Chapter udf

Syntax of First-Order Logic

syn.1 Introduction

fol:syn:itx: sec In order to develop the theory and metatheory of first-order logic, we must first define the syntax and semantics of its expressions. The expressions of first-order logic are terms and formulas. Terms are formed from variables, constant symbols, and function symbols. Formulas, in turn, are formed from predicate symbols together with terms (these form the smallest, "atomic" formulas), and then from atomic formulas we can form more complex ones using logical connectives and quantifiers. There are many different ways to set down the formation rules; we give just one possible one. Other systems will chose different symbols, will select different sets of connectives as primitive, will use parentheses differently (or even not at all, as in the case of so-called Polish notation). What all approaches have in common, though, is that the formation rules define the set of terms and formulas *inductively*. If done properly, every expression can result essentially in only one way according to the formation rules. The inductive definition resulting in expressions that are *uniquely readable* means we can give meanings to these expressions using the same method—inductive definition.

syn.2 First-Order Languages

fol:syn:fol: sec Expressions of first-order logic are built up from a basic vocabulary containing *variables*, *constant symbols*, *predicate symbols* and sometimes *function symbols*. From them, together with logical connectives, quantifiers, and punctuation symbols such as parentheses and commas, *terms* and *formulas* are formed.

Informally, predicate symbols are names for properties and relations, constant symbols are names for individual objects, and function symbols are names for mappings. These, except for the identity predicate $=$, are the *non-logical symbols* and together make up a language. Any first-order language $\mathcal{L}$ is determined by its non-logical symbols. In the most general case, $\mathcal{L}$ contains infinitely many symbols of each kind. explanation

In the general case, we make use of the following symbols in first-order logic:

1. Logical symbols
   a) Logical connectives: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (conditional), $\leftrightarrow$ (biconditional), $\forall$ (universal quantifier), $\exists$ (existential quantifier).
   b) The propositional constant for falsity $\bot$.
   c) The propositional constant for truth $\top$.
   d) The two-place identity predicate $=$.
   e) A denumerable set of variables: v_0, v_1, v_2, ...
2. Non-logical symbols, making up the *standard language* of first-order logic
   a) A denumerable set of n-place predicate symbols for each $n > 0$: A_0^n, A_1^n, A_2^n, ...
   b) A denumerable set of constant symbols: c_0, c_1, c_2, ...
   c) A denumerable set of n-place function symbols for each $n > 0$: f_0^n, f_1^n, f_2^n, ...
3. Punctuation marks: (,), and the comma.

Most of our definitions and results will be formulated for the full standard language of first-order logic. However, depending on the application, we may also restrict the language to only a few predicate symbols, constant symbols, and function symbols.

Example syn.1. The language $\mathcal{L}_A$ of arithmetic contains a single two-place predicate symbol $<$, a single constant symbol 0, one one-place function symbol $\prime$, and two two-place function symbols $+$ and $\times$.

Example syn.2. The language of set theory $\mathcal{L}_Z$ contains only the single two-place predicate symbol $\in$.

Example syn.3. The language of orders $\mathcal{L}_{\leq}$ contains only the two-place predicate symbol $\leq$.

Again, these are conventions: officially, these are just aliases, e.g., $<$, $\in$, and $\leq$ are aliases for A_0^2, 0 for c_0, $\prime$ for f_0^1, $+$ for f_0^2, $\times$ for f_1^2.

.

intro You may be familiar with different terminology and symbols than the ones we use above. Logic texts (and teachers) commonly use either $\sim$, $\neg$, and ! for "negation", $\wedge$, $\cdot$, and & for "conjunction". Commonly used symbols for the "conditional" or "implication" are $\rightarrow$, $\Rightarrow$, and $\supset$. Symbols for "biconditional," "bi-implication," or "(material) equivalence" are $\leftrightarrow$, $\Leftrightarrow$, and $\equiv$. The $\bot$ symbol is variously called "falsity," "falsum,", "absurdity,", or "bottom." The $\top$ symbol is variously called "truth," "verum,", or "top."

