

Chapter udf

Syntax and Semantics

syn.1 Introduction

fol:syn:int:
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.

Giving the meaning of expressions is the domain of semantics. The central concept in semantics is that of satisfaction in **a structure**. **A structure** gives meaning to the building blocks of the language: **a domain** is a non-empty set of objects. The quantifiers are interpreted as ranging over this domain, **constant symbols** are assigned elements in the domain, **function symbols** are assigned functions from the **domain** to itself, and **predicate symbols** are assigned relations on the **domain**. The **domain** together with assignments to the basic vocabulary constitutes **a structure**. **Variables** may appear in **formulas**, and in order to give a semantics, we also have to assign **elements** of the **domain** to them—this is a variable assignment. The satisfaction relation, finally, brings these together. **A formula** may be satisfied in **a structure** \mathfrak{M} relative to a variable assignment s , written as $\mathfrak{M}, s \models \varphi$. This relation is also defined by induction on the structure of φ , using the truth tables for the logical connectives to define, say, satisfaction of $\varphi \wedge \psi$ in terms of satisfaction (or not) of φ and

ψ . It then turns out that the variable assignment is irrelevant if the **formula** φ is a **sentence**, i.e., has no free variables, and so we can talk of **sentences** being simply satisfied (or not) in **structures**.

On the basis of the satisfaction relation $\mathfrak{M} \models \varphi$ for sentences we can then define the basic semantic notions of validity, entailment, and satisfiability. A sentence is valid, $\models \varphi$, if every structure satisfies it. It is entailed by a set of **sentences**, $\Gamma \models \varphi$, if every **structure** that satisfies all the **sentences** in Γ also satisfies φ . And a set of sentences is satisfiable if some **structure** satisfies all **sentences** in it at the same time. Because **formulas** are inductively defined, and satisfaction is in turn defined by induction on the structure of **formulas**, we can use induction to prove properties of our semantics and to relate the semantic notions defined.

syn.2 First-Order Languages

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.

fol:syn:fol:
sec

explanation

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.

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** \perp .
 - 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, \dots
2. Non-logical symbols, making up the *standard language* of first-order logic
 - a) A **denumerable** set of n -place **predicate symbols** for each $n > 0$: $A_0^n, A_1^n, A_2^n, \dots$
 - b) A **denumerable** set of **constant symbols**: c_0, c_1, c_2, \dots
 - c) A **denumerable** set of n -place **function symbols** for each $n > 0$: $f_0^n, f_1^n, f_2^n, \dots$

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** $!$, 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 , $!$ for f_0^1 , $+$ for f_0^2 , \times for f_1^2 .

You may be familiar with different terminology and symbols than the ones intro 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 \perp 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. explanation 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 $\{\neg, \vee\}$, $\{\neg, \wedge\}$, and $\{\neg, \rightarrow\}$ —these can be combined with either quantifier for an expressively complete first-order language.

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*.

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

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*.

explanation 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 $n = 0$.

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

1. \perp 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, \dots, 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 $(\varphi \wedge \psi)$ is a **formula**.
7. If φ and ψ are **formulas**, then $(\varphi \vee \psi)$ 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**.

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 , \perp , $R(t_1, \dots, t_n)$ and $=(t_1, t_2)$. “Atomic **formula**” thus means any **formula** of this form. explanation

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 $\psi * \chi$.

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. intro

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 $A_0^2(t_1, t_2)$, $f_0^2(t_1, t_2)$, $A_0^2(t_1, t_2)$ and $f_0^1(t)$, 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., $\varphi \equiv (\psi \vee \chi)$ means: the string of symbols φ is the same string as the one obtained by concatenating an opening parenthesis, the string ψ , the \vee 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 \vee , etc.

syn.4 Unique Readability

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 interpretation—and that is clearly something we want to explanation

fol:syn:unq:
sec

avoid. But more importantly, without this property, most of the definitions and proofs we are going to give will not go through.

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 $\varphi \rightarrow \psi$.

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

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

Correspondingly, there are two ways to “read” the **formula** $\theta \rightarrow \theta \rightarrow \theta$. It is of the form $\psi \rightarrow \chi$ where ψ is θ and χ is $\theta \rightarrow \theta$, but *it is also* of the form $\psi \rightarrow \chi$ with ψ being $\theta \rightarrow \theta$ 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 $\psi \rightarrow \chi$, the **main operator** is the indicated occurrence of \rightarrow . But if we can match the formula $\theta \rightarrow \theta \rightarrow \theta$ with $\psi \rightarrow \chi$ in the two different ways mentioned above, then in one case we get the first occurrence of \rightarrow 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. $\varphi \equiv \perp$: 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. $\varphi \equiv \neg\psi$: 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 .

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

Proof. Exercise.

□

Problem syn.1. Prove Lemma syn.9.

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

1. $\varphi \equiv \perp$.
2. $\varphi \equiv \top$.
3. $\varphi \equiv R(t_1, \dots, t_n)$ where R is an n -place predicate symbol, t_1, \dots, t_n are terms, and each of R, t_1, \dots, 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 $(\psi \wedge \chi)$.
4. φ is of the form $(\psi \vee \chi)$.

5. φ is of the form $(\psi \rightarrow \chi)$.
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 $(\psi \rightarrow \chi)$ and $(\psi' \rightarrow \chi')$.

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**, \perp , \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](#). \square

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 $\neg\varphi$ is \neg , the main operator of $(\varphi \vee \psi)$ is \vee , 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. $\varphi \equiv \neg\psi$: the **main operator** of φ is \neg .
3. $\varphi \equiv (\psi \wedge \chi)$: the **main operator** of φ is \wedge .
4. $\varphi \equiv (\psi \vee \chi)$: the **main operator** of φ is \vee .
5. $\varphi \equiv (\psi \rightarrow \chi)$: the **main operator** of φ is \rightarrow .
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 $((\theta \rightarrow \alpha) \rightarrow (\alpha \rightarrow \theta))$ is of the form $(\psi \rightarrow \chi)$ where ψ is $(\theta \rightarrow \alpha)$ and χ is $(\alpha \rightarrow \theta)$, the second occurrence of \rightarrow is the **main operator**.

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. explanation

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

Main operator	Type of formula	Example
none	atomic (formula)	$\perp, \top, R(t_1, \dots, t_n), t_1 = t_2$
\neg	negation	$\neg\varphi$
\wedge	conjunction	$(\varphi \wedge \psi)$
\vee	disjunction	$(\varphi \vee \psi)$
\rightarrow	conditional	$(\varphi \rightarrow \psi)$
\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. $\varphi \equiv \neg\psi$: 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. $\varphi \equiv \neg\psi$: 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**.

explanation

Note the subtle difference in how we have defined immediate **subformulas** 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. $\varphi \equiv \neg\psi$: 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 all occurrences of x which are free in ψ are said to be *bound by* the mentioned quantifier occurrence.

