2 The Syntax of the First-Order Logic

Now that we have some intuitive understanding of predicates, quantifiers, terms, truth values, and the common logical connectives, we are ready to define the *syntax* of our new language, which we will call the *first-order logic.* A grammatically correct expression will be referred to as a *well-formed formula.*[1][2] We will specify this grammar *recursively* by defining what a well-formed formula is in terms of other well-formed formulae. In order for this to make any sense, we will specify our grammar using two different kinds of rules. The first kind of rules give us explicit examples of *wffs* without reference to the phrase *"well-formed formula."*[3] These account for the first three rules in the definition below. The rest of the rules attempt to define what a *wff* is by referring to another *wff.*[4]

[1] This is often abbreviated *"wff."*

[2] The word *"formula"* is synonymous with the word *"expression"* for our purposes here.

[3] These are often called *"base cases"* or *"bases"* in a recursive definition.

[4] Usually called *"recursive cases."*

Definition 2.1: Well-Formed Formulae.
We define an expression λ as a *well-formed formula* if and only if λ is syntactically equal[5] to any of the following expressions.

1. $\top$.
2. $\bot$.
3. $\pi(t_1, t_2, ..., t_n)$, where n is a positive natural number, π is an n-ary predicate, and t_1, t_2, …, t_n are terms—constant or variable symbols representing objects.
4. $\neg(\varphi)$, where φ is a *wff.*
5. $(\varphi) \wedge (\psi)$, where φ and ψ are both *wffs.*
6. $(\varphi) \vee (\psi)$, where φ and ψ are both *wffs.*
7. $(\varphi) \to (\psi)$, where φ and ψ are both *wffs.*
8. $(\varphi) \leftrightarrow (\psi)$, where φ and ψ are both *wffs.*
9. $\forall x(\varphi)$, where φ is a *wff* in which x does not appear as a bound variable.
10. $\exists x(\varphi)$, where φ is a *wff* in which x does not appear as a bound variable. ⌟

[5] In this definition, when we say that λ is *syntactically equal* to another expression, we are referring to *string equality:* two strings are equal precisely when they have exactly the same characters arranged in exactly the same order. For example, the sentence $\neg\neg\top$ is logically equivalent to $\top$, but those two sentences are not *equal as strings* because the first uses the $\neg$ symbol, which doesn't appear in the second.

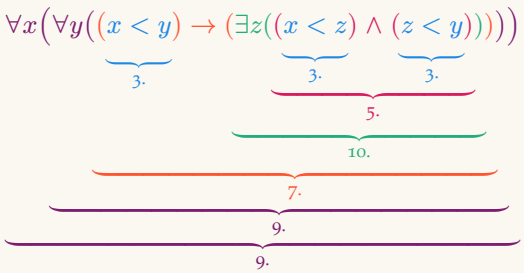
For an example of a well-formed formula, let's first define the following predicates.

$$< (x, y) \coloneqq \text{"}x \text{ is less than } y\text{"}$$
$$= (x, y) \coloneqq \text{"}x \text{ is the same object as } y\text{"}$$

Now consider the expression below, which says *"between any two distinct numbers, there lies another number"* if we take a universe of discourse containing numbers.

$$\forall x \forall y \big(x < y \to \exists z (x < z \wedge z < y)\big)$$

This expression fulfills our definition of a well-formed formula because the *grammar* that *Definition 2.1* recursively lays out is capable of *parsing* this sentence. We demonstrate this below by tagging each part of the sentence according to the numbered rule from *Definition 2.1* that justifies why that portion of the sentence is a *wff.*



At the bottom of the hierarchy are the *predicates applied to terms*, which are *wffs* because of *rule 3.* By combining two such predicate blocks using a *conjunction*, we obtain a *wff* thanks to *rule 5.* If we take this conjunctive block and *existentially quantify* one of its variables, then *rule 10.* tells us that is also a *wff*. Now, we can construct a *wff* according to *rule 7.* by placing a predicate block in a *conditional statement* that has the existential block as its conclusion. Finally, we can take this conditional block and *quantify two of its variables* to produce a *wff* by two successive applications of *rule 9.*

Note that there are several pairs of parentheses in this parsed expression that were not present in the original; however, these parentheses were not strictly required in order to unambiguously parse the expression. Since they were not required, we will permit ourselves to drop them, and we will still understand this to be a well-formed formula. That's not to say that we can drop any pairs of parentheses that we feel like, because some *are* required.[1]

$$\forall x \forall y \forall z (x < y \land y < z \to x < z)$$

The above expression is not well-formed because we don't know if the fragment inside the parentheses is meant to be parsed as $((x < y) \land (y < z)) \to (x < z)$ or as $(x < y) \land ((y < z) \to (x < z))$, and those say very different things. Similarly, expressions that involve a mix of $\land$ and $\lor$ *require* parentheses.

$$\forall x \forall y \forall z (x < y \land y < z \land x < z)$$

However, it is common to see expressions like the one above written with less parentheses than you might expect because $\land$ is an *associative* logical connective.[2] We don't necessarily need parentheses around the individual predicate blocks above, like $(x < y) \land (y < z) \land (x < z)$ precisely because $<$ is a *predicate* and $\land$ is a *logical connective*, and these two kinds of things take different types of inputs. Trying to parse that expression like $(x < (y \land y) < z) \land (x < z)$ would be nonsensical because y is sometimes treated like a *term* and sometimes like a *formula*, and it can't be both. In general, symbols that don't associate or that can't be disambiguated based on *"types"* like these *require parentheses!*[3]

