Fundamenta Informaticae (1997)
IOS Press

Logic Programming from the Perspective of Algebraic Semantics*

Jan A. Plaza
Department of Mathematics and Computer Science
University of Miami
Coral Gables, Florida 33124
U.S.A.

janplaza@math.cs.miami.edu
http://www.cs.miami.edu/~janplaza

November 18, 1996

To the memory of Professor Helena Rasiowa

Abstract. We present an approach to foundations of logic programming in which the connection with algebraic semantics becomes apparent. The approach is based on omega-Herbrand models instead of conventional Herbrand models. We give a proof of Clark's theorem on completeness of SLD-resolution by methods of the algebraic semantics. We prove the existence property for definite programs.

Keywords: Rasiowa-Sikorski lemma, algebraic semantics, logic programming, completeness of SLD-resolution.

1. Introduction

A part of work of Professor Helena Rasiowa was devoted to the development of methods of algebraic semantics. The very possibility of applying algebraic methods to logics containing quantifiers is due to a fundamental result which is commonly called the Rasiowa-Sikorski lemma.

The Rasiowa-Sikorski lemma made it possible to investigate algebraic aspects of many fundamental problems in first-order logics (classical and non-classical) and in the set theory. In the former domain they included Gödel's completeness theorem, completeness theorems for various non-classical logics, Skolem-Löwenheim theorem, Craig interpolation theorem, and existence and disjunction properties for intuitionistic logic, (cf. [15], [12] and references therein.) In the later domain the lemma can be used in the proofs of independence of axioms from a formal theory of sets. The lemma will be of a crucial importance for the considerations of this paper.

The algebraic semantics was the area of the most active research in 1950's and 1960's. Logic programming came into being in 1970's. Although in the theory of logic programming

*Work partially supported by NSF grant IRI 9308970.

three valued models are sometimes considered, the two domains never met in a non-superficial way.

This paper is an invitation for logicians familiar with the algebraic semantics to explore foundations of logic programming. It is also an invitation for those working on foundations of logic programming to consider possible uses of the methods of algebraic semantics.

We intend to make this paper possibly self-contained, so in Sections 2-4 we will recall basic definitions of universal algebra, Boolean algebras and algebraic semantics. In Section 5 we will define omega-Herbrand interpretations and reformulate the foundations of logic programming to make use of this notion. In Section 6 we will give two auxiliary results whose proofs are very straightforward if one uses methods of algebraic semantics. These two results will lead in Section 7 to corollaries concerning logic programming. Among the corollaries there is Clark's completeness theorem as well as a number of new results.

Assumption 1.1. Throughout the paper we consider only first order languages with finite alphabets and without equality, unless explicitly assumed otherwise.

2. Universal Algebra

Universal algebra is concerned with analysis of properties which are common to all algebras independently of their type. Its definitions apply equally well to groups, rings, or Boolean algebras. This last case will be of importance to this paper.

By an *algebra* one understands any tuple $\langle A, op_1, \dots op_n \rangle$ where A is a nonempty set and $op_1, \dots op_n$ are arbitrary functions on A. The set is called a *universe*, the functions are called *operations*. Operations of arity 0 are allowed, and they are called *constants*. By the *signature* of the algebra above one understands the sequence $\langle arity(op_1), \dots, arity(op_n) \rangle$.

Given two algebras of $\langle A, op_1, \dots op_n \rangle$ and $\langle A', op'_1, \dots op'_n \rangle$ of the same signature, one defines a *homomorphism* as a function $h : A \longrightarrow A'$ which preserves all the operations i.e. satisfies the conditions $h(op_i(x_1, \dots, x_{arity(op_i)})) = op'_i(h(x_1), \dots, h(x_{arity(op_i)}))$.

By a *congruence* one understands an equivalence relation on the universe which is compatible with all the operations: if $x_1 \sim x'_1, \dots, x_{arity(op_i)} \sim x'_{arity(op_i)}$ then $op_i(x_1, \dots, x_{arity(op_i)}) \sim op_i(x'_1, \dots, x'_{arity(op_i)})$. The equivalence class determined by an element x will be denoted by $\| x \|$.

Given a congruence $\sim$ on an algebra $\boldsymbol{A}$, by a *quotient algebra* $\boldsymbol{A}/_\sim$ one understands the algebra whose signature is the same as that of $\boldsymbol{A}$, whose universe is the set of equivalence classes of $\sim$, and whose operations are defined in the natural way:

$$op_i(\| x_1 \|, \dots, \| x_{arity(op_i)} \|) = \| op_i(x_1, \dots, x_{arity(op_i)}) \|.$$

The function h which maps every element x to its equivalence class $\| x \|$ is a homomorphism from $\boldsymbol{A}$ onto the quotient algebra $\boldsymbol{A}/_\sim$ and it is called a *canonical quotient homomorphism*.

3. Boolean Algebras

In this section we will sketch certain basic definitions and results needed for the algebraic semantics for first-order classical logic. For more details the reader is referred to [15].

Of all Boolean algebras, the 2-element one is most commonly used in computer science:

$$\mathbf{2} = \langle \{0,1\}, \bot, \top, \vee, \wedge, - \rangle$$

(where $\bot$ is interpreted as 0 and $\top$ is interpreted as 1.) The five operations are called respectively: *zero, unit, join, meet* and *complement.*

To define Boolean algebras in full generality, consider the set of all equalities which are satisfied in **2**. By a *Boolean algebra* one understands any algebra of the same signature as **2** which satisfies all these equalities. (A finite axiomatization could be given as well.) A partial order can be induced on any Boolean algebra: $x \leq y$ iff $x \wedge y = x$. The join operation $\vee$ corresponds in this order to the least upper bound of two elements. The meet operation $\wedge$ corresponds in this order to the greatest lower bound of two elements. Additional operations $\rightarrow$ and $\leftrightarrow$ can be defined from basic operations of the Boolean Algebra: $x \rightarrow y = -x \vee y$ and $x \leftrightarrow y = (x \rightarrow y) \wedge (y \rightarrow x)$. A Boolean algebra is called *non-degenerate* if $\top \neq \bot$.

