Herbrand Semantics

Michael Genesereth
Eric Kao

Abstract: The traditional semantics for First Order Logic (sometimes called Tarskian semantics) is based on the notion of interpretations of constants. Herbrand semantics is an alternative semantics based directly on truth assignments for ground sentences rather than interpretations of constants. Herbrand semantics is simpler and more intuitive than Tarskian semantics; and, consequently, it is easier to teach and learn. Moreover, it is more expressive. For example, while it is not possible to finitely axiomatize integer arithmetic with Tarskian semantics, this can be done easily with Herbrand Semantics. The downside is a loss of some common logical properties, such as compactness and completeness. However, there is no loss of inferential power. Anything that can be proved according to Tarskian semantics can also be proved according to Herbrand semantics. In this article, we define Herbrand semantics and report in detail on its properties.

1. Introduction

First-Order Logic is perhaps the most well-known logic in use today. It provides a language for encoding information about objects and relations; and it provides a deductive mechanism for deriving conclusions from premises expressed in this language. The semantics of first-order logic gives meaning to sentences in this language. This helps us judge whether our sentences say what we think they say, and it helps us to justify the rules of inference we use in deductive reasoning.

The traditional semantics for First Order Logic (sometimes called Tarskian semantics) is based on the notion of interpretations. An interpretation consists of an arbitrary set of objects (called the universe of discourse) and an interpretation function (1) that maps object constants into elements of this set, (2) that maps function constants into functions on this set, and (3) that maps relation constants into relations on this set.

Herbrand semantics is an alternative semantics for First Order Logic based on truth assignments for ground sentences rather than interpretations for object, function, and relation constants. A model is simply a truth assignment for the ground atoms in our language. (Equivalently, it is an arbitrary subset of the ground atoms in our language.) In Herbrand semantics, there is no external universe and no interpretation function for constants. In effect, all ground terms are treated as opaque - they "represent" themselves.

One important advantage of Herbrand semantics is simplicity. It is conceptually simpler than Tarskian semantics; and, consequently, it is easier to teach and to learn. A second important feature of Herbrand semantics is expressiveness. For example, while it is not possible to finitely axiomatize integer arithmetic with Tarskian semantics, this can be done easily with Herbrand Semantics.

The downside of Herbrand semantics is a loss of some logical properties possessed by Tarskian semantics. However, as we shall argue, Herbrand semantics preserves the properties that matter for most purposes.

Note that Herbrand semantics is not new here. It is widely used in discussions of automated reasoning and logic programming and databases. However, in most cases, it is treated (justifiably) as a special case of first-order semantics. In this paper, we turn things around - we start with Herbrand semantics, we study its logical and computational properties, and we then talk about how it can be extended to First Order Logic.

We begin the paper with a definition of the syntax of our Logic. We then introduce Herbrand semantics. We discuss its expressiveness and equivalence of various subsets to other logics. We then discuss its proof theory. Finally, we talk about the relationship between Herbrand semantics and Tarskian semantics.

2. Syntax

The vocabulary of our language has two classes of words, *variables* and *constants*. By convention, we write variables as strings of alphanumeric characters beginning with a letter from the end of the alphabet, i.e. *u*, *v*, *w*, *x*, *y*, *z*. We write constants as strings of alphanumeric characters beginning with a letter from the beginning of the alphabet or a digit, e.g. *a*, *b*, *c*, *john*, 123, *r*14, and so forth.

Constants are further subdivided into *object constants*, *function constants*, and *relation constants*. Each function constant and relation constant has an associated positive integer, called its arity, which determines how many arguments it "accepts". Note that there is no inherent syntactic distinction between object constants, function constants, and relation constants. Their type is determined by their usage in larger expressions.

A *functional term* is an expression formed from an *n*-ary function constant and *n* terms. In what follows, we write functional terms in traditional mathematical notation - the function constant followed by its *arguments* enclosed in parentheses and separated by commas. For example, if *f* is a binary function constant, if *a* is an object constant, and if *y* is a variable, then $f(a,y)$ is a functional term. A *term* is a variable, an object constant, or a functional term.

There are three types of *sentences* in our language, viz. atoms, logical sentences, and quantified sentences.

An *atom* is an expression formed from an *n*-ary relation constant and *n* terms. For example, if *q* is a relation constant with arity 2 and if *a* and *y* are terms, then $q(a,y)$ is a syntactically legal relational sentence.

There are five types of logical sentences, viz. negations, conjunctions, disjunctions, implications, and biconditionals. A *negation* is an expression of the form $\neg\varphi$, where φ is an arbitrary sentence. A *conjunction* is an expression of the form $(\varphi_1 \land ... \land \varphi_n)$, where each φ_i is a sentence. A *disjunction* is an expression of the form $(\varphi_1 \lor ... \lor \varphi_n)$, where each φ_i is a sentence. An *implication* is an expression of the form $(\varphi \Rightarrow \psi)$, where each φ and ψ are sentences. A *biconditional* is an expression of the form $(\varphi \Leftrightarrow \psi)$, where each φ and ψ are sentences.

Quantified sentences are formed from a *quantifier*, a variable, and an embedded sentence. The embedded sentence is called the *scope* of the quantifier. There are two types of quantified sentences, viz. universally quantified sentences and existentially quantified sentences. A *universally quantified sentence* is used to assert that all objects have a certain property. For example, if φ is a sentence with free variable *x*, then $\forall x.\varphi[x]$ is a universally quantified sentence asserting that φ asserts of everything. An *existentially quantified sentence* is used to assert that some object has a certain property. For example, if φ is a sentence with free variable *x*, then $\exists x.\varphi[x]$ is an existentially quantified sentence asserting that φ holds of at least one thing.

In what follows, we drop unneeded parentheses, relying on precedence to disambiguate the structure of unparenthesized sentences. In Herbrand Logic, the precedence relations of the logical operators are the same as in Propositional Logic, and quantifiers have higher precedence than logical operators.

An expression is *ground* if and only if it contains no variables. For example, the sentence $p(a)$ is ground, whereas the sentence $\forall x.p(x)$ is not. An occurrence of a variable is *free* if and only if it is not in the scope of a quantifier of that variable. Otherwise, it is *bound*. A sentence is *open* if and only if it has free variables. Otherwise, it is *closed*. For example, the sentence $p(x)$ is open and the sentence $\forall x.p(x)$ is closed.

3. Herbrand Semantics

The *Herbrand vocabulary* for a logical language is the set of all object constants, a set of function constants, a set of relation constants in the language. (Note that this definition differs from that used in database circles, where a vocabulary includes a specification of relation constants but not object constants, whereas our definition includes all three types of constants.)

The *Herbrand base* for a Herbrand vocabulary is the set of all ground atoms that can be formed from the constants of the language. Said another way, it is the set of all sentences of the form $r(t_1,...,t_n)$, where r is an n-ary relation constant and t_1, ... , t_n are ground terms.