It is conventional to use lower case letters (e.g., a, b, c) from the beginning of the Latin alphabet for constant symbols (sometimes called names), and lower case letters from the end (e.g., x, y, z) for variables. Quantifiers combine with variables, e.g., x; notational variations include $\forall x$, $(\forall x)$, (x), Πx, $\bigwedge_x$ for the universal quantifier and $\exists x$, $(\exists x)$, (Ex), Σx, $\bigvee_x$ for the existential quantifier.

We might treat all the propositional operators and both quantifiers as primitive symbols of the language. We might instead choose a smaller stock of primitive symbols and treat the other logical operators as defined. "Truth functionally complete" sets of Boolean operators include $\{\neg, \lor\}$, $\{\neg, \land\}$, and $\{\neg, \rightarrow\}$—these can be combined with either quantifier for an expressively complete first-order language. explanation

You may be familiar with two other logical operators: the Sheffer stroke $|$ (named after Henry Sheffer), and Peirce's arrow $\downarrow$, also known as Quine's dagger. When given their usual readings of "nand" and "nor" (respectively), these operators are truth functionally complete by themselves.

syn.3 Terms and Formulas

fol:syn:frm:sec

Once a first-order language $\mathcal{L}$ is given, we can define expressions built up from the basic vocabulary of $\mathcal{L}$. These include in particular *terms* and *formulas*.

fol:syn:frm:defn:terms

Definition syn.4 (Terms). The set of *terms* $\mathrm{Trm}(\mathcal{L})$ of $\mathcal{L}$ is defined inductively by:

1. Every variable is a term.
2. Every constant symbol of $\mathcal{L}$ is a term.
3. If f is an n-place function symbol and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.
4. Nothing else is a term.

A term containing no variables is a *closed term*.

The constant symbols appear in our specification of the language and the terms as a separate category of symbols, but they could instead have been included as zero-place function symbols. We could then do without the second clause in the definition of terms. We just have to understand $f(t_1, \dots, t_n)$ as just f by itself if $n = 0$. explanation

fol:syn:frm:defn:formulas

Definition syn.5 (Formula). The set of *formulas* $\mathrm{Frm}(\mathcal{L})$ of the language $\mathcal{L}$ is defined inductively as follows:

1. $\bot$ is an atomic formula.
2. $\top$ is an atomic formula.
3. If R is an n-place predicate symbol of $\mathcal{L}$ and t_1, …, t_n are terms of $\mathcal{L}$, then $R(t_1, \dots, t_n)$ is an atomic formula.

4. If t_1 and t_2 are terms of $\mathcal{L}$, then $=(t_1, t_2)$ is an atomic formula.

5. If φ is a formula, then $\neg\varphi$ is formula.

6. If φ and ψ are formulas, then $(\varphi \land \psi)$ is a formula.

7. If φ and ψ are formulas, then $(\varphi \lor \psi)$ is a formula.

8. If φ and ψ are formulas, then $(\varphi \rightarrow \psi)$ is a formula.

9. If φ and ψ are formulas, then $(\varphi \leftrightarrow \psi)$ is a formula.

10. If φ is a formula and x is a variable, then $\forall x\, \varphi$ is a formula.

11. If φ is a formula and x is a variable, then $\exists x\, \varphi$ is a formula.

12. Nothing else is a formula.

explanation

The definitions of the set of terms and that of formulas are *inductive definitions*. Essentially, we construct the set of formulas in infinitely many stages. In the initial stage, we pronounce all atomic formulas to be formulas; this corresponds to the first few cases of the definition, i.e., the cases for $\top$, $\bot$, $R(t_1, \dots, t_n)$ and $=(t_1, t_2)$. "Atomic formula" thus means any formula of this form.

The other cases of the definition give rules for constructing new formulas out of formulas already constructed. At the second stage, we can use them to construct formulas out of atomic formulas. At the third stage, we construct new formulas from the atomic formulas and those obtained in the second stage, and so on. A formula is anything that is eventually constructed at such a stage, and nothing else.