Example syn.19. Consider the following formula:

$$\exists v_0 \underbrace{A_0^2(v_0, v_1)}_{\psi}$$

ψ represents the scope of $\exists v_0$. The quantifier binds the occurrence of v_0 in ψ , but does not bind the occurrence 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 :

$$\forall v_0 \underbrace{(A_0^1(v_0) \rightarrow A_0^2(v_0, v_1))}_{\psi} \rightarrow \exists v_1 \underbrace{(A_1^2(v_0, v_1) \vee \forall v_0 \underbrace{\neg A_1^1(v_0)}_{\theta})}_{\chi}$$

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 $y \neq x$.
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 $\exists v_3 A_4^2(v_3, v_1)$
2. $f_1^2(v_1, v_2)$ is *not* free for v_0 in $\forall v_2 A_4^2(v_0, v_2)$

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. $\varphi \equiv \perp$: $\varphi[t/x]$ is \perp .
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. $\varphi \equiv (\psi \wedge \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \wedge \chi[t/x])$.
7. $\varphi \equiv (\psi \vee \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \vee \chi[t/x])$.
8. $\varphi \equiv (\psi \rightarrow \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \rightarrow \chi[t/x])$.
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 with a free variable x , we 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 $\varphi(x)[t/x]$.

syn.9 Structures for First-order Languages

fol:syn:str:
sec

First-order languages are, by themselves, *uninterpreted*: the **constant symbols**, **function symbols**, and **predicate symbols** have no specific meaning attached to them. Meanings are given by specifying a *structure*. It specifies the *domain*, i.e., the objects which the **constant symbols** pick out, the **function symbols** operate on, and the quantifiers range over. In addition, it specifies which **constant symbols** pick out which objects, how a **function symbol** maps objects to objects, and which objects the **predicate symbols** apply to. **Structures** are the basis for *semantic* notions in logic, e.g., the notion of consequence, validity, satisfiability. They are variously called “structures,” “interpretations,” or “models” in the literature.

explanation

Definition syn.25 (Structures). A *structure* \mathfrak{M} , for a language \mathcal{L} of first-order logic consists of the following elements:

1. *Domain*: a non-empty set, $|\mathfrak{M}|$
2. *Interpretation of constant symbols*: for each **constant symbol** c of \mathcal{L} , an **element** $c^{\mathfrak{M}} \in |\mathfrak{M}|$
3. *Interpretation of predicate symbols*: for each n -place **predicate symbol** R of \mathcal{L} (other than $=$), an n -place relation $R^{\mathfrak{M}} \subseteq |\mathfrak{M}|^n$
4. *Interpretation of function symbols*: for each n -place **function symbol** f of \mathcal{L} , an n -place function $f^{\mathfrak{M}}: |\mathfrak{M}|^n \rightarrow |\mathfrak{M}|$

Example syn.26. A *structure* \mathfrak{M} for the language of arithmetic consists of a set, an element of $|\mathfrak{M}|$, $o^{\mathfrak{M}}$, as interpretation of the **constant symbol** o , a one-place function $\iota^{\mathfrak{M}}: |\mathfrak{M}| \rightarrow |\mathfrak{M}|$, two two-place functions $+^{\mathfrak{M}}$ and $\times^{\mathfrak{M}}$, both $|\mathfrak{M}|^2 \rightarrow |\mathfrak{M}|$, and a two-place relation $<^{\mathfrak{M}} \subseteq |\mathfrak{M}|^2$.

An obvious example of such a structure is the following:

1. $|\mathfrak{N}| = \mathbb{N}$
2. $o^{\mathfrak{N}} = 0$
3. $\iota^{\mathfrak{N}}(n) = n + 1$ for all $n \in \mathbb{N}$
4. $+^{\mathfrak{N}}(n, m) = n + m$ for all $n, m \in \mathbb{N}$
5. $\times^{\mathfrak{N}}(n, m) = n \cdot m$ for all $n, m \in \mathbb{N}$
6. $<^{\mathfrak{N}} = \{(n, m) : n \in \mathbb{N}, m \in \mathbb{N}, n < m\}$

The structure \mathfrak{N} for \mathcal{L}_A so defined is called the *standard model of arithmetic*, because it interprets the non-logical constants of \mathcal{L}_A exactly how you would expect.

However, there are many other possible **structures** for \mathcal{L}_A . For instance, we might take as the domain the set \mathbb{Z} of integers instead of \mathbb{N} , and define the interpretations of o , ι , $+$, \times , $<$ accordingly. But we can also define structures for \mathcal{L}_A which have nothing even remotely to do with numbers.

Example syn.27. A structure \mathfrak{M} for the language \mathcal{L}_Z of set theory requires just a set and a single-two place relation. So technically, e.g., the set of people plus the relation “ x is older than y ” could be used as a **structure** for \mathcal{L}_Z , as well as \mathbb{N} together with $n \geq m$ for $n, m \in \mathbb{N}$.

A particularly interesting **structure** for \mathcal{L}_Z in which the **elements** of the domain are actually sets, and the interpretation of \in actually is the relation “ x is an **element** of y ” is the **structure** $\mathfrak{H}\mathfrak{F}$ of *hereditarily finite sets*:

1. $|\mathfrak{H}\mathfrak{F}| = \emptyset \cup \wp(\emptyset) \cup \wp(\wp(\emptyset)) \cup \wp(\wp(\wp(\emptyset))) \cup \dots$;
2. $\in^{\mathfrak{H}\mathfrak{F}} = \{\langle x, y \rangle : x, y \in |\mathfrak{H}\mathfrak{F}|, x \in y\}$.

digression

The stipulations we make as to what counts as a **structure** impact our logic. For example, the choice to prevent empty domains ensures, given the usual account of satisfaction (or truth) for quantified sentences, that $\exists x (\varphi(x) \vee \neg\varphi(x))$ is valid—that is, a logical truth. And the stipulation that all **constant symbols** must refer to an object in the domain ensures that the existential generalization is a sound pattern of inference: $\varphi(a)$, therefore $\exists x \varphi(x)$. If we allowed names to refer outside the domain, or to not refer, then we would be on our way to a *free logic*, in which existential generalization requires an additional premise: $\varphi(a)$ and $\exists x x = a$, therefore $\exists x \varphi(x)$.

syn.10 Covered **Structures** for First-order Languages

explanation

Recall that a term is *closed* if it contains no **variables**.

fol:syn:cov:
sec

Definition syn.28 (**Value** of closed terms). If t is a closed term of the language \mathcal{L} and \mathfrak{M} is a **structure** for \mathcal{L} , the **value** $\text{Val}^{\mathfrak{M}}(t)$ is defined as follows:

1. If t is just the **constant symbol** c , then $\text{Val}^{\mathfrak{M}}(c) = c^{\mathfrak{M}}$.
2. If t is of the form $f(t_1, \dots, t_n)$, then

$$\text{Val}^{\mathfrak{M}}(t) = f^{\mathfrak{M}}(\text{Val}^{\mathfrak{M}}(t_1), \dots, \text{Val}^{\mathfrak{M}}(t_n)).$$

Definition syn.29 (**Covered structure**). A **structure** is *covered* if every element of the domain is the **value** of some closed term.

Example syn.30. Let \mathcal{L} be the language with **constant symbols** *zero*, *one*, *two*, \dots , the binary **predicate symbol** $<$, and the binary **function symbols** $+$ and \times . Then a **structure** \mathfrak{M} for \mathcal{L} is the one with domain $|\mathfrak{M}| = \{0, 1, 2, \dots\}$ and assignments $zero^{\mathfrak{M}} = 0$, $one^{\mathfrak{M}} = 1$, $two^{\mathfrak{M}} = 2$, and so forth. For the binary relation symbol $<$, the set $<^{\mathfrak{M}}$ is the set of all pairs $\langle c_1, c_2 \rangle \in |\mathfrak{M}|^2$ such that c_1 is less than c_2 : for example, $\langle 1, 3 \rangle \in <^{\mathfrak{M}}$ but $\langle 2, 2 \rangle \notin <^{\mathfrak{M}}$. For the binary **function symbol** $+$, define $+^{\mathfrak{M}}$ in the usual way—for example, $+^{\mathfrak{M}}(2, 3)$ maps to 5, and similarly for the binary **function symbol** \times . Hence, the **value** of

