First Order Logic - Syntax

Piyush Patil

September 20, 2017

Recall that the propositional language we defined in the last chapter consisted of propositional symbols, which were meant to serve as variables with truth values, such as the deductive propositions to which we are accustomed in mathematics, and logical connectives, which allow us ways of combining propositions together into compound propositions. The first-order logical language, also known as *predicate logic*, augments $\mathcal{L}_0$ by including *logical quantifiers*, which provide a means of referring to the elements of a propositional structure, and *predicates*, which are generalizations of propositions that can accept arguments and specify logical relationships between *variables* (indeed, a proposition, the defining concept of propositional logic, is a predicate with zero arguments).

As before, our language will consist of finite sequences of symbols, with certain pre-defined notions of what allowable symbols and ways of concatenating them are. Specifically, the symbols are the following:

1. *Logical symbols*: $($ $)$ $\neg$ $\rightarrow$ $\forall$
2. *Equality symbol*: $\hat{=}$
3. *Variable symbols*: $x_i,$ for $i \in \mathbb{N}$
4. *Constant symbols*: $c_i,$ for $i \in \mathbb{N}$
5. *Function symbols*: $F_i,$ for $i \in \mathbb{N}$
6. *Predicate symbols*: $P_i,$ for $i \in \mathbb{N}$

To specify the *arity* of a function or predicate symbol is the number of arguments it takes, and we fix a special function

$$\pi : \{F_i, i \in \mathbb{N}\} \cup \{P_i, i \in \mathbb{N}\} \rightarrow \mathbb{N}$$

which maps each function and predicate symbol to its arity. Having set up the necessary framework of symbols, let's move on to describing the terms of the language itself, and the production rules for defining sequences of symbols that characterize the language.

1 Terms

As before, we denote (finite) sequences of symbols as $\langle s_1, \cdots, s_n \rangle$ for for symbols s_i over $1 \leq i \leq n$, with the sum of two symbols denoting their concatenation. Let's jump right in and define the terms of the predicate language.

> **(Definition) Terms**: *The set of* ***terms*** *is the smallest set T of finite sequences of symbols satisfying the following properties.*
>
> 1. $\forall i \in \mathbb{N} : \langle x_i \rangle \in T$
> 2. $\forall i \in \mathbb{N} : \langle c_i \rangle \in T$
> 3. $\forall i \in \mathbb{N} :$ *if* $\tau_1, \cdots, \tau_n \in T$ *then*
>
> $$\langle F_i \rangle + \langle (\rangle + \tau_1 + \cdots + \tau_n + \langle) \rangle \in T$$
>
> *where* $n = \pi(F_i)$.

As convenient shorthand, we'll typically refer to unit-length terms (e.g. $\langle x_i \rangle$) by omitting the brackets (e.g. by referring to x_i as a term), and we'll informally use $F_i(\tau_1, \cdots, \tau_n)$ to refer to terms of the third form above. Similar to the propositional language, we can show that terms are "generated" by starting at unit-length sequences and applying our production rules. Thus, whereas previously we started at symbols and put then together to generate compound propositions using implications and negations, terms are generated by starting at variables and constants and hierarchically applying functions to them.

(Theorem) Unique readability: *Every term in T is either, for some $i \in \mathbb{N}$, a variable x_i, a constant c_i, or of the form $F_i(\tau_1, \cdots, \tau_{\pi(F_i)})$ for $\tau_j \in T, 1 \leq j \leq \pi(F_i)$. Further, exactly one of these conditions is true.*
Proof: TODO same as unique readability proof for $\mathcal{L}_0$ $\square$

Because of the structure of our terms and the way they are generated hierarchically by continuing to "wrap" inside functions, we have another analogous theorem on the sequence structure of terms.

(Theorem) No proper initial segments: *For any $\tau \in T$, no proper initial segment of τ is in T.*
Proof: TODO same as proof for $\mathcal{L}_0$ $\square$

Having defined a base "field" of terms, let's turn to the actual formulas, on which truth values will be defined, which are created using these terms.

2 Formulas

Let's define the set of formulas, and then motivate this definition.

