Leary & Kristiansen, *A Friendly Introduction to Mathematical Logic*

Danny Nygård Hansen

21st December 2022

1 • Structures and Languages

Languages: A **first-order language** $\mathcal{L}$ is actually an (infinite) alphabet consisting of the usual symbols. Notice that we have a *countable* number of variables. These variables are common to all first-order languages, so we use the symbol *Vars*, which does not depend on which language we are considering, to denote the set of variables.

On the other hand we can have an arbitrary number of constant, function and relation symbols. I am not sure if the number of these symbols makes any difference to the theory. Instead of having a separate category for constants we may consider a constant as a function with arity 0. The arity of function symbols may, as in universal algebra, be formalised as a partial function $\rho\colon \mathcal{L} \nrightarrow \mathbb{N}$, whose domain is the set of function and relation symbols.

Terms: Strings can be terms of a particular first-order language $\mathcal{L}$. We think of terms as picking out an element of the universe under consideration. These are obtained either as variables or as the images of functions in $\mathcal{L}$ (or as constants if these are considered a class of their own).

Formulas: As with terms, strings can be formulas of a particular first-order language $\mathcal{L}$. Formulas are supposed to represent propositions about the objects in the universe. We may first construct formulas from $\mathcal{L}$-terms by applying relations (e.g. equality) to a list of terms. Afterwards we may apply either quantifiers or the usual connectives from propositional logic to the resulting formulas. Notice that the particular language $\mathcal{L}$ only determines which terms are allowed and does not affect how these terms can be used to construct formulas.

A formula is said to be **atomic** if it is on the form $= t_1 t_2$ or $R t_1 \cdots t_n$ for an n-ary relation symbol R and terms $t_1, \dots, t_n$.

We say that a string ψ is a **subformula** of a formula φ if ψ itself is a formula, and if it is a substring of φ. By unique readability for formulas (Exercise 1.4.8) each formula φ that is not atomic can be broken up into smaller formulas in a unique way. These give rise to a (unique) construction tree for φ, and the subformulas are precisely the formulas that appear on this tree.

L&K call the symbol $\forall$ itself a quantifier, but by **quantifier** we will mean $\forall v$ for a variable v. The **scope** of a quantifier $\forall v$ occurring in a formula $(\forall v)(\alpha)$ is the subformula α.

Free variables: We define recursively what it means for a variable v to be **free** in a formula φ. Alternatively we may simply say that v is free in φ if it is not in the scope of any quantifier $\forall v$ occurring in φ.

This also makes it easy to define what it means for an **occurrence** of a variable v in φ to be free: Namely that this occurrence does not lie in the scope of any quantifier $\forall v$.

Sentences: A **sentence** in $\mathcal{L}$ is a formula of $\mathcal{L}$ with no free variables.

Structures: An **$\mathcal{L}$-structure** $\mathfrak{A}$ consists of a nonempty set A, the **universe** of $\mathfrak{A}$, along with functions $f^{\mathfrak{A}} \colon A^{\rho(f)} \to A$ and relations $R^{\mathfrak{A}} \in A^{\rho(R)}$ for each function symbol f and relation symbol R in $\mathcal{L}$. The particular language $\mathcal{L}$ does not place any constraints on the universe A.

Compare this to universal algebra: Here we usually specify a collection $\mathcal{F}$ of operation symbols and a type $\rho \colon \mathcal{F} \to \mathbb{N}$ specifying the arity of each symbol. A structure $\mathfrak{A}$ of type ρ is then a nonempty set A along with $\rho(f)$-ary operations $f^{\mathfrak{A}}$ on A for each $f \in \mathcal{F}$. In first-order logic we also allow relations, but otherwise the definition of a structure is basically the same.

Variable assignment functions: A **(variable) assignment function** into an $\mathcal{L}$-structure $\mathfrak{A}$ is a function *Vars* $\to A$. This is the first step towards connecting the syntax and semantics of $\mathcal{L}$. Notice that regardless of the particular structure or language, it is always the variables in *Vars* that we need to assign values to. The codomain A however is determined by the structure $\mathfrak{A}$ in question, which itself depends on the language $\mathcal{L}$ (though A does not as such depend on $\mathcal{L}$).

Given $s \colon \mathit{Vars} \to A$, a variable x and a particular value $a \in A$, we define an **x-modification** of s denoted $s[x \mid a]$ with $s[x \mid a](x) = a$.

Term assignment function: Given $s \colon \mathit{Vars} \to A$ we define the **term assignment function** $\bar{s}$ generated by s. This is defined by recursion on terms, and it is clearly the unique extension s' of s to all terms satisfying that

$$s'(f t_1 \cdots t_n) = f^{\mathfrak{A}}(s'(t_1), \ldots, s'(t_n))$$

for all function symbols f and all terms $t_1, \ldots, t_n$. Indeed, Lemma 1.7.6 shows that only the variables occurring in a particular term t determines the value of $\bar{s}$ in t.

Satisfaction: We define what it means for an $\mathcal{L}$-structure $\mathfrak{A}$ to **satisfy** an $\mathcal{L}$-formula φ with assignment $s\colon \mathit{Vars} \to A$ recursively. In the affirmative case we write $\mathfrak{A} \vDash_s \varphi$.