four is just 4, and the **value** of $\times(\textit{two}, +(\textit{three}, \textit{zero}))$ (or in infix notation, $\textit{two} \times (\textit{three} + \textit{zero})$) is

$$\begin{aligned}
 \text{Val}^{\mathfrak{M}}(\times(\textit{two}, +(\textit{three}, \textit{zero}))) &= \\
 &= \times^{\mathfrak{M}}(\text{Val}^{\mathfrak{M}}(\textit{two}), \text{Val}^{\mathfrak{M}}(\textit{two}, +(\textit{three}, \textit{zero}))) \\
 &= \times^{\mathfrak{M}}(\text{Val}^{\mathfrak{M}}(\textit{two}), +^{\mathfrak{M}}(\text{Val}^{\mathfrak{M}}(\textit{three}), \text{Val}^{\mathfrak{M}}(\textit{zero}))) \\
 &= \times^{\mathfrak{M}}(\textit{two}^{\mathfrak{M}}, +^{\mathfrak{M}}(\textit{three}^{\mathfrak{M}}, \textit{zero}^{\mathfrak{M}})) \\
 &= \times^{\mathfrak{M}}(2, +^{\mathfrak{M}}(3, 0)) \\
 &= \times^{\mathfrak{M}}(2, 3) \\
 &= 6
 \end{aligned}$$

Problem syn.4. Is \mathfrak{N} , the standard model of arithmetic, covered? Explain.

syn.11 Satisfaction of a Formula in a Structure

fol:syn:sat:
sec

The basic notion that relates expressions such as terms and **formulas**, on the one hand, and **structures** on the other, are those of **value** of a term and **satisfaction** of a **formula**. Informally, the **value** of a term is an **element** of a **structure**—if the term is just a constant, its **value** is the object assigned to the constant by the **structure**, and if it is built up using **function symbols**, the **value** is computed from the **values** of constants and the functions assigned to the functions in the term. A **formula** is *satisfied* in a **structure** if the interpretation given to the predicates makes the **formula** true in the domain of the **structure**. This notion of satisfaction is specified inductively: the specification of the **structure** directly states when atomic **formulas** are satisfied, and we define when a complex **formula** is satisfied depending on the main connective or quantifier and whether or not the immediate **subformulas** are satisfied. The case of the quantifiers here is a bit tricky, as the immediate **subformula** of a quantified **formula** has a free **variable**, and **structures** don't specify the **values** of **variables**. In order to deal with this difficulty, we also introduce *variable assignments* and define satisfaction not with respect to a **structure** alone, but with respect to a **structure** plus a **variable** assignment.

explanation

Definition syn.31 (Variable Assignment). A *variable assignment* s for a **structure** \mathfrak{M} is a function which maps each **variable** to an element of $|\mathfrak{M}|$, i.e., $s: \text{Var} \rightarrow |\mathfrak{M}|$.

A **structure** assigns a **value** to each **constant symbol**, and a variable assignment to each variable. But we want to use terms built up from them to also name **elements** of the **domain**. For this we define the **value** of terms inductively. For **constant symbols** and variables the value is just as the **structure** or the variable assignment specifies it; for more complex terms it is computed recursively using the functions the **structure** assigns to the **function symbols**.

explanation

Definition syn.32 (Value of Terms). If t is a term of the language \mathcal{L} , \mathfrak{M} is a **structure** for \mathcal{L} , and s is a **variable assignment** for \mathfrak{M} , the **value** $\text{Val}_s^{\mathfrak{M}}(t)$ is defined as follows:

1. $t \equiv c$: $\text{Val}_s^{\mathfrak{M}}(t) = c^{\mathfrak{M}}$.
2. $t \equiv x$: $\text{Val}_s^{\mathfrak{M}}(t) = s(x)$.
3. $t \equiv f(t_1, \dots, t_n)$:

$$\text{Val}_s^{\mathfrak{M}}(t) = f^{\mathfrak{M}}(\text{Val}_s^{\mathfrak{M}}(t_1), \dots, \text{Val}_s^{\mathfrak{M}}(t_n)).$$

Definition syn.33 (x -Variant). If s is a **variable assignment** for a **structure** \mathfrak{M} , then any **variable assignment** s' for \mathfrak{M} which differs from s at most in what it assigns to x is called an x -variant of s . If s' is an x -variant of s we write $s \sim_x s'$.

explanation

Note that an x -variant of an assignment s does not *have* to assign something different to x . In fact, every assignment counts as an x -variant of itself.

Definition syn.34 (Satisfaction). Satisfaction of a **formula** φ in a **structure** \mathfrak{M} relative to a **variable assignment** s , in symbols: $\mathfrak{M}, s \models \varphi$, is defined recursively as follows. (We write $\mathfrak{M}, s \not\models \varphi$ to mean “not $\mathfrak{M}, s \models \varphi$.”)

fol:syn:sat:
defn:satisfaction

1. $\varphi \equiv \perp$: $\mathfrak{M}, s \not\models \varphi$.
2. $\varphi \equiv \top$: $\mathfrak{M}, s \models \varphi$.
3. $\varphi \equiv R(t_1, \dots, t_n)$: $\mathfrak{M}, s \models \varphi$ iff $\langle \text{Val}_s^{\mathfrak{M}}(t_1), \dots, \text{Val}_s^{\mathfrak{M}}(t_n) \rangle \in R^{\mathfrak{M}}$.
4. $\varphi \equiv t_1 = t_2$: $\mathfrak{M}, s \models \varphi$ iff $\text{Val}_s^{\mathfrak{M}}(t_1) = \text{Val}_s^{\mathfrak{M}}(t_2)$.
5. $\varphi \equiv \neg\psi$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$.
6. $\varphi \equiv (\psi \wedge \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \psi$ and $\mathfrak{M}, s \models \chi$.
7. $\varphi \equiv (\psi \vee \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \psi$ or $\mathfrak{M}, s \models \chi$ (or both).
8. $\varphi \equiv (\psi \rightarrow \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$ or $\mathfrak{M}, s \models \chi$ (or both).
9. $\varphi \equiv (\psi \leftrightarrow \chi)$: $\mathfrak{M}, s \models \varphi$ iff either both $\mathfrak{M}, s \models \psi$ and $\mathfrak{M}, s \models \chi$, or neither $\mathfrak{M}, s \models \psi$ nor $\mathfrak{M}, s \models \chi$.
10. $\varphi \equiv \forall x \psi$: $\mathfrak{M}, s \models \varphi$ iff for every x -variant s' of s , $\mathfrak{M}, s' \models \psi$.
11. $\varphi \equiv \exists x \psi$: $\mathfrak{M}, s \models \varphi$ iff there is an x -variant s' of s so that $\mathfrak{M}, s' \models \psi$.

explanation

The variable assignments are important in the last two clauses. We cannot define satisfaction of $\forall x \psi(x)$ by “for all $a \in |\mathfrak{M}|$, $\mathfrak{M} \models \psi(a)$.” We cannot define satisfaction of $\exists x \psi(x)$ by “for at least one $a \in |\mathfrak{M}|$, $\mathfrak{M} \models \psi(a)$.” The reason is that a is not symbol of the language, and so $\psi(a)$ is not a **formula** (that is, $\psi[a/x]$ is undefined). We also cannot assume that we have **constant symbols**

or terms available that name every **element** of \mathfrak{M} , since there is nothing in the definition of **structures** that requires it. Even in the standard language the set of **constant symbols** is **denumerable**, so if $|\mathfrak{M}|$ is not **enumerable** there aren't even enough **constant symbols** to name every object.

