logical entailment. The following program

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

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

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

this intuition leads to difficulties, for their completions [2] 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, "incorrect" logic programs have to be given a semantics.

3 Preliminaries

Throughout the article, a fixed first-order language $\mathcal{L}$ is considered, in which all theories and programs are defined. Except when otherwise stated it is assumed to be without equality, and to have the following logical symbols: $\wedge$, $\vee$, $\rightarrow$, $\forall$, $\exists$, and $\bot$. The symbols $\neg$, $\top$, and $\leftrightarrow$ will be used for shorthand notations and are defined by $\neg F := (F \rightarrow \bot)$, $\top := \neg\bot$, and $F \leftrightarrow G := (F \rightarrow G) \wedge (G \rightarrow F)$. $G \rightarrow F$ will also be written $F \leftarrow G$.

Atoms, literals, formulas, the Herbrand base $HB(\mathcal{L})$ of a first-order language $\mathcal{L}$ etc. are defined as usual. The *extended Herbrand base* $EHB(\mathcal{L})$ of $\mathcal{L}$ is the set containing $\bot$ and all ground *literals* of $\mathcal{L}$. $\bot$ denotes a special formula which is satisfied in no interpretations. $\bot$ is considered not to be an atom (so as not to modify the usual definition of the Herbrand base). If A is an atom or $\bot$, it *occurs* in $B_1 \vee ... \vee B_n$ ($B_1 \wedge ... \wedge B_n$, resp.) ($n \geq 1$) in case $A = B_i$ for some $i \in \{1, ..., n\}$.

A *clause* is an expression of the form $B_1 \wedge ... \wedge B_n \rightarrow H_1 \vee ... \vee H_m$ with $n \geq 1$ and $m \geq 1$, where the B_i are literals, or $\top$ if $n = 1$, and the H_j are atoms, or $\bot$ if $m = 1$ and if $n \neq 1$ or $B_1 \neq \top$. The clause given above 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 conclusion or *head* is $\bot$ is 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 or $\top$, *definite*, if $m = 1$, and *disjunctive*, if $m \geq 2$.

A *program* is a finite set of clauses. A *constrained* (*unconstrained*, resp.) program is a program containing some (no, resp.) denials. A *positive* (*definite*, resp.) program is a program containing only positive (definite, resp.) clauses. A *disjunctive* program is a program containing some disjunctive clauses.

4 Proof Theory

A system of natural deduction for classical logic [6, 11, 15] is briefly recalled.

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. 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 $O(\mathcal{D}^{F_k})$. $\mathcal{D}^F := \dfrac{\mathcal{D}^{F_1} \ \ldots \ \mathcal{D}^{F_n}}{F}$ is a deduction of F in $\mathcal{T}$ if $\dfrac{F_1 \ \ldots \ F_n}{F}$ 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).

Introduction rules:

$\wedge$I $\quad \dfrac{A \quad B}{A \wedge B}$

$\vee$I$_r$ $\quad \dfrac{A}{A \vee B}$

$\vee$I$_l$ $\quad \dfrac{B}{A \vee B}$

$\rightarrow$I $\quad \dfrac{\begin{matrix}(A)\\ B\end{matrix}}{A \rightarrow B}$

$\forall$I $\quad \dfrac{A}{\forall y A[y/x]}$

$\exists$I $\quad \dfrac{A[t/x]}{\exists x A}$

Elimination rules:

$\vee$E $\quad \dfrac{A \vee B \quad \begin{matrix}(A)\\ C\end{matrix} \quad \begin{matrix}(B)\\ C\end{matrix}}{C}$

$\wedge$E$_r$ $\quad \dfrac{A \wedge B}{A}$

$\wedge$E$_l$ $\quad \dfrac{A \wedge B}{B}$

$\rightarrow$E $\quad \dfrac{A \quad A \rightarrow B}{B}$

$\forall$E $\quad \dfrac{\forall x A}{A[t/x]}$

$\exists$E $\quad \dfrac{\exists y A[y/x] \quad \begin{matrix}(A)\\ B\end{matrix}}{B}$

Absurdity rule:

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

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 $\{\exists x A, B\} \cup O(\mathcal{D}^B)$.

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.