For example, the algebra $\boldsymbol{P}(X)$ of subsets of a set X is a Boolean algebra (with the five operations interpreted respectively as $\emptyset$, X, $\cup$, $\cap$ and $-$). If the set X is non-empty then $\boldsymbol{P}(X)$ is non-degenerate. Also the algebra of all finite and co-finite subsets of an infinite set is a Boolean algebra. Another example: a Cartesian product of two Boolean algebras (with pointwise operations) is a Boolean algebra.

A *Boolean homomorphism* is a function between two Boolean algebras which preserves the five operations. Recall that in the domain of abstract algebra, ideals of rings are defined to be kernels of homomorphisms. In the case of algebraic semantics a dual notion is especially useful: *filters* are co-kernels of homomorphisms, i.e. every filter $\nabla \subseteq \boldsymbol{A}$ is determined by a homomorphism $h : \boldsymbol{A} \longrightarrow \boldsymbol{A'}$ in such a way that ∇ it is the set of all the elements of $\boldsymbol{A}$ which are mapped to $\top$. The following characterization is useful: $\nabla \subseteq \boldsymbol{A}$ is a filter iff

1. $\nabla \neq \emptyset$,
2. $x \in \nabla$ and $x \leq y$ implies $y \in \nabla$,
3. ∇ is closed under $\wedge$.

In Boolean algebras *maximal filters* are characterized by any of the following additional properties:

4. $x \vee y \in \nabla$ implies $x \in \nabla$ or $y \in \nabla$, and $\bot \notin \nabla$,

4'. for every $x \in \boldsymbol{A}$ exactly one of x and $-x$ belongs to ∇.

According to the definition above, Boolean algebra contains only binary operations of joins and meets. In some situations infinite joins or meets make sense. For instance in $\boldsymbol{P}(X)$ for every family of elements $B \subseteq \boldsymbol{P}(X)$ we can define $\bigvee B = lubB$. Still, if $\boldsymbol{P}(X)$ is treated as a Boolean algebra it is not possible to express these infinite joins or meets in its language. An *infinite join* can be coded as $\langle \bigvee, \{a_i \mid i \in I\}, a \rangle$ meaning that $\bigvee \{a_i \mid i \in I\} = a$, and similarly for *infinite meets.* One can consider a Boolean algebra with an additional set Q of infinite joins and meets; such an algebra will be called a *Q-Boolean algebra.*

Given a Q-Boolean algebra $\boldsymbol{A}$ and a Q'-Boolean algebra $\boldsymbol{A'}$, by a *Q-homomorphism* one understands a Boolean homomorphism from $\boldsymbol{A}$ into $\boldsymbol{A'}$ which preserves the joins and meets from Q. Not every Boolean homomorphism from a Q-Boolean algebra $\boldsymbol{A}$ into a Q'-Boolean algebra $\boldsymbol{A'}$ is a Q-homomorphism.

By a *Q-filter in $\boldsymbol{A}$* one understands a co-kernel of a Q-homomorphism from $\boldsymbol{A}$ onto **2**. By another characterization, a Q-filer is a maximal (!) filter which preserves the infinite joins and meets from the set Q:

1. if $\langle \bigvee, \{a_i \mid i \in I\}, a \rangle \in Q$ and $a \in \nabla$ then there exists i such that $a_i \in \nabla$,
2. if $\langle \bigwedge, \{a_i \mid i \in I\}, a \rangle \in Q$ and $a_i \in \nabla$ for every $i \in I$ then $a \in \nabla$.

Every filter determines a congruence relation: $x \sim x'$ iff $(x \leftrightarrow x') \in \nabla$. Thus, one can define a *quotient of a Boolean algebra by a filter* as the quotient by the corresponding congruence: $\boldsymbol{A}/\nabla = \boldsymbol{A}/_{\sim}$. This idea generalizes in a natural way to the case of algebras with infinite joins and meets: one can divide a Q-Boolean algebra by a Q-filter. A quotient of a Boolean algebra by a maximal filter results in the two-element Boolean algebra; as every Q-filter is maximal, the quotient of a Q-Boolean algebra by a Q-filter results in the two-element Boolean algebra.

Now we are ready to state a fundamental result of this theory.

Theorem 3.1. (Rasiowa and Sikorski, 1950)
Let $\boldsymbol{A}$ be a non-degenerate Boolean algebra with a distinguished countable set Q of infinite joins or meets. Then the following conditions hold:
1. The set of all Q-filters is of the second category in the Stone space $S(\boldsymbol{A})$.
2. Every non-zero element x of $\boldsymbol{A}$ belongs to a Q-filter.
3. There exists a Q-isomorphical embedding of $\boldsymbol{A}$ into the algebra $\boldsymbol{P}(X)$ of all subsets of a set X. □

Point 1 is called the Rasiowa-Sikorski lemma; points 2 and 3 follow from 1. The conceptual importance of the lemma is well visible in point 3, which spans the category of algebras and the category of sets. For a logician, point 2 provides a bridge between Boolean algebras (which represent syntactical aspects the classical logic) and the Tarski semantics.

A crucial corollary to point 2 is that, under the assumptions of the theorem, there exists a Q-homomorphism from $\boldsymbol{A}$ onto $\mathbf{2}$. Indeed, by point 1 there exists a Q-filter ∇ in $\boldsymbol{A}$, and one can verify that the canonical homomorphism from $\boldsymbol{A}$ onto the quotient $\boldsymbol{A}/\nabla$ satisfies the required conditions.

4. Algebraic Semantics

Now, as an example representative of the algebraic semantics, we will sketch the definitions and constructions leading to the proof of the model existence theorem: every consistent theory has a (two-valued) model. From this presentation the reader should be able to see the main ideas behind the algebraic semantics. It is however not in the scope of this paper to show how this ideas can be further developed and extended onto non-classical logics. For more details the reader is referred to [15].

As mentioned before, we will consider a first order language without equality and assume that the alphabet contains finitely or countably many function symbols and predicate symbols and countably many individual variables $x_0, x_1, \ldots$. (Let us mention that by using additional machinery, results mentioned in this section could be extended to arbitrary, possibly uncountable, first order languages.)

