The Herbrand Manifesto
Thinking Inside the Box

Michael Genesereth and Eric Kao
Computer Science Department
Stanford University

Keynote address delivered at RuleML 2015, Berlin, Germany

Abstract: The traditional semantics for relational logic (sometimes called *Tarskian* semantics) is based on the notion of interpretations of constants in terms of objects external to the logic. *Herbrand* semantics is an alternative that is based on truth assignments for ground sentences without reference to external objects. Herbrand semantics is simpler and more intuitive than Tarskian semantics; and, consequently, it is easier to teach and learn. Moreover, it is stronger than Tarskian semantics. 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 inferential completeness. However, there is no loss of inferential power - anything that can be deduced according to Tarskian semantics can also be deduced according to Herbrand semantics. Based on these results, we argue that there is value in using Herbrand semantics for relational logic in place of Tarskian semantics. It alleviates many of the current problems with relational logic and ultimately may foster a wider use of relational logic in human reasoning and computer applications.

Citation: M. R. Genesereth, E. Kao: The Herbrand Manifesto - Thinking Inside the Box, in Bassiliades N., Gottlob G., Sadri F., Paschke A., Roman D. (eds) Rule Technologies: Foundations, Tools, and Applications. RuleML 2015. Lecture Notes in Computer Science, vol 9202. Springer Cham, 2015.

---

1. Introduction

One of the main strengths of relational logic is that it provides us with a well-defined language for expressing complex information about objects and their relationships. We can write negations, disjunctions, implications, quantified sentences, and so forth. Logic also provides us with precise rules for deriving conclusions from sentences expressed within this language while avoiding the derivation of sentences that are *not* logical conclusions.

What makes it all work is that the language has a clearly defined semantics, which gives meaning to logical connectives and quantifiers. This allows us to know that we are using those connectives and quantifiers correctly; and it allows us to know that, in our reasoning, we are deriving conclusions that follow from our premises and avoiding those that do not.

The basis for almost all treatments of logical semantics is the notion of a model. A *model* is a mathematical structure that tells us which sentences are true and which are false. And this is the

basis for logical entailment. We say that a set of premises *logically entails* a conclusion if and only if every model that satisfies the premises also satisfies the conclusion. In other words, the conclusion must be true whenever the premises are true.

Tarskian semantics is the traditional approach to defining models in relational logic. In Tarskian semantics, a model consists of an arbitrary set of objects (called the universe of discourse) and an interpretation function that (1) maps object constants into elements of this set, (2) maps function constants into functions on this set, and (3) maps relation constants into relations on this set.

As an example, consider the model defined below. Our language in this case consists of the object constants a and b, the unary function constant f and the binary function constant r. Our universe of discourse consists of the natural numbers. Our interpretation maps a into 1 and b into 2; it maps f into a function on these numbers; and it maps r into a set of 2-tuples.

Vocabulary:	$\{a, b, f, r\}$
Universe:	$\{1, 2, 3, 4, ...\}$
Interpretation:	$i(a) = 1$
	$i(b) = 2$
	$i(f) = \{1 \rightarrow 2, 2 \rightarrow 4, ...\}$
	$i(r) = \{<1,2>, <2,3>, <3,4>, ...\}$

A model of this sort completely determines the truth or falsity of all sentences in the language. And it gives us a definition of logical entailment. Note, however, that there are unboundedly many interpretations for any language, and entailment is defined over all conceivable universes - finite, countably infinite, and beyond. It also requires an understanding of relations as set of tuples of objects.

Herbrand semantics is simpler. We start out with the notion of a Herbrand base, i.e. the set of ground atoms in our language. A model is simply a subset of the Herbrand base, viz. the elements that are deemed to be true.

As an example, consider the model defined below. Our language here consists of the object constants a and b, the unary relation constant p, and the binary relation constant q. The Herbrand base corresponding to this vocabulary has just six elements, as shown. Any subset of these elements is a model.

Vocabulary:	$\{a, b, p, q\}$
Herbrand Base:	$\{p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b)\}$
Herbrand Model:	$\{p(a), q(a,b)\}$

As with Tarskian semantics, a Herbrand model completely determines the truth or falsity of all sentences in the language, not just the ground atoms. And it gives us a definition of logical entailment. One important difference from Tarskian semantics is that Herbrand semantics is less open-ended. There is no external universe, only symbols and sentences in the language. In a sense, it is thinking inside the box.

