First Order Logic - Semantics

Piyush Patil

September 28, 2017

Any language consists of a set of valid sentences, subject to certain logical and linguistic rules and constraints. The two main ideas governing the validity of sentences and statements are *syntax* and *semantics*. Syntax refers to the gramatically structure of different elements of a sentence, and valid arrangements of these element types, whereas semantics refers to the actual meaning of these elements and how these different meanings of elements consolidate through a syntactical structure to form a sentence with a cohesive meaning. The previous chapter covered the syntax of first-order logic, and the symbols, valid sequences of symbols, and structure of these sequences therein. In this chapter, we'll explore the actual meaning of the elements of the predicate language (the first-order analog to the propositional language).

1 Formulas and Structures

If, when determining the semantics of first-order logic, our goal is to identify meanings of the elements of our logical language in such a way as to form valid, and meaningful sentences, we'll start by laying down an interpretation of our logical system. Concretely, this means we'll associate every non-constant logical symbol with some object, and specify a *domain of discourse* - essentially just a set of some objects - over which our variables will range. Intuitively, formulas in first-order logic are logical statements on these objects. Each term is assigned an object that it represents, while sentences are assigned truth values.

First, let's restrict ourselves to a possible subset of the full predicate language, since using the full predicate language might be overkill. If we have some set $\mathcal{A}$ of constant, predicate, and function symbols from $\mathcal{L}$, then a formula $\phi \in \mathcal{L}$ is said to be an $\mathcal{L}_{\mathcal{A}}$-formula if every constant, predicate, or function symbol apppearing in ϕ is in $\mathcal{A}$. Next, to actually formally specify an interpretation of the kind we want above, we'll define a *structure*, which is essentially a specified universe of abstract objects we want to make logical statements about, and a function associating logical symbols from our chosen sub-language $\mathcal{A}$ to logical operators and statements on those objects, interpreting the symbols in the most natural and canonical way possible.

(Definition) Structure: *Given a set $\mathcal{A}$ of constant, predicate, function symbols, an **$\mathcal{L}_{\mathcal{A}}$-structure**, which we often refer to as a **structure over $\mathcal{L}$**, is a pair (M, I), where M is a set and I is a function over $\mathcal{A}$, such that*

1. *For every constant symbol $c_i \in \mathcal{A}$, $I(c_i) \in M$.*
2. *For every function symbol $F_i \in \mathcal{A}$, $I(F_i)$ is a function from $M^{\pi(F_i)}$ to M.*
3. *For every predicate symbol $P_i \in \mathcal{A}$, $I(P_i) \subseteq M^{\pi(P_i)}$.*

The idea here is that M is our universe of symbols and I is the mapping we use to associate terms from our sub-language $\mathcal{A}$ with operations and relations on our objects. To construct this association in a natural way, we require that every constant symbol in $\mathcal{A}$ have an associated constant symbol in M, that every function symbol behave as an n-ary function over M (where n is the function's arity), and every predicate symbols behave as a boolean operator. Indeed, we often refer to M as the structure's *universe* and I as the structure's *interpretation mapping.*

The first two conditions are straightforward to "implement" in the definition of a structure above, but the third is a bit more nebulous; what we do in the third condition is use I to map a predicate symbol, which we'd like the interpret as an n-ary boolean function (where n is the predicate's arity), to the set of n-tuples of objects which make the predicate true. After all, a predicate is a statement depending on n arguments, and has a well-defined truth value; our interpretation, then, for predicate symbol P_i, is

$$I(P_i) = \{(m_1, \cdots, m_n) \text{ s.t. } m_j \in M \textit{ for } 1 \leq j \leq n \textit{ and } P_i(m_1, \cdots, m_n) = \text{ T}\}$$

2 The Satisfaction Relation

Now that we've rigorously cemented a way to imbue the terms of our language with a consistent, meaningful interpretation by casting constant, function, and predicate terms as different kinds of static operations and relations on some set of objects, we next want to give the variables in formulas specific meaning and domains of discourse ranging over the objects in question.

This will in turn allow us to interpret formulas as logical statements with truth values. Hence, when we extend satisfiability over the propositional language to the predicate langauge - or more accurately, structures over the predicate langauge - we'll first need to specify not only the interpretation of the language (which is handled by the structure) as well as the assigning of variables to objects and domains of discourse over which they range.

> **(Definition) Assignment**: *Let $\mathcal{M}$ be a structure over $\mathcal{L}$. An $\mathcal{M}$-assignment is a mapping $\nu : \{x_i, i \in \mathbb{N}\} \to M$.*

If we fix a structure and an assignment on that structure, then we'll have associated constants and variables with objects from our universe, and function and predicate symbols with natural analogous operations and relations on and between objects from our universe, all towards building a meaningful interpretation of our first order sub-language. Finally, we can recursively define the truth values of formulas over our sub-language in terms of our interpretation.

