Artificial Intelligence and Symbolic Mathematical Computation
J. Calmet, J.A. Campbell and J. Pfalzgraf (eds.)
Springer Verlag, LNCS vol. 1138, 1996, pp. 354-364.

Soundness and Completeness versus Lifting Property[1]

Jan A. Plaza

University of Miami
Department of Mathematics and Computer Science
P.O. Box 249085
Coral Gables, Florida 33124
U.S.A.

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

Abstract. We give new formulations of the property of soundness and completeness of a resolution system and of the lifting lemma, and we discuss their relationship.
The discussion points out why certain resolution systems are not complete, and that there is a simple method for showing that a resolution system is "absolutely incomplete" – that there is no notion of program completion and no logic which could give soundness and completeness. The method is demonstrated on the case of basic variants of the SLDNF-resolution. The same approach can be used to analyze other resolution systems.

Keywords: soundness and completeness, lifting lemma, program completion, SLDNF-resolution.

1 Introduction

One of the ideas underlying the development of logic programming is that programming should be to a greater extent declarative. Declarativeness of logic programming is characterized in a formal way, by so called, soundness and completeness properties. Many sound resolution systems have been proposed as possible bases for logic programming; [AB94] gives a broad overview. For most of these systems it has been shown that they do not satisfy a particular completeness property for the full class of normal programs. Still, one could ask whose fault it is: is it a deficiency of the resolution system or a deficiency of a particular formulation of the soundness and completeness property. This paper presents a simple method which can be used to put the blame entirely on the resolution system.

In *The Craft of Prolog* [O'Keefe90] wrote: "Elegance is not optional," and then repeated: "Elegance is not optional." Let us add that elegance is not optional not only in programming but also in the presentations of a theory. It is our intention to make in this paper a step in the direction of presenting issues of completeness in an elegant way. For this purpose we will modify the definition of Herbrand universes; calling the resulting structures omega-Herbrand universes. The concept of the omega-Herbrand universe is *not* a generalzation of that of Herbrand universe. The change in the definition is not a mere technicality, and will have far reaching consequences: without it the universe would not have desired strong compactness properties. These compactness properties, in turn, will be crucial for showing equivalence of conventional and new formulations of soundness and completeness, and lifting properties. References to other non-technical properties of omeaga-Herbrand universes will be mentioned in Conclusion. Unlike conventional formulations of lifting, which provided just technical tools to be used in proofs of other theorems, the lifting property in our new formulation will have a clear philosophical reading. These ideas will be presented using a notation which allows to think of answers returned by a resolution system as subsets of a universe.

In the paper we assume that the reader is familiar with basic notions in foundations of logic programming, as presented, for instance, in chapters 1-3 of [Lloyd87]. Following conventions of the set theory, the set of natural numbers $\{0, 1, 2, \ldots\}$ will be denoted by ω.

2 Terms and Substitutions as Subsets of the omega-Herbrand Universe

A resolution system, such as the SLDNF, when given a program and a query, returns substitutions as answers. In order to discuss resolution systems conveniently, we will need to introduce notation which will allow us to think of such answers as of subsets of (a power of) a universe. The universe will be somewhat different from the Herbrand universe:

Definition 1 (omega-Herbrand universe).
Let $\mathcal{L}$ be a first-order language and let $\mathcal{L}^{\mathcal{K}}$ result by adding a 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*. □

Notice that free elements of the omega-Herbrand universe for language $\mathcal{L}$ do not have names in $\mathcal{L}$. So the difference between the ω-*Herbrand universe* and the conventional Herbrand universe results from the presence of these unnamed elements. The reason for such a definition will be explained in the next section.

Now, with any term t we will associate the set $[\![t]\!]$ of all its instances in $U^\omega_\mathcal{L}$. For instance: $[\![f(c_1, g(c_2))]\!] = \{f(c_1, g(c_2))\}$, $[\![x]\!] = U^\omega_\mathcal{L}$ and $[\![g(x)]\!] = \{g(e) \mid e \in U^\omega_\mathcal{L}\}$. As any substitution can be represented as a sequence of terms, not only terms but also substitutions can be represented using the same idea. Before we give a formal definition, a few words about substitutions and valuations.