For a vocabulary with object constants a and b, no function constants, and relation constants p and q where p has arity 1 and q has arity 2, the Herbrand base is shown below.

$$\{p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b)\}$$

It is worthwhile to note that, for a given relation constant and a finite set of terms, there is an upper bound on the number of ground relational sentences that can be formed using that relation constant. In particular, for a set of terms of size b, there are b^n distinct n-tuples of object constants; and hence there are b^n ground relational sentences for each n-ary relation constant. Since the number of relation constants in a vocabulary is finite, this means that the Herbrand base is also finite.

Of course, not all Herbrand bases are finite. In the presence of function constants, the number of ground terms is infinite; and so the Herbrand base is infinite. For example, in a language with a single object constant a and a single unary function constant f and a single unary relation constant p, the Herbrand base consists of the sentences shown below.

$$\{p(a), p(f(a)), p(f(f(a))), ...\}$$

A *truth assignment* for a Herbrand Logic language is a function that maps each ground relational sentence in the Herbrand base to a truth value. In what follows, we use the digit 1 as a synonym for true and 0 as a synonym for false; and we refer to the value assigned to a ground relational sentence by writing the relational sentence with the name of the truth assignment as a superscript. For example, the truth assignment i defined below is an example for the case of the language mentioned a few paragraphs above.

$$\begin{array}{l} p(a)^i = 1 \\ p(b)^i = 0 \\ q(a,a)^i = 1 \\ q(a,b)^i = 0 \\ q(b,a)^i = 1 \\ q(b,b)^i = 0 \end{array}$$

Once we have a truth assignment for the ground relational sentences of a language, the semantics of our operators prescribes a unique extension of that assignment to the complex sentences of the language.

The rules for logical sentences in Herbrand semantics are simple. A truth assignment *i* satisfies a negation $\neg\varphi$ if and only if *i* does not satisfy φ. Truth assignment *i* satisfies a conjunction $(\varphi_1 \land ... \land \varphi_n)$ if and only if *i* satisfies every φ_i. Truth assignment *i* satisfies a disjunction $(\varphi_1 \lor ... \lor \varphi_n)$ if and only if *i* satisfies at least one φ_i. Truth assignment *i* satisfies an implication $(\varphi \Rightarrow \psi)$ if and only if *i* does not satisfy φ or does satisfy ψ. Truth assignment *i* satisfies an equivalence $(\varphi \Leftrightarrow \psi)$ if and only if *i* satisfies both φ and ψ or it satisfies neither φ nor ψ.

In order to define satisfaction of quantified sentences, we need the notion of instances. An *instance* of an expression is an expression in which all variables have been consistently replaced by ground terms. Consistent replacement here means that, if one occurrence of a variable is replaced by a ground term, then all occurrences of that variable are replaced by the same ground term.

A universally quantified sentence is true for a truth assignment if and only if *every* instance of the scope of the quantified sentence is true for that assignment. An existentially quantified sentence is true for a truth assignment if and only if *some* instance of the scope of the quantified sentence is true for that assignment.

As an example of these definitions, consider the sentence $\forall x.(p(x) \Rightarrow q(x,x))$. What is the truth value under the truth assignment shown above? According to our definition, a universally quantified sentence is true if and only every instance of its scope is true. For this language, there are just two instances. See below.

$$p(a) \Rightarrow q(a,a)$$
$$p(b) \Rightarrow q(b,b)$$

We know that $p(a)$ is true and $q(a,a)$ is true, so the first instance is true. $q(b,b)$ is false, but so is $p(b)$ so the second instance is true as well. Since both instances are true, the original quantified sentence is true.

Now let's consider a case with nested quantifiers. Is $\forall x.\exists y.q(x,y)$ true or false for the truth assignment shown above? As before, we know that this sentence is true if every instance of its scope is true. The two possible instances are shown below.

$$\exists y.q(a,y)$$
$$\exists y.q(b,y)$$

To determine the truth of the first of these existential sentences, we must find at least one instance of the scope that is true. The possibilities are shown below. Of these, the first is true; and so the first existential sentence is true.

$$q(a,a)$$
$$q(a,b)$$

Now, we do the same for the second existentially quantified. The possible instances follow. Of these, again the first is true; and so the second existential sentence is true.

$$q(b,a)$$
$$q(b,b)$$

Since both existential sentences are true, the original universally quantified sentence must be true as well.

A truth assignment *i satisfies* a sentence with free variables if and only if it satisfies every instance of that sentence. A truth assignment *i satisfies* a set of sentences if and only if *i* satisfies every sentence in the set.

4. Expressiveness

One important feature of Herbrand semantics is its expressiveness. In this section, we show that several concepts which are not expressible in first-order logic under Tarskian semantics are expressible under Herbrand semantics.

We start by showing how to finitely axiomatize Modular Arithmetic, which is also expressible under Tarskian semantics. Then we show how to finitely define Peano Arithmetic, transitive closure, and the minimal model of safe, stratified programs, all of which are not expressible under Tarskian semantics.

Modular arithmetic

In this example, we show how to characterize Modular Arithmetic in Herbrand Logic. In Modular Arithmetic, there are only finitely many objects. For example, in Modular Arithmetic with modulus 4, we would have just four integers - 0, 1, 2, 3 - and that's all. Our goal here to define the addition relation. Admittedly, this is a modest goal; but, once we see how to do this; we can use the same approach to define other arithmetic relations.

Let's start with the *same* relation, which is true of every number and itself and is false for numbers that are different. We can completely characterize the *same* relation by writing ground relational sentences, one positive sentence for each number and itself and negative sentences for all of the other cases.

$same(0,0)$	$\neg same(0,1)$	$\neg same(0,2)$	$\neg same(0,3)$
$\neg same(1,0)$	$same(1,1)$	$\neg same(1,2)$	$\neg same(1,3)$
$\neg same(2,0)$	$\neg same(2,1)$	$same(2,2)$	$\neg same(2,3)$
$\neg same(3,0)$	$\neg same(3,1)$	$\neg same(3,2)$	$same(3,3)$

Now, let's axiomatize the *next* relation, which, for each number, gives the next larger number, wrapping back to 0 after we reach 3.

$$next(0,1)$$
$$next(1,2)$$
$$next(2,3)$$
$$next(3,0)$$

Properly, we should write out the negative literals as well. However, we can save that work by writing a single axiom asserting that *next* is a functional relation, i.e., for each member of the Herbrand base, there is just one successor.

$$\forall x.\forall y.\forall z.(next(x,y) \wedge next(x,z) \Rightarrow same(y,z))$$

In order to see why this saves us the work of writing out the negative literals, we can write this axiom in the equivalent form shown below.

$$\forall x.\forall y.\forall z.(next(x,y) \land \neg same(y,z) \Rightarrow \neg next(x,z))$$