In much of the literature, Herbrand semantics is treated (somewhat understandably) as a special case of Tarskian semantics - the case where we look at so-called Herbrand interpretations [4]. One downside of this is that Herbrand semantics has not been given as much theoretical attention as Tarskian semantics. In this paper, we turn things upside down, focussing on Herbrand semantics in

its own right instead of treating it as a special case of Tarskian semantics. The results are interesting. We no longer have many of the nice features of Tarskian semantics - compactness, inferential completeness, and semidecidability. On the other hand, there are some real benefits to doing things this way. Most importantly, Herbrand semantics is conceptually a lot simpler than Tarskian semantics; and, as a result, Herbrand semantics is easier to teach and learn. It has equivalent or greater inferential power. And more things are definable with Herbrand semantics than with Tarskian semantics. In the remainder of this paper, we demonstrate with examples some of the power and the properties of Herbrand semantics; each point is explained in detail in a companion paper [3].

2. Nuts and Bolts

Let's start with the basics. As mentioned earlier, a Herbrand base is the set of ground atoms in our language; and a model is an arbitrary subset of this set.

Given a model Δ, we say that a ground atom ϕ is true iff ϕ is in Δ. Here, h is the truth assignment corresponding to Δ. We use 1 to represent truth and 0 to represent falsity.

$$h(\phi) = 1 \text{ iff } \phi \in \Delta$$

The truth values of logical sentences are defined the same as with Tarskian semantics. A negation is true iff the negated sentence is false. A conjunction is true iff the conjuncts are both true. And so forth.

$$\begin{aligned}
&h(\neg\phi) = 1 && \text{iff } h(\phi) = 0\\
&h(\phi \wedge \psi) = 1 && \text{iff } h(\phi) = 1 \text{ and } h(\psi) = 1\\
&h(\phi \vee \psi) = 1 && \text{iff } h(\phi) = 1 \text{ or } h(\psi) = 1\\
&h(\phi \Rightarrow \psi) = 1 && \text{iff } h(\phi) = 0 \text{ or } h(\psi) = 1\\
&h(\phi \Leftrightarrow \psi) = 1 && \text{iff } h(\phi) = h(\psi)
\end{aligned}$$

Finally, a universally quantified sentence is true if and only all of the instances are true.

$$h(\forall x.\phi(x)) = 1 \text{ iff } h(\phi(\tau)) = 1 \text{ for every ground term } \tau$$

Despite many similarities, this definition does *not* produce the same results as Tarskian semantics. To illustrate this point, let's look at an example that illustrates the difference.

Here is a popular question from Stanford 's doctoral comprehensive exam. Suppose we are given a set Δ of sentences in the language of relational logic such that Δ logically entails $\phi(\tau)$ for every ground term τ in the language. Is it the case that Δ logically entails $\forall x.\phi(x)$?

The question is not difficult if one understands Tarskian semantics, but apparently not everyone does. The most common answer to this question is 'yes'; people seem to think that, if Δ logically entails every ground instance of ϕ, it must entail the universally quantified version. Of course, under Tarskian semantics, that answer is wrong. There can be some unnamed element of the universe of discourse for which the sentence is false.

However, the popularity of the "incorrect" answer suggests that perhaps our semantics does not capture our intuitions about logic. Maybe it should. The good news is that, with Herbrand

semantics, the answer to this question is 'yes'. (See the definition of satisfaction for universally quantified sentences above.)

As another example of a difference between Tarskian semantics and Herbrand semantics, consider the problem of axiomatizing Peano Arithmetic. As we know from Gödel, a finite axiomatization is not possible in relational logic with Tarskian semantics. Interestingly, with Herbrand semantics there is such a finite axiomatization.

Since there are infinitely many natural numbers, we need infinitely many terms. A common 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.

Unfortunately, even with this representation, axiomatizing Peano Arithmetic is a bit challenging. 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 necessary to characterize our relations in finite space.

Let's look at equality first. The axioms shown here define equality in terms of 0 and the *s* function. For all *x*, *equal*(*x*,*x*). For all *x*, 0 is not equal to *s*(*x*) and *s*(*x*) is not equal to 0. For all *x* and for all *y*, if *x* is not equal to *y*, then *s*(*x*) is not equal to *s*(*y*).

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

It is easy to see that these axioms completely characterize equality. By the first axiom, the equality 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 equal to 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 equality relation. Viewed the other way around, to see that two non-identical terms are not equal, 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 equal, and so the original terms are not equal.