> **(Definition) Extended assignment mapping**: *Let $\mathcal{A}$ be a set of constant, predicate, and function symbols, $\mathcal{M} = (M, I)$ be an $\mathcal{L}_{\mathcal{A}}$-structure, and ν be an $\mathcal{M}$-assignment. We define the* ***extended assignment function*** *by extending ν to a $\overline{\nu}$ given by*
>
> $$\overline{\nu} : \{\tau \text{ s.t. } \tau \textit{ is an } \mathcal{L}_{\mathcal{A}}\textit{-term}\} \to M$$
>
> *We define the behavior of $\overline{\nu}$ inductively as follows:*
>
> $$\textit{for every } \mathcal{L}_{\mathcal{A}}\textit{-terms } \tau : \overline{\nu}(\tau) = \begin{cases} \nu(x_i), & \textit{if } \tau = \langle x_i \rangle \textit{ for some } i \\ I(c_i), & \textit{if } \tau = \langle c_i \rangle \textit{ for some } i \\ I(F_i)\left(\overline{\nu}(\tau_1), \cdots, \overline{\nu}(\tau_{\pi(F_i)})\right), & \textit{if } \tau = F_i(\tau_1, \cdots, \tau_{\pi(T_i)}) \textit{ for some } i \end{cases}$$

It follows by unique readability that this is a well-defined extension. We have now concluded laying the groundwork for the semantics of first-order logic. Once choose a sub-language $\mathcal{A}$, specify a structure on $\mathcal{A}$ (which associates our abstract language of symbols with a concrete universe of objects about which to make statements, an a function for mapping terms to their interpreted meanings under our interpretation), and choose an assignment of variables, we've fully specified an interpretation of our language and its built-in logical rules in terms of our chosen domain of objects. Constants represent static, unchanging objects we want to refer to, variables are placeholders used to range over sets of objects using quantifiers. Functions allow us to combine objects into a new object, extending our universe and allowing us to add layers of abstraction to the objects. Predicates allow us to make arbitrary logical statements about objects, whereas the equality symbol, the implication symbol, and the negation symbol all give us building blocks with which to build logical statements, referring to the respective ideas of equality between objects, implication between formulas, and the negation of a formula. We are now in a position to start using structures and (extended) assignments on that structure to map formulas and sentences to truth values, giving us an analogous concept of satisfiability from predicate logic.

> **(Definition) Satisfaction relation**: *Let $\mathcal{M} = (M, I)$ be a structure over $\mathcal{L}$ and ν be an $\mathcal{M}$-assignment. Define the satisfaction relation, denoted $(\mathcal{M}, \nu) \vDash \phi$ for formula $\phi \in \mathcal{L}$, to hold under with the following conditions.*
>
> 1. *Atomic case: Suppose ϕ is atomic. Then we have one of the following cases.*
>   (a) *Suppose $\phi = P_i(\tau_1, \cdots, \tau_{\pi(P_i)})$ for some i and terms $\tau_j, 1 \le j \le \pi(P_i)$. Then $(\mathcal{M}, \nu) \vDash \phi$ if we have*
>
> $$\left(\overline{\nu}(\tau_1), \cdots, \overline{\nu}(\tau_{\pi(P_i)})\right) \in I(P_i)$$
>
>   (b) *Suppose $\phi = (\sigma \hat{=} \tau)$ for terms σ, τ. Then $(\mathcal{M}, \nu) \vDash \phi$ if we have*
>
> $$\overline{\nu}(\sigma) = \overline{\nu}(\tau)$$
>
> 2. *Non-atomic case: Suppose ϕ is not atomic. Then we have one of the following cases.*
>   (a) *Suppose $\phi = (\neg\psi)$ for formula ψ. Then $(\mathcal{M}, \nu) \vDash \phi$ if we have*
>
> $$(\mathcal{M}, \nu) \nvDash \psi$$
>
>   (b) *Suppose $\phi = (\psi_1 \to \psi_2)$ for formulas ψ_1, ψ_2. Then $(\mathcal{M}, \nu) \vDash \phi$ if we have either*
>
> $$(\mathcal{M}, \nu) \nvDash \psi_1 \textit{ or } (\mathcal{M}, \nu) \vDash \psi_2$$
>
>   (c) *Suppose $\phi = (\forall x_i \psi)$ for variable x_i and formula ψ. Then $(\mathcal{M}, \nu) \vDash \phi$ if we have*
>
> $$\textit{for all } \mathcal{M}\textit{-assignments } \mu \textit{ that agree with } \nu \textit{ on the free variables of } \phi \ : (\mathcal{M}, \mu) \vDash \psi$$

Above, when we say that an $\mathcal{M}$-assignment μ agrees with an $\mathcal{M}$-assignment ν on the free variables of a formula ϕ, we simply mean that for every free variable x_i appearing in ϕ, $\mu(x_i) = \nu(x_i)$. This is simply a straightforward way of checking when two assignments are in fact equal, accounting for only free variables since those are the only variables we care about assigning (after all, recall that bound variables are associated to a particular domain of discourse anyways, and so are in a sense already assigned). It's easy to prove that if μ and ν agree on the free variables of ϕ, then $\overline{mu}(\phi) = \overline{nu}(\phi)$.

The satisfaction relation above is the crux of our extending the concept of satisfiability in the propositional language to first-order logic. It explicitly sets up the motivation we had in mind when we defined structures and assignments, capturing our intuition for predicate, function, negation, and implication symbols, as well as the universal logical quantifier $\forall$. As expected, an assignment on a structure satisfies a formula consisting of a predicate whenever the assignment "makes the predicate evaluate to true", i.e. when it's contained in $I(P_i)$, which was our interpretation for the true values of a predicate P_i. Further, equality formulas are satisfied only for terms that are equivalent under an assignment, negation formulas are satisfied precisely when the internal formula is not satisfied, and implication formulas are satisfied whenever the antecedent is not satisfied (equivalent to $\text{F} \to \text{anything}$) or when the consequent is satisfied (equivalent to $\text{anything} \to \text{T}$). Finally, the universal quantifier is satisfied whenever every equivalent assignment satisfies the sub-formula; because x_i is bound in ϕ, the values of all the μ's on x_i respectively range over every value in M, cementing our motivation for the $\forall$ quantifier.

3 Substitution and the Satisfaction Relation