The addition table for Modular Arithmetic is the usual addition table for arbitrary numbers except that we wrap around whenever we get past 3. For such a small arithmetic, it is easy to write out the ground relational sentences for addition, as shown below.

plus(0,0,0)	*plus*(1,0,1)	*plus*(2,0,2)	*plus*(3,0,3)
plus(0,1,1)	*plus*(1,1,2)	*plus*(2,1,3)	*plus*(3,1,0)
plus(0,2,2)	*plus*(1,2,3)	*plus*(2,2,0)	*plus*(3,2,1)
plus(0,3,3)	*plus*(1,3,0)	*plus*(2,3,1)	*plus*(3,3,2)

As with *next*, we avoid writing out the negative literals by writing a suitable functionality axiom for *plus*.

$$\forall x.\forall y.\forall z.\forall w.(plus(x,y,z) \land \neg same(z,w) \Rightarrow \neg plus(x,y,w))$$

That's one way to do things, but we can do better. Rather than writing out all of those relational sentences, we can use Herbrand Logic to define *plus* in terms of *same* and *next* and use that axiomatization to deduce the ground relational sentences. The definition is shown below. First, we have an identity axiom. Adding 0 to any number results in the same number. Second we have a successor axiom. If *z* is the sum of *x* and *y*, then the sum of the successor of *x* and *y* is the successor of *z*. Finally, we have our functionality axiom once again.

$$\forall y.plus(0,y,y)$$
$$\forall x.\forall y.\forall z.\forall x2.\forall z2.(plus(x,y,z) \land next(x,x2) \land next(z,z2) \Rightarrow plus(x2,y,z2))$$
$$\forall x.\forall y.\forall z.\forall w.(plus(x,y,z) \land \neg same(z,w) \Rightarrow \neg plus(x,y,w))$$

One advantage of doing things this way is economy. With these sentences, we do not need the ground relational sentences about *plus* given above. They are all logically entailed by our sentences about *next* and the definitional sentences. A second advantage is versatility. Our sentences define *plus* in terms of *next* for arithmetic with any modulus, not just modulus 4.

Peano arithmetic

Now, let's look at the problem of encoding arithmetic for all of the natural numbers. Since there are infinitely many natural numbers, we need infinitely many terms.

One way to get infinitely many terms is to have a vocabulary with infinitely many object constants. While there is nothing wrong with this in principle, it makes the job of axiomatizing arithmetic effectively impossible, as we would have to write out infinitely many literals to capture our successor relation.

An alternative approach is to represent numbers using a single object constant (e.g. 0) and a single unary function constant (e.g. *s*). We can then represent every number *n* by applying the function constant to 0 exactly *n* times. In this encoding, *s*(0) represents 1; *s*(*s*(0)) represents 2; and so forth. With this encoding, we automatically get an infinite universe of terms, and we can write axioms defining addition and multiplication as simple variations on the axioms of Modular Arithmetic.

Unfortunately, even with this representation, axiomatizing Peano Arithmetic is more challenging than axiomatizing Modular Arithmetic. We cannot just write out ground relational sentences to characterize our relations, because there are infinitely many cases to consider. For Peano Arithmetic, we must rely on logical sentences and quantified sentences, not just because they are more economical but because they are the only way we can characterize our relations in finite space.

Let's look at the same relation first. The axioms shown here define the *same* relation in terms of 0 and *s*.

$$\forall x.same(x,x)$$
$$\forall x.(\neg same(0,s(x)) \wedge \neg same(s(x),0))$$
$$\forall x.\forall y.(\neg same(x,y) \Rightarrow \neg same(s(x),s(y)))$$

It is easy to see that these axioms completely characterize the *same* relation. By the first axiom, the same relation holds of every term and itself. The other two axioms tell us what is not true. The second axiom tells us that 0 is not the same as any composite term. The same holds true with the arguments reversed. The third axiom builds on these results to show that non-identical composite terms of arbitrary complexity do not satisfy the same relation. Viewed the other way around, to see that two non-identical terms are not the same, we just strip away occurrences of s from each term till one of the two terms becomes 0 and the other one is not 0. By the second axiom, these are not the same, and so the original terms are not the same.

Once we have the *same* relation, we can define the other relations in our arithmetic. The following axioms define the plus relation in terms of 0, *s*, and *same*. Adding 0 to any number results in that number. If adding a number *x* to a number *y* produces a number *z*, then adding the successor of *x* to *y* produces the successor of *z*. Finally, we have a functionality axiom for *plus*.

$$\forall y.plus(0,y,y)$$
$$\forall x.\forall y.\forall z.(plus(x,y,z) \Rightarrow plus(s(x),y,s(z)))$$
$$\forall x.\forall y.\forall z.\forall w.(plus(x,y,z) \wedge \neg same(z,w) \Rightarrow \neg plus(x,y,w))$$

The axiomatization of multiplication is analogous. Multiplying any number by 0 produces 0. If a number *z* is the product of *x* and *y* and *w* is the sum of *y* and *z*, then *w* is the product of the successor of *x* and *y*. As before, we have a functionality axiom.

$$\forall y.times(0,y,0)$$
$$\forall x.\forall y.\forall z.\forall w.(times(x,y,z) \wedge plus(y,z,w) \Rightarrow times(s(x),y,w))$$
$$\forall x.\forall y.\forall z.\forall w.(times(x,y,z) \wedge \neg same(z,w) \Rightarrow \neg times(x,y,w))$$

That's all we need - just three axioms for *same* and three axioms for each arithmetic function.

Before we leave our discussion of Peano aritmetic, it is worthwhile to look at the concept of Diophantine equations. A *polynomial equation* is a sentence composed using only addition, multiplication, and exponentiation with fixed exponents (that is numbers not variables). For example, the expression shown below in traditional math notation is a polynomial equation.

$$x^2 + 2y = 4z$$

A *natural Diophantine equation* is a polynomial equation in which the variables are restricted to the natural numbers. For example, the polynomial equation here is also a Diophantine equation and happens to have a solution in the natural numbers, viz. x=4 and y=8 and z=8.

Diophantine equations can be readily expressed as sentences In Peano Arithmetic. For example, we can represent the Diophantine equation above with the sentence shown below.

$$\forall x.\forall y.\forall z.\forall u.\forall v.\forall w.(times(x,x,u) \land times(2,y,v) \land plus(u,v,w) \Rightarrow times(4,z,w))$$

This is a little messy, but it is doable. And we can always clean things up by adding a little syntactic sugar to our notation to make it look like traditional math notation.

Once this mapping is done, we can use the tools of logic to work with these sentences. In some cases, we can find solutions; and, in some cases, we can prove that solutions do not exist. This has practical value in some situations, but it also has significant theoretical value in establishing important properties of Herbrand Logic, a topic that we discuss in a later section.

Transitive Closure

Consider the language consisting of the object constant 0 and the unary function constant s. Let Φ_{TC} be the following set of axioms.