The base cases are the atomic formulas. This completes the connection between the syntax and semantics of $\mathcal{L}$: We have now effectively associated a truth value to each $\mathcal{L}$-formula, provided that we are given an interpretation of all the symbols in such a formula, namely function symbols (i.e. $\mathfrak{A}$) and variables (i.e. s).

Notice that the definition of satisfaction is *informal*, even if it is precise. Clearly we cannot hope to define the semantics of $\mathcal{L}$ in terms of $\mathcal{L}$ itself, so we must resort to a definition in natural language using terms like 'not', 'or' and 'for all'.

Proposition 1.7.7 shows that satisfaction of a formula φ only depends on the free variables occurring in φ.

Models and validity: An $\mathcal{L}$-structure $\mathfrak{A}$ is a **model** of an $\mathcal{L}$-formula φ, written $\mathfrak{A} \vDash \varphi$, if $\mathfrak{A} \vDash_s \varphi$ for all assignment functions s. In in turn $\mathfrak{A} \vDash \varphi$ for all $\mathcal{L}$-structures $\mathfrak{A}$, then we say that φ is **valid** and write $\vDash \varphi$.

If σ is a sentence in $\mathcal{L}$, then $\mathfrak{A} \vDash \sigma$ if and only if $\mathfrak{A} \vDash_s \sigma$ for *any* s, in which case we say that σ is **true in $\mathfrak{A}$**.

Substitutions: We define recursively how to substitute a term in place of a variable, first in terms and then in formulas. L&K use the notation u_t^x for the term u with every occurrence of x replaced with the term t, but the notation $u[t/x]$ is also common. We similarly write φ_t^x for a formula φ.

If φ is a formula, t a term and x a variable, then we define when t is **substitutable** for x in φ. We do this by recursion, and the case when φ is on the form $(\forall y)(\alpha)$ is the interesting one. In this case t is substitutable for x if either of the following holds:

(a) x is not free in φ, which includes the case when $y = x$. In this case every occurrence of x (as a variable and not in a quantifier) in φ lies in the scope of a quantifier $\forall x$. But when performing the substitution φ_t^x the relevant subformula $(\forall x)(\beta)$ is left untouched. Hence there is no danger of making a substitution in a problemating place, since we aren't making any substitutions!

(b) y does not occur in t, and t is substitutable for x in α. A problem could happen if the former is not satisfied, since any free occurrence of y would be bound by the quantifier, which is not desirable. The latter condition is of course necessary.

Logical implication: If Γ and Δ are sets of $\mathcal{L}$-formulas, then we say that Γ **logically implies** Δ, denoted $\Gamma \vDash \Delta$, if $\mathfrak{A} \vDash \Gamma$ implies $\mathfrak{A} \vDash \Delta$ for all $\mathcal{L}$-structures $\mathfrak{A}$.

Note the asymmetry in the roles played by structures and variable assignment functions in the definition of logical implication: It is *not* the case that $\Gamma \vDash \Delta$ means that

$$\mathfrak{A} \vDash_s \Gamma \quad \text{implies} \quad \mathfrak{A} \vDash_s \Delta$$

for all $\mathcal{L}$-structures $\mathfrak{A}$ and variable assignment functions s into $\mathfrak{A}$. This would require us to assign the same values to variables in Γ and in Δ, but this is not needed. However, it also does not mean that

$$\mathfrak{A} \vDash_s \Gamma \quad \text{implies} \quad \mathfrak{A} \vDash_t \Delta$$

for all $\mathcal{L}$-structures $\mathfrak{A}$ and variable assignment functions s, t into $\mathfrak{A}$. This says that we require $\mathfrak{A} \vDash_t \Delta$ to hold for all t as soon as $\mathfrak{A} \vDash_s \Gamma$ holds for *some* s. But we only require $\mathfrak{A} \vDash_t \Delta$ when we assume $\mathfrak{A} \vDash_s \Gamma$ for *all* s. That is, when proving $\mathfrak{A} \vDash_t \Delta$ we may construct whatever s we wish.

If $\emptyset \vDash \varphi$, then we say that φ is **valid** and write $\vDash \varphi$. This agrees with the definition of validity above.

Exercise 1.9.3

Suppose that φ is an $\mathcal{L}$-formula and x is a variable. Prove that φ is valid if and only if $(\forall x)(\varphi)$ is valid. Thus, if φ has free variables x, y, and z, φ will be valid if and only if $\forall x \forall y \forall z \varphi$ is valid. The sentence $\forall x \forall y \forall z \varphi$ is called the **universal closure** of φ.

Solution. Notice that $\vDash (\forall x)(\varphi)$ if and only if $\mathfrak{A} \vDash_s (\forall x)(\varphi)$ for all $\mathcal{L}$-structures $\mathfrak{A}$ and variable assignment functions s into $\mathfrak{A}$. This is the case if and only if $\mathfrak{A} \vDash_{s[x|a]} \varphi$ for all $\mathfrak{A}$, s and elements a in the universe A. If φ is valid then this follows.