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. $\vdash_c \bot \rightarrow A$. Since minimal logic does not treat the formula $\bot$ differently from an atom, $\nvdash_m \bot \rightarrow A$ [1]. Thus, inconsistencies are "local" in minimal logic because it does not satisfy the "ex falso quodlibet" principle. Since no absurdity rule is allowed for deduction in minimal logic, minimal logic is not complete for first-order logic [1]. Although incomplete for first-order logic, minimal logic is 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, like e.g. a proof of A obtained from a proof of $A \wedge B$ by application of $\wedge \mathrm{E}_r$. Deduction without detours, called *normal deductions,* have been formalized [11]. For the purpose of this article, their definition is not needed. The proof of Theorem 4.1 below relies on two well-known theorems, the "Normal Deduction Theorem" and the "Subformula Property" [11, 1]. Given a program P we shall abuse the notation and also denote by P the set of formulas $\{\forall^* C \mid C \in P\}$.

Theorem 4.1 (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 $P \vdash_m F$.*

Theorem 4.1 is established in [14] for definite – i.e. non-disjunctive – positive unconstrained programs. Related results are given in [13, 10].

4.4 Procedural Semantics

The following Theorem 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 4.2 (Isomorphy Theorem) *Let P be a positive, definite, and unconstrained program, and F an atom or a conjunction of atoms. There exist 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 4.1 and 4.2, minimal logic and SLD-resolution are two equivalent definitions of the procedural semantics of positive, definite, and unconstrained programs. By Theorem 4.2, 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 4.1 it provides with a procedural semantics for disjunctive programs as well.

Like SLD-resolution, minimal logic can be extended with various forms of "negation as failure" [2] 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 formalizing SLD-resolution.

5 Model Theory

Weak interpretations, in which local inconsistencies are possible, are proposed as counterparts to classical logic's interpretations. Although not faithful to minimal logic, they are appropriate for logic programs and databases.

5.1 Weak Models

Definition 5.1 (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* $\bot^{\mathcal{I}}_{pos} \in \{\mathbf{t}, \mathbf{f}\}$ *and* $\bot^{\mathcal{I}}_{neg} = \mathbf{t}$ *to* $\bot$.[1]
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* $\mathbf{t} \in \{p^{\mathcal{I}}_{pos}, p^{\mathcal{I}}_{neg}\}$.[1]
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* $p^{\mathcal{I}}_{pos} \cup p^{\mathcal{I}}_{neg} = D^n$.[1]

A weak interpretation $\mathcal{I}$ *of* $\mathcal{L}$ *is* consistent *if and only if:* $\bot^{\mathcal{I}}_{pos} = \mathbf{f}$, *for every predicate symbol* p *of arity* 0 *in* $\mathcal{L}$, $p^{\mathcal{I}}_{pos} \neq p^{\mathcal{I}}_{neg}$, *and 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 symbols 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 5.2 (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* $\bot$, p *a predicate symbol of arity* 0, q *a predicate symbol of arity* $n \geq 1$, *and* $t_1, ..., t_n$, n *terms.*

1. $\bot^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $\bot^{\mathcal{I}}_{pos} = \mathbf{t}$. $\neg\bot^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
2. $p^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $p^{\mathcal{I}}_{pos} = \mathbf{t}$. $\neg p^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $p^{\mathcal{I}}_{neg} = \mathbf{t}$.
3. $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} \in q^{\mathcal{I}}_{pos}$. $\neg q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $q(t_1, ..., t_n)^{\mathcal{I}}_{\mathcal{V}} \in p^{\mathcal{I}}_{neg}$.
4. $(F \wedge G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. $\neg(F \wedge G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *or* $\neg G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
5. $(F \vee G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *or* $G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. $\neg(F \vee G)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $\neg G^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
6. $(F \rightarrow H)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff if* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$, *then* $H^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$. $\neg(F \rightarrow H)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *and* $\neg H^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
7. $(\forall x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff for all variable assignments* $\mathcal{W}$ *that coincide with* $\mathcal{V}$ *on every variable except possibly on* x, $F^{\mathcal{I}}_{\mathcal{W}} = \mathbf{t}$. $\neg(\forall x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $(\exists x \neg F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
8. $(\exists x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff for some variable assignment* $\mathcal{W}$ *that coincide with* $\mathcal{V}$ *on every variable except possibly on* x, $F^{\mathcal{I}}_{\mathcal{W}} = \mathbf{t}$. $\neg(\exists x F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $(\forall x \neg F)^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.
9. $\neg\neg F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ *iff* $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$.

[1] Possibly $\bot^{\mathcal{I}}_{pos} = \mathbf{t}$, $p^{\mathcal{I}}_{pos} = p^{\mathcal{I}}_{neg} = \mathbf{t}$, and $p^{\mathcal{I}}_{pos} \cap p^{\mathcal{I}}_{neg} \neq \emptyset$.

Note that Definition 5.2 directly defines the truth of negated formulas $\neg F$ without referring to the definition $\neg F := F \rightarrow \bot$.