In the Tarski semantics, which is standard for the first-order classical logic, one considers interpretations of languages in relational structures. A relational structure has a nonempty universe D, on which functions are defined, which correspond to the function symbols of the language, and on which relations are defined, which correspond to the predicate symbols. The relations have values in the two-element Boolean algebra and serve as interpretations of predicate symbols from the language. Given a predicate symbol p, with a corresponding relation $\boldsymbol{p}$, and a tuple $\bar{e}$ of elements of the universe, $\boldsymbol{p}(\bar{e})$ has either value *true* or *false*, i.e. a value in the algebra $\mathbf{2}$. This interpretation of predicate symbols extends by using the operations of the algebra $\mathbf{2}$ to the interpretation of arbitrary formulas. For every formula B and a valuation $\nu : Var \longrightarrow D$ of variables, $B^I[\nu]$ obtains a value in the algebra $\mathbf{2}$.

In order to obtain the definition of *algebraic interpretations*, allow in the Tarski definition arbitrary non-degenerate Q-Boolean algebra $\boldsymbol{A}$, allow $\boldsymbol{p}(\bar{e})$ to have arbitrary value in $\boldsymbol{A}$ but require that for every formula $B(\bar{x}_1, \bar{x}_2)$, and every tuple $\bar{e}_2$, the following infinite meets and joins are in Q:

$$(\forall_{\bar{x}_1} B(\bar{x}_1, \bar{x}_2))^I[\bar{e}_2/\bar{x}_2] = \bigwedge\nolimits_{\bar{e}_1} (B(\bar{x}_1, \bar{x}_2))^I[\bar{e}_1/\bar{x}_1, \bar{e}_2/\bar{x}_2]$$
$$(\exists_{\bar{x}_1} B(\bar{x}_1, \bar{x}_2))^I[\bar{e}_2/\bar{x}_2] = \bigvee\nolimits_{\bar{e}_1} (B(\bar{x}_1, \bar{x}_2))^I[\bar{e}_1/\bar{x}_1, \bar{e}_2/\bar{x}_2]$$

These infinite meets and joins are used to interpret $\forall$ and $\exists$. An algebraic interpretation is an *algebraic model for a formula* B if it gives the value $\top$ to the universal closure of this formula.

The set $Form(\mathcal{L})$, of all formulas of a first-order language $\mathcal{L}$, can be considered to be an algebra with operations $\top, \bot, \wedge, \vee, -$, for instance, the operation $\wedge$ applied to two formulas B_1 and B_2 results in the formula $B_1 \wedge B_2$. This algebra is not a Boolean algebra; notice that the operation $\wedge$ is not commutative: applying $\wedge$ to B_1 and B_2 yields a result different than applying it to B_2 and B_1; the two results are logically equivalent, but they are different.

Given a theory $\mathcal{T}$ one can consider the following relation on formulas: $B_1 \sim B_2$ iff $\mathcal{T} \vdash B_1 \equiv B_2$. This relation is a congruence in the algebra of formulas and the quotient algebra $Form(\mathcal{L})/_\sim$ is a Boolean algebra. This algebra is denoted by $\boldsymbol{A}(\mathcal{T})$ and is called a *Lindenbaum algebra of $\mathcal{L}$ modulo $\mathcal{T}$*. The elements of $\boldsymbol{A}(\mathcal{T})$ are classes of formulas which are equivalent in $\mathcal{T}$, thus different elements of $\boldsymbol{A}(\mathcal{T})$ can be intuitively understood as abstract ideas which do not coincide in the context of $\mathcal{T}$. For instance formulas $B_1 \wedge B_2$ and $B_2 \wedge B_1$ represent the same element of $\boldsymbol{A}(\mathcal{T})$ and we could say that they represent the same idea. If the theory $\mathcal{T}$ is consistent, the Lindenbaum algebra $\boldsymbol{A}(\mathcal{T})$ is non-degenerate.

We will see that every consistent theory $\mathcal{T}$ has a natural algebraic interpretation I. The universe of this interpretation is the set of all terms in the language of $\mathcal{T}$ (the terms may contain variables.) The function symbols are interpreted on this universe in the natural way: given a function symbol f, for any tuple $\bar{t}$ of elements of I, the interpretation is $\boldsymbol{f}(\bar{t}) = f(\bar{t})$. The relations on the universe are interpreted in the Lindenbaum algebra $\boldsymbol{A}(\mathcal{T})$: given a predicate symbol p, for any tuple $\bar{t}$ of elements of I, the interpretation is $\boldsymbol{p}(\bar{t}) = \| p(\bar{t}) \|$. This interpretation is a model for $\mathcal{T}$ and it is called *the canonical algebraic model for $\mathcal{T}$*.

By the construction of the canonical model we see that the statement: "Every consistent theory has an algebraic model" is obvious; a more interesting question is whether consistent theory has a two valued model. Given the Rasiowa-Sikorski lemma, the construction of such a model is immediate. By point 2 of Theorem 3.1. there exists a Q-filter ∇ in $\boldsymbol{A}(\mathcal{T})$. Now consider the canonical algebraic model for $\mathcal{T}$ and divide the algebra $\boldsymbol{A}(\mathcal{T})$ by ∇.

5. Foundations of Logic Programming using omega-Herbrand Interpretations

In this section we will sketch certain basic definitions and results of the theory of logic programming. We will concentrate on the material relevant to the Clark's completeness theorem for the SLD-resolution. The reader is referred to [8] or [1] for more details, and to [2] for an overview of the new directions of research.

Unlike other presentations of this topic, we will use omega-Herbrand interpretations in place of the conventional Herbrand interpretations. This change will allow us to make a connection with the algebraic semantics.

Following conventions of the set theory, the set of natural numbers $\{0, 1, 2, \ldots\}$ will be denoted by ω – the Greek letter omega; the formula $i < \omega$ means the same as $i \in \omega$. According to the standard convention for first-order languages, individual constants are considered to be function symbols of arity 0. In logical formulas the symbol $\leftarrow$ stands for the reversed implication $A \leftarrow B$ is read A *provided* B, and means $B \rightarrow A$; the formula $\leftarrow B$ means $\bot \leftarrow B$ which is equivalent to $\neg B$; the formula $A \leftarrow$ means $A \leftarrow \top$ which is equivalent to A. According to the terminology of logic programming, a *ground term* is a term with no variables, a *ground formula* is a formula without variables.

By a *definite clause* or a *Horn clause* one understands any formula $A \leftarrow A_1 \wedge \ldots \wedge A_n$ where $A, A_1, \ldots, A_n$ are atomic formulas ($n = 0$ is allowed, in which case the clause is equivalent to A and is called a *fact*.) By a *definite program* one understands a finite set of definite clauses. By a *definite goal* one understands any formula $\leftarrow A_1 \wedge \ldots \wedge A_n$ where $A_1, \ldots, A_n$ are atomic formulas.