By a *substitution* we understand any partial function from the set $Var = \{x_i \mid i < \omega\}$ of individual variables of a language $\mathcal{L}$, into the set of terms of $\mathcal{L}$. Substitutions are written as sets of elementary substitutions, e.g. $\theta = \{t_i/x_i \mid i \in I\}$ where I

is a set of natural numbers. Any substitution $\{t_i/x_i \mid i \in I\}$ which is not total, is identified with the following total substitution: $\{t_i/x_i \mid i \in I\} \cup \{x_i/x_i \mid i \in (\omega - I)\}$. The symbol ϵ stands for the *identity substitution*: $\epsilon = \{x_i/x_i \mid i < \omega\}$. Notice that, by the convention above, the empty substitution (i.e. Prolog's *YES*) is identified with the identity substitution.

Given an interpretation of a language $\mathcal{L}$, by a *valuation* we always understand a total function from the set *Var* of variables of $\mathcal{L}$, into the domain of the interpretation.

Definition 2 (Terms and substitutions as subsets of $U_{\mathcal{L}}^{\omega}$).

1. Let t be a term of a first-order language $\mathcal{L}$. We define: $[\![t]\!] = \{t[\nu] \in U_{\mathcal{L}}^{\omega} \mid \nu : Var \longrightarrow U_{\mathcal{L}}^{\omega}\}$. Always $[\![t]\!] \subseteq U_{\mathcal{L}}^{\omega}$.
2. Let $x_i : i < \omega$ be a fixed enumeration of all the variables of $\mathcal{L}$, let $\theta = \{t_i/x_i \mid i < \omega\}$ be a substitution in $\mathcal{L}$, and let $\tilde{\theta} = \langle t_0, t_1, t_2, \ldots\rangle$ be the sequence of terms substituted for consecutive variables. We define: $[\![\theta]\!] = \{\tilde{\theta}[\nu] \in (U_{\mathcal{L}}^{\omega})^{\omega} \mid \nu : Var \longrightarrow U_{\mathcal{L}}^{\omega}\}$. Always $[\![\theta]\!] \subseteq (U_{\mathcal{L}}^{\omega})^{\omega}$. □

The meaning of $[\![t]\!]$ and $[\![\theta]\!]$ depends on the chosen language $\mathcal{L}$. Notice however that for every language $\mathcal{L}$ we have $[\![c]\!] \neq [\![x]\!]$. (For a language with a single individual constant c and no other function symbols, there wouldn't be any difference between $[\![c]\!]$ and $[\![x]\!]$ if one defined these notions using conventional Herbrand universes.) Notice also that we may write $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ without specifying the language of the substitutions: If θ_1, θ_2 are substitutions in language $\mathcal{L}$, and if $\mathcal{L} \subseteq \mathcal{L}'$, then the relation $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ does not depend on whether the substitutions are interpreted as subsets of $U_{\mathcal{L}}^{\omega}$ or of $U_{\mathcal{L}'}^{\omega}$.

The following proposition shows that the notation with $[\![t]\!]$ and $[\![\theta]\!]$ can concisely express basic notions related to unification.

Proposition 3.
1. $[\![t_1]\!] \subseteq [\![t_2]\!]$ *iff there exists a substitution θ such that $t_1 = t_2\theta$.*
2. $[\![t_1]\!] \cap [\![t_2]\!] = [\![t]\!]$ *iff t is the result of a most general unification of t_1 and t_2.*
3. $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ *iff there exists a substitution θ such that $\theta_1 = \theta_2\theta$*
(which means that θ_2 is a more general substitution than θ_1.)

The proof is straightforward and will be omitted.

In topology, the compactness property says that every (open) cover contains a finite subcover. In the theory of logics intermediate between the classical and intuitionistic logic, one defines strongly compact pseudo-Boolean algebras as those in which every cover contains a single covering element. By analogy, we will call the properties listed in the following theorem a strong compactness.