Definition 5.3 *Let F be a closed formula and $\mathcal{S}$ a set of closed formulas. F is* true *in $\mathcal{I}$, denoted $F^{\mathcal{I}}$:t, iff $F^{\mathcal{I}}_{\mathcal{V}} = \mathbf{t}$ for some variable assignment $\mathcal{V}$. F is* false *in $\mathcal{I}$ iff $\neg F^{\mathcal{I}}$:t. F is* inconsistent *in $\mathcal{I}$ iff F is both, true and false in $\mathcal{I}$. Otherwise, F is* consistent *in $\mathcal{I}$. F is* consistently true *(*consistently false*, resp.) in $\mathcal{I}$, iff F is true (false, resp.) and consistent in $\mathcal{I}$. A weak interpretation $\mathcal{I}$* satisfies *F ($\mathcal{S}$, resp.) iff 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.) iff F (every formula in $\mathcal{S}$, resp.) is consistently true in $\mathcal{I}$.*

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

Proposition 5.4 *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 iff $(\ (A \rightarrow B) \vee (A \rightarrow C)\ )^{\mathcal{I}}$:t.*
2. *$A \vee B$ is consistently true in $\mathcal{I}$ iff at least one of A and B is consistently true in $\mathcal{I}$.*
3. *$A \wedge B$ is consistently true in $\mathcal{I}$ iff both, A and B are consistently true in $\mathcal{I}$.*
4. *If $(H \leftarrow D)^{\mathcal{I}}$:t and if H is consistently false in $\mathcal{I}$, then D is consistently false in $\mathcal{I}$.*

Example 1 In the following examples, a weak interpretation is specified by the atoms (and possibly $\bot$) it satisfies, consistently or not.

1. $\neg\neg p = \bot \leftarrow \neg p$ is satisfied by $\{p, \neg p, \bot\}$ or $\{p\}$.
2. $\{a \leftarrow \neg b\ ;\ b \leftarrow \neg a\}$ is satisfied by $\{a, \neg a, b, \neg b, \bot\}$, 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 $\{a, \neg a, b, \neg b, c, \neg c, \bot\}$, or $\{a, b, \neg b, c, \neg c, \bot\}$, or $\{a, b, c, \neg c, \bot\}$, or $\{a, b, c\}$, or $\{a, b\}$.
4. $\{\bot \leftarrow b\ ;\ b \leftarrow a\ ;\ a\}$ is satisfied by $\{a, \neg a, b, \neg b, \bot\}$ or $\{a, b, \neg b, \bot\}$.
5. $\{b \leftarrow a\ ;\ \bot \leftarrow a\ ;\ a\}$ is satisfied by $\{a, \neg a, b, \bot\}$ or $\{a, \neg a, b, \neg b, \bot\}$.
6. $\{a \leftarrow b\ ;\ b \leftarrow a\}$ is satisfied by $\{a, \neg a, b, \neg b, \bot\}$, or $\{a, b, \neg b, \bot\}$, or $\{a, b\}$, or $\emptyset$.
7. $\{a \vee b\ ;\ \bot \leftarrow a\}$ is satisfied by $\{a, \neg a, \bot\}$ or $\{b\}$.

5.2 Intended 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 5.5 *Let $S \subseteq EHB(\mathcal{L})$. Let A be a ground atom or $\bot$. $\mathcal{H}_w(S)$ 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 $\neg A \in S$.*

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, classical Herbrand interpretations can be represented as usual [8].

Theorem 5.6 *Let P be a set of clauses, $\mathcal{I}$ a weak interpretation, and let $S_\mathcal{I} = \{L \mid L \in EHB(\mathcal{L}) \wedge L^\mathcal{I}\mathbf{:t}\}$. $\mathcal{I}$ is a weak model of P if and only if $\mathcal{H}_w(S_\mathcal{I})$ is a weak Herbrand model of P.*

$\mathcal{H}_w(EHB(\mathcal{L}))$ 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.

Definition 5.7 *Let $\mathcal{H}_w(S_1)$ and $\mathcal{H}_w(S_1)$ be two weak Herbrand interpretations. $\mathcal{H}_w(S_1) \preceq \mathcal{H}_w(S_2)$ iff $S_1 \subseteq S_2$. $\mathcal{H}_w(S_1) \cap \mathcal{H}_w(S_2) := \mathcal{H}_w(S_1 \cap S_2)$.*

Clearly, $\preceq$ extends to weak Herbrand interpretations the usual order relation defined on classical Herbrand interpretations [8]. A (classical) Herbrand model of program P which is minimal for the order $\preceq$ of Definition 5.7 clearly is also minimal in the classical sense. Thus, the intended model of a positive, unconstrained and definite program (as usually defined [8]) is its (unique) minimal weak model. However, not all minimal weak models of a non-positive program convey its intuitive meaning. An example is the minimal model $\mathcal{H}_w(\{p\})$ of $P = \{p \leftarrow \neg p\}$.