(Definition) Formulas: *The let of **formulas**, denoted $\mathcal{L}$, is defined as the smallest set that satisfies the following conditions.*

1. *Predicate formulas: For any predicate symbol P_i and terms $\tau_1, \cdots, \tau_{\pi(P_i)}$, $P_i(\tau_1, \cdots, \tau_{\pi(P_i)}) \in \mathcal{L}$*
2. *Closed under equality: For any terms τ_1, τ_2, $(\tau_1 \hat{=} \tau_2) \in \mathcal{L}$*
3. *Closed under negation: For any formula $\phi \in \mathcal{L}$, $(\neg\phi) \in \mathcal{L}$*
4. *Closed under implication: For any formulas $\phi_1, \phi_2 \in \mathcal{L}$, $(\phi_1 \to \phi_2) \in \mathcal{L}$*
5. *Closed under quantifier application: For any formula $\phi \in \mathcal{L}$ and variable x_i, $(\forall x_i \phi) \in \mathcal{L}$*

*Predicate symbols and equality formulas are called **atomic formulas**.*

The central motivation behind predicate logic, why we defined the symbols we did and why we defined the above production rules for terms and formulas, is to construct a language under which we can express the truth and falsity of logical statements. Of course, this was precisely the motivation behind propositional logic, but propositional logic has certain fundamental shortcomings that limit its expressevity.

In propositional logic, our goal was to study the internal structure of propositions, in terms of their definition using logical connectives and sub-propositions. The propositional constants A_n represent atomic propositions which can't be broken down further through logical connectives, and based on the truth values of the propositional constants composing a compound proposition, we can ascertain the truth value of the entire proposition.

However, propositional logic is limited to the finite; we can only combine propositions in finite ways, and atomic propositions can only describe finite truth variables. With the addition of objects such as predicates, which act on variables that span entire, potentially infinite domains, and quantifiers that allow us to speak on the nature of the entirety of these domains at once, predicate logic extends propositional logic to the handle statements dealing with the infinite. As such, we define terms to be the constituents of our formulas; terms don't have truth values themselves, but rather are simply placeholders for expressions. Variables represent fluid terms that refer to an entire domain, whereas constants are immutable referents. Functions exist as means of combining terms into more complex terms, enabling the predicate language to make statements on more complex statements then simply constants and variables, which form the base level. The equality operator exists to allow our logical system to tell when two terms refer to the same underlying expression. Formulas, then, are the sentences of our language, which do have their own truth values. As before, we can take sub-formulas and combine them to form higher-order, compound formulas using logical operators such as negation and implication. In accordance with our motivation on extending propositional logic, we also include (1) logical quantifiers, which allow us to make truth statements not on just propositional constants but on entire domains, captured by variables, and (2) predicates, which we can essentially conceptualize as boolean functions that act on terms by assigning them one cummulative truth value.

3 Subformulas

As before, we prove readability results on the predicate language, confirming that indeed the language consists of expressions that are recursively generated using our production rules, starting at the bottom with our terms (which themselves have a recursive readable structure).

(Theorem) Unique Readability: *Let ϕ be a formula. Then exactly one of the following conditions is true.*

1. $\phi = P_i(\tau_1, \cdots, \tau_{\pi(P_i)})$ *for some* $i \in \mathbb{N}$ *and* $\tau_j \in T$.
2. $\phi = (\tau_1 \hat{=} \tau_2)$ *for some* $\tau_1, \tau_2 \in T$.
3. $\phi = (\neg\psi)$ *for some* $\psi \in \mathcal{L}$.
4. $\phi = (\psi_1 \to \psi_2)$ *for some* $\psi_1, \psi_2 \in \mathcal{L}$.
5. $\phi = (\forall x_i \psi)$ *for some variable* $x_i \in T$ *and* $\psi \in \mathcal{L}$.

Further, in any of these cases the expression is unique.

Proof: TODO same □

As before, the proof of the uniqueness claim in the readability theorem hinges on the analogous lemma that no proper initial segment of a formula is also a formula. Let's now move on to consider ideas of scope and different kinds of variables, characterized by their scope.

4 Free and Bound Variables

First, consider that the use of logical quantifiers in formula is meant to, in a sense, define self-contained formulas of their own. Thus, formulas which contain quantifiers inside them have a notion of "scope" to them - the quantifiers make statements on sub-formulas that are embedded in the entire formula, which is wrapped around and makes a statement on the sub-formula. The following theorem proves this formally.

(Theorem) Quantifiers appear in subformulas: *Let ϕ be a formula of the form*

$$\phi = s + \langle \forall, x_i \rangle + t$$

for finite sequences of symbols s, t. Then there exist finite sequences of symbols s', t', where $s = s' + \langle (\rangle$, and formula ψ for which

$$\phi = s' + \psi + t'$$

Moreover, ψ is unique.

Proof: This theorem is stating exactly what the intuition above was directed towards - formulas in which quantifiers appear in the middle can be framed as containing sub-formulas inside a scope (this is why $s = s' + \langle (\rangle$) around which the formula is wrapped.

Uniqueness is easy to prove, since if there were distinct formulas ψ_1, ψ_2 for which $\phi = s' + \psi_1 + t'_1 = s' + \psi_2 + t'_2$ then one would be a proper initial segment of the other, which is impossible (note that we don't need to consider separate s'_1 and s'_2 since s' is defined in terms of the fixed s).