Once we have the *equal* relation, we can define the other relations in our arithmetic. The following axioms define the plus relation in terms of 0, *s*, and *equal*. 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.

$$\forall y.times(0,y,0)$$

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

Under Herbrand semantics, this axiomatization is complete since we have defined truth for all ground atoms and thus all sentences. By contrast, Gödel's incompleteness theorem tells us that these axioms are not complete under Tarskian semantics. Note that the Incompleteness Theorem assumes semi-decidability of logical entailment. Relational logic with Tarskian semantics is semi-decidable; with Herbrand semantics, it is not semi-decidable, as we shall see shortly. So, there is no contradiction here.

3. No Free Lunch

Unfortunately, the additional expressive power of Herbrand semantics comes with a price. We lose some nice features that we have with Tarskian semantics.

First of all, there is compactness. A logic is *compact* if and only if every unsatisfiable set of sentences has a finite subset that is unsatisfiable.

Relational logic with Tarskian semantics turns out to be compact. The upshot is that it is possible to demonstrate unsatisfiability in finite space; alternatively, all proofs are finite.

By contrast, relational logic with Herbrand semantics is not compact - there are infinite sets of sentences that are unsatisfiable while every finite subset is satisfiable. Consider the set of sentences shown here. It is clearly unsatisfiable under Herbrand semantics; but, if we remove any one sentence, it becomes satisfiable.

$$\{p(0), p(s(0)), p(s(s(0))), \ldots, \exists x.\neg p(x)\}$$

The upshot is that relational logic with Herbrand semantics is not compact. Fortunately, this does not cause any practical difficulties, since in all cases of practical interest we are working with finite sets of premises.

More disturbing is that there is no complete proof procedure for relational logic with Herbrand semantics. Gödel's incompleteness theorem tells us that the set of all true sentences of Peano Arithmetic is not computably enumerable. Our axiomatization is complete using Herbrand semantics. If Herbrand entailment were semi-decidable, the set of all true sentences would be enumerable. Consequently, there is no complete (semi-decidable) proof procedure for relational logic with Herbrand semantics.

However, this is not as bad as it seems. It turns out that everything that is true under Tarskian semantics is also true under Herbrand semantics, so we can use the same rules of inference. The upshot here is that we lose nothing by switching to Herbrand semantics. In fact, we can add some additional rules of inference. It is not that relational logic with Herbrand semantics is weaker. In fact, it is stronger. There are more things that are true. We cannot prove them all, but we can prove everything we could prove before.

Some may be disturbed by the fact that Herbrand entailment is not semi-decidable. However, Tarskian semantics is not perfect either. Although it is semi-decidable, it is not decidable; a proof procedure might still run forever if a proposed conclusion does not follow from a set of premises.

There is one other limitation that some may find even more disturbing. Since Herbrand semantics

is effectively limited to countable universes, it would appear that we can no longer use the logic to axiomatize uncountable sets, such as the real numbers. This is true. However, it is not that much of a limit. For one, most CS applications involve finite or countably infinite domains. Remember that there are at most countably many floating point numbers.

Arguably one might want to axiomatize the reals even without converting to floating point numbers. However, even here, Tarskian semantics is limited because of the Löwenheim-Skolem theorem. This theorem states that, under Tarskian semantics, if a set of sentences has an infinite model of any cardinality, then it has a countable model. In particular, any theory of the real numbers has a countable model - everything one can say about the real numbers in relational logic is also true of some countable model.

4. Curiouser and Curiouser

The power of Herbrand semantics is, in large part, due to the implicit property of domain closure - there are no objects in the universe except for ground terms. This allows us to give complete definitions to things that cannot be completely defined with Tarskian semantics. We have already seen Peano Arithmetic. It turns out that, under Herbrand semantics, we can also define some other useful concepts that are not definable with Tarskian semantics, and we can do so without resort to more complex logical mechanisms, such as negation as failure.

Let's look at transitive closure first [5]. Let's say that we have a binary relation p and we want to axiomatize its transitive closure q. The typical approach in relational logic would be to write the definition shown here.

$$\forall x.\forall z.(q(x,z) \Leftrightarrow p(x,z) \vee \exists y.(p(x,y) \wedge q(y,z)))$$

It is easy to see that q contains the transitive closure of p. The problem is that, in general, it can contain additional elements as well, corresponding to various non-standard models. For example, the universe of discourse might contain an object that does not have any p relationships at all. However, if we link all other objects to this object via q, this satisfies our definition. The upshot is that we have a model of our sentence that is a proper superset of the transitive closure of p. Not good.

By contrast, we *can* define the transitive closure of a relation in relational logic with Herbrand semantics. It is not as simple or intuitive as the definition above, but it is theoretically possible. The trick is to exploit the enumerability of the Herbrand universe. Suppose we have the object constant 0, an arbitrary unary relation constant s; and suppose our job is to define q as the transitive closure of p.