With every definite program P one associates a set of formulas P^* obtained in the following way. Transform every clause: $p(\bar{t}) \leftarrow B$ in P to $p(\bar{x}) \leftarrow \exists_{\bar{y}}((\bar{x} = \bar{t}) \wedge B)$, where all the variables in the sequence $\bar{x}$ are distinct, and where $\bar{y}$ is the sequence of all free variables that occur in $(\bar{x} = \bar{t}) \wedge B$ but do not occur in $p(\bar{x})$. Then for every predicate p, list all statements defining this predicate:

$$p(\bar{x}) \leftarrow B_1 \ , \ \ldots \ , \ p(\bar{x}) \leftarrow B_n,$$

and form $p(\bar{x}) \leftarrow B_1 \vee \ldots \vee B_n$. If there are no statements defining p, form $p(\bar{x}) \leftarrow \bot$. Now turn every $\leftarrow$ into $\equiv$. The set of formulas obtained in this way is denoted by P^*. (This construction was a part of the process of obtaining program completion $Comp(P)$ in [4].)

The *SLD-resolution*, introduced in [7], can be viewed as a nondeterministic algorithm which takes as input a definite program and goal $P, \leftarrow B$ and attempts to construct so called *SLD-refutation*. This construction is either successful or fails or does not halt. If the construction of the SLD-refutation is successful the algorithm returns as output a substitution for the variables of B. We will not quote the exact definition, but we will mention these properties of SLD-resolution which are of importance to our considerations.

The *soundness theorem for the SLD-resolution* of Clark [5] states that for every definite program and goal $P, \leftarrow B$, if θ is a substitution computed by the SLD-resolution then $P \vdash B\theta$. Clark proved also completeness of SLD-resolution i.e. a certain version of the converse of this implication. Later in this paper we will give a proof of Clark's completeness result using methods of algebraic semantics. The soundness and completeness together can be paraphrased in an informal way by saying that the procedural meaning of the program P, as determined by the SLD-resolution, is the same as the declarative meaning of the set of formulas P in classical logic.

Now let us recall the definition of Herbrand universes. Consider a first-order language $\mathcal{L}$. If $\mathcal{L}$ contains at least one function symbol define $\mathcal{L}'$ as $\mathcal{L}$, otherwise define $\mathcal{L}'$ as an extension of $\mathcal{L}$ obtained by adding to its alphabet one new individual constant. By the *Herbrand universe* $U_{\mathcal{L}}$ *for* $\mathcal{L}$ one understands the set of all ground terms of the language $\mathcal{L}'$.

Herbrand universes and related notions are basic objects used in the theory of logic programming. One could argue however that many properties expected of Herbrand interpretations fail to hold, and that the proofs which use Herbrand universes are not always elegant and contain many seemingly unavoidable technical details. To change this situation we introduced in [9, 10] a notion of an omega-Herbrand universe.

Definition 5.1. [omega-Herbrand universe]
Let $\mathcal{L}$ be a first-order language and let $\mathcal{L}^{\mathcal{K}}$ result by adding countable set $\mathcal{K} = \{k_i \mid i < \omega\}$ of new individual constants to the alphabet of $\mathcal{L}$. By the *omega-Herbrand universe* $U^{\omega}_{\mathcal{L}}$ *for* $\mathcal{L}$ we understand the set of all ground terms of the language $\mathcal{L}^{\mathcal{K}}$. We refer to members of $U^{\omega}_{\mathcal{L}}$ as *elements*. Members of the set $\mathcal{K}$ will be called *free elements*.

Let $\mathcal{L}'$ be be the language obtained by removing from $\mathcal{L}$ all predicate symbols except equality; by $\mathbf{U}^{\omega}_{\mathcal{L}}$ we denote the (two-valued) interpretation for the language $\mathcal{L}'$ whose universe is $U^{\omega}_{\mathcal{L}}$ and in which function symbols are interpreted in the natural way. □

In algebraic terms, both the Herbrand universe $U_{\mathcal{L}}$ and the omega-Herbrand universe $U^{\omega}_{\mathcal{L}}$ are free algebras, but the difference between them lies in the number of the free generators: there are ω free generators in $U^{\omega}_{\mathcal{L}}$ but only one or none in $U_{\mathcal{L}}$. This detail causes significant differences in the properties of these notions. We will come back to this issue in Section 7.

The first-order theory which characterizes omega-Herbrand universes was considered in [9] and [16].

Equipped with the notion of omega-Herbrand universe we define notions analogous to those for conventional Herbrand universes.

Definition 5.2.

1. By the *omega-Herbrand base for* $\mathcal{L}$ we understand the set $B^{\omega}_{\mathcal{L}}$ of all ground atomic formulas in $\mathcal{L}^{\mathcal{K}}$.

2. By an *omega-Herbrand interpretation for* $\mathcal{L}$ we understand any subset of $B^\omega_\mathcal{L}$. (Let us remark that omega-Herbrand interpretations should be viewed as conventional Tarski interpretations whose universe is $U^\omega_\mathcal{L}$ and in which the function symbols are interpreted in the natural way. The subset of $B^\omega_\mathcal{L}$ mentioned above specifies the interpretation of predicate symbols in this universe. Identifying an omega-Herbrand interpretation with a subset of $B^\omega_\mathcal{L}$ could paraphrased in the language of model theory by saying that the omega-Herbrand interpretation is uniquely determined by its diagram.)
3. Let Γ be a set of formulas. By an *omega-Herbrand model for* Γ we understand any omega-Herbrand interpretation that is a model for Γ.
4. The class of omega-Herbrand interpretations for $\mathcal{L}$ (treated as subsets of $B^\omega_\mathcal{L}$) are ordered by inclusion. If among omega-Herbrand models for P there is the least model in this order, it is denoted M^ω_P and is called the *least omega-Herbrand model for* P.
5. For a definite program P, the operator $T^\omega_P : 2^{B^\omega_\mathcal{L}} \longrightarrow 2^{B^\omega_\mathcal{L}}$ on the lattice of omega-Herbrand interpretations is defined as follows: For any $I \subseteq B^\omega_\mathcal{L}$,
$$T^\omega_P(I) \;=\; \{A[\nu] \in B^\omega_\mathcal{L} \mid (A \leftarrow B) \in P \text{ and } I \models B[\nu]\}.$$
6. If P is a definite program in language $\mathcal{L}$, then by the *SLD-ω-success-set of* P we understand the set $SLDss^\omega_\mathcal{L}(P) = \{A \in B^\omega_\mathcal{L} \mid P; \leftarrow A \text{ has an SLD-refutation }\}$. □

In the new setting we still have results analogous to those for conventional Herbrand interpretations.