Conversely, notice that $s = s[x \mid s(x)]$, so every variable assignment function into $\mathfrak{A}$ is on the form $s[x \mid a]$ for some s and a in A. Hence if $(\forall x)(\varphi)$ is valid, then the above implies that $\mathfrak{A} \vDash_s \varphi$ for all $\mathfrak{A}$ and s. Thus $\mathfrak{A} \vDash \varphi$ for all $\mathfrak{A}$, so $\vDash \varphi$. □

Exercise 1.9.4

(a) Assume that $\vDash (\varphi \to \psi)$. Show that $\varphi \vDash \psi$.

(b) Suppose that φ is $x < y$ and ψ is $z < w$. Show that $\varphi \vDash \psi$ but $\nvDash (\varphi \to \psi)$.

SOLUTION. (a) Let $\mathfrak{A}$ be a structure such that $\mathfrak{A} \vDash \varphi$, and let s be any variable assignment function into $\mathfrak{A}$. In particular we thus have $\mathfrak{A} \vDash_s \varphi$. Furthermore, $\mathfrak{A} \vDash_s (\varphi \to \psi)$, which is the case if and only if either $\mathfrak{A} \nvDash_s \varphi$ or $\mathfrak{A} \vDash_s \psi$. The former is impossible, so the latter holds. Since s was arbitrary we have $\mathfrak{A} \vDash \psi$, so $\varphi \vDash \psi$ as desired.

(b) We first show that $\varphi \vDash \psi$, so let $\mathfrak{A}$ be a structure such that $\mathfrak{A} \vDash \varphi$. We must then show that $\mathfrak{A} \vDash \psi$. If $<^{\mathfrak{A}} = A \times A$ then this is obvious. Assume thus that there exist $a, b \in A$ such that $(a,b) \notin <^{\mathfrak{A}}$, i.e. such that $a <^{\mathfrak{A}} b$ is false. Let s be a variable assignment function into $\mathfrak{A}$ with $s(x) = a$ and $s(y) = b$. Then $\mathfrak{A} \nvDash_s \varphi$, so nothing has to be proved. This shows that $\varphi \vDash \psi$.

We next show that $\mathfrak{N} \nvDash (\varphi \to \psi)$. Let r be a variable assignment function into $\mathfrak{N}$ with $r(x) = 0$, $r(y) = 1$, $r(z) = 1$, and $r(w) = 0$. Then $\mathfrak{N} \vDash_r \varphi$ but $\mathfrak{N} \nvDash_r \psi$, so $\mathfrak{N} \nvDash_r (\varphi \to \psi)$ as desired. □

2 • Deductions

Deductions: We fix a language $\mathcal{L}$, a set Λ of *logical axioms*, and a set of ordered pairs (Γ, φ) called *rules of inference*. We further have a set Σ of *nonlogical axioms*, whose content depends on the application. A *deduction from* Σ is then a finite sequence of $\mathcal{L}$-formulas such that each formula is either an axiom or follows from the previous formulas by the rules of inference. If there exists a deduction from Σ whose last element is φ, then this deduction is called a *deduction from* Σ *of* φ, and we write $\Sigma \vdash \varphi$.

We denote by Thm_Σ the set of formulas φ such that $\Sigma \vdash \varphi$. This is clearly the smallest set of formulas containing the axioms that is closed under application of the rules of inference.

Logical axioms: The logical axioms include:

(E_1) Reflexivity of variables, i.e. $x = x$ for every variable x.

(E_2) Substitution in function symbols: If two variables are the same, we can substitute them as arguments to any function symbol.

(E_3) Substitution in relation symbols: Same as above, but for relation symbols.

For instance, for each n-ary function symbol f in $\mathcal{L}$ and each collection $x_1, \ldots, x_n, y_1, \ldots, x_n$ of variables we get an axiom of the second kind above. Hence even if there is only one function symbol in $\mathcal{L}$, as long as this as positive arity there are infinitely many axioms of this kind: If f is unary, then we have an axiom

$$v_i = v_j \to f(v_i) = f(v_j)$$

for all $i, j \in \mathbb{Z}_+$. Hence we already have a potentially massive collection of axioms.

We also have axioms for quantifiers. For every variable x, term t and formula φ such that t is substitutable for x in φ we have the following:

(Q_1) Universal instantiation: $(\forall x\varphi) \to \varphi_t^x$.

(Q_2) Existential generalisation: $\varphi_t^x \to (\exists x\varphi)$.

Propositional consequence: Consider the restricted language $\mathcal{P}$ only containing a set of propositional variables and the logical connectives. Formulas of $\mathcal{P}$ are given recursively in the usual way, and the semantics of $\mathcal{P}$ is that of standard propositional logic. In particular, a formula of $\mathcal{P}$ is a *tautology* if every assignment of truth values to the propositional variables makes it true.

Given an $\mathcal{L}$-formula φ, we convert it to a $\mathcal{P}$-formula φ_P as follows:

(a) Systematically replace with propositional variables every occurrence of subformulas on the form $(\forall x)(\alpha)$ that are not in the scope of another quantifier.

(b) Systematically replace in a systematic way with propositional variables every remaining atomic subformula.