Definition 5.8 (Supported Weak Model) *Let P be a set of clauses and $\mathcal{M}$ be a weak model of P. Let A denote a ground atom which is consistently true in $\mathcal{M}$. A is* supported in $\mathcal{M}$ (wrt P) *iff there exists a ground instance $H \leftarrow B$ of a clause in P such that A occurs in H and B is consistently true in $\mathcal{M}$. $\mathcal{M}$ is a* supported weak model of P *iff every ground atom which is consistently true in $\mathcal{M}$ is supported in $\mathcal{M}$ (wrt P).*

$\mathcal{H}_w(EHB(\mathcal{L}))$ is a supported weak Herbrand model of every set of clauses. If $\mathcal{M}$ is a supported weak model of a program P, then by Proposition 5.4 for every ground instance $H \leftarrow B$ of a clause in P, if H is consistently false, then so is also B. However, H might be inconsistent in $\mathcal{M}$ and B, consistently true, as e.g. the second weak model of Example 1 (4) shows.

Definition 5.9 (Intended Models 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 5.10 *Every set of clauses, in particular every program, has an intended model.*

Example 2 Consider the programs of Example 1.

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

Note that disjunctive and non-positive programs might have several intended models. The following proposition shows that intended models interpret programs according to a programmer's intuition.

Proposition 5.11 *Let P be a program, $\mathcal{I}$ an intended model of P, and A a ground atom or the formula $\bot$.*

1. *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 $\mathcal{I}$.*
2. *If A is true in $\mathcal{I}$, then there exists a ground instance $H \leftarrow B$ of a clause in P such that A occurs in H and B is true in $\mathcal{I}$.*

Note that by Definition 5.9, if a ground atom A is consistently true in an intended model $\mathcal{I}$ of a program P, then there exists a ground instance $H \leftarrow B$ of a clause in P such that A occurs in H and B is consistently true in $\mathcal{I}$.

Since for (disjunctive as well as definite) positive and unconstrained programs minimal models (as usually defined [8]) coincide with the intended models of Definition 5.9, Theorem 4.1 establishes the completeness of the minimal logic proof calculus for such programs with respect to the "Compositional Semantics" as defined by Definition 5.9. A completeness proof for programs referring to negation as finite failure is out of the scope of this article, for it would have to refer not only to the proof calculus, but also to the search strategy.

5.3 Compositionality

In this section, the "composition principle" mentioned in Section 2.2 is formalized and the semantics defined in Section 5.2 is shown to fulfill it.

Definition 5.12 *Let A be an atom or the formula $\bot$, and let P and Q be sets of clauses. P* defines *A iff A occurs in an instance of the head of a clause in P. P* depends on *Q iff an atom occurring (positively or negatively) in the body of a clause of P is defined by Q. $EHB(\mathcal{L})_P$ denotes the subset of $EHB(\mathcal{L})$ consisting of formulas defined in P.*

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. For a semantics $\mathcal{S}$, these models will be referred to as *$\mathcal{S}$-intended models.*

Definition 5.13 (Compositional Semantics) *Let $\mathcal{S}$ be a semantics. The semantics $\mathcal{S}$ is* compositional *iff for every programs P and Q such that P does not depend on Q, and 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 $EHB(\mathcal{L})_P$ true (false, inconsistent, resp.) in $\mathcal{M}_P$ is also true (false, inconsistent, resp.) in $\mathcal{M}_{P\cup Q}$.*

Theorem 5.14 *The semantics specified by Definition 5.9 is compositional.*

P and Q might satisfy the conditions of the theorem, and an atom might be consistent in P and inconsistent in $P\cup Q$. A strengthening of Definition 5.13 precluding such cases would require inconsistencies to be global.

5.4 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 extends to disjunctive, possibly constrained programs P as follows [9]: 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 (wrt 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 all the clauses in P, in the heads of which p occurs. The *completed definition of p (wrt 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 program P consists in the union of the following sets: The set of the completed definitions of the predicates occurring in P, Clark's equality theory [2, 8] CET, and $\{\forall^* \ C \mid C \in P\}$.

Theorem 5.15 (Completion Semantics) *Let P be a program and $\mathcal{I}$ a weak Herbrand interpretation. If $\mathcal{I}$ is consistent, then $\mathcal{I}$ is a minimal weak model of $Comp(P)$ iff $\mathcal{I}$ is an intended model of P.*

The following example shows that inconsistent minimal weak models of the completion of a program P are not necessarily supported models of P.