Example syn.35. Let $=\{a, b, f, R\}$ where a and b are **constant symbols**, f is a two-place **function symbol**, and R is a two-place **predicate symbol**. Consider the **structure** \mathfrak{M} defined by:

1. $|\mathfrak{M}| = \{1, 2, 3, 4\}$
2. $a^{\mathfrak{M}} = 1$
3. $b^{\mathfrak{M}} = 2$
4. $f^{\mathfrak{M}}(x, y) = x + y$ if $x + y \leq 3$ and $= 3$ otherwise.
5. $R^{\mathfrak{M}} = \{\langle 1, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 2, 4 \rangle\}$

The function $s(x) = 1$ that assigns $1 \in |\mathfrak{M}|$ to every **variable** is a variable assignment for \mathfrak{M} .

Then

$$\text{Val}_s^{\mathfrak{M}}(f(a, b)) = f^{\mathfrak{M}}(\text{Val}_s^{\mathfrak{M}}(a), \text{Val}_s^{\mathfrak{M}}(b)).$$

Since a and b are **constant symbols**, $\text{Val}_s^{\mathfrak{M}}(a) = a^{\mathfrak{M}} = 1$ and $\text{Val}_s^{\mathfrak{M}}(b) = b^{\mathfrak{M}} = 2$. So

$$\text{Val}_s^{\mathfrak{M}}(f(a, b)) = f^{\mathfrak{M}}(1, 2) = 1 + 2 = 3.$$

To compute the value of $f(f(a, b), a)$ we have to consider

$$\text{Val}_s^{\mathfrak{M}}(f(f(a, b), a)) = f^{\mathfrak{M}}(\text{Val}_s^{\mathfrak{M}}(f(a, b)), \text{Val}_s^{\mathfrak{M}}(a)) = f^{\mathfrak{M}}(3, 1) = 3,$$

since $3 + 1 > 3$. Since $s(x) = 1$ and $\text{Val}_s^{\mathfrak{M}}(x) = s(x)$, we also have

$$\text{Val}_s^{\mathfrak{M}}(f(f(a, b), x)) = f^{\mathfrak{M}}(\text{Val}_s^{\mathfrak{M}}(f(a, b)), \text{Val}_s^{\mathfrak{M}}(x)) = f^{\mathfrak{M}}(3, 1) = 3,$$

An atomic **formula** $R(t_1, t_2)$ is satisfied if the tuple of values of its arguments, i.e., $\langle \text{Val}_s^{\mathfrak{M}}(t_1), \text{Val}_s^{\mathfrak{M}}(t_2) \rangle$, is **an element** of $R^{\mathfrak{M}}$. So, e.g., we have $\mathfrak{M}, s \models R(b, f(a, b))$ since $\langle \text{Val}_s^{\mathfrak{M}}(b), \text{Val}_s^{\mathfrak{M}}(f(a, b)) \rangle = \langle 2, 3 \rangle \in R^{\mathfrak{M}}$, but $\mathfrak{M} \not\models R(x, f(a, b))$ since $\langle 1, 3 \rangle \notin R^{\mathfrak{M}}[s]$.

To determine if a non-atomic formula φ is satisfied, you apply the clauses in the inductive definition that applies to the main connective. For instance, the main connective in $R(a, a) \rightarrow (R(b, x) \vee R(x, b))$ is the \rightarrow , and

$$\begin{aligned} \mathfrak{M}, s \models R(a, a) \rightarrow (R(b, x) \vee R(x, b)) \text{ iff} \\ \mathfrak{M}, s \not\models R(a, a) \text{ or } \mathfrak{M}, s \models R(b, x) \vee R(x, b) \end{aligned}$$

Since $\mathfrak{M}, s \models R(a, a)$ (because $\langle 1, 1 \rangle \in R^{\mathfrak{M}}$) we can't yet determine the answer and must first figure out if $\mathfrak{M}, s \models R(b, x) \vee R(x, b)$:

$$\begin{aligned} \mathfrak{M}, s \models R(b, x) \vee R(x, b) &\text{ iff} \\ \mathfrak{M}, s \models R(b, x) &\text{ or } \mathfrak{M}, s \models R(x, b) \end{aligned}$$

And this is the case, since $\mathfrak{M}, s \models R(x, b)$ (because $\langle 1, 2 \rangle \in R^{\mathfrak{M}}$).

Recall that an x -variant of s is a variable assignment that differs from s at most in what it assigns to x . For every **element** of $|\mathfrak{M}|$, there is an x -variant of s : $s_1(x) = 1$, $s_2(x) = 2$, $s_3(x) = 3$, $s_4(x) = 4$, and with $s_i(y) = s(y) = 1$ for all variables y other than x are all the x -variants of s for the structure \mathfrak{M} . Note, in particular, that $s_1 = s$ is also an x -variant of s , i.e., s is an x -variant of itself.

To determine if an existentially quantified **formula** $\exists x \varphi(x)$ is satisfied, we have to determine if $\mathfrak{M}, s' \models \varphi(x)$ for at least one x -variant s' of s . So,

$$\mathfrak{M}, s \models \exists x (R(b, x) \vee R(x, b)),$$

since $\mathfrak{M}, s_1 \models R(b, x) \vee R(x, b)$ (s_3 would also fit the bill). But,

$$\mathfrak{M}, s \not\models \exists x (R(b, x) \wedge R(x, b))$$

since for none of the s_i , $\mathfrak{M}, s_i \models R(b, x) \wedge R(x, b)$.

To determine if a universally quantified **formula** $\forall x \varphi(x)$ is satisfied, we have to determine if $\mathfrak{M}, s' \models \varphi(x)$ for all x -variants s' of s . So,

$$\mathfrak{M}, s \models \forall x (R(x, a) \rightarrow R(a, x)),$$

since $\mathfrak{M}, s_i \models R(x, a) \rightarrow R(a, x)$ for all s_i ($\mathfrak{M}, s_1 \models R(a, x)$ and $\mathfrak{M}, s_j \not\models R(a, x)$ for $j = 2, 3$, and 4). But,

$$\mathfrak{M}, s \not\models \forall x (R(a, x) \rightarrow R(x, a))$$

since $\mathfrak{M}, s_2 \not\models R(a, x) \rightarrow R(x, a)$ (because $\mathfrak{M}, s_2 \models R(a, x)$ and $\mathfrak{M}, s_2 \not\models R(x, a)$).

For a more complicated case, consider

$$\forall x (R(a, x) \rightarrow \exists y R(x, y)).$$

Since $\mathfrak{M}, s_3 \not\models R(a, x)$ and $\mathfrak{M}, s_4 \not\models R(a, x)$, the interesting cases where we have to worry about the consequent of the conditional are only s_1 and s_2 . Does $\mathfrak{M}, s_1 \models \exists y R(x, y)$ hold? It does if there is at least one y -variant s'_1 of s_1 so that $\mathfrak{M}, s'_1 \models R(x, y)$. In fact, s_1 is such a y -variant ($s_1(x) = 1$, $s_1(y) = 1$, and $\langle 1, 1 \rangle \in R^{\mathfrak{M}}$), so the answer is yes. To determine if $\mathfrak{M}, s_2 \models \exists y R(x, y)$ we have to look at the y -variants of s_2 . Here, s_2 itself does not satisfy $R(x, y)$ ($s_2(x) = 2$, $s_2(y) = 1$, and $\langle 2, 1 \rangle \notin R^{\mathfrak{M}}$). However, consider $s'_2 \sim_y s_2$ with $s'_2(y) = 3$.