Let us remark that for a definite program P the least omega-Herbrand model M^ω_P always exists and can be obtained as intersection of all omega-Herbrand models for P.

Before we formulate the next theorem we need to explain the notation. One defines $T^\omega_P{\uparrow}\alpha$ as the result of iterating the operator T^ω_P α times starting from the least element of the lattice $2^{B^\omega_\mathcal{L}}$. More formally: $T^\omega_P{\uparrow}0 = \bot$, $T^\omega_P{\uparrow}\alpha + 1 = T^\omega_P(T^\omega_P{\uparrow}\alpha)$, and for limit ordinals λ, $T^\omega_P{\uparrow}\lambda = \bigcup_{\alpha<\lambda} T^\omega_P{\uparrow}\alpha$. The symbol $lfpT^\omega_P$ is used only if the operator T^ω_P has the least-fix point, and it denotes this least fix-point.

The following theorem parallels the results of [17] and [3] using omega-Herbrand interpretations instead of conventional Herbrand interpretations.

Theorem 5.1.
For any definite program P,
$$SLDss^\omega_\mathcal{L}(P) \;=\; T^\omega_P{\uparrow}\omega \;=\; lfpT^\omega_P \;=\; M^\omega_P \;=\; \{A \in B^\omega_\mathcal{L} \mid P \vdash A\}$$ □

Proofs of these equalities are analogous to the proofs in [17],[3] or in [8].

Now let us consider the semantical implication $\models$ restricted to omega-Herbrand interpretations.

Definition 5.3. Let $\Gamma \cup \{B\}$ be a set of formulas in first-order language $\mathcal{L}$ (possibly containing the symbol of equality). We write $\Gamma {\models}^\omega_\mathcal{L} B$ to mean: for every omega-Herbrand interpretation I for $\mathcal{L}$, $I \models \Gamma$ implies $I \models B$. □

By virtue of the next proposition, in all situations which are of a practical importance for logic programming we can write $\Gamma {\models}^\omega B$ instead of $\Gamma {\models}^\omega_\mathcal{L} B$.

Proposition 5.1. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two countable first order languages containing a set of formulas $\Gamma \cup \{B\}$. (The languages may contain the symbol of equality.) Then, the conditions $\Gamma {\models}^\omega_{\mathcal{L}_1} B$ and $\Gamma {\models}^\omega_{\mathcal{L}_2} B$ are equivalent. □

Before we give the proof let us recall that given languages $\mathcal{L}_1 \subseteq \mathcal{L}_2$, an interpretation I_1 for $\mathcal{L}_1$ and an interpretation I_2 for $\mathcal{L}_2$, one calls I_2 an *extension* of I_1 if the universe of I_1 is contained in the universe of I_2, and functions and relations of I_2 restricted to the universe of I_1 coincide with those of I_1; one calls I_2 an *expansion* of I_1 if I_2 is an extension of I_1 and the universes of I_1 and I_2 are the same.

Proof:
Without loosing generality we may assume that the predicate symbols of $\mathcal{L}_1$ are the same as the predicate symbols of $\mathcal{L}_2$ and they are precisely the predicate symbols occurring in $\Gamma \cup \{B\}$; this is true because for any two languages $\mathcal{L}_1'$ and $\mathcal{L}_2'$ containing $\Gamma \cup \{B\}$ and such that their alphabets contain the same function symbols, $\Gamma \models^{\omega}_{\mathcal{L}_1'} B$ and $\Gamma \models^{\omega}_{\mathcal{L}_2'} B$ are equivalent.

Also, without loosing generality we may assume that $\mathcal{L}_1 \subseteq \mathcal{L}_2$; indeed, if the equivalence is established for languages satisfying this additional assumption then for arbitrary $\mathcal{L}_1'$ and $\mathcal{L}_2'$ containing $\Gamma \cup \{B\}$ we can reason: $\Gamma \models^{\omega}_{\mathcal{L}_1'} B$ iff $\Gamma \models^{\omega}_{\mathcal{L}(\Gamma \cup \{B\})} B$ iff $\Gamma \models^{\omega}_{\mathcal{L}_2'} B$ (where $\mathcal{L}(\Gamma \cup \{B\})$ denotes the language containing exactly the symbols of $\Gamma \cup \{B\}$).

For the remaining part of the proof these additional assumptions will be in force.

We will show that $\Gamma \models^{\omega}_{\mathcal{L}_1} B$ implies $\Gamma \models^{\omega}_{\mathcal{L}_2} B$. Assume that $\Gamma \models^{\omega}_{\mathcal{L}_1} B$. Let I be an interpretation for $\mathcal{L}_2$ which expands $\mathbf{U}^{\omega}_{\mathcal{L}_2}$ such that $I \models \Gamma$. The task will be to show that $I \models B$. Let $I' = I|_{\mathcal{L}_1}$ be the reduct of I to the language $\mathcal{L}_1$. Thus I' is an interpretation for $\mathcal{L}_1$, over $U^{\omega}_{\mathcal{L}_2}$ with the natural interpretation of functions from $\mathcal{L}_1$ and such that $I' \models \Gamma$. There exists an isomorphism $h : \mathbf{U}^{\omega}_{\mathcal{L}_1} \longrightarrow \mathbf{U}^{\omega}_{\mathcal{L}_2}|_{\mathcal{L}_1}$. We will define an interpretation I'' for $\mathcal{L}_1$ which expands $U^{\omega}_{\mathcal{L}_1}$: for every predicate letter $p^{I''}(e_1, ..., e_n)$ iff $p^{I'}(h(e_1), ..., h(e_n))$. By induction on the structure of formula γ one can show that

$$(*) \quad I'' \models \gamma[e_1/x_1, e_2/x_2, ...] \text{ iff } I' \models \gamma[h(e_1)/x_1, h(e_2)/x_2, ...].$$

Thus I'' is an interpretation for $\mathcal{L}_1$ which expands $\mathbf{U}^{\omega}_{\mathcal{L}_1}$ such that $I'' \models \Gamma$. By $(*)$, $I' \models B$, and obviously $I \models B$. This completes the proof of the implication to the right.