By convention, we write $=$ between its arguments and leave out the parentheses: $t_1 = t_2$ is an abbreviation for $=(t_1, t_2)$. Moreover, $\neg{=}(t_1, t_2)$ is abbreviated as $t_1 \neq t_2$. When writing a formula $(\psi * \chi)$ constructed from ψ, χ using a two-place connective $*$, we will often leave out the outermost pair of parentheses and write simply $\psi * \chi$.

intro

Some logic texts require that the variable x must occur in φ in order for $\exists x\, \varphi$ and $\forall x\, \varphi$ to count as formulas. Nothing bad happens if you don't require this, and it makes things easier.

If we work in a language for a specific application, we will often write two-place predicate symbols and function symbols between the respective terms, e.g., $t_1 < t_2$ and $(t_1 + t_2)$ in the language of arithmetic and $t_1 \in t_2$ in the language of set theory. The successor function in the language of arithmetic is even written conventionally *after* its argument: t'. Officially, however, these are just conventional abbreviations for $A^2_0(t_1, t_2)$, $f^2_0(t_1, t_2)$, $A^2_0(t_1, t_2)$ and $f^1_0(t)$, respectively.

Definition syn.6 (Syntactic identity). The symbol $\equiv$ expresses syntactic identity between strings of symbols, i.e., $\varphi \equiv \psi$ iff φ and ψ are strings of symbols of the same length and which contain the same symbol in each place.

The $\equiv$ symbol may be flanked by strings obtained by concatenation, e.g., $\varphi \equiv (\psi \lor \chi)$ means: the string of symbols φ is the same string as the one obtained by concatenating an opening parenthesis, the string ψ, the $\lor$ symbol, the string χ, and a closing parenthesis, in this order. If this is the case, then we know that the first symbol of φ is an opening parenthesis, φ contains ψ as a substring (starting at the second symbol), that substring is followed by $\lor$, etc.

syn.4 Unique Readability

fol:syn:unq:sec The way we defined formulas guarantees that every formula has a *unique reading*, i.e., there is essentially only one way of constructing it according to our formation rules for formulas and only one way of "interpreting" it. If this were not so, we would have ambiguous formulas, i.e., formulas that have more than one reading or intepretation—and that is clearly something we want to avoid. But more importantly, without this property, most of the definitions and proofs we are going to give will not go through. explanation

Perhaps the best way to make this clear is to see what would happen if we had given bad rules for forming formulas that would not guarantee unique readability. For instance, we could have forgotten the parentheses in the formation rules for connectives, e.g., we might have allowed this:

> If φ and ψ are formulas, then so is $\varphi \to \psi$.

Starting from an atomic formula θ, this would allow us to form $\theta \to \theta$. From this, together with θ, we would get $\theta \to \theta \to \theta$. But there are two ways to do this:

1. We take θ to be φ and $\theta \to \theta$ to be ψ.

2. We take φ to be $\theta \to \theta$ and ψ is θ.

Correspondingly, there are two ways to "read" the formula $\theta \to \theta \to \theta$. It is of the form $\psi \to \chi$ where ψ is θ and χ is $\theta \to \theta$, but *it is also* of the form $\psi \to \chi$ with ψ being $\theta \to \theta$ and χ being θ.

If this happens, our definitions will not always work. For instance, when we define the main operator of a formula, we say: in a formula of the form $\psi \to \chi$, the main operator is the indicated occurrence of $\to$. But if we can match the formula $\theta \to \theta \to \theta$ with $\psi \to \chi$ in the two different ways mentioned above, then in one case we get the first occurrence of $\to$ as the main operator, and in the second case the second occurrence. But we intend the main operator to be a *function* of the formula, i.e., every formula must have exactly one main operator occurrence.

Lemma syn.7. *The number of left and right parentheses in a formula φ are equal.*

Proof. We prove this by induction on the way φ is constructed. This requires two things: (a) We have to prove first that all atomic formulas have the property in question (the induction basis). (b) Then we have to prove that when we construct new formulas out of given formulas, the new formulas have the property provided the old ones do.

Let $l(\varphi)$ be the number of left parentheses, and $r(\varphi)$ the number of right parentheses in φ, and $l(t)$ and $r(t)$ similarly the number of left and right parentheses in a term t. We leave the proof that for any term t, $l(t) = r(t)$ as an exercise.