By ‘systematically’ we mean that multiple occurrences of the same subformula is replaced with the same propositional variable.

If Γ_P is a set of $\mathcal{P}$-formulas and φ_P is a $\mathcal{P}$-formula, then φ_P is a *propositional consequence* of Γ_P if φ_P is true under every truth assignment making every formula in Γ_P true.

If Γ is a *finite* set of $\mathcal{L}$-formulas and φ is an $\mathcal{L}$-formula, then φ is a *propositional consequence* of Γ if φ_P is a propositional consequence of Γ_P. I’m not sure exactly how significant the finiteness condition is. Perhaps it has to do with us having to go through every formula in Γ and convert each to a propositional formula? Perhaps the corresponding rules of inference will not be decidable.

Rules of inference: We first have the following:

(PC) If an $\mathcal{L}$-formula φ is a propositional consequence of a finite set Γ of $\mathcal{L}$-formulas, then (Γ, φ) is a rule of inference.

We also have the rules

$$\big(\{\psi \to \varphi\}, \psi \to (\forall x)(\varphi)\big) \quad \text{and} \quad \big(\{\varphi \to \psi\}, (\exists x)(\varphi) \to \psi\big)$$

of type (QR), where x is not free in ψ. This assumption is supposed to formalise that we make no assumptions about x. Intuitively, if x occurs in ψ then we may simply make the substitution ψ_y^x, where $y \neq x$ is a variable that does not occur in ψ. If we can then somehow prove a formula φ in which x is free, then this x cannot have come from ψ. Hence φ must hold whatever x is.

The second rule says that if x somehow 'disappears' when proving ψ, then it didn't matter what x was. Hence it's enough that there exists some x.

Soundness: If $\Sigma \vdash \varphi$, then $\Sigma \vDash \varphi$. The proof goes through the following steps:

(a) *The logical axioms are valid*: The equality axioms are easy, and the quantifier axioms are clear enough but require a technical lemma (Theorem 2.6.2).

(b) *If* (Γ, θ) *is a rule of inference, then* $\Gamma \vDash \theta$: For rules of type (PC) this follows by applying Exercise 2.4.3, then Exercise 2.4.6, and finally Exercise 1.9.4. For the rules of type (QR), this is shown by explicitly considering structures, applying Proposition 1.7.7 to make use of free variables (here we use the assumption that x is not free in ψ).

(c) *Proof of soundness*: We show that $\mathrm{Thm}_\Sigma \subseteq C := \{\varphi \mid \Sigma \vDash \varphi\}$ by induction, showing that C contains both Σ and Λ, and that it is closed under application of any rule of inference (Γ, θ). The former is obvious.

For the latter, assume that $\Gamma \subseteq C$, so that $\Sigma \vDash \Gamma$. For any structure $\mathfrak{A}$ with $\mathfrak{A} \vDash \Sigma$ we thus have $\mathfrak{A} \vDash \Gamma$. Since $\Gamma \vDash \theta$ it thus follows that $\mathfrak{A} \vDash \theta$.

Exercise 2.4.3

Prove Lemma 2.4.2: If $\Gamma_P = \{(\gamma_1)_P, (\gamma_2)_P, \dots, (\gamma_n)_P\}$ is a nonempty finite set of propositional formulas and φ_P is a propositional formula, then φ_P is a propositional consequence of Γ_P if and only if

$$\Big[(\gamma_1)_P \wedge (\gamma_2)_P \wedge \ldots \wedge (\gamma_n)_P\Big] \to \varphi_P$$

is a tautology.

Solution. Obvious, since the above formula is true if and only if either all $(\gamma_i)_P$ and φ_P are true, or if at least one $(\gamma_i)_P$ is false. □

Exercise 2.4.6

Prove that if θ is not valid, then θ_P is not a tautology. Deduce that if θ_P is a tautology, then θ is valid.

Solution. We prove this by induction on the (propositional) complexity of θ (i.e. the number of connectives in θ). We in fact prove the following stronger claim:

If θ is not valid, then θ_P is not a tautology. And if $\mathfrak{A} \vDash_s \theta$ for some structure $\mathfrak{A}$ and assignment function s, then θ_P is not a contradiction.

If θ is atomic or on the form $(\forall x)(\alpha)$ then this is obvious by the definition of θ_P. Otherwise assume that the above holds for all formulas with smaller complexity than θ. Notice that θ is not valid if and only if $\mathfrak{A} \nvDash_s \theta$ for some $\mathfrak{A}$ and s.

$\theta :\equiv (\neg\alpha)$: First assume that θ is not valid. Notice that $\mathfrak{A} \nvDash_s \theta$ implies that $\mathfrak{A} \vDash_s \alpha$, so by induction α_P is not a contradiction. Hence $\theta_P = (\neg\alpha_P)$ is not a tautology.

On the other hand, if $\mathfrak{A} \vDash_s \theta$ then $\mathfrak{A} \nvDash_s \alpha$. Hence α is not valid, so by induction α_P is not a tautology. Thus θ_P is not a contradiction.