We start by defining a helper relation qh as shown below. The basic idea here is that $qh(x,z,n)$ is the (partial) transitive closure in which no intermediate variable is bigger than n. Once we have qh, we can easily define q in terms of qh. q is true of two elements if and only if there is a level at which qh becomes true of those elements.

$$qh(x,z,0) \Leftrightarrow p(x,z) \vee p(x,0) \wedge p(0,z)$$
$$qh(x,z,s(n)) \Leftrightarrow qh(x,z,n) \vee (qh(x,s(n),n) \wedge qh(s(n),z,n))$$

It is easy to see that q is exactly the transitive closure of p. The only disadvantage of this axiomatization is that we need the helper relation qh. But that causes no significant problems.

$$\forall x.\forall z.(q(x,z) \Leftrightarrow \exists n.qh(x,z,n))$$

But wait. There's more! As we know, it is possible to encode some relations in rule systems that cannot be encoded in relational logic with Tarskian semantics. Rule systems get this power from the use of negation as failure to minimize those relations. The cool thing is that, even without any form of negation as failure, it is possible to encode those relations in relational logic with Herbrand semantics. Moreover, various minimization policies can result from different axiomatizations.

Consider a logic program like the one shown here. There are two rules defining p and one rule defining q.

$$p(0,1)$$
$$p(X,Y) \text{ :- } q(X,0), p(Y,Z)$$
$$q(X,Y) \text{ :- } p(X,0), q(Y,Z)$$

The first step of our conversion is to normalize the program so that the head of every rule consists of distinct variables. This is easy to do using equality (defined as we did earlier in Peano Arithmetic). We then combine the bodies of the resulting rules using the disjunction operator.

$$p(X,Y) \text{ :- } X=0, Y=1$$
$$p(X,Y) \text{ :- } q(X,0), p(Y,Z)$$
$$q(X,Y) \text{ :- } p(X,0), q(Y,Z)$$

Next we transform the normalized program as follows. The first two sentences here are the result of transforming the original axioms using our helper relations. The other axioms are the additional axioms defining the helper relations and defining the target relations in terms of these helper relations. For each rule defining an n-ary relation in the normalized program, we define an $(n+1)$-ary auxiliary relation as shown here. ph is true of x and y and 0 iff $x=0$ and $y=1$. ph is true of x and y and s(n) iff there is a z that the definition holds of elements on step n. Finally, if ph is true is true of x and y on step n, then it is also true on step s(n). Ditto for qh.

$$ph(x,y,0) \Leftrightarrow x=0 \wedge y=1$$
$$ph(x,y,s(n)) \Leftrightarrow \exists z.(qh(x,0,n) \wedge ph(y,z,n))$$
$$ph(x,y,s(n)) \Leftarrow ph(x,y,n)$$

$$\sim qh(x,y,0)$$
$$qh(x,y,s(n)) \Leftrightarrow \exists z.(ph(x,0,n) \wedge qh(y,z,n))$$
$$qh(x,y,s(n)) \Leftarrow qh(x,y,n)$$

Finally, we define p and q in terms of ph and qh, as we did in the transitive closure example.

$$p(x,y) \Leftrightarrow \exists n.ph(x,y,n)$$
$$q(x,y) \Leftrightarrow \exists n.qh(x,y,n)$$

Now, the interesting thing is that it turns out that we can do this transformation in general (for arbitrary logic programs so long as they are safe and stratified). Let P be an arbitrary safe, stratified program over the relations R. Let M be the unique minimal model of P under stratified semantics [6, 1]. Then, this transformation has a unique model M' under Herbrand semantics such that M' =

M over *R*. Voila - minimization without negation as failure.

One consequence of this result is that we can treat :- as syntactic sugar for definitions requiring minimization. There is no need for a different logic. Which does not mean that :- is useless. In fact, the oddity of our definitions makes clear the value of :- in expressing definitions intuitively.

I think there is also another, more subtle benefit of this theorem. One possible practical consequence of this work concerns the relationship between rule systems and ordinary logic. Rules and ordinary logic are often seen as alternatives. Herbrand semantics has the potential to bring these two fields closer together in a fruitful way. This upshot could be a possible re-prioritization of research in these two areas.

The power and beauty of rule systems is their suitability for writing complete definitions. We start with some completely specified base relations and define other relations in terms of these base relations, working our way up the definitional hierarchy. At every point in time we have a complete model of the world.