1. $\varphi \equiv \bot$: φ has 0 left and 0 right parentheses.
2. $\varphi \equiv \top$: φ has 0 left and 0 right parentheses.
3. $\varphi \equiv R(t_1, \dots, t_n)$: $l(\varphi) = 1 + l(t_1) + \dots + l(t_n) = 1 + r(t_1) + \dots + r(t_n) = r(\varphi)$. Here we make use of the fact, left as an exercise, that $l(t) = r(t)$ for any term t.
4. $\varphi \equiv t_1 = t_2$: $l(\varphi) = l(t_1) + l(t_2) = r(t_1) + r(t_2) = r(\varphi)$.
5. $\varphi \equiv \lnot\psi$: By induction hypothesis, $l(\psi) = r(\psi)$. Thus $l(\varphi) = l(\psi) = r(\psi) = r(\varphi)$.
6. $\varphi \equiv (\psi * \chi)$: By induction hypothesis, $l(\psi) = r(\psi)$ and $l(\chi) = r(\chi)$. Thus $l(\varphi) = 1 + l(\psi) + l(\chi) = 1 + r(\psi) + r(\chi) = r(\varphi)$.
7. $\varphi \equiv \forall x\, \psi$: By induction hypothesis, $l(\psi) = r(\psi)$. Thus, $l(\varphi) = l(\psi) = r(\psi) = r(\varphi)$.
8. $\varphi \equiv \exists x\, \psi$: Similarly. □

Definition syn.8 (Proper prefix). A string of symbols ψ is a *proper prefix* of a string of symbols φ if concatenating ψ and a non-empty string of symbols yields φ.

Lemma syn.9. *If φ is a formula, and ψ is a proper prefix of φ, then ψ is not a formula.* fol:syn:unq: lem:no-prefix

Proof. Exercise. □

Problem syn.1. Prove Lemma syn.9.

Proposition syn.10. *If φ is an atomic formula, then it satisfes one, and only one of the following conditions.* fol:syn:unq: prop:unique-atomic

1. $\varphi \equiv \bot$.
2. $\varphi \equiv \top$.

3. $\varphi \equiv R(t_1, \dots, t_n)$ *where* R *is an* n*-place predicate symbol,* t_1*, ...,* t_n *are terms, and each of* R*,* t_1*, ...,* t_n *is uniquely determined.*

4. $\varphi \equiv t_1 = t_2$ *where* t_1 *and* t_2 *are uniquely determined terms.*

Proof. Exercise. □

Problem syn.2. Prove Proposition syn.10 (Hint: Formulate and prove a version of Lemma syn.9 for terms.)

Proposition syn.11 (Unique Readability). *Every formula satisfies one, and only one of the following conditions.*

1. φ *is atomic.*

2. φ *is of the form* $\neg\psi$*.*

3. φ *is of the form* $(\psi \land \chi)$*.*

4. φ *is of the form* $(\psi \lor \chi)$*.*

5. φ *is of the form* $(\psi \to \chi)$*.*

6. φ *is of the form* $(\psi \leftrightarrow \chi)$*.*

7. φ *is of the form* $\forall x\, \psi$*.*

8. φ *is of the form* $\exists x\, \psi$*.*

Moreover, in each case ψ*, or* ψ *and* χ*, are uniquely determined. This means that, e.g., there are no different pairs* ψ*,* χ *and* ψ'*,* χ' *so that* φ *is both of the form* $(\psi \to \chi)$ *and* $(\psi' \to \chi')$*.*

Proof. The formation rules require that if a formula is not atomic, it must start with an opening parenthesis (, $\neg$, or with a quantifier. On the other hand, every formula that start with one of the following symbols must be atomic: a predicate symbol, a function symbol, a constant symbol, $\bot$, $\top$.

So we really only have to show that if φ is of the form $(\psi * \chi)$ and also of the form $(\psi' *' \chi')$, then $\psi \equiv \psi'$, $\chi \equiv \chi'$, and $* = *'$.