$\theta :\equiv (\alpha \vee \beta)$: Assume that θ is not valid, so that $\mathfrak{A} \nvDash_s \theta$. Then both $\mathfrak{A} \nvDash_s \alpha$ and $\mathfrak{A} \nvDash_s \beta$, i.e. neither α nor β is valid. It follows by induction that α_P and β_P are not tautologies, so neither is $\theta_P :\equiv (\alpha_P \vee \beta_P)$.

Next, if $\mathfrak{A} \vDash_s \theta$ then either $\mathfrak{A} \vDash_s \alpha$ or $\mathfrak{A} \vDash_s \beta$. Without loss of generality assume the former. By induction, α_P is then not a contradiction, so neither is θ_P. □

EXERCISE 2.7.4

Suppose that η is a sentence. Prove that $\Sigma \vdash \eta$ if and only if $\Sigma \cup (\neg\eta) \vdash \bot$. Notice that this exercise tells us that our deductive system allows us to do proofs by contradiction.

Here $\bot$ denotes the contradiction $[(\forall x)(x = x)] \wedge \neg[(\forall x)(x = x)]$.

SOLUTION. First assume that $\Sigma \vdash \eta$. Notice then that $\Sigma \cup (\neg\eta) \vdash \eta$, and we also clearly have $\Sigma \cup (\neg\eta) \vdash (\neg\eta)$. Hence $\Sigma \cup (\neg\eta) \vdash \{\eta, (\neg\eta)\}$. But notice that $\{\eta, (\neg\eta)\} \vdash \bot$ by (PC) since no truth assignment makes $\{\eta_P, (\neg\eta_P)\}$ true. This implies that $\Sigma \cup (\neg\eta) \vdash \bot$ as desired.

Conversely assume that $\Sigma \cup (\neg\eta) \vdash \bot$. But $\{\bot\} \vdash \eta$ since any formula is a propositional consequence of $\{\bot\}$. Hence $\Sigma \vdash \eta$. □

3 • Completeness and Compactness

Completeness

Reducing to a weaker statement: To prove that $\Sigma \vDash \varphi$ implies $\Sigma \vdash \varphi$, we first show that it suffices to prove a seemingly weaker statement:

(a) In $\Sigma \vdash \varphi$ we may assume that φ is a sentence, since by Lemma 2.7.2, $\Sigma \vdash \varphi$ if and only if $\Sigma \vdash \varphi'$, where φ' is the universal closure of φ.

The same is true for $\Sigma \vDash \varphi$: Here we must show that this implies $\Sigma \vDash \varphi'$. It suffices to show that $\Sigma \vDash (\forall x)(\varphi)$, i.e. where we add a single quantifier. Let $\mathfrak{A}$ be a structure such that $\mathfrak{A} \vDash \Sigma$. We must then show that $\mathfrak{A} \vDash_s (\forall x)(\varphi)$ for all assignment functions s, i.e. that $\mathfrak{A} \vDash_{s[x|a]} \varphi$ for all $a \in A$. But this is clear since $\mathfrak{A} \vDash \varphi$.

(b) By Lemma 2.7.3 we may in $\Sigma \vdash \varphi$ also assume that every element of Σ is a sentence: Let Σ' be the set of universal closures of elements in Σ. Then we claim that if $\Sigma' \vdash \varphi$, then $\Sigma \vdash \varphi$. For any derivation D from Σ', say of the formula φ, let Σ'_D be the subset of Σ' of all elements from Σ' appearing in D. Then D is also a derivation from Σ'_D to φ, so $\Sigma'_D \vdash \varphi$. But Σ'_D is finite since D is, so the lemma implies that also $\Sigma_D \vdash \varphi$, where Σ_D is the set of elements from Σ whose universal closures are the elements in Σ'_D.[1] Furthermore, $\Sigma_D \subseteq \Sigma$, so $\Sigma \vdash \Sigma_D$. It follows that $\Sigma \vdash \varphi$ as claimed.

Again the same is true for $\Sigma \vDash \varphi$, namely that this implies that $\Sigma' \vDash \varphi$. It suffices to show that $\Sigma' \vDash \Sigma$, and it further suffices to show that $\{\sigma'\} \vDash \sigma$ for all $\sigma \in \Sigma$. But this is obvious.

That is, we have proved the following implications:

$$\Sigma \vDash \varphi \quad \Rightarrow \quad \Sigma' \vDash \varphi', \quad \text{and} \quad \Sigma' \vdash \varphi' \quad \Rightarrow \quad \Sigma \vdash \varphi.$$

Hence for completeness it suffices to show that $\Sigma' \vDash \varphi'$ implies $\Sigma' \vdash \varphi'$. We may make the following further simplification:

(c) It suffices to consider the case where φ is the sentence $\bot$: For assume that $\Sigma \vDash \bot$ implies $\Sigma \vdash \bot$, and that $\Sigma \vDash \varphi$. Then $\Sigma \cup (\neg\varphi) \vDash \bot$, implying that $\Sigma \cup (\neg\varphi) \vdash \bot$ by the assumption. This in turn implies that $\Sigma \vdash \varphi$ by Exercise 2.7.4.

We prove the contrapositive: If Σ is consistent, then there is a model of Σ.