Theorem 4. (Strong compactness)
1. If $[\![t]\!] \subseteq \bigcup_{i\in I}[\![t_i]\!]$ *then there exists $i_0 \in I$ such that* $[\![t]\!] \subseteq [\![t_{i_0}]\!]$.
2. If $[\![\theta]\!] \subseteq \bigcup_{i\in I}[\![\theta_i]\!]$ *then there exists $i_0 \in I$ such that* $[\![\theta]\!] \subseteq [\![\theta_{i_0}]\!]$.

Proof.

1. Assume $[\![t]\!] \subseteq \bigcup_{i\in I}[\![t_i]\!]$. Consider valuation $\iota = \{k_i/x_i \mid i < \omega\}$. Now, $t[\iota]$ is a ground term and we have $t[\iota] \in \bigcup_{i\in I}[\![t_i]\!]$. Take i_0 such that $t[\iota] \in [\![t_{i_0}]\!]$. There exists valuation ν such that $t[\iota] = t_{i_0}[\nu]$ and there exists substitution θ' such that $\nu = \theta'[\iota]$. Thus $t = t_{i_0}\theta'$ and $[\![t]\!] \subseteq [\![t_{i_0}]\!]$.
2. Assume $[\![\theta]\!] \subseteq \bigcup_{i\in I}[\![\theta_i]\!]$. Consider valuation $\iota = \{k_i/x_i \mid i < \omega\}$. Now, $\tilde{\theta}[\iota]$ is a sequence of ground terms and we have $\tilde{\theta}[\iota] \in \bigcup_{i\in I}[\![\theta_i]\!]$. Take i_0 such that $\tilde{\theta}[\iota] \in [\![\theta_{i_0}]\!]$. There exists valuation ν such that $\tilde{\theta}[\iota] = \tilde{\theta}_{i_0}[\nu]$ and there exists substitution θ' such that $\nu = \theta'[\iota]$. Thus $\tilde{\theta} = \tilde{\theta}_{i_0}(\theta')$ and $[\![\theta]\!] \subseteq [\![\theta_{i_0}]\!]$. □

Remark. The choice of omega-Herbrand universes instead of conventional Herbrand universes in Definition 2 is essential. The definition has been constructed so that it guarantees that Theorem 4 on strong compactness holds. In [Kunen87] a notion analogous to $[\![t]\!]$ is defined using the conventional Herbrand universe $U_{\mathcal{L}}$; this change causes that the statement analogous to Theorem 4 is not true, even under Kunen's assumption that the language $\mathcal{L}$ contains infinitely many function symbols or constants. Indeed, for every such language $\mathcal{L}$ we would have $[\![x]\!] \subseteq \bigcup_{t\in U_{\mathcal{L}}}[\![t]\!]$ but there would be no single term $t_0 \in U_{\mathcal{L}}$ such that $[\![x]\!] \subseteq [\![t_0]\!]$. Lemma 1 in [Kunen87] states a property much weaker than 4. (One could also notice that having *infinitely* many free elements in the universe is essential for Theorem 4.) □

Remark. In constraint logic programming over a domain of finite terms one can consider substitutions constrained by formulas involving no predicates except equality. For instance,

$$\{c/x_1,\ f(x_5)/x_2,\ x_4/x_3\} \text{ WHERE } (x_5{\neq}x_4 \wedge \forall_y x_5{\neq}f(y)).$$

Also such substitutions can be interpreted as subsets of the omega-Herbrand universe:

$$[\![\theta \text{ WHERE } S]\!] = \{\tilde{\theta}[\nu] \in (U^\omega_{\mathcal{L}})^\omega \mid U^\omega_{\mathcal{L}} \models S[\nu]\}.$$

but the compactness property does not hold. Indeed,

$$[\![\epsilon]\!] = \textstyle\bigcup_{i<\omega}[\![\{f^i(x_1)/x_0\} \text{ WHERE } (\forall_y x_1{\neq}f(y))]\!]$$

but there is no finite $I \subseteq \omega$ such that

$$[\![\epsilon]\!] = \textstyle\bigcup_{i\in I}[\![\{f^i(x_1)/x_0\} \text{ WHERE } (\forall_y x_1{\neq}f(y))]\!].$$ □

