Mathematical Problems of Computer Science 36, 7 12, 2012.

The Degree of Unsolvability of the Completion Semantics for General Logic Programs

Levon A. Haykazyan

Yerevan State University
e-mail levon@rock.com

Abstract

The completion semantics considers interpretations that satisfy a special first-order theory was first introduced in [1] . These interpretations include but are not limited to Herbrand interpretations. Nevertheless, in logic programming the restriction to Herbrand interpretations is very desirable. As [2] remarks, however, this results in a non-recursively enumerable semantics. In this paper we show the Π_1^1-completeness of the completion semantics with restriction to Herbrand interpretations.

1. Preliminaries

Fix a first order language $\mathcal{L}$ with countable sets of predicate and functional symbols of each arity. We denote predicate symbols by $p, q, \ldots$, functional symbols by $f, g, h, \ldots$ and variables by $x, y, z, \ldots$ (all possibly subscripted and superscripted). For simplicity we assume that formulas are built using $\exists$, $\vee$ and $\neg$ and other connectives and quantifiers are defined through those.

A *program clause* is a formula of the form $\forall(A \leftarrow L_1 \wedge \ldots \wedge L_n)$, where A is an atom, $n \geq 0$ and $L_1, \ldots, L_n$ are literals. As a common convention we will omit the universal quantifier in a program clauses and simply write $A \leftarrow L_1 \wedge \ldots \wedge L_n$. The atom A is called the head and $L_1 \wedge \ldots \wedge L_n$ the body of the clause. A *(logic) program* is a finite set of program clauses. A *query* is a formula of the form $\exists(S_1 \wedge \ldots \wedge S_n)$. where $n > 0$ and $S_1, \ldots, S_n$ are literals.

Given a program P, we construct a set $comp(P)$ of formulas as follows. First we rewrite each clause $p(t_1, \ldots, t_n) \leftarrow L_1 \wedge \ldots \wedge L_m$ as

$$p(x_1, \ldots, x_n) \leftarrow \exists y_1, \ldots, y_k(x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge L_1 \wedge \ldots \wedge L_m),$$

where $x_1, \ldots, x_n$ are new variables, $y_1, \ldots, y_k$ are the variables of the original clause and $=$ is a new predicate symbol. that is interpreted as the equality. If

$$\begin{gathered} p(x_1, \ldots, x_n) \leftarrow E_1 \\ \vdots \\ p(x_1, \ldots, x_n) \leftarrow E_l \end{gathered}$$

are all the general forms of clauses in P with the predicate symbol p in the head. then $\forall(p(x_1, \ldots, x_n) \rightarrow E_1 \vee \ldots \vee E_l)$ is called the definition of p. As usual. if $l = 0$. the notation

$E_1 \vee \ldots \vee E_l$ is understood as a logical falsehood. The completion *comp*(P) of P is the set of definitions of all predicate symbols used in P.

Definition 1: *The completion semantics is the binary relation between logic programs and queries that holds if and only if the query is true in all Herbrand models of the completion of the program.*

The completion semantics was first introduced in [1]. However instead of Herbrand interpretations. [1] considers all first-order interpretations that satisfy a special equality theory called CET. Nevertheless, CET is rightfully considered to be too general. The restriction to Herbrand interpretations seems the most natural one in logic programming. But as [2] notes, the set of negative ground literals which are true in all Herbrand models of *comp*(P) for a definite program P, may not be recursively enumerable. In this paper we show the exact complexity of the completion semantics (with restriction to Herbrand interpretations).

2. Arithmetic in Logic Programming

In this section we show that the completion semantics is not arithmetical. To do that we present a technique of coding arithmetical formulas in logic programs, thereby reducing arithmetical truth to the completion semantics.

First we remind the Lloyd-Topor transformation from [3]. If we look at the algorithm of forming *comp*(P) from a logic program P, we see that it does not depend on bodies of clauses being conjunctions of literals. It is tempting to generalise logic programs by allowing arbitrary formulas in bodies of clauses. Such programs are called *extended* programs and their completions are formed exactly as those of ordinary programs. Lloyd-Topor transformation constructs an ordinary program P' from an extended one P. such that *comp*(P') is a conservative extension of *comp*(P).