Extending $\mathcal{L}$ and Σ recursively: The model of Σ we will construct will have a universe containing (equivalence classes of) variable-free terms, so we must ensure that $\mathcal{L}$ has at least some constants. Furthermore, if c is a constant and P a unary predicate, then the set $\{\exists x P(x), \neg P(c)\}$ is consistent, but we must ensure the existence of a constant c' such that $P(c')$ is true for this set to have a model. Hence we also add the axiom $\exists x P(x) \to P(c')$.

We extend $\mathcal{L}$ and Σ by using the following constructions:

(a) We first *extend $\mathcal{L}$ by constants*: Let c_n be a constant not in $\mathcal{L}$ for all $n \in \mathbb{Z}_+$, and let $\tilde{\mathcal{L}} = \mathcal{L} \cup \{c_n \mid n \in \mathbb{Z}_+\}$. The constants c_n are called *Henkin constants*.

[1] Two elements in Σ may have the same universal closure, so in constructing Σ_D we make an arbitrary choice of a single element. Since Σ'_D is finite, this does not require the Axiom of Choice. Alternatively, we may choose any enumeration (hence well-ordering) of Σ, which is possible since Σ is countable, and pick the smallest relevant element.

Notice that since $\mathcal{L}$ is countable, so is $\tilde{\mathcal{L}}$. Furthermore, Lemma 3.2.3 shows that if Σ is a consistent set of $\mathcal{L}$-sentences, then it is also a consistent set of $\tilde{\mathcal{L}}$-sentences.

(b) Since $\mathcal{L}$ is countable, we may enumerate all sentences in $\mathcal{L}$ on the form $(\exists x)(\theta)$. Notice that since these are sentences, the only free variable (if any) in θ is x. Denote these by $(\exists x)(\theta_n)$ for $n \in \mathbb{Z}_+$ and define the collection

$$\tilde{H} = \{(\exists x)(\theta_n) \to \theta_n(c_n) \mid (\exists x)(\theta_n) \text{ is an } \mathcal{L}\text{-sentence}\}$$

of *Henkin axioms*, where $\theta_n(c_n)$ is shorthand for $(\theta_n)^x_{c_n}$.

(c) Finally we define the set $\tilde{\Sigma} = \Sigma \cup \tilde{H}$ of $\tilde{\mathcal{L}}$-sentences. Lemma 3.2.4 shows that if Σ is consistent, then $\tilde{\Sigma}$ is also consistent.

Notice that we must add a Henkin constant for each existential sentence: Otherwise if P is a unary predicate and c a Henkin constant, then it might happen that $\exists x P(x) \to P(c)$ and $\exists x \neg P(x) \to \neg P(c)$ are Henkin axioms. The set $\{\exists x P(x), \exists x \neg P(x)\}$ is consistent, but adding these axioms makes it inconsistent.

Furthermore, it is not enough to perform this procedure once: We have only added a Henkin axiom for each existential $\mathcal{L}$-sentence, but in extending $\mathcal{L}$ to $\tilde{\mathcal{L}}$ we might have added more existential sentences. Hence we apply the constructions above recursively as follows: Let $\mathcal{L}_0 = \mathcal{L}$ and $\Sigma_0 = \Sigma$. For $n \in \mathbb{N}$ we do the following:

(a') Given a (countable) language $\mathcal{L}_n$, extend it by constants to obtain $\mathcal{L}_{n+1}$. Then Σ_n is a consistent set of $\mathcal{L}_{n+1}$-sentences.

(b') Construct the set H_{n+1} of Henkin axioms in the language $\mathcal{L}_{n+1}$.

(c') Define $\Sigma_{n+1} = \Sigma_n \cup H_{n+1}$. Then Σ_{n+1} is a consistent set of $\mathcal{L}_{n+1}$-sentences.

Finally let $\mathcal{L}' = \bigcup_{n\in\mathbb{N}} \mathcal{L}_n$ and $\hat{\Sigma} = \bigcup_{n\in\mathbb{N}} \Sigma_n$. We claim that Σ_n as well as $\hat{\Sigma}$ are consistent sets of $\mathcal{L}'$-sentences: For Σ_n, notice that $\mathcal{L}'$ can be obtained from $\mathcal{L}_n$ by extending by constants, so Lemma 3.2.3 implies that Σ_n is consistent. For $\hat{\Sigma}$, this is the content of Exercise 3.2.2

Maximal extension of $\hat{\Sigma}$: Since $\mathcal{L}'$ is countable, there exists an enumeration $\{\sigma_k\}_{k\in\mathbb{Z}_+}$ of all $\mathcal{L}'$-sentences. Let $\Sigma^0 = \hat{\Sigma}$ and let

$$\Sigma^{k+1} = \begin{cases} \Sigma^k \cup \{\sigma_{k+1}\}, & \text{if } \Sigma^k \cup \{\sigma_{k+1}\} \text{ is consistent,} \\ \Sigma^k \cup \{\neg\sigma_{k+1}\}, & \text{otherwise.} \end{cases}$$

Let $\Sigma' = \bigcup_{k\in\mathbb{N}} \Sigma^k$. Then each Σ^k is consistent by Exercise 3.2.3, and Σ' is consistent by Exercise 3.2.2. Furthermore, Σ' is the maximal consistent extension of $\hat{\Sigma}$ since it contains all $\mathcal{L}'$-sentences possible for consistency, and this is sufficient by Lemma 2.7.3. This also has the property that if σ is a sentence, then $\sigma \in \Sigma'$ if and only if $\Sigma' \vdash \sigma$ by Exercise 3.2.5.