Example 3 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 wrt P in $\mathcal{H}_w(S)$.

6 Conclusion

In this article, a semantics has been proposed according to which, logic programs and deductive databases might specify *local* inconsistencies, thus departing from classical logic in which inconsistencies are *global.*

Local inconsistencies have been motivated on the one hand by practical considerations on database query answering and program composition, on the other hand by proof theoretic considerations. It has been shown that classical logic's global inconsistencies, that make refutation proofs possible, are not needed for proving the completeness of SLD-resolution for positive unconstrained programs. Minimal logic, a natural deduction style weakening of classical logic's proof theory which precludes refutation proofs, has been shown to be sufficient for generalized programs corresponding to deductive databases and disjunctive logic programs, and to formalize SLD-resolution proofs, thus generalizing a former result [14].

In order to provide logic programs and deductive databases with a model theory conforming to the rejection of refutation proofs, a nonclassical model theory allowing local inconsistencies has been proposed. Relying on this model theory, a notion of intended model for a generalization of logic programs has been proposed. The proposed intended models are minimal models specified by sets of ground literals, thus naturally extending the usual definitions for positive, unconstrained programs. The proposed notion of intended model has also been characterized in terms of the completion of a generalized program, thus extending the standard completion semantics. Finally, the proposed semantics has been shown to be *compositional* in the sense that it fulfills a natural program composition requirement. Interestingly, both, the proof and the model theories need no extensions to account for integrity constraints and disjunctive clauses.

The Compositional Semantics seems to well pass the following tests:

- Which models are defined as the one intended by the programmer?
- How simple and intuitive is the definition of these models?

Indeed, by Theorem 5.15 and Proposition 5.11, the Compositional Semantics complies with a programmer's intuition. It has a simple declarative definition in terms of notions well established in logic programming and databases.

Further research on the subject is needed.

First, an inductive (or fixpoint-like) definition of the proposed intended models has to be investigated.

Second, the Compositional Semantics need to be tested against practical database applications and logic programs. Also, it is desirable to investigate how far it fulfills natural requirements. [3, 4] proposes a catalogue of such requirements, one of which, modularity, corresponds to compositionality. Other properties, such as stability under partial evaluation, seem to be fulfilled by the Compositional Semantics.

Third, a detailed comparison of the semantics proposed in this article with other approaches is desirable. In particular, a comparison with the Stable Model [5] and Well-Founded [16] Semantics deserves further research. It seems possible to adapt the definition of stable models to weak interpretations, what would yield another declarative definition of the Compositional Semantics. Although it is compositional, the Well-Founded Semantics is more difficult to compare with the Compositional Semantic, because its notion of model is weaker: It formalizes in fact entailment in minimal models. It seems that the well-founded model of a program is related to the intersection of its (Compositional Semantics) intended models.

Fourth, *consistent programs,* i.e. programs with no inconsistent intended models, deserve investigations. Syntactical characterizations of consistent programs would be interesting. Call-consistency [12] seems to be a necessary condition for program consistency.

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 a program.

Acknowledgements

Norbert Eisinger, Heribert Schütz, and the anonymous referees are thanked for useful comments on a preliminary version of this paper, 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. *A Compositional Semantics for Logic Programs and Deductive Databases.* Res. Report PMS-FB-4, Comp. Sc. Dpt., Munich University, 1996. http://www.pms.informatik.uni-muenchen.de/publikationen/

[2] K. Clark. Negation as Failure. *Logic and Databases.* 293-322, Plenum Press, New York, 1978.

[3] J. Dix. Classifying Semantics of Disjunctive Logic Programs. *Proc. Joint Int. Conf. and Symp. on Logic Programming.* 798-812, MIT Press, 1992.

[4] J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: I. Strong Properties, II. Weak Properties. *Fundamentae Informatica.* Vol. 22, No. 3, 227-288, 1995.

[5] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. *Proc. 5th Int. Conf. and Symp. on Logic Programming.* 1070-1080, MIT Press, 1988.

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

[7] K. Kunen. Declarative Semantics of Logic Programming. *Bull. Europ. Assoc. Theoretical Computer Science.* Vol. 44, 147-167, 1991.

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

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

[10] 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.

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

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

[13] H. Schwichtenberg. Minimal From Classical Proofs. *Computer Science Logic.* 326-328, Springer Verlag LNCS 626, 1992.

[14] R. F. Stärk. A Direct Proof for the Completeness of SLD-Resolution. *Computer Science Logic.* 382-383, Springer-Verlag LNCS 440, 1990.

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

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