3 Soundness and Completeness versus Lifting Lemma

Now we are ready to start analyzing resolution systems. For SLD-resolution, definite program P and goal $\leftarrow B$, we could define the *SLD-answer set* as: $SLD(P,B) = \bigcup_{i\in I}[\![\theta_i]\!]$, where $\theta_i : i \in I$ is the collection of all SLD-computed answers for P and $\leftarrow B$. Similarly, for SLDNF-resolution, normal program P and goal $\leftarrow B$, the *SLDNF-answer set* is: $SLDNF(P,B) = \bigcup_{i\in I}[\![\theta_i]\!]$, where $\theta_i : i \in I$ is the collection of all SLDNF-computed answers for P and $\leftarrow B$. The next definition generalizes these notions.

Definition 5. Let *Resolution* be a resolution system which, if given a program P and a goal $\leftarrow B$, returns a sequence of computed answer substitutions $\theta_i : i \in I$. Then, by the *Resolution-answer set* for P and $\leftarrow B$ we understand the set: $Resolution(P,B) = \bigcup_{i\in I}[\![\theta_i]\!]$. □

Soundness and completeness of SLD-resolution for the class of definite programs and goals are usually[2] formulated as two separate implications, cf. [Clark79], [Lloyd87]:

> Soundness: If θ is an SLD-computed answer for P and $\leftarrow B$, then $Comp(P) \vdash B\theta$.
> Completeness: If $Comp(P) \vdash B\theta$ then there exists a substitution θ', more general than θ, which is an SLD-computed answer for P and $\leftarrow B$.

These statements could not be combined into one equivalence because of a technical difficulty with the "substitution θ' that is more general than θ". However if we use answer sets from Definition 5 we may conveniently formulate soundness and completeness as a single equivalence:

$$SLD(P, B) \supseteq [\![\theta]\!] \quad \text{iff} \quad Comp(P) \vdash B\theta.$$

This is explained in the following proposition.

Proposition 6. *The following formulations of the soundness and completeness property are equivalent:*

1. *If θ is a Resolution-computed answer for P and $\leftarrow B$ then $Completion(P) \vdash B\theta$;*
   If $Completion(P) \vdash B\theta$ then there exists a substitution θ' more general than θ, which is a Resolution-computed answer for P and $\leftarrow B$.
2. *$Resolution(P, B) \supseteq [\![\theta]\!]$ iff $Completion(P) \vdash B\theta$*

The reader should notice that that this proposition critically depends on the Theorem 4 on strong compactness.

Proof.

$1 \Rightarrow 2$. For the proof of the implication to the right assume that $Resolution(P, B) \supseteq [\![\theta]\!]$. This means that $[\![\theta]\!]$ is covered by the $Resolution$-computed answers for P and $\leftarrow B$. By Theorem 4 on strong compactness there exists a single $Resolution$-computed answer $[\![\theta']\!]$ such that $[\![\theta']\!] \supseteq [\![\theta]\!]$. Then, by 1, $Completion(P) \vdash B\theta'$ and of course $Completion(P) \vdash B\theta$.
For the proof of the implication to the left assume that $Completion(P) \vdash B\theta$. Then, by 1, $Resolution$ computes an answer θ' more general than θ. We have $Resolution(P, B) \supseteq [\![\theta']\!] \supseteq [\![\theta]\!]$.

$2 \Rightarrow 1$. For the proof of the first implication assume that θ is a $Resolution$-computed answer for P and $\leftarrow B$. Thus $Resolution(P, B) \supseteq [\![\theta]\!]$ and by 2, $Completion(P) \vdash B\theta$.
For the proof of the second implication assume that $Completion(P) \vdash B\theta$. Then, by 2, $Resolution(P, B) \supseteq [\![\theta]\!]$ and by Theorem 4 on strong compactness there exists a single $Resolution$-computed answer $[\![\theta']\!]$ such that $[\![\theta']\!] \supseteq [\![\theta]\!]$ which means that θ' is more general than θ. $\square$

[2] perhaps using just P instead of Clark's program completion $Comp(P)$.