$\mathfrak{M}, s'_2 \models R(x, y)$ since $\langle 2, 3 \rangle \in R^{\mathfrak{M}}$, and so $\mathfrak{M}, s_2 \models \exists y R(x, y)$. In sum, for every x -variant s_i of s , either $\mathfrak{M}, s_i \not\models R(a, x)$ ($i = 3, 4$) or $\mathfrak{M}, s_i \models \exists y R(x, y)$ ($i = 1, 2$), and so

$$\mathfrak{M}, s \models \forall x (R(a, x) \rightarrow \exists y R(x, y)).$$

On the other hand,

$$\mathfrak{M}, s \not\models \exists x (R(a, x) \wedge \forall y R(x, y)).$$

The only x -variants s_i of s with $\mathfrak{M}, s_i \models R(a, x)$ are s_1 and s_2 . But for each, there is in turn a y -variant $s'_i \sim_y s_i$ with $s'_i(y) = 4$ so that $\mathfrak{M}, s'_i \not\models R(x, y)$ and so $\mathfrak{M}, s_i \not\models \forall y R(x, y)$ for $i = 1, 2$. In sum, none of the x -variants $s_i \sim_x s$ are such that $\mathfrak{M}, s_i \models R(a, x) \wedge \forall y R(x, y)$.

Problem syn.5. Let $\mathcal{L} = \{c, f, A\}$ with one **constant symbol**, one one-place **function symbol** and one two-place **predicate symbol**, and let the **structure** \mathfrak{M} be given by

1. $|\mathfrak{M}| = \{1, 2, 3\}$
2. $c^{\mathfrak{M}} = 3$
3. $f^{\mathfrak{M}}(1) = 2, f^{\mathfrak{M}}(2) = 3, f^{\mathfrak{M}}(3) = 2$
4. $A^{\mathfrak{M}} = \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 3 \rangle\}$

(a) Let $s(v) = 1$ for all variables v . Find out whether

$$\mathfrak{M}, s \models \exists x (A(f(z), c) \rightarrow \forall y (A(y, x) \vee A(f(y), x)))$$

Explain why or why not.

(b) Give a different structure and variable assignment in which the formula is not satisfied.

syn.12 Variable Assignments

fol:syn:ass:
sec

A **variable** assignment s provides a value for *every* variable—and there are infinitely many of them. This is of course not necessary. We require **variable** assignments to assign values to all **variables** simply because it makes things a lot easier. The value of a term t , and whether or not a **formula** φ is satisfied in a **structure** with respect to s , only depend on the assignments s makes to the **variables** in t and the free **variables** of φ . This is the content of the next two propositions. To make the idea of “depends on” precise, we show that any two variable assignments that agree on all the variables in t give the same value, and that φ is satisfied relative to one iff it is satisfied relative to the other if two variable assignments agree on all free variables of φ .

explanation

fol:syn:ass:
prop:valindep

Proposition syn.36. *If the **variables** in a term t are among x_1, \dots, x_n , and $s_1(x_i) = s_2(x_i)$ for $i = 1, \dots, n$, then $\text{Val}_{s_1}^{\mathfrak{M}}(t) = \text{Val}_{s_2}^{\mathfrak{M}}(t)$.*

Proof. By induction on the complexity of t . For the base case, t can be a **constant symbol** or one of the variables x_1, \dots, x_n . If $t = c$, then $\text{Val}_{s_1}^{\mathfrak{M}}(t) = c^{\mathfrak{M}} = \text{Val}_{s_2}^{\mathfrak{M}}(t)$. If $t = x_i$, $s_1(x_i) = s_2(x_i)$ by the hypothesis of the proposition, and so $\text{Val}_{s_1}^{\mathfrak{M}}(t) = s_1(x_i) = s_2(x_i) = \text{Val}_{s_2}^{\mathfrak{M}}(t)$.

For the inductive step, assume that $t = f(t_1, \dots, t_k)$ and that the claim holds for t_1, \dots, t_k . Then

$$\begin{aligned} \text{Val}_{s_1}^{\mathfrak{M}}(t) &= \text{Val}_{s_1}^{\mathfrak{M}}(f(t_1, \dots, t_k)) = \\ &= f^{\mathfrak{M}}(\text{Val}_{s_1}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s_1}^{\mathfrak{M}}(t_k)) \end{aligned}$$

For $j = 1, \dots, k$, the **variables** of t_j are among x_1, \dots, x_n . So by induction hypothesis, $\text{Val}_{s_1}^{\mathfrak{M}}(t_j) = \text{Val}_{s_2}^{\mathfrak{M}}(t_j)$. So,

$$\begin{aligned} \text{Val}_{s_1}^{\mathfrak{M}}(t) &= \text{Val}_{s_2}^{\mathfrak{M}}(f(t_1, \dots, t_k)) = \\ &= f^{\mathfrak{M}}(\text{Val}_{s_1}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s_1}^{\mathfrak{M}}(t_k)) = \\ &= f^{\mathfrak{M}}(\text{Val}_{s_2}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s_2}^{\mathfrak{M}}(t_k)) = \\ &= \text{Val}_{s_2}^{\mathfrak{M}}(f(t_1, \dots, t_k)) = \text{Val}_{s_2}^{\mathfrak{M}}(t). \end{aligned}$$

□

Proposition syn.37. *If the free **variables** in φ are among x_1, \dots, x_n , and $s_1(x_i) = s_2(x_i)$ for $i = 1, \dots, n$, then $\mathfrak{M}, s_1 \models \varphi$ iff $\mathfrak{M}, s_2 \models \varphi$.*

*fol:syn:ass:
prop:satind*

Proof. We use induction on the complexity of φ . For the base case, where φ is atomic, φ can be: \top , \perp , $R(t_1, \dots, t_k)$ for a k -place predicate R and terms t_1, \dots, t_k , or $t_1 = t_2$ for terms t_1 and t_2 .

1. $\varphi \equiv \top$: both $\mathfrak{M}, s_1 \models \varphi$ and $\mathfrak{M}, s_2 \models \varphi$.
2. $\varphi \equiv \perp$: both $\mathfrak{M}, s_1 \not\models \varphi$ and $\mathfrak{M}, s_2 \not\models \varphi$.
3. $\varphi \equiv R(t_1, \dots, t_k)$: let $\mathfrak{M}, s_1 \models \varphi$. Then

$$\langle \text{Val}_{s_1}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s_1}^{\mathfrak{M}}(t_k) \rangle \in R^{\mathfrak{M}}.$$

For $i = 1, \dots, k$, $\text{Val}_{s_1}^{\mathfrak{M}}(t_i) = \text{Val}_{s_2}^{\mathfrak{M}}(t_i)$ by [Proposition syn.36](#). So we also have $\langle \text{Val}_{s_2}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s_2}^{\mathfrak{M}}(t_k) \rangle \in R^{\mathfrak{M}}$.

4. $\varphi \equiv t_1 = t_2$: suppose $\mathfrak{M}, s_1 \models \varphi$. Then $\text{Val}_{s_1}^{\mathfrak{M}}(t_1) = \text{Val}_{s_1}^{\mathfrak{M}}(t_2)$. So,

$$\begin{aligned} \text{Val}_{s_2}^{\mathfrak{M}}(t_1) &= \text{Val}_{s_1}^{\mathfrak{M}}(t_1) && \text{(by Proposition syn.36)} \\ &= \text{Val}_{s_1}^{\mathfrak{M}}(t_2) && \text{(since } \mathfrak{M}, s_1 \models t_1 = t_2 \text{)} \\ &= \text{Val}_{s_2}^{\mathfrak{M}}(t_2) && \text{(by Proposition syn.36),} \end{aligned}$$

so $\mathfrak{M}, s_2 \models t_1 = t_2$.