- $p(x,z) \Leftrightarrow q_{h(x,z,0)}$
- $q_h(x,z,n) \lor q_h(x,y,n) \land p(y,z) \Leftrightarrow q_h(x,z,s(n))$
- $q(x,z) \Leftrightarrow \exists n.q_h(x,z,n)$

Proposition. For any model M over the relation p, there is exactly one model M^+ of Φ_{TC} (over p, q, q_h) that extends M. Furthermore, q is the transitive closure of p in M+.

Minimal Model of Safe, Stratified Programs

Using Herbrand semantics, the unique model of a safe, stratified program can be axiomatized in the language of first-order logic without minimization constructs. In this section, we show how a straightforward transformation from a program to a finite theory in the language of first-order logic captures the stratified minimal model semantics.

First, we develop the theory and methods assuming the constants in the language consists of the object constant 0 and the unary function constant s. Then, at the end of the section, we generalize the result by removing the constraint on the object constants and function constants of the language.

Program normalization

Let Π be a program over the relations **R**.

Define the N[.] transformation (normalize) as follows.

For each relation r in **R**, replace all rules with r in the head by a single rule r(X) :- $b_1(T_1) \lor \ldots \lor b_n(T_n)$, where each bn is a conjunction of atoms, X is a vector of variables, and each T_i is a vector of terms, and for every i, Vars(X) subsetof Vars(T_i) [Note, a simple atom r(T) is treated as a rule r(X) :- T = X]

One can do this in a way that preserves equivalence to the input program. The process is essentially one of collecting separate rules into one and standardizing variables.

For example, the following program:

```
p(0,1)
p(X,Y) :- q(X,0) & p(Y,Z)
```

```
q(X,Y) :- p(X,0) & q(Y,Z)
```

Normalizes into the program:

```
p(X,Y) :- (X,Y) = (0,1) ∨ q(X,0) & p(Y,Z)
q(X,Y) :- p(X,0) & q(Y,Z)
```

Semi-positive programs

Let Π be a safe semi-positive program over the relations **R** and the constants ⟨ 0 , s(.) ⟩.

Define the H[.] transformation as follows.

Take a normalized program and perform the following steps:

1. For each rule r(X) :- $b_1(T_1)$ ∨ ... ∨ $b_n(T_n)$, we do the following:
    a. replace each atom s(T) in a body with the helper predicate sh(T,N) [T is a vector of terms, equality atoms are left unchanged.]
    b. replace the head r(X) with r_h(X,s(N)), where r_h is a new helper relation not in **R**
    c. For each $b_i(T_i)$, existentially quantify the variables in T_i but not in X. The quantifier scopes over the disjunct.
    d. replace the :- with <=>
2. For each relation r ∈ **R**, add the axioms:
    - r_h(X,s(N)) <= r_h(X,N)
    - r(X) <=> ∃N.r_h(X,N)
    - ¬r_h(X,0)

For example, the normalized program:

```
p(X,Y) :- (X,Y) = (0,1) ∨ q(X,0) & p(Y,Z)
q(X,Y) :- p(X,0), q(Y,Z)
```

Axiomatizes into:

- p_h(X,Y,s(N)) <=> ∃Z. [(X,Y) = (0,1) ∨ q_h(X,0,N) & p_h(Y,Z,N)]
- q_h(X,Y,s(N)) <=> ∃Z. [p_h(X,0,s(N)) & q_h(Y,Z,s(N))]
- p_h(X,Y,s(N)) <= p_h(X,Y,N)
- p(X,Y) <=> ∃N.p_h(X,Y,N)
- ¬p_h(X,Y,0)
- q_h(X,Y,s(N)) <= q_h(X,Y,N)
- q(X,Y) <=> ∃N.q_h(X,Y,N)
- ¬q_h(X,Y,0)

Let the transformation H[[N[.]] be called AxiomSP[.].

Theorem 9.1. (Herbrand Axiomatization of Semi-positive Programs): Let Π be a safe semi-positive program over ⟨**R**, 0, s(.)⟩. Let M be the unique minimal model of Π. Then AxiomSP[Π] has a unique model M^+ in under Herbrand semantics, and M^+ = M over **R**.

Proof.

1. (Existence) We show that AxiomSP[Π] has a model M^+.

   Let M be the unique minimal model of the program Π. For a set of ground atoms G, let T(G) be the inflation of G adding to G every ground atom that is the head of a rule instance of Π whose body is satisfied by G. Let M_0, M_1, ..., M_n be defined as follows:

   - $M_0 = \varnothing$
   - $M_{k+1} = T(M_k)$

   Let M^+ be defined as follows:

   - For each r ∈ **R**, for each tuple t, for each nonnegative integer k, $r_h(t,k)$ is true in M^+ if and only if r(t) is true in M_k.
   - For each r ∈ **R**, for each tuple t, r(t) is true in M^+ if and only if there exists a nonnegative integer k such that $r_h(t,k)$ is true in M^+. It can be checked that M^+ is a model of AxiomSP[Π].

2. (Agreement) We show that M^+ = M over **R**. Let M^+ be a model of AxiomSP[Π].

   a. We show that for every relation r ∈ **R** and every nonnegative integer k, $r_h(X,k)$ is true in M^+ if and only if r(X) is true in M_k.

      Base case:
      For every r ∈ **R** and every instance t of X, $\neg r_h(t,0)$ is false by the axioms $\neg r_h(X,0)$, agreeing with $M_0 = \varnothing$.

      As induction hypothesis, assume that (a) holds for k = j. (IH) As induction step, we show that (a) holds for k = j + 1.

      (<=) Let r(t) ∈ M_{j+1}. We consider the two cases:

      case i: r(t) ∈ M_j. By (IH), $r_h(t,j)$ is true in M^+. Then by the axiom ($r_h(X,s(N))$ <= $r_h(X,N)$ ∈ AxiomSP[Π], $r_h(t,j+s)$ is true in M^+.

      case ii: r(t) ∈ (M_{j+1} - M_j). There is a ground instance of a rule in N[Π] that adds r(t) to M_{j+1}. Then by H.1 in the H transformation, there is a corresponding axiom in H[N[Π]] that ensures $r_h(t,j+1)$ is true in M^+.

      (=>) Let $r_h(t,j+1)$ is true in M^+. Again we consider two cases:

      case i: $r_h(t, j)$ is true in M^+. Then by (IH), r(t) is true in M_j. By definition of M_{j+1}, r(t) is also true in M_{j+1}.

      case ii: $r_h(t, j)$ is false in M^+. By (IH), { $r_h(. , j)$ | r ∈ **R** } agrees with M_j. According to H.1 in the H transformation, there is a single axiom $r_h(X,s(N))$ <=> Φ(N) that uniquely determines $r_h(. , j+1)$ in terms of { $r_h(. , j)$ | r ∈ **R** }. Since $r_h(t, j+1)$ is true in M^+ by the axiom $r_h(X,s(N)) <=> \Phi(N)$, the corresponding rule in N[Π], when evaluated on M_j, makes r(t) true in M_{j+1}.