Now, we will show that $\Gamma \models^{\omega}_{\mathcal{L}_2} B$ implies $\Gamma \models^{\omega}_{\mathcal{L}_1} B$. Assume that $\Gamma \models^{\omega}_{\mathcal{L}_2} B$. Let I be an interpretation for $\mathcal{L}_1$, expanding $\mathbf{U}^{\omega}_{\mathcal{L}_1}$ such that $I \models \Gamma$. The task will be to show that $I \models B$. Let $I' = I|_{\mathcal{L}_1}$ be the reduct of I to the language $\mathcal{L}_1$. Thus I' is an interpretation for $\mathcal{L}_1$, over $U^{\omega}_{\mathcal{L}_2}$ with the natural interpretation of functions from $\mathcal{L}_1$ and such that $I' \models \Gamma$. There exists an isomorphism $h : \mathbf{U}^{\omega}_{\mathcal{L}_2}|_{\mathcal{L}_1} \longrightarrow \mathbf{U}^{\omega}_{\mathcal{L}_1}$. We will define an interpretation I' for $\mathcal{L}_1$, over $U^{\omega}_{\mathcal{L}_2}$: for every predicate symbol p we define: $p^{I'}(e_1, ..., e_n)$ iff $p^{I}(h(e_1), ..., h(e_n))$. By induction on the structure of formula γ one can show that

$$(**) \quad I' \models \gamma[e_1/x_1, e_2/x_2, ...] \text{ iff } I \models \gamma[h(e_1)/x_1, h(e_2)/x_2, ...].$$

Thus I' is an interpretation for $\mathcal{L}_1$, over $U^{\omega}_{\mathcal{L}_2}$, extending $\mathbf{U}^{\omega}_{\mathcal{L}_1}$ such that $I' \models \Gamma$. By $(**)$, $I' \models \Gamma$. Let I'' be an expansion of I' in which the functions of $\mathcal{L}_2 - \mathcal{L}_1$ are interpreted in a natural way. I'' is an interpretation for $\mathcal{L}_2$, expanding $U^{\omega}_{\mathcal{L}_2}$, such that $I'' \models \Gamma$. By the initial assumption of this part of the proof, $I'' \models B$ and obviously also $I' \models B$. By $(**)$, $I \models B$. This completes the proof of the implication to the left. $\square$

Remark 5.1. One could define relation $\models^{0}_{\mathcal{L}}$ as the restriction of $\models$ to the class of those interpretations of $\mathcal{L}$ which expand the conventional Herbrand universe for $\mathcal{L}$. However this relation would essentially depend on the choice $\mathcal{L}$. Indeed, if $\mathcal{L}_1$ is the first-order language with the alphabet containing unary predicate symbol p and a constant c, and if $\mathcal{L}_2$ contains in addition a constant c', then we have $p(c) \models^{0}_{\mathcal{L}_1} \forall_x p(x)$ but $p(c) \not\models^{0}_{\mathcal{L}_2} \forall_x p(x)$. This is one of the reasons why we needed to reformulate the theory of logic programming, basing it on omega-Herbrand interpretations instead of conventional Herbrand interpretations. $\square$

Remark 5.2. (omega-Herbrand interpretations versus canonical algebraic interpretations)
Let Γ be a consistent set of formulas in a first-order language $\mathcal{L}$. Consider the canonical algebraic model I for Γ. The universe of the interpretation I is the set of all terms (possibly with variables) of $\mathcal{L}$. The algebra of interpretation is the Lindenbaum algebra $\boldsymbol{A}(\Gamma)$. Notice that the universe is isomorphic with the omega-Herbrand universe $U^{\omega}_{\mathcal{L}}$; the isomorphism is obtained by replacing every variable x_i by the free element k_i. Now, take any Q-filter ∇ in $\boldsymbol{A}(\Gamma)$ and divide I by ∇. The resulting interpretation is a (two-valued) omega-Herbrand model for Γ. Additional properties of this model will depend of the choice of the Q-filter ∇. $\square$

6. Two Straightforward Results of the Algebraic Semantics

Theorem 6.1. [Completeness with respect to omega-Herbrand semantics]
Let $\Gamma \cup \{B\}$ be a set of formulas in a countable first-order language $\mathcal{L}$ without equality.
1. If $\Gamma \not\vdash B$ then there exists an omega-Herbrand interpretation I for $\mathcal{L}$ such that $I \models \Gamma$ and $I \not\models B$.
2. $\Gamma \vdash B$ iff $\Gamma \models^{\omega} B$. $\square$

Proof:

1. Assume that $\Gamma \not\vdash B$. Consider the canonical algebraic interpretation I for Γ, whose universe is the set of all terms of $\mathcal{L}$, with logical values in the Lindenbaum algebra $\boldsymbol{A}(\Gamma)$ of formulas of $\mathcal{L}$ modulo Γ. I is a model for Γ: for every $\gamma \in \Gamma$ and every valuation ν, $\gamma^I[\nu] = \top$. As $\Gamma \not\vdash B$, for the identity valuation $\iota = \{x_i/x_i \mid i < \omega\}$ we have $B^I[\iota] \neq \top$, and $\neg B^I[\iota] \neq \bot$. As $\mathcal{L}$ is countable, by Rasiowa-Sikorski lemma, there exists a Q-filter $\nabla \subseteq \boldsymbol{A}(\Gamma)$ such that $B_I[\iota] \notin \nabla$. Consider quotient $I' = I/\nabla$. I' is an algebraic interpretation in the set of all terms of $\mathcal{L}$ and in the two-element Boolean algebra, such that $I' \models \Gamma$ and $I' \not\models B[\iota]$. By identifying each variable x_i with the free constant k_i, the set of all terms of $\mathcal{L}$ can be identified with $U^{\omega}_{\mathcal{L}}$. This ends the proof of 1.
2. For the implication to the right: $\Gamma \vdash B$ implies $\Gamma \models B$ which implies $\Gamma \models^{\omega} B$.
   For the implication to the left: assume $\Gamma \not\vdash B$; by 1 there exists omega-Herbrand interpretation I for $\mathcal{L}$ such that $I \models \Gamma$ and $I \not\models B$; therefore $\Gamma \not\models^{\omega} B$. $\square$

Let us remark that in the theorem above the assumption that the language does not contain the symbol of equality is essential. If $=$ is allowed to be present in the language of $\Gamma \cup \{B\}$, the relations $\Gamma \models B$ and $\Gamma \models^{\omega} B$ do not coincide. Indeed, for $\Gamma = \{p(x) \leftarrow \forall_y x \neq f(y),\ p(f(x)) \leftarrow p(x)\}$ we have $\Gamma \models^{\omega} \forall_x p(x)$ but $\Gamma \not\models \forall_x p(x)$.