Unfortunately, complete theories are not always possible; and in such situations we need to provide for expressing incomplete information. In an attempt to deal with incomplete information, researchers have proposed various extensions to rule systems, e.g. negations, disjunctions, and existentials in the heads of rules, unstratified rules systems, and so forth. Unfortunately, extensions like these mar the beauty of rule systems and ruin their computational properties.

The alternative is to switch to relational logic in such situations. Unfortunately, relational logic with Tarskian semantics is more complex and fails to provide minimization or negation as failure.

Our argument is that Herbrand semantics for ordinary logic gives us an ideal middle ground between rules and relational logic, allowing us to combine rules with relational logic without losing the benefits that each brings to the table. We can use rules for definitions and ordinary logical operators for constraints. The two can co-exist. In fact, as I have suggested, we can even formalize negation as failure and various minimization policies within relational logic, so long as we are using Herbrand semantics.

Now, I do not know whether this is practically possible or not. However, I think it is an idea worthy of study, considering the lack of a unifying semantics today.

5. Conclusion

In conclusion, let's return to the theme of simplicity. The fact is that Tarskian semantics is more difficult to understand than Herbrand semantics.

First of all, in Tarskian semantics, there are unboundedly many interpretations for any language, and entailment is defined over all conceivable universes - finite, countably infinite, and beyond.

Second, Tarskian semantics requires an understanding of relations as sets of tuples, which is a novel concept for many students. In Herbrand semantics, everything is defined in terms of sentences, which are more concrete and which students already understand.

Finally, in Tarskian semantics, there is also greater complexity in the definition of satisfaction. Here is the definition in Tarskian semantics. "An interpretation *i* and a variable assignment *s* *satisfy* a universally quantified sentence if and only if *i* and *s* satisfy the scope of the sentence for every version of the variable assignment. A *version* $s[v \rightarrow x]$ of a variable assignment *s* is a variable

assignment that assigns *v* the value *x* and agrees with *s* on all other variables." That's a mouthful. Now, compare the definition in Herbrand semantics. "A model *satisfies* a universally quantified sentence if an only if it satisfies every instance." That's it. Shorter and easier to understand.

These ideas confuse students. As a result, they feel insecure and are all too often turned off on logic. This is sad because we should be teaching more logic and not turning people away. In Greek times, logic was one of the three basic disciplines that students learned. Today, it is taught in only a very small percentage of schools. Instead, we are taught geometry. We are taught how to bisect angles in high school, but we are not taught logic. Only few of us need to bisect angles in our daily lives, but many of us use logic in our professional lives and in our private live, e.g. to understand political arguments, court cases, and so forth. Perhaps, if we could make logic more useful and easier to teach, this could change.

To test the value of Herbrand semantics in this regard, we recently switched Stanford's introductory logic course from Tarskian semantics to Herbrand semantics. The results have been gratifying. Students get semantics right away. They do better on quizzes. And there are fewer complaints about feeling lost. It is clear that many students come away from the course feeling empowered and intent on using logic. More so than before anyway.

The logic course is now available as a MOOC and an associated book [2]. It was one of the first MOOCs taught at Stanford. We teach it each year in the Fall. Typical enrollment is now almost 100,000 students per session. To date, more than 500,000 students have enrolled in all. As is typical with MOOCs, only a fraction of these students finish. Even so, more students have seen this than have graduated from Stanford's math program in its entire history.

In a previous keynote address at RuleML, we talked about Logical Spreadsheets. On that occasion, we mentioned the goal of popularizing logic and suggested that what we need is a way to make logic more accessible and we need tools that make it clear that logic is useful, not just as an intellectual tool but as a practical technology as well.

This time, we have talked about a way to make logic more accessible - a way to teach people enough logic so that they can use logical spreadsheets and other logic-based technologies. If logic is easy to learn, our hope is that we can make it more popular. Not just to promote our interests as researchers but also to benefit society with the fruits of our research.

References

1. Chandra, A.K., Harel, D.: Horn clause queries and generalizations. The Journal of Logic Programming 2(1), 1-15 (1985),

2. Genesereth, M., Kao, E.J.Y.: Introduction to Logic, Morgan-Claypool (2013).

3. Genesereth, M., Kao, E.J.Y.: Herbrand semantics (2015).

4. Goubault-Larrecq, J., Mackie, I.: Proof theory and automated deduction, Applied Logic Series, vol. 6. Springer Netherlands, 1 edn. (1997).

5. Keller, U.: Some remarks on the definability of transitive closure in first-order logic and datalog. Tech. rep., Digital Enterprise Research Institute (DERI) (2004).

6. Van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the ACM (JACM) 23(4), 733-742 (1976).