By induction, we conclude (a).

b. We show that $M^+ = M$ over **R**. For any r ∈ **R**, r ∈ **R**, r(t) ∈ M if and only if there exists k s.t. r(t) ∈ M_k (by the inflationary fixed point semantics of semi-positive programs), if and only if there exists k s.t. $r_h(t,k)$ as true in M^+ (by (a)), if and only if r(t) is true in M^+ (by the axiom $r(X) <=> \exists N.r_h(X,N)$ in AxiomSP[Π]).
c. (Uniqueness) We show that a model of AxiomOP[Π] is unique. By (1.a), $r_h(. , .)$ is uniquely determined for every r ∈ **R**. By (1.b), r(.) is uniquely determined for every r ∈ **R**. So the entire model is uniquely determined.

Safe, stratified programs

First we slightly generalize AxiomSP[.] for open programs.

Define AxiomOPSP[Π] as follows. [Π is an open, semi-positive program] Every step is the same as AxiomSP, except that we restrict attention to relations that occur in the head of some rule in N[Π]. Specifically:

1. For each rule `r(X) :- b`$_1$`(T`$_1$`) ∨ ... ∨ b`$_n$`(T`$_n$`)`, we do the following:
   a. For each atom s(T) in a body, *if relation s occurs in a head*, replace s(T) with the helper predicate $s_h(T,N)$ [equality atoms are left unchanged.]
2. For each relation r, *if relation r occurs in a head*, add the axioms:
   - $(r_h(X,s(N)) <= r_h(X,N)$
   - $r(X) <=> \exists N.r_h(X,N)$
   - $\neg r_h(X,0)$

It is straightforward to generalize theorem 9.1 to the following:

Theorem 9.2. (Herbrand Axiomatization of Open, Semi-positive Programs) Let Π be an open, safe semi-positive program over ⟨ **R**, 0, s(.) ⟩ . Let $\mathbf{R}_E$ be the set of all relations in **R** that do not occur in the head of a rule in Π. Let M_E be a model over $\mathbf{R}_E$ (written as a set of ground atoms over $\mathbf{R}_E$). Let M be the unique minimal model of Π ∪ M_E. Then AxiomOPSP[Π], under Herbrand semantics, has a unique model M^+ that agrees with M_E. Furthermore, $M^+ = M$ over **R**.

Finally, we are ready to define the axiomatization for a general safe, stratified program.

Given a safe, stratified program Π, with a stratification ⟨ $(\Pi^0, \Pi^1,...,\Pi^n)$, sigma(.) ⟩. (For definition, see pages 381-382 in [AHV95] or slide 11 in [Abi09], (with the slight change that Π^0 defines ebd(Π)).

Define Axiom[Π] to be $\cup_k$ AxiomOPSP(Π^k)

Finally, we state the main theorem:

Theorem 9.3. (Herbrand Axiomatization of Safe, Stratified Programs) Let Π be a safe, stratified program over $\langle \mathbf{R}, 0, s(.) \rangle$. Let M be the unique minimal model of Π under stratified semantics. Then Axiom[Π] has a unique model M^+ under Herbrand semantics, and $M^+ = M$ over $\mathbf{R}$.

Proof:

Let T = Axiom[Π].

Let $\langle (\Pi^0,...,\Pi^n), sigma(.) \rangle$ be the stratification of Π used in producing T. Let $\mathbf{R}_0$ = edb(Π). Let $\mathbf{R}_{k+1} = \mathbf{R}_k \cup \{ r : r \in \mathbf{R}$ and sigma(r) = k + 1}.

Define a sequence of models M_0, ..., M_n inductively as follows:

Let M_0 be the unique minimal model of Π^0 according to the standard semantics for semi-positive programs. For each k ∈ {1, ..., n}, let M_k be the unique minimal model of $\Pi^k \cup M_{k-1}$ according to the standard semantics for semi-positive programs (where M_{k-1} is interpreted as a set of ground atoms).

Let M = M_n. M is the unique minimal model of Π under stratified semantics. (For reference, see pages 381-382 in [AHV95] or slide 11 in [Abi09])

We can apply theorem 2 inductively to show the existence of a sequence of models M^+_0, ..., M^+_n as follows. Applying theorem 2 with $\Pi = \Pi^0$, $\mathbf{R} = \mathbf{R}_0$, and $\mathbf{R}_E$ = {}, we conclude there exists a unique model M^+_0 of AxiomSP[Π^0] over $\mathbf{R}_0$. Furthermore, $M^+_0 = M_0$ over $\mathbf{R}_0$. Applying theorem 2 with $\Pi = \Pi^1$, $\mathbf{R} = \mathbf{R}_1$, and $\mathbf{R}_E = \mathbf{R}_0$, we conclude there exists a unique model M^+_1 over $\mathbf{R}_1$ that extends M^+_0 and satisfies AxiomSP[Π^1]. Furthermore, $M^+_1 = M_1$ over $\mathbf{R}_1$.

For each k ∈ {2, ..., n}, we apply theorem 2 with $\Pi = \Pi^k$, $\mathbf{R} = \mathbf{R}_k$, and $\mathbf{R}_E = \mathbf{R}_{k-1}$, we conclude there exists a unique model M^+_k over $\mathbf{R}_k$ that extends M^+_{k-1} and satisfies AxiomSP[Π^k]. Furthermore, $M^+_k = M_k$ over $\mathbf{R}_k$.

Let $M^+ = M^+_n$. M^+ is a unique model of T over $\mathbf{R}$. Furthermore, M^+ = M over $\mathbf{R}$.

General language

So far in this section, several expressiveness results rely on the underlying language consisting of exactly one object constant and exactly one unary function constant. In this section, we show how we generalize these results to any language that contains at least one object constant and at least one function constant (of any arity above 0).

The approach is simple, for any language that contains an object constant (say 0) and a unary function constant (sayit s(.)). We introduce a new helper relation w(.) and introduce additional axioms so that w(.) is true for exactly the terms 0, s(0), s(s(0)), ...

Once w(.) is captured, every quantification over a {0, s(0), s(s(0)), ...} can be expressed with the aid of w(.).

Axiomatization of w(.)

To axiomatize w(.), we first need to axiomatize the identity relation (id(. , .))such that two ground terms are identical if and only if they are syntactically identical.

For simplicity, consider a language with the object constants 0,1 and the function constants s(.), f(.,.). The following axioms axiomatized the id(.,.) relation.