Before we formulate the next lemma, recall that $k_0, k_1, k_2, \ldots$ are new individual constants used to form omega-Herbrand universe $U^{\omega}_{\mathcal{L}}$. The lemma justifies the name "free elements", that is used for $k_0, k_1, k_2, \ldots$.

Lemma 6.1. Let P be a definite program and let A be an atomic formula in a first-order language $\mathcal{L}$ without equality. Then:

$$M^{\omega}_P \models A(\bar{x})[\bar{t}(k_1, \ldots, k_n)/\bar{x}] \quad \text{implies} \quad P \vdash A(\bar{t}(x_1, \ldots, x_n)). \qquad \square$$

Proof:
Assume $M^{\omega}_P \models A(\bar{x})[\bar{t}(k_1, \ldots, k_n)/\bar{x}]$. Recall that M^{ω}_P is the intersection of all omega-Herbrand models for P. Thus in any omega-Herbrand model I for P, we have $I \models A(\bar{x})[\bar{t}(k_1, \ldots, k_n)/\bar{x}]$.

Consider the canonical algebraic model I_0 for P, whose universe is the set of all terms of $\mathcal{L}$ and whose truth values are in the Lindenbaum algebra $\boldsymbol{A}(P)$ of formulas of $\mathcal{L}$ modulo P. Consider any Q-filter $\nabla \subseteq \boldsymbol{A}(P)$. As I_0/∇ can be identified with an omega-Herbrand model I for P, and as $I \models A(\bar{t})[k_1/x_1, \ldots, k_n/x_n]$ we see that $\| A(\bar{t}(x_1, \ldots, x_n)) \| \in \nabla$.

Thus for every Q-filter $\nabla \subseteq \boldsymbol{A}(P)$ we have $\| A(\bar{t}(x_1, \ldots, x_n)) \| \in \nabla$. By the Rasiowa-Sikorski lemma 3.1. (2) the only element which belongs to every Q-filter is $\top$. Thus $\| A(\bar{t}(x_1, \ldots, x_n)) \|$ is the top element in $\boldsymbol{A}(P)$. Therefore $P \vdash A(\bar{t}(x_1, \ldots, x_n))$. $\square$

7. Corollaries for Logic Programming

Remark 7.1. The following property is used in the proof of Theorem 5.1..

Let P be a set of definite clauses in $\mathcal{L}$, then:

P is unsatisfiable iff P has no Herbrand model.

The straightforward example with P being $\{p(c), \exists_x(\neg p(x))\}$ shows that one cannot drop the assumption that formulas in P are clauses. However if we consider omega-Herbrand models, by Theorem 6.1. we obtain the desirable strengthening. This is specified in the next proposition. $\Box$

Proposition 7.1. Let Γ be an arbitrary set of formulas in a countable first-order language without equality. Then, Γ is unsatisfiable iff Γ has no omega-Herbrand model. $\Box$

Proof:

Implication to the right is obvious. For the implication to the left: If Γ has no omega-Herbrand model then $\Gamma \models^{\omega} \perp$, and by Theorem 6.1. $\Gamma \models \perp$, so Γ has no model. $\Box$

Remark 7.2. Consider a first-order language $\mathcal{L}$ without equality. Let P be a definite program, and let A be a ground atomic formula. The *Herbrand rule* is defined as:

If $P^* \cup \{A\}$ has no Herbrand model, infer $\neg A$.

Recall that the Reiter Closed World Assumption (CWA) is the following rule:

If $P \not\vdash A$, infer $\neg A$.

The Herbrand rule is strictly weaker than CWA. Indeed consider definite program P containing clauses: $p(c) \leftarrow q$, $p(f(x)) \leftarrow p(x)$, $q \leftarrow p(x)$. Notice that $\neg q$ follows from P under CWA, but $\neg q$ cannot be derived from P using the Herbrand rule. As the next proposition shows the situation changes if we use omega-Herbrand interpretations instead of conventional ones. $\Box$

Proposition 7.2. Let the *omega-Herbrand rule* be an inference rule defined as follows:

If $P^* \cup \{A\}$ has no countable omega-Herbrand model, infer $\neg A$

Then, for definite programs P and ground atomic formulas A the results of the omega-Herbrand rule coincide with those of CWA. $\Box$

Proof:

As P^* always has a countable omega-Herbrand model the fact that $P^* \cup \{A\}$ does not have any such model is equivalent to saying that $P^* \not\models^{\omega} A$. By Theorem 6.1.this is equivalent to $P^* \not\vdash A$. It is known (cf. [8] pp. 80-82) that for definite programs P and ground atomic formulas A, $P^* \not\vdash A$ is equivalent to $P \not\vdash A$. $\Box$

Remark 7.3. Let P be a definite program and let A be a ground atomic formula in a first-order language without equality, then: $P \vdash A$ iff $M_P \models A$. The straightforward counter-example with P being $\{p(a) \leftarrow \top\}$ and A being $p(x)$ shows that one can not drop the assumption that A is ground. The next theorem shows that the situation changes dramatically if we allow omega-Herbrand models. $\Box$

Theorem 7.1. Let P be a definite program and let A be an atomic formula in a first-order language without equality. Then: $P \vdash A$ iff $M_P^{\omega} \models A$. $\Box$

Proof:

$P \vdash A$ iff
$P \vdash \forall A$ iff
$P \cup \{\neg\forall A\}$ does not have a model iff (by 6.1.)
$P \cup \{\neg\forall A\}$ does not have an omega-Herbrand model iff
$\neg\forall A$ is false in all omega-Herbrand models for P iff

$\forall A$ is true in all omega-Herbrand models for P iff
$\forall A$ is true in the intersection of all omega-Herbrand models for P iff
$\forall A$ is true in M_P^ω iff
$M_P^\omega \models A$ □

Another property of M_P^ω, which does not have a counterpart with conventional Herbrand models is given in the next theorem.