Now assume $\mathfrak{M}, s_1 \models \psi$ iff $\mathfrak{M}, s_2 \models \psi$ for all **formulas** ψ less complex than φ . The induction step proceeds by cases determined by the main operator of φ . In each case, we only demonstrate the forward direction of the **biconditional**; the proof of the reverse direction is symmetrical. In all cases except those for the quantifiers, we apply the induction hypothesis to sub-**formulas** ψ of φ . The free variables of ψ are among those of φ . Thus, if s_1 and s_2 agree on the free variables of φ , they also agree on those of ψ , and the induction hypothesis applies to ψ .

1. $\varphi \equiv \neg\psi$: if $\mathfrak{M}, s_1 \models \varphi$, then $\mathfrak{M}, s_1 \not\models \psi$, so by the induction hypothesis, $\mathfrak{M}, s_2 \not\models \psi$, hence $\mathfrak{M}, s_2 \models \varphi$.
2. $\varphi \equiv \psi \wedge \chi$: if $\mathfrak{M}, s_1 \models \varphi$, then $\mathfrak{M}, s_1 \models \psi$ and $\mathfrak{M}, s_1 \models \chi$, so by induction hypothesis, $\mathfrak{M}, s_2 \models \psi$ and $\mathfrak{M}, s_2 \models \chi$. Hence, $\mathfrak{M}, s_2 \models \varphi$.
3. $\varphi \equiv \psi \vee \chi$: if $\mathfrak{M}, s_1 \models \varphi$, then $\mathfrak{M}, s_1 \models \psi$ or $\mathfrak{M}, s_1 \models \chi$. By induction hypothesis, $\mathfrak{M}, s_2 \models \psi$ or $\mathfrak{M}, s_2 \models \chi$, so $\mathfrak{M}, s_2 \models \varphi$.
4. $\varphi \equiv \psi \rightarrow \chi$: if $\mathfrak{M}, s_1 \models \varphi$, then $\mathfrak{M}, s_1 \not\models \psi$ or $\mathfrak{M}, s_1 \models \chi$. By the induction hypothesis, $\mathfrak{M}, s_2 \not\models \psi$ or $\mathfrak{M}, s_2 \models \chi$, so $\mathfrak{M}, s_2 \models \varphi$.
5. $\varphi \equiv \psi \leftrightarrow \chi$: if $\mathfrak{M}, s_1 \models \varphi$, then either $\mathfrak{M}, s_1 \models \psi$ and $\mathfrak{M}, s_1 \models \chi$, or $\mathfrak{M}, s_1 \not\models \psi$ and $\mathfrak{M}, s_1 \not\models \chi$. By the induction hypothesis, either $\mathfrak{M}, s_2 \models \psi$ and $\mathfrak{M}, s_2 \models \chi$ or $\mathfrak{M}, s_2 \not\models \psi$ and $\mathfrak{M}, s_2 \not\models \chi$. In either case, $\mathfrak{M}, s_2 \models \varphi$.
6. $\varphi \equiv \exists x \psi$: if $\mathfrak{M}, s_1 \models \varphi$, there is an x -variant s'_1 of s_1 so that $\mathfrak{M}, s'_1 \models \psi$. Let s'_2 be the x -variant of s_2 that assigns the same thing to x as does s'_1 . The free variables of ψ are among x_1, \dots, x_n , and x . $s'_1(x_i) = s'_2(x_i)$, since s'_1 and s'_2 are x -variants of s_1 and s_2 , respectively, and by hypothesis $s_1(x_i) = s_2(x_i)$. $s'_1(x) = s'_2(x)$ by the way we have defined s'_2 . Then the induction hypothesis applies to ψ and s'_1, s'_2 , so $\mathfrak{M}, s'_2 \models \psi$. Hence, there is an x -variant of s_2 that satisfies ψ , and so $\mathfrak{M}, s_2 \models \varphi$.
7. $\varphi \equiv \forall x \psi$: if $\mathfrak{M}, s_1 \models \varphi$, then for every x -variant s'_1 of s_1 , $\mathfrak{M}, s'_1 \models \psi$. Take an arbitrary x -variant s'_2 of s_2 , let s'_1 be the x -variant of s_1 which assigns the same thing to x as does s'_2 . The free variables of ψ are among x_1, \dots, x_n , and x . $s'_1(x_i) = s'_2(x_i)$, since s'_1 and s'_2 are x -variants of s_1 and s_2 , respectively, and by hypothesis $s_1(x_i) = s_2(x_i)$. $s'_1(x) = s'_2(x)$ by the way we have defined s'_1 . Then the induction hypothesis applies to ψ and s'_1, s_2 , and we have $\mathfrak{M}, s'_2 \models \psi$. Since s'_2 is an arbitrary x -variant of s_2 , every x -variant of s_2 satisfies ψ , and so $\mathfrak{M}, s_2 \models \varphi$.

By induction, we get that $\mathfrak{M}, s_1 \models \varphi$ iff $\mathfrak{M}, s_2 \models \varphi$ whenever the free **variables** in φ are among x_1, \dots, x_n and $s_1(x_i) = s_2(x_i)$ for $i = 1, \dots, n$. \square

Problem syn.6. Complete the proof of [Proposition syn.37](#).

explanation

Sentences have no free variables, so any two variable assignments assign the same things to all the (zero) free variables of any sentence. The proposition just proved then means that whether or not a **sentence** is satisfied in a structure relative to a variable assignment is completely independent of the assignment. We'll record this fact. It justifies the definition of satisfaction of a **sentence** in a **structure** (without mentioning a variable assignment) that follows.

Corollary syn.38. *If φ is a sentence and s a variable assignment, then $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s' \models \varphi$ for every variable assignment s' .*

fol:syn:ass:
cor:sat-sentence

Proof. Let s' be any variable assignment. Since φ is a **sentence**, it has no free variables, and so every variable assignment s' trivially assigns the same things to all free variables of φ as does s . So the condition of **Proposition syn.37** is satisfied, and we have $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s' \models \varphi$. \square

Definition syn.39. If φ is a **sentence**, we say that a **structure** \mathfrak{M} *satisfies* φ , $\mathfrak{M} \models \varphi$, iff $\mathfrak{M}, s \models \varphi$ for all variable assignments s .

fol:syn:ass:
defn:satisfaction

If $\mathfrak{M} \models \varphi$, we also simply say that φ is *true* in \mathfrak{M} .

Proposition syn.40. *Let \mathfrak{M} be a structure, φ be a sentence, and s a variable assignment. $\mathfrak{M} \models \varphi$ iff $\mathfrak{M}, s \models \varphi$.*

fol:syn:ass:
prop:sentence-sat-true

Proof. Exercise. \square

Problem syn.7. Prove **Proposition syn.40**

Proposition syn.41. *Suppose $\varphi(x)$ only contains x free, and \mathfrak{M} is a structure. Then:*

fol:syn:ass:
prop:sat-quant

1. $\mathfrak{M} \models \exists x \varphi(x)$ iff $\mathfrak{M}, s \models \varphi(x)$ for at least one variable assignment s .
2. $\mathfrak{M} \models \forall x \varphi(x)$ iff $\mathfrak{M}, s \models \varphi(x)$ for all variable assignments s .

Proof. Exercise. \square

Problem syn.8. Prove **Proposition syn.41**.