For an arbitrary formula F, the clause $A \leftarrow F$ is transformed using the following rules: $A \leftarrow \exists F$ is transformed into $A \leftarrow F$; $A \leftarrow F_1 \vee F_2$ is transformed into two clauses $A \leftarrow F_1$ and $A \leftarrow F_2$; and $A \leftarrow \neg F$ is transformed into two clauses $p(x_1, \ldots, x_n) \leftarrow F$ and $A \leftarrow \neg p(x_1, \ldots, x_n)$ (here p is a new predicate symbols and $x_1, \ldots, x_n$ are new variables).

By repeatedly applying this transformation to clauses of an extended program P we get an ordinary program P'. It is shown in [3] that *comp*(P') is a conservative extension of *comp*(P). This means that every model of *comp*(P') is a model of *comp*(P) and conversely given a model of *comp*(P) we can obtain a model of *comp*(P') by uniquely interpreting predicate symbols not occurring in P.

By the *language of arithmetic* $\mathcal{L}_A$ we mean the language consisting of the nullary functional symbol 0. the unary functional symbol s. the binary predicate symbol eq and the ternary predicate symbols p_+ and $p_\times$. Consider the interpretation $\mathcal{N}$ for this language. whose domain is the set $\mathbb{N}$ of natural numbers with the following interpretations for predicate and functional symbols: 0 is interpreted as the natural number 0; s is interpreted as the successor function (i.e. $x \mapsto x + 1$); eq is interpreted as the equality; p_+ is interpreted as the graph of addition; $p_\times$ is interpreted as the graph of multiplication. It is well known that the set of formulas true in $\mathcal{N}$ is not arithmetical.

Let p_N be a new unary predicate symbol. Consider the following program:

$$\begin{array}{rcl} p_N(0) & & \\ p_N(s(x)) & \leftarrow & p_N(x) \\ eq(x,x) & \leftarrow & p_N(x) \\ p_+(x,0,x) & \leftarrow & p_N(x) \\ p_+(x,s(y),s(z)) & \leftarrow & p_+(x,y,z) \\ p_\times(x,0,0) & \leftarrow & p_N(x) \\ p_\times(x,s(y),z) & \leftarrow & p_\times(x,y,u) \wedge p_+(u,x,z) \end{array}$$

We will denote it by A. Note that A does not use negation, so *comp*(A) is consistent. In particular the least Herbrand model of A is a model of *comp*(A) (see for example [4]). Let $\mathcal{I}$ be a Herbrand model of *comp*(A). Define $\mathcal{I}_N$ to be the interpretation (for $\mathcal{L}_A$) whose domain is the set $\{s^n(0) : n \in \mathbb{N}\}$ with predicate and functional symbols interpreted as in $\mathcal{I}$.

Proposition 1: *For every Herbrand model $\mathcal{I}$ of comp(A), $\mathcal{I}_N$ is isomorphic to $\mathcal{N}$.*

Proof: It is straightforward to check that the correspondence $n \mapsto s^n(0)$ for $n \in \mathcal{N}$ is an isomorphism between $\mathcal{N}$ and $\mathcal{I}_N$. In what follows we identify the natural number n with the term $s^n(0)$. ∎

Now we can express arithmetic formulas in logic programs. Let F be a closed formula in the language of arithmetic and q be a new unary predicate symbol. Denote by F' the formula obtained from F by restricting all quantifiers to p_N[1]. Let P_F denote the extended program obtained as the union of A and the clause $q \leftarrow F'$.