Theorem 7.2. [Existence property]
Let P be a definite program and let A be an atomic formula in a first-order language $\mathcal{L}$ without equality. Then the following conditions are equivalent.

1. $M_P^\omega \models \exists_{\bar{x}} A(\bar{x})$.
2. $P \vdash \exists_{\bar{x}} A(\bar{x})$.
3. There exists a sequence $\bar{t}$ of terms of $\mathcal{L}$ such that $P \vdash A(\bar{t})$. □

Proof:
Notice that $3 \Rightarrow 2 \Rightarrow 1$. It remains to prove that $1 \Rightarrow 3$. Assume $M_P^\omega \models \exists_{\bar{x}} A(\bar{x})$. Then there exists a sequence $\bar{t}$ of terms and free elements $k_1, \ldots, k_n$ such that $M_P^\omega \models A(\bar{x})[\bar{t}(k_1, \ldots, k_n)/\bar{x}]$. By Lemma 6.1. we have $P \vdash A(\bar{t})$. □

Problems resulting from the use of conventional Herbrand interpretations mentioned in the remarks in this section cause some complications in the standard proof of completeness of SLD-resolution. Below we give a proof which uses omega-Herbrand interpretations instead.

Theorem 7.3. (Clark's completeness theorem for SLD-resolution)
Let $P, \leftarrow A$ be a definite program and goal. Then, if $P \vdash A\theta$ then there exists a substitution θ', more general than θ, which can be computed by the SLD-resolution as an answer to $P, \leftarrow A$. □

Proof:
Assume that $P \vdash A\theta$. Let $\iota = \{k_i/x_i\}$ be a substitution in an extended language. We have $P \vdash A\theta\iota$. As $A\theta\iota$ is ground, by Theorem 5.1., SLD-resolution given $P, \leftarrow A\theta\iota$ returns answer *YES*. By the lifting lemma (cf. [8], [11]) we deduce that SLD given $P, \leftarrow A$ can return an answer θ' which is more general than $\theta\iota$. As θ' is a substitution in the original language of $P, \leftarrow A$, it must be more general than θ. □

8. Conclusion

We have shown an approach to foundations of logic programming in which the connection with algebraic semantics becomes apparent. In this setting various results concerning logic programming become straightforward corollaries to the Rasiowa-Sikorski lemma.

The approach is based on omega-Herbrand models instead of conventional Herbrand models. We compared certain aspects of the two approaches. We demonstrated how by using algebraic semantics certain annoying technicalities can be eliminated from the proof of Clark's completeness theorem of SLD-resolution. We formulated an proved the existence property for definite programs.

Let us mention that omega-Herbrand interpretations, which are crucial for the presented approach, possess also the following desirable properties: the theory of equality determined by $\mathbf{U}_\mathcal{L}^\omega$ is decidable and the operators on lattices of omega-Herbrand interpretations associated with positive programs with $(\forall, =, \text{etc.})$ reach their least-fix points after ω iterations, cf. [9, 10].

As algebraic semantics is a well established domain of mathematical logic, applying its methods to the much younger domain of logic programming can bring new interesting results – we hope that this direction is pursued by researchers specializing in algebraic semantics.

References

[1] K. R. Apt, Introduction to Logic Programming, in *Handbook of Theoretical Computer Science*, edited by J. Van Leeuwen, MIT Press and Elsevier, 1989, vol. 1, pp. 493-574.

[2] K. R. Apt and R. N. Bol, Logic Programming And Negation: A Survey, *Journal of Logic Programming* Vol.19/20, pp. 9-71, 1994.

[3] K. R. Apt and M. H. van Emden, Contributions to the Theory of Logic Programming, *J. ACM* July 1982, vol. 29, no. 3, pp. 841-862.

[4] K. L. Clark, Negation as Failure, in *Logic and Databases*, edited by H. Gallaire and J. Minker, Plenum Press, New York, 1978, pp. 193-322.

[5] K. L. Clark, Predicate Logic as a Computational Formalism, *Research Report DOC 79/59*, Dept. of Computing, Imperial College, 1979.

[6] S. Feferman, Review of the paper: H. Rasiowa and R. Sikorski, A proof of the completeness theorem of Gödel, *Journal of Symbolic Logic* 17, 1952, p. 72.

[7] R. A. Kowalski, Predicate Logic as a Programming Language, in *Information Processing '74*, Stockholm, North Holland, 1974, pp. 569-574.

[8] J. W. Lloyd, *Foundations of Logic Programming*, Second extended edition, Springer Verlag, 1987.

[9] J. A. Plaza, *Fully Declarative Programming with Logic – Mathematical Foundations*, Ph.D. Dissertation, City University of New York, July 1990.

[10] J. A. Plaza, Operators on Lattices of omega-Herbrand Interpretations, in: A. Nerode and M. Taitslin (eds.) *Logical Foundations of Computer Science – Tver '92, Second International Symposium, Tver, Russia, July 1992, Proceedings,* Springer Verlag, LNCS 620, 1992, pp. 358-369.

[11] J. A. Plaza, Soundness and Completeness versus Lifting Property, submitted for publication.

[12] H. Rasiowa, *Non-classical logics, An Algebraic Approach*, North Holland, 1974.

[13] H. Rasiowa and R. Sikorski, A proof of Completeness Theorem of Gödel, *Fundamenta Mathematicae 37*, 1950, pp. 193-200.

[14] H. Rasiowa and R. Sikorski, Algebraic treatment of the notion of satisfiability, *Fundamenta Mathematicae 40*, 1953, pp. 62-95.

[15] H. Rasiowa and R. Sikorski, *The Mathematics of Metamathematics*, PWN - Polish Scientific Publishers, Warsaw, first edition – 1963, third edition – 1970.

[16] J. C. Shepherdson, *Language and equality Theory in Logic Programming*, Report No. PM-91-02, University of Bristol, School of Mathematics, England, 1991.

[17] M. H. van Emden and R. A. Kowalski, The Semantics of Predicate Logic as a Programming Language, *J. ACM* 23, 4 (Oct. 1976), pp. 733-742.