Another important property of a resolution system is related to the lifting lemma. The lifting lemma, in its many variants, has been used as a technical tool in the proofs of completeness of resolution systems: it allows to "lift" ground derivations to the general derivations involving variables, cf. [Robinson65], [Clark79], [Lloyd87] and [Doets94]. In our new formulation the lifting lemma reads:

$$\mathit{Resolution}(P, B) \supseteq [\![\theta]\!] \quad \text{iff} \quad \mathit{Resolution}(P, B\theta) \supseteq [\![\epsilon]\!].$$

This can be understood as:

> "*Resolution* is capable of computing answer θ, exactly when it accepts θ as appropriate".

Indeed, by Theorem 4 on strong compactness, $\mathit{Resolution}(P, B) \supseteq [\![\theta]\!]$ means that θ can be computed, and $\mathit{Resolution}(P, B\theta) \supseteq [\![\epsilon]\!]$ means that given P and $\leftarrow B\theta$ the resolution returns the identity substitution, i.e. *YES*, thus accepting θ as appropriate. As our formulation has a clear intuitive meaning we will no longer call it a lemma, referring to it as to a *lifting property.* We believe that the lifting property in this formulation is interesting for its own sake, independently of its applications in proofs of other theorems. The next proposition explains how this version is related to other versions.

Proposition 7. *Consider the following formulations of lifting properties.*

1. *If θ' is a Resolution-computed answer for P and $\leftarrow B\theta$, then there exists a substitution θ'' more general than $\theta\theta'$, such that θ'' is a Resolution-computed answer for P and $\leftarrow B$.*
   (This is the conventional formulation, cf. [Lloyd87], lm. 8.2 p. 47.)
2. $\mathit{Resolution}(P, B\theta) \supseteq [\![\theta_0]\!]$ *implies* $\mathit{Resolution}(P, B) \supseteq [\![\theta\theta_0]\!]$.
3. $\mathit{Resolution}(P, B) \supseteq [\![\theta]\!]$ *iff* $\mathit{Resolution}(P, B\theta) \supseteq [\![\epsilon]\!]$

Formulations 1 and 2 are equivalent. Formulation 3 implies each of 1 and 2.

The reader should notice that that this proposition critically depends on the Theorem 4 on strong compactness.

Proof.

$1 \Rightarrow 2$. Assume that $\mathit{Resolution}(P, B\theta) \supseteq [\![\theta_0]\!]$. Thus by Theorem 4 on strong compactness *Resolution* with P and $\leftarrow B\theta$ computes θ' such that $[\![\theta']\!] \supseteq [\![\theta_0]\!]$. By 1, there exists θ'' such that $[\![\theta'']\!] \supseteq [\![\theta\theta']\!]$ and $\mathit{Resolution}(P, B) \supseteq [\![\theta'']\!]$. Thus $\mathit{Resolution}(P, B) \supseteq [\![\theta'']\!] \supseteq [\![\theta\theta']\!] \supseteq [\![\theta\theta_0]\!]$.

$2 \Rightarrow 1$. Assume that *Resolution* with P and $\leftarrow B\theta$ computes θ'. Then, by 2, $\mathit{Resolution}(P, B) \supseteq [\![\theta\theta']\!]$. By Theorem 4 on strong compactness *Resolution* with P and $\leftarrow B\theta$ computes θ'' which is more general than $\theta\theta'$.

$3 \Rightarrow 2$. Assume that $\mathit{Resolution}(P, B\theta) \supseteq [\![\theta_0]\!]$. By 3, $\mathit{Resolution}(P, B\theta\theta_0) \supseteq [\![\epsilon]\!]$. By 3 again (using the other implication), $\mathit{Resolution}(P, B) \supseteq [\![\theta\theta_0]\!]$. □

The following theorem offers a method for showing that a resolution system is "absolutely incomplete."

Theorem 8. (the test of lifting)
If a resolution system does not satisfy the lifting property:

$$(*) \quad Resolution(P, B) \supseteq [\![\theta]\!] \quad \textit{iff} \quad Resolution(P, B\theta) \supseteq [\![\epsilon]\!]$$

then there is no notion of program completion Completion and no logic **L** *for which the soundness and completeness property:*

$$(**) \quad Resolution(P, B) \supseteq [\![\theta]\!] \quad \textit{iff} \quad Completion(P) \vdash_{\mathbf{L}} B\theta$$

could hold.