Problem syn.9. Suppose \mathcal{L} is a language without **function symbols**. Given a **structure** \mathfrak{M} , c a **constant symbol** and $a \in |\mathfrak{M}|$, define $\mathfrak{M}[a/c]$ to be the **structure** that is just like \mathfrak{M} , except that $c^{\mathfrak{M}[a/c]} = a$. Define $\mathfrak{M} \models \varphi$ for **sentences** φ by:

1. $\varphi \equiv \perp$: not $\mathfrak{M} \models \varphi$.
2. $\varphi \equiv \top$: $\mathfrak{M} \models \varphi$.
3. $\varphi \equiv R(d_1, \dots, d_n)$: $\mathfrak{M} \models \varphi$ iff $\langle d_1^{\mathfrak{M}}, \dots, d_n^{\mathfrak{M}} \rangle \in R^{\mathfrak{M}}$.
4. $\varphi \equiv d_1 = d_2$: $\mathfrak{M} \models \varphi$ iff $d_1^{\mathfrak{M}} = d_2^{\mathfrak{M}}$.
5. $\varphi \equiv \neg\psi$: $\mathfrak{M} \models \varphi$ iff not $\mathfrak{M} \models \psi$.

6. $\varphi \equiv (\psi \wedge \chi)$: $\mathfrak{M} \models \varphi$ iff $\mathfrak{M} \models \psi$ and $\mathfrak{M} \models \chi$.
7. $\varphi \equiv (\psi \vee \chi)$: $\mathfrak{M} \models \varphi$ iff $\mathfrak{M} \models \psi$ or $\mathfrak{M} \models \chi$ (or both).
8. $\varphi \equiv (\psi \rightarrow \chi)$: $\mathfrak{M} \models \varphi$ iff not $\mathfrak{M} \models \psi$ or $\mathfrak{M} \models \chi$ (or both).
9. $\varphi \equiv (\psi \leftrightarrow \chi)$: $\mathfrak{M} \models \varphi$ iff either both $\mathfrak{M} \models \psi$ and $\mathfrak{M} \models \chi$, or neither $\mathfrak{M} \models \psi$ nor $\mathfrak{M} \models \chi$.
10. $\varphi \equiv \forall x \psi$: $\mathfrak{M} \models \varphi$ iff for all $a \in |\mathfrak{M}|$, $\mathfrak{M}[a/c] \models \psi[c/x]$, if c does not occur in ψ .
11. $\varphi \equiv \exists x \psi$: $\mathfrak{M} \models \varphi$ iff there is an $a \in |\mathfrak{M}|$ such that $\mathfrak{M}[a/c] \models \psi[c/x]$, if c does not occur in ψ .

Let x_1, \dots, x_n be all free variables in φ , c_1, \dots, c_n constant symbols not in φ , $a_1, \dots, a_n \in |\mathfrak{M}|$, and $s(x_i) = a_i$.

Show that $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}[a_1/c_1, \dots, a_n/c_n] \models \varphi[c_1/x_1] \dots [c_n/x_n]$.

(This problem shows that it is possible to give a semantics for first-order logic that makes do without variable assignments.)

Problem syn.10. Suppose that f is a function symbol not in $\varphi(x, y)$. Show that there is a structure \mathfrak{M} such that $\mathfrak{M} \models \forall x \exists y \varphi(x, y)$ iff there is an \mathfrak{M}' such that $\mathfrak{M}' \models \forall x \varphi(x, f(x))$.

(This problem is a special case of what's known as Skolem's Theorem; $\forall x \varphi(x, f(x))$ is called a *Skolem normal form* of $\forall x \exists y \varphi(x, y)$.)

syn.13 Extensionality

fol:syn:ext:
sec

Extensionality, sometimes called relevance, can be expressed informally as follows: the only factors that bears upon the satisfaction of formula φ in a structure \mathfrak{M} relative to a variable assignment s , are the size of the domain and the assignments made by \mathfrak{M} and s to the elements of the language that actually appear in φ .

One immediate consequence of extensionality is that where two structures \mathfrak{M} and \mathfrak{M}' agree on all the elements of the language appearing in a sentence φ and have the same domain, \mathfrak{M} and \mathfrak{M}' must also agree on whether or not φ itself is true.

fol:syn:ext:
prop:extensionality

Proposition syn.42 (Extensionality). *Let φ be a formula, and \mathfrak{M}_1 and \mathfrak{M}_2 be structures with $|\mathfrak{M}_1| = |\mathfrak{M}_2|$, and s a variable assignment on $|\mathfrak{M}_1| = |\mathfrak{M}_2|$. If $c^{\mathfrak{M}_1} = c^{\mathfrak{M}_2}$, $R^{\mathfrak{M}_1} = R^{\mathfrak{M}_2}$, and $f^{\mathfrak{M}_1} = f^{\mathfrak{M}_2}$ for every constant symbol c , relation symbol R , and function symbol f occurring in φ , then $\mathfrak{M}_1, s \models \varphi$ iff $\mathfrak{M}_2, s \models \varphi$.*

Proof. First prove (by induction on t) that for every term, $\text{Val}_s^{\mathfrak{M}_1}(t) = \text{Val}_s^{\mathfrak{M}_2}(t)$. Then prove the proposition by induction on φ , making use of the claim just proved for the induction basis (where φ is atomic). \square

Problem syn.11. Carry out the proof of [Proposition syn.42](#) in detail.

Corollary syn.43 (Extensionality for Sentences). *Let φ be a sentence and $\mathfrak{M}_1, \mathfrak{M}_2$ as in [Proposition syn.42](#). Then $\mathfrak{M}_1 \models \varphi$ iff $\mathfrak{M}_2 \models \varphi$.* fol:syn:ext:
cor:extensionality-sent

Proof. Follows from [Proposition syn.42](#) by [Corollary syn.38](#). □

Moreover, the value of a term, and whether or not a structure satisfies a formula, only depends on the values of its subterms.

Proposition syn.44. *Let \mathfrak{M} be a structure, t and t' terms, and s a variable assignment. Let $s' \sim_x s$ be the x -variant of s given by $s'(x) = \text{Val}_s^{\mathfrak{M}}(t')$. Then $\text{Val}_s^{\mathfrak{M}}(t[t'/x]) = \text{Val}_{s'}^{\mathfrak{M}}(t)$.* fol:syn:ext:
prop:ext-terms

Proof. By induction on t .

1. If t is a constant, say, $t \equiv c$, then $t[t'/x] = c$, and $\text{Val}_s^{\mathfrak{M}}(c) = c^{\mathfrak{M}} = \text{Val}_{s'}^{\mathfrak{M}}(c)$.
2. If t is a variable other than x , say, $t \equiv y$, then $t[t'/x] = y$, and $\text{Val}_s^{\mathfrak{M}}(y) = \text{Val}_{s'}^{\mathfrak{M}}(y)$ since $s' \sim_x s$.
3. If $t \equiv x$, then $t[t'/x] = t'$. But $\text{Val}_{s'}^{\mathfrak{M}}(x) = \text{Val}_s^{\mathfrak{M}}(t')$ by definition of s' .
4. If $t \equiv f(t_1, \dots, t_n)$ then we have:

$$\begin{aligned}
 \text{Val}_s^{\mathfrak{M}}(t[t'/x]) &= \\
 &= \text{Val}_s^{\mathfrak{M}}(f(t_1[t'/x], \dots, t_n[t'/x])) \\
 &\quad \text{by definition of } t[t'/x] \\
 &= f^{\mathfrak{M}}(\text{Val}_s^{\mathfrak{M}}(t_1[t'/x]), \dots, \text{Val}_s^{\mathfrak{M}}(t_n[t'/x])) \\
 &\quad \text{by definition of } \text{Val}_s^{\mathfrak{M}}(f(\dots)) \\
 &= f^{\mathfrak{M}}(\text{Val}_{s'}^{\mathfrak{M}}(t_1), \dots, \text{Val}_{s'}^{\mathfrak{M}}(t_n)) \\
 &\quad \text{by induction hypothesis} \\
 &= \text{Val}_{s'}^{\mathfrak{M}}(t) \text{ by definition of } \text{Val}_{s'}^{\mathfrak{M}}(f(\dots))
 \end{aligned}$$