Let's quickly look at one more example, and let's take a universe of discourse consisting of the all of the natural numbers 0, 1, 2, … and so on.

$$\forall y (x < y \lor x = y)$$

This expression is clearly a well-formed formula—it's a universally quantified predicate, which is an application of *rule 10.* to an instance of *rule 3.* Yet, we should be able to see that this sentence *has no truth value* because it contains a *free variable*—namely, x. In order to figure out what this means, we need to resolve this ambiguity by either

[1] It is possible for us to impose a *"presedence"* on the symbols in our language so that any expression can be unambiguously parsed without quantifiers by, for example, saying that $\land$ takes higher presedence than $\to$ (as is often the practice) in a similar way that, in grade school, you may have been taught that multiplication takes higher presedence than addition. We will not do this, but be aware that this is the case in some other contexts.

[2] More on this later.

[3] If ever in doubt, err on the side of using *more* rather than *less* parentheses to ensure your expression is well-formed and therefore unambiguous.

quantifying x or replacing x with a constant term. Importantly, even if we concede that x is a variable (as opposed to a constant), we can't infer the truth value of this statement because we don't understand what role x is playing in that expression. If x is interpreted as being arbitrary,[1] then this would read $\forall x \forall y(x < y \vee x = y)$, which is clearly *false* because $1 \not< 0$ and $1 \neq 0$. If we interpret x as being unspecified but particular,[2] then this would read $\exists x \forall y(x < y \vee x = y)$, which is *true* because every natural number is greater than or equal to 0, so we could fix $x := 0$ and verify that $\forall y(0 < y \vee 0 = y)$ is *true*.

[1] This corresponds with universally quantifying x. There are some people —some of them may even be intelligent—who interpert free variables as being implicitly universally quantified. We will not be following in this morally dubious tradition.

[2] This corresponds with existential quantification. I am not aware of anyone who interprets free variables as implicitly existentially quantified, thank God.

The well-formed formulae are the schema for those sentences that are grammatically correct, but in order to carry *meaning*—in order to carry a *truth value*— a well-formed formula must contain *no free variables!* These are the actual kinds of utterances that we want to communicate when we speak in our language, so let's give them a name.

Definition 2.2: Proposition.
We call a formula λ a *proposition* if and only if λ is a *wff* with no free variables. ⌟

2.1 Notes on Quantification of Variables

In the same way that Modern English lexicography has simplified and standardized throughout the hundreds of years of the language's development, there are several considerations and *"best practices"* that we should be aware of in order to make sure we fully understand the meaning of first-order logic's syntax.

First of all, we will ***never*** re-quantify a variable that has already been quantified within a given scope. Consider the following example.

$$\forall x \forall y\big((0 < x \wedge x < y) \to \exists x(x = y)\big)$$

In the above expression, in the inner $x = y$ fragment, does x refer to the one that was quantified *existentially* (by $\exists x$) or the one that was quantified *universally* (by $\forall x$)? The statement would resolve to *true* in the former case, but *false* in the latter case![3] We *could* disambiguate this problem in several ways that would allow us to re-quantify variables that are already quantified in the same scope,[4] but there is absolutely no reason to even allow for the possibility of this level of confusion. If you ever find yourself in such a situation, just ***use a different symbol!*** This is explicitly enforced in *rule 9.* and *rule 10.* of *Definition 2.1*.

[3] Supposing we take a universe of discourse consisting of all natural numbers as we have normally been doing.

[4] In fact, this is a somewhat common perspective particularly among those who take inspiration from the semantics of programming languages, where it is more common to allow rebinding variables in nested scopes (*e.g.,* Haskell). This is sometimes referred to as *"shadowing,"* and is distinct from just reassigning values to a variable because it takes into account lexical scoping. However, different programming languages resolve lexical scoping in different and potentially incompatible ways, and some programming languages even explicitly disallow this kind of behavior (*e.g.,* Erlang). Also, the first-order logic isn't a programming language.

Second, we should be careful to note the ***order of quantification*** because it can severely impact the meaning of a sentence. Let's consider the following proposition.

$$\forall x \exists y(x < y)$$

Reading this in order from left to right: *"For each x, there exists a y such that $x < y$."* This means that for each x, we can find a corresponding y that is greater than that particular x; different values of x may have different corresponding y values. This is clearly *true* in the natural number universe since, given x, we could let $y := x + 1$. In other words, *y is dependent on x* because y is quantified *after* x. Contrast that with the sentence below, which simply swaps the order of the two quantifiers.

$$\exists y \forall x(x < y)$$

Again, reading this in order: *"There exists a y such that, for all x, we have $x < y$."* This means that there exists a y that is *greater than any x!* In this situation, there is only one value of y that must work *for all* of the different values x can take. This is clearly *false* in the natural number universe because, if we thought we had such a value of y, then $y + 1$ would be even greater, and when $x := y + 1$ we would see $y + 1 \not< y$. Here, y does not depend on x; it's x who depends on y because of the order of quantification.