Proof. It is enough to notice that the soundness and completeness $(**)$ implies the lifting property $(*)$. Indeed:

$$\begin{array}{ll} Resolution(P, B) \supseteq [\![\theta]\!] & \text{iff (by } (**)) \\ Completion(P) \vdash_{\mathbf{L}} B\theta & \text{iff} \\ Completion(P) \vdash_{\mathbf{L}} (B\theta)\epsilon & \text{iff (by } (**)) \\ Resolution(P, B\theta) \supseteq [\![\epsilon]\!]. & \end{array}$$

□

Theorem 8 can be paraphrased as: lifting $(*)$ is a necessary condition for soundness and completeness $(**)$. The question whether it is a sufficient condition cannot be considered unless one defines what constitutes a program completion or a logic and what doesn't. Considering this issue is not in the scope of this paper.

Remark. The proof above does not use the strong compactness property, so Theorem 8, on the test of lifting, generalizes to the case of resolution systems for constraint logic programming over universes of terms. □

As a consequence of the soundness and completeness of SLD-resolution, from Theorem 8 we obtain the following expected corollary:

Corollary 9. *For definite programs and goals, the SLD-resolution satisfies the lifting property:*

$$SLD(P, B) \supseteq [\![\theta]\!] \quad \textit{iff} \quad SLD(P, B\theta) \supseteq [\![\epsilon]\!]$$

4 "Absolute Incompleteness" of the SLDNF-resolution

The SLD-resolution is sound and complete for definite programs and goals cf. [Clark79], [Lloyd87]. We will formulate this in the following two ways:

$$SLD(P, B) \supseteq [\![\theta]\!] \quad \text{iff} \quad P \vdash B\theta$$
$$SLD(P, B) \supseteq [\![\theta]\!] \quad \text{iff} \quad Comp(P) \vdash B\theta$$

where *Comp* is Clark's program completion. Now let us consider basic generalizations of SLD-resolution which can handle programs with negation.

In [Clark78] SLDNF-resolution was introduced and proved sound for normal programs and goals. We will formulate this as:

$$SLDNF(P, B) \supseteq [\![\theta]\!] \quad \text{implies} \quad Comp(P) \vdash B\theta.$$

It is well known that for normal programs and goals the SLDNF-resolution is not complete with respect to Clark's program completion and inferences in classical logic. The reason for this lack of completeness of has been identified as related to the safeness condition.

Let us recall that the safeness condition disallows selecting a non-ground negative literal for processing. It is a reasonable idea to relax this condition. We paraphrase the description from [Lloyd87], (p. 94):

It is possible to weaken the safeness condition a little and still maintain soundness. Consider the following weaker safeness condition. Non-ground negative literals can be selected for processing. If the subgoal $\neg A$ is selected, a "lemma" is created with a single subgoal A. If A fails, then $\neg A$ succeeds. However, if A succeeds, a check is made to make sure no bindings were made to any variables in the top-level goal of the corresponding refutation. If no such binding was made, then $\neg A$ fails. But, if such a binding was made, then a different literal is selected and $\neg A$ is delayed in the hope that more of its variables will be bound later. If we are left with only negative literals, neither of which can be processed, the system halts without returning an answer – i.e. flounders.

Let us call such a resolution SLDNF$^+$. SLDNF$^+$ is a generalization of SLDNF-resolution: $\mathit{SLDNF}(P, B) \subseteq \mathit{SLDNF}^+(P, B)$. It is known that for normal programs and goals SLDNF$^+$ resolution is sound with respect to Clark's program completion and inferences in classical logic:

$$\mathit{SLDNF}^+(P, B) \supseteq [\![\theta]\!] \quad \text{implies} \quad \mathit{Comp}(P) \vdash B\theta$$

but the converse implication still does not hold.