- ∀x.id(x,x)
- ∀x.∀y.(id(x,y) <=> id(y,x))
- ¬id(0,1)
- ∀x.(¬id(s(x),0) ∧ ¬id(s(x), 1)
- ∀x.∀y.(¬id(f(x,y),0) ∧ ¬id(f(x,y),1))
- ∀x.∀y.∀z.(¬id(f(x,y),s(z)))
- ∀x.∀x'.(id(s(x),s(x') => id(x,x'))
- ∀x.∀y.∀x'.∀y'.(id(f(x,y),f(x',y') => id(x,x') ∧ id(y,y'))

The axioms can be suitably generalized to any language *L* with a finite number of object constants and a finite number of function constants. Call the set of axioms I_L.

Let W be the set consisting of the following axioms.

1. w(0)
2. ∀x.(w(x) <=> w(s(x)))
3. ∀x.(w(x) => (id(x,0) ∨ ∃y. id(x,s(y))))

Lemma. Let *L* be a lanugage consisting of finitely many object constants, including 0, and a finitely many function constants, including s(.). Let M be any model of I_L ∪ W under Herbrand semantics. Then for any ground term t, w(t) is true in M if and only if t ∈ {0, s(0), s(s(0)), ...}.

Proof.

1. It's clear that if t ∈ { 0, s(0), ... } then w(t) is true in M.
2. Assume w(t) and t ∉ { 0, s(0), ... } (+)

   Case 1: t is not 0 and t is not identical to s(t') for any t'. Then (c.) is contradicted.

   Case 2: t = s(t') for some t'. Then there exists t'' s.t. t = s(s(... s(t''))), where t'' is not identical to s(t''') for any t'''. Also by (+), t'' is not identical to 0. Then (c.) is contradicted by applying Case 1 to t'' in place of t.

   By contradiction, we have that if w(t) is true in M then t ∈ { 0, s(0), ... }.

Having axiomatized w(.), axiomatizations that relied on quantifying over the entire universe spanned by ⟨ 0, s(.) ⟩ can now be achieved by quantifying over w(.), even if the actual universe contains more terms than those spanned by 0 and s(.).

For example, consider the transitive closure axiomatization from section 4. Φ_{TC} is the following set of axioms.

- p(x,z) <=> $q_{h(x,z,0)}$

- $q_h(x,z,n) \lor q_h(x,y,n) \land p(y,z) <=> q_h(x,z,s(n))$
- $q(x,z) <=> \exists n.q_h(x,z,n)$

We can amend the axiomatization by replacing the third axiom by $q(x,z) <=> \exists n.(w(n) \land q_h(x,z,n))$

. The amended axiomatization now axiomatizes transitive closure in the more general setting.

Proposition. Let *L* be a finite language consisting of at least one object constant (say 0) and at least one unary function constant (say s). For any model M over the relation p, there is exactly one model M^+ of $\Phi_{TC} \cup I_L \cup W$ (over p, q, q_h, id, w) that extends M. Furthermore, q is the transitive closure of p in M+.

If the language chosen does not contain a unary function constant, but only a higher arity function constant (say f(.,.)), then the same effect can be achieved be using f(t,t) in place of s(t) everywhere s(t) appears.

5. Model Theory

Recall that compactness means that, if an infinite set of sentences is unsatisfiable, there is some finite subset that is unsatisfiable. It guarantees finite proofs.

Theorem 5 (NonCompactness). : Herbrand logic is not compact. Proof. Consider the following infinite set of sentences.

p(a)
p(f (a))
p(f (f (a))) p(f (f (f (a))))
...

Assuming the vocabulary is {p, a, f }, the ground terms are a, f (a), f (f (a)), . . . , and this set of sentences entails âˆ€x.p(x). Add in the sentence âˆƒx.Â¬p(x). Clearly, the infinite set is unsatisfiable. However, every finite subset is satisfiable with respect to the vocabulary {p,a,f}. (Every finite subset is missing either âˆƒx.Â¬p(x) or one of the sentences above. If it is the former, the set is satisfiable, and if it is the latter, the set can be satisfied by making the missing sentence false.) Thus, compactness does not hold.

Model theory in Herbrand logic is much simpler than it is in first-order logic.

Unlike first-order logic where two models A and B can be equivalent with varying strengths, i.e. elementarily equivalent, secondarily equivalent, ..., isomorphic, and equal, there is no such hierarchy in Herbrand Logic. Two models either satisfy all the same sentences because they are the same exact model, or they disagree on some ground atom, i.e. one satisfies it, and the other satisfies its negation. Thus, the set of ground literals satisfied by a model uniquely identify it.

In our discussion of model theory, we use the usual notation for the set of models that satisfy a given sentence set âˆ†: Mod[âˆ†]. The set of sentences entailed by a set of sentences âˆ† is denoted Cn[âˆ†]. Also, the set of sentences true in all models in the set M is denoted as usual by Th[M].

Theorem 7. : Let S be the set of ground literals satisfied by a Herbrand model M. S is satisfied by exactly one Herbrand model: M. Equivalently, Mod[Th[M]] = {M}.

Corollary 8. : Lowenheim-Skolem-Tarski does not hold in Herbrand logic.

Proof. Lowenheim-Skolem-Tarski is a theorem in first-order logic that states if a set of sentences has a model of any infinite size, it has a model of every infinite size. Clearly this theorem does not hold in Herbrand logic, since every infinite model has countable size.

Unlike Finite Herbrand Logic, not all theories are finitely axiomatizable.

Theorem 8. Some Herbrand models are not finitely axiomatizable.

Proof. Consider a vocabulary with the set of terms {a,f(a),f(f(a)),f(f(f(a))),...}, and a single, unary relation constant p. The set of all Herbrand models over this vocabulary is a subset of the ground atoms {p(a),p(f(a)),p(f(f(a)),...}. There are 2Ï‰ such subsets, which is uncountably infinite.

Since there are only countably many sentences from a countable vocabulary, there are models without finite axiomatizations.

This statement in this Corollary was made earlier with the condition that checking whether a candidate proof actually proves a conjecture is decidable. There is no such condition on this theorem. Skolemization is another mainstay of many logics that does not have the usual effect in Herbrand logic.

Theorem 6. In Herbrand logic, Skolemization does not preserve satisfiability. Proof. The following set of sentences are unsatisfiable.

p(a) âˆƒx.Â¬p(x)

Skolemizing produces a satisfiable set of sentences: p(a), Â¬p(k)

This result is not surprising given the dependence of Herbrand satisfaction on the vocabulary.

6. Proof Theory

[TODO: Domain Closure and Induction Rules.]

Theorem 10.4 (Soundness of Herbrand Fitch): Given a set of sentences Σ and a sentence φ, if Σ |-$_{HL}$ φ then Σ |=$_{HL}$ φ.

Theorem 10.5 (Incompleteness of Herbrand Fitch): There exists a set of sentences Σ and a sentence φ such that Σ |=$_{HL}$ φ but it is not the case that Σ |-$_{HL}$ φ.

Corollary 7 (Infinite Proofs). : In Herbrand logic, Some entailed sentences admit only infinite proofs.