Model of Σ': First, let T be the set of variable-free terms of $\mathcal{L}'$, and let $t_1 \sim t_2$ if $(t_1 = t_2) \in \Sigma'$. Exercise 3.2.7 shows that this is an equivalence relation. We construct the model $\mathfrak{A}$ as follows:

(a) *Universe*: The universe A is the quotient $T/{\sim}$.

(b) *Constants*: For each constant $c \in \mathcal{L}'$ (including Henkin constants), we simply let $c^{\mathfrak{A}} = [c]$, i.e. the $\sim$-equivalence class of c.

(c) *Functions*: If f is an n-ary function symbol, then we let

$$f^{\mathfrak{A}}\big([t_1],\dots,[t_n]\big) = [f t_1 \cdots t_n].$$

This is shown to be well-defined by an argument similar to that of Exercise 3.2.7, showing that $\vdash (t_1 = t_2 \to f(t_1) = f(t_2))$.

(d) *Relations*: Given an n-ary relation symbol R, we say that $R^{\mathfrak{A}}([t_1],\dots,[t_n])$ is true if $Rt_1 \cdots t_n \in \Sigma'$. We show in Exercise 3.2.8 that this is well-defined.

Proposition 3.2.6 now shows that $\mathfrak{A} \models \Sigma'$ by showing that $\sigma \in \Sigma'$ if and only if $\mathfrak{A} \models \sigma$ for all sentences σ. This is done as follows:

(a) If σ is atomic, then this follows by definition of $\mathfrak{A}$.

(b) If σ is on the form $\neg\alpha$ or $\alpha \vee \beta$, then this easily follows by induction.

(c) If $(\forall x)\varphi(x) \in \Sigma'$, then we must show that $\mathfrak{A} \models_s \varphi(t)$ for all s and t, i.e. that $\mathfrak{A} \models \varphi(t)$ since $\varphi(t)$ is a sentence. Since $(\forall x)\varphi(x) \to \varphi(t)$ is an axiom and $\Sigma' \vdash (\forall x)\varphi(x)$, then $\Sigma' \vdash \varphi(t)$. By induction, $\mathfrak{A} \models \varphi(t)$.

Conversely, if $(\forall x)\varphi(x) \notin \Sigma'$ then $\neg(\forall x)\varphi(x) \in \Sigma'$, and so $(\exists x)\neg\varphi(x) \in \Sigma'$. There is a Henkin constant c such that $[(\exists x)\neg\varphi(x) \to \neg\varphi(c)] \in \Sigma'$, so $\neg\varphi(c) \in \Sigma'$. By induction, $\mathfrak{A} \models \neg\varphi(c)$, i.e. $\mathfrak{A} \not\models \varphi(c)$, which implies that $\mathfrak{A} \not\models (\forall x)\varphi(x)$.

In particular $\mathfrak{A} \models \Sigma$.

Restricting $\mathfrak{A}$ to $\mathcal{L}$: Next define the restriction $\mathfrak{A}|_{\mathcal{L}}$ of $\mathfrak{A}$ to $\mathcal{L}$ in the obvious way. Then Lemma 3.2.7 shows that if σ is an $\mathcal{L}$-sentence, then $\mathfrak{A}|_{\mathcal{L}} \models \sigma$ if and only if $\mathfrak{A} \models \sigma$. We prove this by showing that $\mathfrak{A}|_{\mathcal{L}} \models \sigma$ if and only if $\sigma \in \Sigma'$ as in the proof of Proposition 3.2.6. Thus we have $\mathfrak{A}|_{\mathcal{L}} \models \Sigma$ as desired.

Remark 3.1: $\bar{s}(t) = [t]$.

Let t be a variable-free $\mathcal{L}'$-term, and let s be any assignment function into $\mathfrak{A}$. Then we claim that $\bar{s}(t) = [t]$. (Of course, by Lemma 1.7.6 the value of $\bar{s}(t)$ should not depend on s, since t contains no variables.) We show this by induction.

If c is a constant, then by definition $\bar{s}(c) = [c]$. If $t :\equiv f t_1 \cdots t_n$, then by induction we have

$$\bar{s}(f t_1 \cdots t_n) = f^{\mathfrak{A}}(\bar{s}(t_1),\dots \bar{s}(t_n)) = f^{\mathfrak{A}}\big([t_1],\dots,[t_n]\big) = [f t_1 \cdots t_n].$$

Notice that we here use the definition of $f^{\mathfrak{A}}$: It is not immediately obvious where in the proof of Proposition 3.2.6 this is used, but this definition ensures that $R^{\mathfrak{A}}$ is well-defined (see Exercise 3.2.8). ⌟

Exercise 3.2.2

Assume that $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \cdots$ are such that each Σ_n is a consistent set of sentences in a language $\mathcal{L}$. Show $\hat{\Sigma} = \bigcup_{n\in\mathbb{N}} \Sigma_n$ is consistent.