To prove the existence of ψ, we proceed by induction on the length of ϕ. The base case, of formulas of unit length, is trivially true as no such formulas exist. Next, take as the inductive hypothesis that the theorem is true for formulas up to length n. By readability, we can decompose any formula ϕ of the required form and with length $n+1$ into five cases. In the first two cases, where ϕ is either a predicate formula on terms or the equality between two terms, ϕ is atomic and hence composed only of terms and not subformulas, which means it has no occurrence of the string $\langle \forall, x_i \rangle$. Thus, let's consider the remaining three cases.

First, suppose that either $\phi = (\neg\theta)$ (for formula θ) or $\phi = (\theta_1 \to \theta_2)$ (for formulas θ_1, θ_2). Then, seeing as none of the symbols in ϕ that aren't in θ are quantifiers or variables, it follows that since ϕ contains the sequence $\langle \forall, x_i \rangle$, in the first case θ does too, and in the second case, one of θ_1 or θ_2 does too (clearly, since $\langle \forall, x_i \rangle$ doesn't contain $\to$, it can't be split between θ_1 and θ_2). In either case, since any subformula of ϕ has a shorter length than ϕ, by the inductive hypothesis we can find a unique ψ as required, which is embedded in the subformula and hence in ϕ.

Lastly, suppose instead that $\phi = (\forall x_j \theta)$. If we have $j = i$, then the ψ we seek is precisely ϕ, with s', t' being empty sequences. Otherwise, if $j \neq i$, then the occurrence of $\langle \forall, x_i \rangle$ must be in θ, and as above we apply the inductive hypothesis. This considers all possible cases, and so the proof is complete. □

Recall that the whole point of introducing logical quantifiers was to allow us to make truth statements about entire domains of discourse, with variables serving the role of "running" over the domain, allowing the corresponding statement to refer to the variable. Thus, it makes sense that if we defined our language properly to encode this intuition, occurrences of logical quantifiers in formulas must be either in the "outermost" formula, or embedded in a subformula. This lends a recursive structure to the formulas of $\mathcal{L}$, wherein every occurrence of a logical quantifier defines a subformula one level of embedding deeper than the last quantifier. This is precisely what we referred to earlier as the intuitive notion of "scope", which was meant to capture the idea of statements that contain sub-statements with their own local domains of discourse, which then

becomes an object in the containing statement. Let's now formalize all this into a concrete definition. For notational convenience, we'll often say that "$\forall x_i$ occurs in a formula" when really we mean $\langle \forall, x_i \rangle$ occurs in the formula.

(Definition) Scope: *The* ***scope*** *of an occurrence of $\forall x_i$, for some i, in a formula $\phi = \langle a_1, \cdots, a_n \rangle$ is the unique interval $[j_1, j_2]$ defined by the properties*

1. *$a_{j_1+1} = \forall$ and $a_{j_1+2} = x_i$ (and, of course, $a_{j_1} = ($ by readability).*
2. *$\langle a_{j_1}, \cdots, a_{j_2} \rangle$ is a formula.*

In other words, occurrence of logical quantifiers have as their scope precisely the most immediate containing subformula (which we proved must exist). Notice that although variables can be used alongside logical quantifiers, as placeholders local to the scope of the quantifier for use in making a statement over the domain of discourse referred to within that scope, they can also appear in other contexts and for other purposes, such as, for instance, in functions or predicates. It's helpful to distinguish between these two uses of variables, since in the former case we're using them in a purely local setting, so that the variable really only has meaning with respect to its surrounding scope and doesn't, in contrast with the variables appearing in functions or predicates, have any intrinsic meaning of referrent of its own. In other words, variables used by logical quantifiers are, in a sense, bound to that quantifier's scope and lose meaning outside that scope. This inspires the following definition.

(Definition) Free and bound variables: *Let ϕ be a formula containing variable x_i. An occurrence of x_i in ϕ is* ***free*** *if it's not within the scope of any occurrence of $\forall x_i$ in ϕ; otherwise, the variable is* ***bound****. If there exists a free occurrence of x_i, then we say x_i is a* ***free variable****; otherwise x_i is a* ***bound variable****.*

Formulas with only bound variables are more self-contained then those containing free variables, since every bound variable concretely refers to a certain domain of discourse. Free variables, on the other hand, refer to some external referrent that's not defined within any scope of the formula, so in order for the formula to have actual meaning as a logical statement, we require its free variables to have well-defined external references. We'll distinguish between these two formulas, motivated by the intuition that formulas with only bound variables are properly defined elements of our predicate language than their counterparts with free variables.

(Definition) Sentence: *A formula is a* ***sentence*** *if it contains no free variables.*