Proof. The above proof demonstrates a set of sentences that entail âˆ€x.p(x). The set of premises in any finite proof will be missing one of the above sentences; thus, those premises do not entail âˆ€x.p(x). Thus no finite proof can exist for âˆ€x.p(x).

7. Decidability

In a previous section, we showed that Herbrand semantics makes the logic highly expressive. The expressiveness directly impacts the computability of the logic. In this section, we first discuss useful fragments that enjoy good computational properties under Herbrand semantics. Then, we show that Herbrand semantics, in generally, is highly undecidable.

When the number of ground terms in a language is finite, the problem of sentence validity is decidable.

Theorem 7.1. Let L be a language with finitely many object constants and zero function constants. Under L, the problem of sentence validity is decidable.

Proof.
In a language with finitely many object constants and no function constants, the set of ground terms is finite. Therefore, the sentence validity problem under Herbrand Semantics can be reduced via grounding to the sentence validity problem in Propositional Logic.

When the number of object constants is countably infinite, the problem of sentence validity is semi-decidable.

Theorem 7.2. Let L be a language with countably infinitely many object constants (and zero to ω function constants). Under L, the problem of sentence validity is semi-decidable.

Proof.
By Corollary 8.6 in the next section, under language L, a sentence is unsatisfiable (or valid) under Herbrand Semantics if and only if it is unsatisfiable (or valid) under First Order Logic semantics. Sentence validity under First Order Logic is, semi-decidable therefore sentence validity under Herbrand Semantics is also semi-decidable.

The next result is a more general version of the previous theorem.

First we define some notations.

For a language L, GroundTerms[L] is the set of all ground terms over the language.

For a set of sentences T, MinLang[T] is the language that consists exactly of the constants (relation constants, object constants, and function constants).

Theorem 7.3. Over a language L, sentence validity is semi-decidable over the class of sentences {φ | GroundTerms[L] - GroundTerms[MinLang[φ]] is infinite}.

Proof.
Sentence validity according to First Order Logic semantics is semi-decidable. By Corollary 8.4, for any sentence φ in the specified class, φ is valid according to First Order Logic semantics if and only if φ is valid according to Herbrand semantics. Therefore, over the specified class of sentences, sentence validity according to Herbrand semantics is semi-decidable.

In general, the sentence validity problem is highly undecidable under Herbrand semantics.

We establish some notations and conventions for the remainder of this section.

Consider the following hierarchy of computational problems.
Let ∅ be a decidable problem.
Let ∅' be the Halting Problem.

Let $\varnothing''$ be the Halting Problem for oracle machines with an $\varnothing'$ oracle.
More generally, let $\varnothing^{(n+1)}$ be the Halting Problem for oracle machines with an $\varnothing^{(n)}$ oracle.

Let $\Sigma_n(L)$ be the set of prenex normal form L-sentences with up to n alternating quantifiers, leading with an existential quantifier.

Let $\Pi_n(L)$ be the set of prenex normal form L-sentences with up to n alternating quantifiers, leading with a universal quantifier.

For a class of sentences C, let |= C be the following problem:
Input: a sentence in the class C.
Output: whether C is a valid sentence in Herbrand Logic.

Theorem 7.4. Let L be a language with finitely many object constants and at least one function constant. The following statements hold:
A1. For all n >= 0, $\models \Sigma_n(L)$ is Turing-reducible to $\varnothing^{(n)}$
A2. For all n >= 0, $\models \Pi_n(L)$ is Turing-reducible to $\varnothing^{(n)}$
B1. For all n >= 0, $\models \Sigma_{n+1}(L)$ is not Turing-reducible to $\varnothing^{(n)}$
B2. For all n >= 0, $\models \Pi_{n+1}(L)$ is not Turing-reducible to $\varnothing^{(n)}$

Proof.
Part A1. Direct proof by induction:
Base case: Each $\Sigma_0(L)$ or $\Pi_0(L)$ sentence is ground and therefore decidable ($\varnothing$) in validity.
Inductive hypothesis: Assume for induction that both $\models\Sigma_i(L)$ and $\models\Pi_i(L)$ are Turing-reducible to $\varnothing^{(i)}$.
Inductive step: Given any $\Sigma_{i+1}(L)$ sentence φ, φ can be written as $\exists\nu.\psi$, where ψ is a $\Pi_i(L)$ sentence. There is an oracle machine with a $\varnothing^{(i)}$ oracle that semi-decides $\models\Sigma_{i+1}(L)$ by checking whether $\psi(\tau)$ is valid (a Π_i problem decidable by the $\varnothing^{(i)}$ oracle) for each $\Sigma_{i+1}(L)$ sentence $\exists\nu.\psi(\nu)$. Si+1(L) is Turing-reducible to $\varnothing^{(i+1)}$. Furthermore, $\models \Pi_{i+1}(L)$ is Turing-reducible to $\models \Sigma_{i+1}(L)$ which is, in turn, Turing reducible to $\varnothing^{(i+1)}$.
Finally, we conclude by induction that both A1 and A2 hold.

Part B1. Let A be a finite set of Π_1 sentences that axiomatizes natural numbers in Herbrand Logic. A $\models \Sigma_{n+1}(L)$ is equivalent to $\models$ (A => $\Sigma_{n+1}(L)$), which is a specialization of $\models \Sigma_{n+1}(L)$. A $\models \Sigma_{n+1}(L)$ is Turing-reducible to $\models \Sigma_{n+1}(L)$.

Suppose $\models \Sigma_{n+1}(L)$ is Turing-reducible to $\varnothing^{(n)}$, then A $\models \Sigma_{n+1}(L)$ is Turing-reducible to $\varnothing^{(n)}$. This contradicts lemma 1 that A $\models \Sigma_{n+1}(L)$ is not Turing-reducible to $\varnothing^{(n)}$.

Part B2. $\models \Sigma_{n+1}(L)$ is Turing-reducible to $\models \Pi_{n+1}(L)$. Suppose $\models \Pi_{n+1}(L)$ is Turing-reducible to $\varnothing^{(n)}$, then we have $\models \Sigma_{n+1}(L)$ Turing-reducible to $\varnothing^{(n)}$, contradicting B1.

Lemma 1. Let L be the language of natural numbers arithmetic. Let A be a finite set of Π_1 sentences that axiomatize the natural numbers in Herbrand Logic. For every n >= 0, A $\models \Sigma_{n+1}(L)$ is not Turing-reducible to any $\varnothing^{(n)}$ for any finite n.

Proof.
A set of natural numbers X (or equivalently, a decision problem X) is of the arithmetic hierarchy $\Sigma_m(L)$ iff it can be characterized by a $\Sigma_m(L)$ formula. formally, X is in $\Sigma_m(L)$ iff there exists φ in $\Sigma_m(L)$ such that n is a member of X iff φ(n) holds over the structure of natural numbers.

Because we know there is an complete axiomatization of the natural numbers under Herbrand semantics, call the set of axioms A, we can state each problem in the arithmetic hierarchy as an equivalent problem in Herbrand Entailment.