So suppose both $\varphi \equiv (\psi * \chi)$ and $\varphi \equiv (\psi' *' \chi')$. Then either $\psi \equiv \psi'$ or not. If it is, clearly $* = *'$ and $\chi \equiv \chi'$, since they then are substrings of φ that begin in the same place and are of the same length. The other case is $\psi \not\equiv \psi'$. Since ψ and ψ' are both substrings of φ that begin at the same place, one must be a proper prefix of the other. But this is impossible by Lemma syn.9. □

syn.5 Main operator of a Formula

explanation It is often useful to talk about the last operator used in constructing a formula φ. This operator is called the *main operator* of φ. Intuitively, it is the "outermost" operator of φ. For example, the main operator of $\lnot\varphi$ is $\lnot$, the main operator of $(\varphi \lor \psi)$ is $\lor$, etc. fol:syn:mai:sec

Definition syn.12 (Main operator). The *main operator* of a formula φ is defined as follows: fol:syn:mai:def:main-op

1. φ is atomic: φ has no main operator.
2. $\varphi \equiv \lnot\psi$: the main operator of φ is $\lnot$.
3. $\varphi \equiv (\psi \land \chi)$: the main operator of φ is $\land$.
4. $\varphi \equiv (\psi \lor \chi)$: the main operator of φ is $\lor$.
5. $\varphi \equiv (\psi \to \chi)$: the main operator of φ is $\to$.
6. $\varphi \equiv (\psi \leftrightarrow \chi)$: the main operator of φ is $\leftrightarrow$.
7. $\varphi \equiv \forall x\, \psi$: the main operator of φ is $\forall$.
8. $\varphi \equiv \exists x\, \psi$: the main operator of φ is $\exists$.

In each case, we intend the specific indicated *occurrence* of the main operator in the formula. For instance, since the formula $((\theta \to \alpha) \to (\alpha \to \theta))$ is of the form $(\psi \to \chi)$ where ψ is $(\theta \to \alpha)$ and χ is $(\alpha \to \theta)$, the second occurrence of $\to$ is the main operator.

explanation This is a *recursive* definition of a function which maps all non-atomic formulas to their main operator occurrence. Because of the way formulas are defined inductively, every formula φ satisfies one of the cases in Definition syn.12. This guarantees that for each non-atomic formula φ a main operator exists. Because each formula satisfies only one of these conditions, and because the smaller formulas from which φ is constructed are uniquely determined in each case, the main operator occurrence of φ is unique, and so we have defined a function.

We call formulas by the following names depending on which symbol their main operator is:

Main operator	Type of formula	Example
none	atomic (formula)	$\bot$, $\top$, $R(t_1, \dots, t_n)$, $t_1 = t_2$
$\lnot$	negation	$\lnot\varphi$
$\land$	conjunction	$(\varphi \land \psi)$
$\lor$	disjunction	$(\varphi \lor \psi)$
$\to$	conditional	$(\varphi \to \psi)$
$\forall$	universal (formula)	$\forall x\, \varphi$
$\exists$	existential (formula)	$\exists x\, \varphi$

syn.6 Subformulas

fol:syn:sbf:sec It is often useful to talk about the formulas that "make up" a given formula. explanation
We call these its *subformulas*. Any formula counts as a subformula of itself; a subformula of φ other than φ itself is a *proper subformula*.

Definition syn.13 (Immediate Subformula). If φ is a formula, the *immediate subformulas* of φ are defined inductively as follows:

1. Atomic formulas have no immediate subformulas.

2. $\varphi \equiv \lnot\psi$: The only immediate subformula of φ is ψ.

3. $\varphi \equiv (\psi * \chi)$: The immediate subformulas of φ are ψ and χ ($*$ is any one of the two-place connectives).

4. $\varphi \equiv \forall x\, \psi$: The only immediate subformula of φ is ψ.

5. $\varphi \equiv \exists x\, \psi$: The only immediate subformula of φ is ψ.

Definition syn.14 (Proper Subformula). If φ is a formula, the *proper subformulas* of φ are recursively as follows:

1. Atomic formulas have no proper subformulas.

2. $\varphi \equiv \lnot\psi$: The proper subformulas of φ are ψ together with all proper subformulas of ψ.