□

Proposition syn.45. *Let \mathfrak{M} be a structure, φ a formula, t a term, and s a variable assignment. Let $s' \sim_x s$ be the x -variant of s given by $s'(x) = \text{Val}_s^{\mathfrak{M}}(t)$. Then $\mathfrak{M}, s \models \varphi[t/x]$ iff $\mathfrak{M}, s' \models \varphi$.* fol:syn:ext:
prop:ext-formulas

Proof. Exercise. □

Problem syn.12. Prove [Proposition syn.45](#)

syn.14 Semantic Notions

fol:syn:sem:
sec

Give the definition of **structures** for first-order languages, we can define some basic semantic properties of and relationships between sentences. The simplest of these is the notion of *validity* of a sentence. A sentence is valid if it is satisfied in every **structure**. Valid sentences are those that are satisfied regardless of how the non-logical symbols in it are interpreted. Valid sentences are therefore also called *logical truths*—they are true, i.e., satisfied, in any **structure** and hence their truth depends only on the logical symbols occurring in them and their syntactic **structure**, but not on the non-logical symbols or their interpretation.

explanation

Definition syn.46 (Validity). A sentence φ is *valid*, $\models \varphi$, iff $\mathfrak{M} \models \varphi$ for every **structure** \mathfrak{M} .

Definition syn.47 (Entailment). A set of sentences Γ *entails* a sentence φ , $\Gamma \models \varphi$, iff for every **structure** \mathfrak{M} with $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$.

Definition syn.48 (Satisfiability). A set of sentences Γ is *satisfiable* if $\mathfrak{M} \models \Gamma$ for some **structure** \mathfrak{M} . If Γ is not satisfiable it is called *unsatisfiable*.

Proposition syn.49. *A sentence φ is valid iff $\Gamma \models \varphi$ for every set of sentences Γ .*

Proof. For the forward direction, let φ be valid, and let Γ be a set of sentences. Let \mathfrak{M} be a **structure** so that $\mathfrak{M} \models \Gamma$. Since φ is valid, $\mathfrak{M} \models \varphi$, hence $\Gamma \models \varphi$.

For the contrapositive of the reverse direction, let φ be invalid, so there is a **structure** \mathfrak{M} with $\mathfrak{M} \not\models \varphi$. When $\Gamma = \{\top\}$, since \top is valid, $\mathfrak{M} \models \Gamma$. Hence, there is a **structure** \mathfrak{M} so that $\mathfrak{M} \models \Gamma$ but $\mathfrak{M} \not\models \varphi$, hence Γ does not entail φ . \square

fol:syn:sem:
prop:entails-unsat

Proposition syn.50. *$\Gamma \models \varphi$ iff $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable.*

Proof. For the forward direction, suppose $\Gamma \models \varphi$ and suppose to the contrary that there is a **structure** \mathfrak{M} so that $\mathfrak{M} \models \Gamma \cup \{\neg\varphi\}$. Since $\mathfrak{M} \models \Gamma$ and $\Gamma \models \varphi$, $\mathfrak{M} \models \varphi$. Also, since $\mathfrak{M} \models \Gamma \cup \{\neg\varphi\}$, $\mathfrak{M} \models \neg\varphi$, so we have both $\mathfrak{M} \models \varphi$ and $\mathfrak{M} \models \neg\varphi$, a contradiction. Hence, there can be no such **structure** \mathfrak{M} , so $\Gamma \cup \{\varphi\}$ is unsatisfiable.

For the reverse direction, suppose $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable. So for every **structure** \mathfrak{M} , either $\mathfrak{M} \not\models \Gamma$ or $\mathfrak{M} \models \varphi$. Hence, for every **structure** \mathfrak{M} with $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$, so $\Gamma \models \varphi$. \square

Problem syn.13. 1. Show that $\Gamma \models \perp$ iff Γ is unsatisfiable.

2. Show that $\Gamma \cup \{\varphi\} \models \perp$ iff $\Gamma \models \neg\varphi$.

3. Suppose c does not occur in φ or Γ . Show that $\Gamma \models \forall x \varphi$ iff $\Gamma \models \varphi[c/x]$.

Proposition syn.51. *If $\Gamma \subseteq \Gamma'$ and $\Gamma \models \varphi$, then $\Gamma' \models \varphi$.*

Proof. Suppose that $\Gamma \subseteq \Gamma'$ and $\Gamma \models \varphi$. Let \mathfrak{M} be such that $\mathfrak{M} \models \Gamma'$; then $\mathfrak{M} \models \Gamma$, and since $\Gamma \models \varphi$, we get that $\mathfrak{M} \models \varphi$. Hence, whenever $\mathfrak{M} \models \Gamma'$, $\mathfrak{M} \models \varphi$, so $\Gamma' \models \varphi$. \square

Theorem syn.52 (Semantic Deduction Theorem). $\Gamma \cup \{\varphi\} \models \psi$ iff $\Gamma \models \varphi \rightarrow \psi$.

fol:syn:sem:
thm:sem-deduction

Proof. For the forward direction, let $\Gamma \cup \{\varphi\} \models \psi$ and let \mathfrak{M} be a **structure** so that $\mathfrak{M} \models \Gamma$. If $\mathfrak{M} \models \varphi$, then $\mathfrak{M} \models \Gamma \cup \{\varphi\}$, so since $\Gamma \cup \{\varphi\}$ entails ψ , we get $\mathfrak{M} \models \psi$. Therefore, $\mathfrak{M} \models \varphi \rightarrow \psi$, so $\Gamma \models \varphi \rightarrow \psi$.

For the reverse direction, let $\Gamma \models \varphi \rightarrow \psi$ and \mathfrak{M} be a **structure** so that $\mathfrak{M} \models \Gamma \cup \{\varphi\}$. Then $\mathfrak{M} \models \Gamma$, so $\mathfrak{M} \models \varphi \rightarrow \psi$, and since $\mathfrak{M} \models \varphi$, $\mathfrak{M} \models \psi$. Hence, whenever $\mathfrak{M} \models \Gamma \cup \{\varphi\}$, $\mathfrak{M} \models \psi$, so $\Gamma \cup \{\varphi\} \models \psi$. \square

Proposition syn.53. Let \mathfrak{M} be a **structure**, and $\varphi(x)$ a **formula** with one free variable x , and t a closed term. Then:

fol:syn:sem:
prop:quant-terms

1. $\varphi(t) \models \exists x \varphi(x)$
2. $\forall x \varphi(x) \models \varphi(t)$

Proof. 1. Suppose $\mathfrak{M} \models \varphi(t)$. Let s be a variable assignment with $s(x) = \text{Val}^{\mathfrak{M}}(t)$. Then $\mathfrak{M}, s \models \varphi(t)$ since $\varphi(t)$ is a **sentence**. By **Proposition syn.45**, $\mathfrak{M}, s \models \varphi(x)$. By **Proposition syn.41**, $\mathfrak{M} \models \exists x \varphi(x)$.

2. Suppose $\mathfrak{M} \models \forall x \varphi(x)$. Let s be a variable assignment with $s(x) = \text{Val}^{\mathfrak{M}}(t)$. By **Proposition syn.41**, $\mathfrak{M}, s \models \varphi(x)$. By **Proposition syn.45**, $\mathfrak{M}, s \models \varphi(t)$. By **Proposition syn.40**, $\mathfrak{M} \models \varphi(t)$ since $\varphi(t)$ is a **sentence**. \square

Problem syn.14. Complete the proof of **Proposition syn.53**.

Photo Credits

Bibliography