Proposition 2: *The formula F is true in $\mathcal{N}$ if and only if q is a consequence of comp(P_F) on Herbrand interpretations.*

Proof: By the definition of the completion *comp*(P_F) = *comp*(A) $\cup \{q \leftrightarrow F'\}$. Let $\mathcal{I}$ be a Herbrand model of *comp*(P_F) (such a model clearly exists). Then it is a model of *comp*(A) and by the previous proposition, $\mathcal{I}_N$ is isomorphic to N. Then $\mathcal{I} \models q$ iff $\mathcal{I} \models F'$ iff $\mathcal{I}_N \models F$ iff $\mathcal{N} \models F$. ∎

As a corollary arithmetical truth is reducible to the completion semantics.

Theorem 1: *The completion semantics is not arithmetical.*

3. The Completion Semantics in Analytical Hierarchy

We have already seen that the completion semantics is not arithmetical. Here we show that it is analytical and in fact Π^1_1. Moreover. we show that the set of closed first-order formulas with equality that are true in all Herbrand interpretations is Π^1_1. To do that we construct a Π^1_1 formula of second-order arithmetic that defines this set. As a common convention. we use lowercase letters for object variables and uppercase letters for set variables.

Proposition 3: *The set of closed first-order formulas with equality that are true in all Herbrand interpretations is Π^1_1.*

Proof: Fix some effective one to one correspondence between the set of closed first-order formulas with equality and the set of natural numbers. The number corresponding to a formula is called its code. Similarly fix a one to one correspondence between the set of ground terms and the set of natural numbers. Each subset S of natural numbers defines a Herbrand interpretation $\mathcal{I}_S$ consisting of ground atoms whose codes are in S. Clearly. for

[1]Subformulas of the form $\exists xG$ are replaced by $\exists x(p_N(x) \wedge G)$ and those of the form $\forall xG$ by $\forall x(p_N(x) \rightarrow G)$.

every Herbrand interpretation there is a subset (and in fact infinitely many) that defines it. The set Y is the set of codes of closed formulas true in the Herbrand interpretation encoded by X if and only if they satisfy the following formula $True(X, Y)$:

$$\begin{gathered}
\forall x(\exists y, z\ Equal(x, y, z) \rightarrow (x \in Y \leftrightarrow y = z)) \wedge \\
\forall x(GroundAtom(x) \rightarrow (x \in Y \leftrightarrow x \in X)) \wedge \\
\forall x(\exists y\ Negation(x, y) \rightarrow (x \in Y \leftrightarrow y \notin Y)) \wedge \\
\forall x(\exists y, z\ Disjunction(x, y, z) \rightarrow (x \in Y \leftrightarrow (y \in Y \vee z \in Y))) \wedge \\
\forall x(Existential(x) \rightarrow (x \in Y \leftrightarrow \exists y(Instance(x, y) \wedge y \in Y)),
\end{gathered}$$

where

$Equal(x, y, z)$ defines the relation "y encodes a ground term t_1, z encodes a ground t_2 and x encodes the formula $t_1 = t_2$;"

$GroundAtom(x)$ defines the set "x encodes a ground atom;"

$Negation(x, y)$ defines the relation "y encodes a closed formula F and x encodes the formula $\neg F$;"

$Disjunction(x, y, z)$ defines the relation "y encodes a closed formula F, z encodes a closed formula G and x encodes the formula $F \vee G$;"

$Existential(x)$ defines the set "x encodes a closed formula of the form $\exists v F$;"

$Instance(x, y)$ defines the relation "x encodes a closed formula $\exists v F$ and y encodes the formula $F\{v/t\}$ for some ground term t."

All the aforementioned relations are recursive and hence can be defined in the first-order arithmetic. Now the set of closed formulas that are true in all Herbrand interpretations can be defined by the formula $TrueEverywhere(x)$:

$$\forall X, Y(True(X, Y) \rightarrow x \in Y). \ \blacksquare$$

Theorem 2: *The completion semantics is a* Π^1_1 *relation.*

Proof: The completion semantics is defined by the following formula $CompSem(x, y)$:

$$\exists z(Completion(x, y, z) \wedge TrueEverywhere(z)),$$

where

$Completion(x, y, z)$ defines the relation "x encodes a program P, y encodes a query Q and z encodes the formula $comp(P) \rightarrow Q$." ∎