3. $\varphi \equiv (\psi * \chi)$: The proper subformulas of φ are ψ, χ, together with all proper subformulas of ψ and those of χ.

4. $\varphi \equiv \forall x\, \psi$: The proper subformulas of φ are ψ together with all proper subformulas of ψ.

5. $\varphi \equiv \exists x\, \psi$: The proper subformulas of φ are ψ together with all proper subformulas of ψ.

Definition syn.15 (Subformula). The subformulas of φ are φ itself together with all its proper subformulas.

Note the subtle difference in how we have defined immediate subformulas explanation
and proper subformulas. In the first case, we have directly defined the immediate subformulas of a formula φ for each possible form of φ. It is an explicit definition by cases, and the cases mirror the inductive definition of the set of formulas. In the second case, we have also mirrored the way the set of all formulas is defined, but in each case we have also included the proper subformulas of the smaller formulas ψ, χ in addition to these formulas themselves. This makes the definition *recursive*. In general, a definition of a function on an inductively defined set (in our case, formulas) is recursive if the cases in the definition of the function make use of the function itself. To be well defined,

we must make sure, however, that we only ever use the values of the function for arguments that come "before" the one we are defining—in our case, when defining "proper subformula" for $(\psi * \chi)$ we only use the proper subformulas of the "earlier" formulas ψ and χ.

syn.7 Free Variables and Sentences

fol:syn:fvs:sec

Definition syn.16 (Free occurrences of a variable). The *free* occurrences of a variable in a formula are defined inductively as follows: fol:syn:fvs:defn:free-occ

1. φ is atomic: all variable occurrences in φ are free.
2. $\varphi \equiv \lnot\psi$: the free variable occurrences of φ are exactly those of ψ.
3. $\varphi \equiv (\psi * \chi)$: the free variable occurrences of φ are those in ψ together with those in χ.
4. $\varphi \equiv \forall x\, \psi$: the free variable occurrences in φ are all of those in ψ except for occurrences of x.
5. $\varphi \equiv \exists x\, \psi$: the free variable occurrences in φ are all of those in ψ except for occurrences of x.

Definition syn.17 (Bound Variables). An occurrence of a variable in a formula φ is *bound* if it is not free.

Problem syn.3. Give an inductive definition of the bound variable occurrences along the lines of Definition syn.16.

Definition syn.18 (Scope). If $\forall x\, \psi$ is an occurrence of a subformula in a formula φ, then the corresponding occurrence of ψ in φ is called the *scope* of the corresponding occurrence of $\forall x$. Similarly for $\exists x$.

If ψ is the scope of a quantifier occurrence $\forall x$ or $\exists x$ in φ, then the free occurrences of x in ψ are bound in $\forall x\, \psi$ and $\exists x\, \psi$. We say that these occurrences are *bound by* the mentioned quantifier occurrence.

Example syn.19. Consider the following formula:

$$\exists v_0\ \underbrace{A^2_0(v_0, v_1)}_{\psi}$$

ψ represents the scope of $\exists v_0$. The quantifier binds the occurence of v_0 in ψ, but does not bind the occurence of v_1. So v_1 is a free variable in this case.

We can now see how this might work in a more complicated formula φ:

$$\forall v_0\ \underbrace{(A^1_0(v_0) \to A^2_0(v_0, v_1))}_{\psi} \to \exists v_1\ \underbrace{(A^2_1(v_0, v_1) \lor \forall v_0\ \overbrace{\lnot A^1_1(v_0)}^{\theta})}_{\chi}$$

ψ is the scope of the first $\forall v_0$, χ is the scope of $\exists v_1$, and θ is the scope of the second $\forall v_0$. The first $\forall v_0$ binds the occurrences of v_0 in ψ, $\exists v_1$ the occurrence of v_1 in χ, and the second $\forall v_0$ binds the occurrence of v_0 in θ. The first occurrence of v_1 and the fourth occurrence of v_0 are free in φ. The last occurrence of v_0 is free in θ, but bound in χ and φ.