Post's theorem relating Turing degrees to the arithmetic hierarchy tells us that a problem B is Turing-reducible to $\varnothing^{(n)}$ if and only if the problem is in $\Delta_{n+1}(L)$.

For any n >= 1, there exists a problem B_n in $\Sigma_n(L)$ - $\Delta_n(L)$. [Basic result about the arithmetic hierarchy]

We conclude that for any $\varnothing^{(n)}$, there exists a $\Sigma_{n+1}(L)$ sentence φ such that {m | φ(m) holds over the structure of natural numbers} is not Turing-reducible to $\varnothing^{(n)}$. So the equivalent problem {m | A |= φ(m)} is not Turing-reducible to $\varnothing^{(n)}$. Finally, the more general problem A |= $\Sigma_{n+1}(L)$ is not Turing-reducible to $\varnothing^{(n)}$.

8. Relationship to First Order Logic

In this section, we explicate some of the relationships between Herbrand Semantics and traditional First Order Logic semantics.

For any given set of equality-free sentences Σ, we know that every satisfying truth assignment in Herbrand Logic corresponds to a satisfying First Order Logic interpretation of Σ. As a result, if an equality-free sentence φ is satisfied in all of the First Order Logic interpretations that satisfy Σ, then φ is also satisfied in all the Herbrand Logic truth assignments of Σ.

From the definition of logical entailment, we obtain the following theorem that whenever a sentence is entailed under First Order Logic, it is also entailed under Herbrand Logic. That is, Herbrand Logic is deductively more powerful than First Order Logic.

Theorem 8.1: Given a set of equality-free sentences Σ and an equality-free sentence φ, if Σ $|=_{FO}$ φ then Σ $|=_H$ φ.

A similar result holds for sentences with equality when we extend Herbrand Logic to incorporate equality.

Herbrand Logic itself is expressively powerful but there is no (recursively enumerable) complete proof procedure for Herbrand Logic. Even so, the proof system for Herbrand Logic remains more powerful than First Order Logic.

Theorem 8.2: Given a set of equality-free sentences Σ and an equality-free sentence φ, if Σ $|=_{FO}$ φ then Σ $|-_H$ φ.

A similar result holds for sentences with equality when we include the two equality rules into Fitch for Herbrand Logic.

A further relationship between Herbrand Logic and First Order Logic is that First Order Logic can be simulated by Herbrand Logic.

Theorem 8.3 [Satisfiability Equivalence to FOL]:
Let L be a language. Let T be a set of equality-free sentences over L. Then if GroundTerms[L] - GroundTerms[MinLang[T]] is infinite, then over the language L, T is satisfiable under Herbrand Semantics if and only if T is satisfiable under First Order (Tarskian) semantics.

Proof.
We first state three well-known results from first-order logic model theory.

Lemma 1: Let T be a set of equality-free sentences. If T has a model under First Order Logic, then T has an infinite model under First Order Logic.

Lemma 2: Let T be a set of equality-free sentences. If T has a model under First Order Logic, then T has a model under First Order Logic that obeys the unique names assumption.

Lemma 3 [Lowenheim-Skolem Theorem]: Let T be a set of sentences in First Order Logic. If T has an infinite model under First Order Logic, then T has a model under First Order Logic of every infinite cardinality.

We now proceed to the main proof.

(<=) Assume T is satisfiable over the language L under Herbrand Semantics. Then there is a Herbrand interpretation that satisfies T. Then T is satisfiable over the language L under First Order Logic semantics.

(=>) Assume T is satisfiable over the language L under First Order Logic semantics.

Let U = GroundTerms[L] - GroundTerms[MinLang[T]]. Let M be a First Order interpretation that satisfies T. By Lemmas 1, 2 and 3, we can assume without loss of generality that M obeys the unique names assumption and that M has the same cardinality as U.

Let O_U = {M(u) | u ∈ U}, the set of objects mapped to by the ground terms U. Let O_N be the set of objects not referred to by any ground term in L. Let O = $O_U \cup O_N$. O has the same cardinality as U. Let f be a bijection that maps U to O.

Let M' be the First Order interpretation that is exactly as M except that for each τ ∈ U, M'(τ) = f(τ). M' is a First Order interpretation that satisfies T and furthermore obeys both the unique names assumption and the domain closure assumption. Therefore, T is satisfiable by a Herbrand interpretation. We conclude that over the language L, T is satisfiable under Herbrand Semantics.

Corollary 8.4 [Validity equivalence to FOL]:
Let L be a language. Let φ be an equality-free sentence over L. Then if GroundTerms[L] - GroundTerms[MinLang[φ]] is infinite, then over the language L, φ is valid under Herbrand Semantics if and only if φ is valid under First Order (Tarskian) semantics.

Proof.
Let φ be an equality-free sentence over L. Let T = {¬φ}. φ is valid according to Herbrand semantics if and only if T not satisfiable according to Herbrand semantics if and only if (by Theorem 8.3) T is not satisfiable according to First Order Logic semantics if and only if φ is valid according te First Order Logic semantics.

Corollary 8.5 [Infinitely many object constants]
Let L be a language with infinitely many object constants. Let T be a finite set of the equality-free sentences over L. Over the language L, T is satisfiable according to Herbrand semantics if and only if T is satisfiable according to First Order Logic semantics.

Proof.
Let T be a finite set of equality-free sentences over L. The set of all object constants that occur in T is finite. GroundTerms[L] - GroundTerms[MinLang[T]] is infinite because it contains an infinite set of object constants that do not occur in T. The of the corollary conclusion follows by Theorem 8.3.

Corollary 8.6 [Infinitely many object constants]
Let L be a language with infinitely many object constants. Let φ an equality-free sentences over L. Over the language L, φ is valid according to Herbrand semantics if and only if φ is valid according to First Order Logic semantics.

Proof.
Let φ be an equality-free sentence over L. Let T = {¬φ}. φ is valid according to Herbrand semantics if and only if T not satisfiable according to Herbrand semantics if and only if (by the previous corollary) T is not satisfiable according to First Order Logic semantics if and only if φ is valid according te First Order Logic semantics.

Corollary 8.7 [Simulation of FOL]
Let T be a (possibly infinite) set of the equality-free sentences over a language L. Let L^+ be the augmented language that consists of a new function constant in addition to the constants in L. Then T is satisfiable over L according to First Order Logic semantics if and only if T is satisfiable over L^+ according to Herbrand semantics.

7. Conclusion

References

[Abi09] Serge Abiteboul (2009). Datalog with negation [presentation]. Retrieved from http://abiteboul.com/TEACHING/DBCOURSE/deductive-neg-datalog.pdf

[AHV95] 1995. *Foundations of Databases: The Logical Level* (1st ed.). Serge Abiteboul, Richard Hull, and Victor Vianu (Eds.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.