4. Π^1_1-completeness of the Completion Semantics

We have seen in the previous section that the completion semantics is Π^1_1. Here we will show that it is in fact Π^1_1-complete. To show this we need to reduce some Π^1_1-complete relation to it. Let $\varphi^{(2)}_0, \varphi^{(2)}_1, \ldots$ be an effective enumeration of binary partial recursive functions. An ordinal α is called *recursive* if there is a recursive binary relation R which well-orders a subset of natural numbers and that well-ordering is isomorphic to α. The index of R is then called the index of α. It is well known that the set of indices of recursive ordinals, that is the set $\{i : \varphi^{(2)}_i$ *is the characteristic function of a well-ordering of a subset of* $\mathbb{N}\}$, is Π^1_1-complete (see [5]). We will reduce this set to the completion semantics.

Let $r \in \mathbb{N}$. Denote by $F_r(x, y, z)$ the formula defining the graph of $\varphi^{(2)}_r$ in the language of first-order arithmetic. That is, for every $n, m, k \in \mathbb{N}$ the following holds: $\varphi^{(2)}_r(n, m) = k$

if and only if $\mathcal{N} \models F_r(s^n(0), s^m(0), s^k(0))$. Clearly F_r can effectively be constructed from r. Let F'_r denote the formula obtained from F_r by restricting all quantifiers to p_N. Let p_R be a new binary, p_S, p_D new unary and q a new nullary predicate symbols. Consider the following formulas:

$$\begin{array}{ll} & \forall x, y((p_D(x) \wedge p_D(y)) \to (p_R(x,y) \vee p_R(y,x)))\wedge \\ TotalOrder - & \forall x, y((p_R(x,y) \wedge p_R(y,x)) \to eq(x,y))\wedge \\ & \forall x, y, z((p_R(x,y) \wedge p_R(y,z)) \to p_R(x,z))\wedge \end{array}$$

$$NoLeastElement - \exists x p_S(x) \wedge \forall x(p_S(x) \to \exists y(p_S(y) \wedge p_R(y,x) \wedge \neg eq(y,x)))$$

$$Characteristic - \forall x, y((p_N(x) \wedge p_N(y)) \to (F'_r(x,y,0) \vee F'_r(x,y,s(0)))).$$

Essentially $TotalOrder$ expresses the fact that p_R is a relation of a total order on the set defined by p_D. $NoLeastElement$ expresses the fact that the set defined by p_S is non-empty and does not contain a least element with respect to p_R and $Characteristic$ expresses the fact that F_r is interpreted in $\mathcal{N}$ as the graph of a characteristic function. Consider the following extended program P_r :

$$\begin{array}{rcl} & A & \\ p_R(x,y) & \leftarrow & F'_r(x,y,s(0)) \wedge p_N(x) \wedge p_N(y) \\ p_D(x) & \leftarrow & \exists y(p_R(x,y) \vee p_R(y,x)) \\ p_S(x) & \leftarrow & p_S(x) \wedge p_D(x) \\ q & \leftarrow & NoLeastElement \vee \neg TotalOrder \vee \neg Characteristic. \end{array}$$

Proposition 4: *The query $\neg q$ is a consequence of $comp(P_r)$ on Herbrand interpretations if and only if r is an index of recursive ordinal.*