Finally let us consider the version of SLDNF in which we drop the safeness condition entirely. This is an idealization of Prolog's computational mechanism and let us call it PROLOG$^+$. Notice however that in Prolog not only the safeness condition is dropped but also, for pragmatic reasons, unification is done without the occurs check, and a deterministic depth first search is implemented. PROLOG$^+$ doesn't make these further concessions — its unification is correct and selection of subgoals and input clauses in derivations is non-deterministic.

It is known that for normal programs and goals, PROLOG$^+$ is neither sound nor complete with respect to Clark's program completion and inferences in classical logic.

The reader already expects that the variants of SLDNF-resolution above are "absolutely incomplete". This term, which was also used in the title of this section, is surrounded by quotes, in order to emphasize that it is not precise. A more precise term would be: absolutely not sound-and-complete for a specific class of programs. A precise formulation is given in the following theorem.

Theorem 10. *For each of the following resolution systems:*

$$\mathit{SLDNF},\ \mathit{SLDNF}^+,\ \mathit{PROLOG}^+$$

there is no notion of program completion Completion *and no logic* $\mathbf{L}$ *for which the soundness and completeness property:*

$$\mathit{Resolution}(P, B) \supseteq [\![\theta]\!] \quad \text{iff} \quad \mathit{Completion}(P) \vdash_{\mathbf{L}} B\theta$$

could hold for the class of normal programs and goals.

Proof. By Theorem 8 it is enough to show that the resolution systems listed above do not satisfy the lifting property $(*)$. Consider the following normal program:

$$p(x) \leftarrow \neg q(x)$$
$$q(a) \leftarrow \mathit{false}$$
$$q(b) \leftarrow \mathit{true}$$

In the case of SLDNF, the derivation for P and $\leftarrow p(x)$ encounters subgoal $\neg q(x)$ and flounders (i.e. halts, due to the safeness condition, without returning any answer.) So $\mathit{SLDNF}(P, p(x)) = \emptyset$. On the other hand $\mathit{SLDNF}(P, p(a)) = [\![\epsilon]\!] = \mathit{YES} = (U^{\omega}_{\mathcal{L}})^{\omega}$. So the lifting property does not hold.

In the case of SLDNF$^+$, the derivation for P and $\leftarrow p(x)$ encounters subgoal $\neg q(x)$ and finds an illegal substitution b/x which it discards, and then flounders. So $\mathit{SLDNF}^+(P, p(x)) = \emptyset$. On the other hand $\mathit{SLDNF}^+(P, p(a)) = [\![\epsilon]\!] = \mathit{YES} = (U^{\omega}_{\mathcal{L}})^{\omega}$. So the lifting property does not hold.

In the case of PROLOG$^+$, the derivation for P and $\leftarrow p(x)$ encounters subgoal $\neg q(x)$ and finds an substitution b/x which causes the subgoal to fail. So $\mathit{PROLOG}^+(P, p(x)) = \emptyset = \mathit{NO}$ and $\mathit{PROLOG}^+(P, p(a)) = [\![\epsilon]\!] = \mathit{YES}$. So the lifting property does not hold. □

Remark. It is known that there exist subclasses of the class of normal programs and goals, for which SLDNF-resolution is sound and complete with respect to Clark's program completion. [Kunen89] contains such a result for allowed programs and inferences in Kleene's three-valued logic. [Plaza90, Plaza91] contains such a result for propositional normal programs, and inferences in classical logic. [DM91] contains such a result for allowed programs and inferences in classical logic. □

¿From the results mentioned in the remark above, by Theorem 8 we obtain the following corollary.

Corollary 11. *For allowed programs and goals, the SLDNF-resolution satisfies the lifting property:*
$$\mathit{SLDNF}(P, B) \supseteq [\![\theta]\!] \quad \textit{iff} \quad \mathit{SLDNF}(P, B\theta) \supseteq [\![\epsilon]\!]$$

One source of the lack of completeness of SLDNF for the full class of normal programs and goals is the lack of symmetry in the way it treats positive and negative subgoals; SLDNF makes substitutions in positive but not in negative subgoals. So to obtain completeness one need a resolution system which never flounders, which can process negative subgoals making substitutions, and when it stops, always returns an answer. Many such resolution systems have been considered, see for instance [Chan88, Chan89], [Przymusiński89], [Plaza90, Plaza92f] and [Drabent95] which consider various forms of constructive negation. For an overview of other directions see [AB94]. In this paper we cannot argue whether these systems satisfy the lifting property $(*)$, but for the researchers working on these systems it should be straightforward to produce appropriate argument.