Solution. We prove that if $\hat{\Sigma}$ is inconsistent, then at least one Σ_n is inconsistent, so let D be a derivation of $\perp$ from $\hat{\Sigma}$. Since D is finite, it contains finitely many formulas from $\hat{\Sigma}$, say $\varphi_1, \ldots, \varphi_k$. Each φ_i then lies in some Σ_i, so let n be the largest of these i. Then all φ_i lie in Σ_n, so D is also a derivation of $\perp$ from Σ_n, and hence Σ_n is inconsistent. □

Exercise 3.2.3

Show that if Π is any consistent set of sentences and σ is a sentence such that $\Pi \cup \{\sigma\}$ is inconsistent, then $\Pi \cup \{\neg\sigma\}$ is consistent.

Solution. We show that if $\Pi \cup \{\neg\sigma\}$ is inconsistent, then so is Π. The former means that $\Pi \cup \{\neg\sigma\} \vdash \perp$, which by Exercise 2.7.4 implies that $\Pi \vdash \sigma$. But since $\Pi \cup \{\sigma\} \vdash \perp$ the same exercise implies that $\Pi \vdash (\neg\sigma)$.[2] Hence $\Pi \vdash \{\sigma, (\neg\sigma)\}$, so $\Pi \vdash \perp$ by the same argument as in Exercise 2.7.4. □

Exercise 3.2.5

Prove Lemma 3.2.5: If σ is a sentence, then $\sigma \in \Sigma'$ if and only if $\Sigma' \vdash \sigma$.

Solution. The 'only if' part is obvious. The 'if' part follows since if $\Sigma' \vdash \sigma$ then $\Sigma' \cup \{\sigma\}$ is consistent, and hence $\sigma \in \Sigma'$ by maximality. □

Exercise 3.2.7

Complete the proof of the claim on page 80 that the relation $\sim$ is an equivalence relation.

Solution. The authors sketch a proof that $\sim$ is symmetric. We give a slightly different proof below. Notice that Theorem 2.7.1 shows that equality of *variables* is an equivalence relation. We first show that equality of *variable-free terms* is also an equivalence relation.

[2] Here we use that we can derive σ from $\neg\neg\sigma$, which is possible since the latter is a propositional consequence of the former.

Let $\varphi(x,y,z)$ be one of the formulas in Theorem 2.7.1, so that $\vdash \varphi(x,y,z)$, and denote by $\varphi(t_1,t_2,t_3)$ the formula obtained by replacing x with the variable-free term t_1, y with t_2, and z with t_3 (notice that t_1 is substitutable for x, etc.). Then Lemma 2.7.2 shows that also $\vdash (\forall x)(\forall y)(\forall z)\varphi(x,y,z)$. The axiom (Q1) then has the instances

$$\begin{aligned}(\forall x)(\forall y)(\forall z)\varphi(x,y,z) &\to (\forall y)(\forall z)\varphi(t_1,y,z),\\ (\forall y)(\forall z)\varphi(t_1,y,z) &\to (\forall z)\varphi(t_1,t_2,z),\\ (\forall z)\varphi(t_1,t_2,z) &\to \varphi(t_1,t_2,t_3),\end{aligned}$$

from which it is easy to derive that $\vdash \varphi(t_1,t_2,t_3)$. Hence equality of terms is an equivalence relation.

Next we prove that $\sim$ is transitive, since reflexivity and symmetry are easier, so assume that $t_1 \sim t_2$ and $t_2 \sim t_3$. By Lemma 3.2.5 (i.e. Exercise 3.2.5), this is the same as $\Sigma' \vdash (t_1 = t_2)$ and $\Sigma' \vdash (t_2 = t_3)$. Then it is clear that also $\Sigma' \vdash [(t_1 = t_2) \land (t_2 = t_3)]$, and by the above this implies that $\Sigma' \vdash (t_1 = t_3)$, i.e. that $t_1 \sim t_3$. □

Exercise 3.2.8

Show that the relation $R^{\mathfrak{A}}$ of the structure $\mathfrak{A}$ is well-defined.

Solution. By Definition 1.7.4 and Definition 1.7.9, we say that $\mathfrak{A} \vDash Rt_1 \cdots t_n$ if $(\bar{s}(t_1),\dots,\bar{s}(t_n)) \in R^{\mathfrak{A}}$ for all assignment functions s into $\mathfrak{A}$. By Remark 3.1 this means that $([t_1],\dots,[t_n]) \in R^{\mathfrak{A}}$, so we in other words define $R^{\mathfrak{A}}$ by

$$R^{\mathfrak{A}} = \left\{ \big([t_1],\dots,[t_n]\big) \in A^n \;\middle|\; Rt_1 \dots t_n \in \Sigma' \right\},$$

and we must show that this is well-defined. An argument similar to that of Exercise 3.2.7 shows that $\vdash [t_1 = t_2 \to (R(t_1) \to R(t_2))]$, which implies well-definition. □