Proof: Assume that $\neg q$ is not a consequence of $comp(P_r)$ on Herbrand interpretations. Then there is a Herbrand model $\mathcal{I}$ of $comp(P_r)$ such that $q \in \mathcal{I}$. Hence $\mathcal{I} \models comp(A)$ and so $\mathcal{I}_N$ is isomorphic to $\mathcal{N}$. The interpretation $\mathcal{I}$ satisfies the definition of p_R. So p_R is interpreted as the set $R = \{(s^n(0), s^m(0)) : \varphi_r^{(2)}(n,m) = 1\}$. The definition of p_D is the formula $\forall x(p_D(x) \leftrightarrow \exists y(p_R(x,y) \vee p_R(y,x))$. So p_D is interpreted as the union of the domain and codomain of R. Denote this set by D. The definition of p_S is the formula $\forall x(p_S(x) \leftrightarrow (p_S(x) \wedge p_D(x)))$ which is logically equivalent to $\forall x(p_S(x) \to p_D(x))$. Thus p_S is interpreted as some subset S of D. Lastly, since $q \in \mathcal{I}$, we have $\mathcal{I} \models NoLeastElement \vee \neg TotalOrder \vee \neg Characteristic$. This implies that either $\varphi_r^{(2)}$ is not the characteristic function of R or R is not a total order on D or the set S is non-empty and does not have a least element. Hence r is not an index of recursive ordinal.

Conversely assume that r is not an index of a recursive ordinal. Take some model of $comp(A)$ (such model exists since A is a definite program). Extend this interpretation by defining interpretations of p_R, p_D, p_S and q as follows. The predicate symbol p_R is interpreted as $R = \{(s^n(0), s^m(0)) : \varphi_r^{(2)}(n,m) = 1\}$. p_D is interpreted as the union of its domain and codomain (denote it by D) and q is interpreted as *true*. If R is not a total order of D or $\varphi_r^{(2)}$ is not its characteristic function, then we can interpret p_S as the empty set. Otherwise there is a non-empty subset of D that does not have a least element. Interpret p_S by that set. It is easy to see that the interpretation satisfies $comp(P_r)$. Therefore $\neg q$ is not a consequence of $comp(P_r)$. ∎

As a corollary we have the following

Theorem 3: *The completion semantics is a Π^1_1-complete relation.*

References

[1] K. L. Clark, "Negation as failure". *In H. Gallaire, J. Minker, Logic and Databeses.* pp 293-323. Plenum, 1978.

[2] J. C. Shepherdson, "Negation as failure, completion and stratification". *In D. M. Gabbay, C. J. Hogger, J. A. Robinson, Logic Programming.* pp 355-419. Clarendon Press, 1998.

[3] J. W. Lloyd, R. W. Topor. "Making prolog more expressive", *Journal of Logic Programming*, vol. 1, pp. 225-240. 1984.

[4] J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1994.

[5] H. Rogers, *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

Ընդհանրացված տրամաբանական ծրագրերի փակման սեմանտիկայի անլուծելիության աստիճանը

Լ. Հայկազյան

Ամփոփում

Փակման սեմանտիկան ներմուծվել է [1] աշխատությունում, ուր դիտարկվում են ինտերպրետացիաներ, որոնք բավարարում են հատուկ առաջին կարգի տեսության: Այս ինտերպրետացիաները ներառում են, բայց չեն սահմանափակվում Հերբրանի ինտերպրետացիաներով: Այնուամենայնիվ, տրամաբանական ծրագրավորման մեջ Հերբրանի ինտերպրետացիաներով սահմանափակումը խիստ ցանկալի է: Սակայն ինչպես նշվում է [2] աշխատությունում, դա բերում է ոչ ռեկուրսիվորեն թվարկելի սեմանտիկայի: Սույն աշխատությունում ցույց է տրվում փակման սեմանտիկայի Π_1^1-լրիվությունը Հերբրանի ինտերպրետացիաներով սահմանափակման դեպքում:

Степень неразрешимости семантики замыкания обобщенных логических программ

Л. Айказян

Аннотация

Семантика замыкания была введена в работе [1], в которой рассматриваются интерпретации, удовлетворяющие специальной теории первого порядка. Эти интерпретации включают в себя Эрбрановские интерпретации, но не ограничиваются ими. Тем не менее в логическом программировании ограничение Эрбрановскими интерпретациями весьма желательно. Однако, как отмечено в работе [2], это приводит к не рекурсивно перечислимой семантике. В данной работе показывается Π_1^1 -полнота семантики замыкания при ограничении Эрбрановскими интерпретациями.