5 Conclusion

We presented omega-Herbrand universes and discussed one of their advantages over conventional Herbrand universes. Further advantages are demonstrated in [Plaza92o] and [Plaza92f].

We gave a formulation (∗∗) of the soundness and completeness property which allows for a convenient analysis of this notion.

We gave a non-technical formulation (∗) of the lifting property which, we believe, makes this property interesting in itself, not only as a tool in proofs of other theorems. The informal reading of the property is: a resolution system is capable of computing an answer exactly when it accepts this answer as correct.

We formulated the test of lifting (Theorem 8) – a condition sufficient for an "absolute incompleteness" of a resolution system.

We showed that three basic variants of the SLDNF-resolution are "absolutely incomplete" (Theorem 10): there is no notion of program completion and no logic which could yield the soundness and completeness property for the class of normal programs and goals.

Acknowledgments

Special thanks to Professor Melvin Fitting.

References

[AB94] K. R. Apt and R. N. Bol, Logic Programming And Negation: A Survey, *Journal of Logic Programming* Vol.19/20, pp. 9-71, 1994.

[Chan88] D. Chan, Constructive Negation Based on Completed Database, in [LPconf88], pp. 111-125.

[Chan89] D. Chan, An extension of constructive negation and its application in corouting, in [LPconfNA89], pp. 477-493.

[Clark78] 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.

[Clark79] K. L. Clark, Predicate Logic as a Computational Formalism, *Research Report DOC 79/59*, Dept. of Computing, Imperial College, 1979.

[Doets94] K. Doets, *From Logic to Logic Programming* MIT Press, 1994.

[DM91] W. Drabent and M.Martelli, Strict Completion of Logic Programs, *New Generation Computing* 9, pp. 69-79, 1991.

[Drabent95] What is Failure? An Approach to Constructive Negation, *Acta Informatica*, to appear.

[Kunen87] K. Kunen, Answer Sets and Negation as Failure, in [LPconf87].

[Kunen89] K. Kunen, Signed Data Dependencies in Logic Programs, *Journal of Logic Programming*, 1989, vol. 7, no. 3, pp.231-247.

[Lloyd87] J. W. Lloyd, *Foundations of Logic Programming*, Second extended edition, Springer Verlag, 1987.

[LPconf87] J-L. Lassez (ed.), *Logic Programming, Proceedings of the Fourth International Conference*, MIT Press, 1987.

[LPconf88] R. A. Kowalski and K. A. Bowen (eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, MIT Press, 1988.

[LPconfNA89] E. L. Lusk and R. A. Overbeek (eds.), *Logic Programming, Proceedings of the North American Conference 1989*, MIT Press, 1989.

[O'Keefe90] R. A. O'Keefe, *The Craft of Prolog*, MIT Press, 1990.

[Plaza90] J. A. Plaza, *Fully Declarative Programming with Logic – Mathematical Foundations* Ph.D. Dissertation, City University of New York, July 1990.

[Plaza91] J. A. Plaza, Completeness for propositional logic programs with negation, in: Methodologies for Intelligent Systems, 6th International Symposium, edited by Z. W. Ras and M. Zemankova, Springer Verlag, LNAI 542, 1991, pp. 600-609.

[Plaza92f] J. A. Plaza, Fully Declarative Logic Programming, in *Programming Language Implementation and Logic Programming, 4th International Symposium*, edited by M. Bruynooghe and M. Wirsing, Springer Verlag, LNCS 631, 1992, pp. 415-427.

[Plaza92o] 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.

[Przymusiński89] T. C. Przymusiński, On Constructive Negation in Logic Programming, in [LPconfNA89], addendum.

[Robinson65] J. A. Robinson, A machine-oriented logic based on the resolution principle, *J. ACM* 12, 1965, pp. 23-41.