Definition syn.20 (Sentence). A formula φ is a *sentence* iff it contains no free occurrences of variables.

syn.8 Substitution

fol:syn:sub:sec

Definition syn.21 (Substitution in a term). We define $s[t/x]$, the result of *substituting* t for every occurrence of x in s, recursively:

1. $s \equiv c$: $s[t/x]$ is just s.
2. $s \equiv y$: $s[t/x]$ is also just s, provided y is a variable and $y \not\equiv x$.
3. $s \equiv x$: $s[t/x]$ is t.
4. $s \equiv f(t_1, \dots, t_n)$: $s[t/x]$ is $f(t_1[t/x], \dots, t_n[t/x])$.

Definition syn.22. A term t is *free for* x in φ if none of the free occurrences of x in φ occur in the scope of a quantifier that binds a variable in t.

Example syn.23.

1. v_8 is free for v_1 in $\exists v_3 A^2_4(v_3, v_1)$
2. $f^2_1(v_1, v_2)$ is *not* free for v_0 in $\forall v_2 A^2_4(v_0, v_2)$

Definition syn.24 (Substitution in a formula). If φ is a formula, x is a variable, and t is a term free for x in φ, then $\varphi[t/x]$ is the result of substituting t for all free occurrences of x in φ.

1. $\varphi \equiv \bot$: $\varphi[t/x]$ is $\bot$.
2. $\varphi \equiv \top$: $\varphi[t/x]$ is $\top$.
3. $\varphi \equiv P(t_1, \dots, t_n)$: $\varphi[t/x]$ is $P(t_1[t/x], \dots, t_n[t/x])$.
4. $\varphi \equiv t_1 = t_2$: $\varphi[t/x]$ is $t_1[t/x] = t_2[t/x]$.
5. $\varphi \equiv \neg\psi$: $\varphi[t/x]$ is $\neg\psi[t/x]$.
6. $\varphi \equiv (\psi \land \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \land \chi[t/x])$.
7. $\varphi \equiv (\psi \lor \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \lor \chi[t/x])$.
8. $\varphi \equiv (\psi \to \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \to \chi[t/x])$.

9. $\varphi \equiv (\psi \leftrightarrow \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \leftrightarrow \chi[t/x])$.

10. $\varphi \equiv \forall y\, \psi$: $\varphi[t/x]$ is $\forall y\, \psi[t/x]$, provided y is a variable other than x; otherwise $\varphi[t/x]$ is just φ.

11. $\varphi \equiv \exists y\, \psi$: $\varphi[t/x]$ is $\exists y\, \psi[t/x]$, provided y is a variable other than x; otherwise $\varphi[t/x]$ is just φ.

explanation

Note that substitution may be vacuous: If x does not occur in φ at all, then $\varphi[t/x]$ is just φ.

The restriction that t must be free for x in φ is necessary to exclude cases like the following. If $\varphi \equiv \exists y\, x < y$ and $t \equiv y$, then $\varphi[t/x]$ would be $\exists y\, y < y$. In this case the free variable y is "captured" by the quantifier $\exists y$ upon substitution, and that is undesirable. For instance, we would like it to be the case that whenever $\forall x\, \psi$ holds, so does $\psi[t/x]$. But consider $\forall x\, \exists y\, x < y$ (here ψ is $\exists y\, x < y$). It is sentence that is true about, e.g., the natural numbers: for every number x there is a number y greater than it. If we allowed y as a possible substitution for x, we would end up with $\psi[y/x] \equiv \exists y\, y < y$, which is false. We prevent this by requiring that none of the free variables in t would end up being bound by a quantifier in φ.

We often use the following convention to avoid cumbersome notation: If φ is a formula which may contain the variable x free, we also write $\varphi(x)$ to indicate this. When it is clear which φ and x we have in mind, and t is a term (assumed to be free for x in $\varphi(x)$), then we write $\varphi(t)$ as short for $\varphi[t/x]$. So for instance, we might say, "we call $\varphi(t)$ an instance of $\forall x\, A(x)$." By this we mean that if φ is any formula, x a variable, and t a term that's free for x in φ, then $\varphi[t/x]$ is an instance of $\forall x\, \varphi$.

Photo Credits

Bibliography