

5.2 Terms, Formulas, Free Variables and Substitution

Section 5.1 read quantified sentences by eye. That is enough to understand them, and not enough to reason with them: every quantifier rule of Sections 5.3 to 5.6, and the rule for using an equation in Section 5.7, has the form “put the term t in for the variable x in φ ”, written $\varphi[t/x]$, and a rule has to say exactly which symbols change. This section makes the language of Section 5.1 exact, in the way Section 4.1 made the propositional language exact, says exactly what $\varphi[t/x]$ does, and names the one way it goes wrong, **capture**.

5.2.1 Which letters are holes

Three sentences, over the domain $\mathbb{N}$:

- “ $x > 2$ ” is not a statement. It is the open sentence of item (e) of Example (Statements and nonstatements), true for some values of x and false for others.
- “ $\forall x (x^2 \geq 0)$ ” is a statement, and a true one. The letter x still appears, but the quantifier has closed its hole: the sentence does not ask which x we mean.
- “ $\forall x (x \geq y)$ ” is neither true nor false as it stands. It is true when y is 0 (every natural number is at least 0) and false when y is 1 (the number 0 is not at least 1). It is an open sentence **in** y , although it begins with a quantifier.

So the question “is this a statement?” is not answered by looking for quantifiers. It is answered by asking which letters are still holes. In the third sentence x is closed by $\forall x$ and y is open. The open letters will be called **free**, and the closed ones **bound**.

Here is why the difference matters for reasoning. Over $\mathbb{N}$, the sentence

$$\exists y (y \neq x)$$

is an open sentence in x , and it is true for every x : given x , the number $y = x + 1$ differs from it. Now suppose that we want to say the same thing about a particular number named by another letter, and we put y in for x without thinking. The result is

$$\exists y (y \neq y),$$

“some number differs from itself”, which is false. Nothing about numbers went wrong. The y that we put in landed inside the scope of $\exists y$, and the quantifier took hold of it: the new y was meant to name a number from outside, and it has become the quantifier’s own placeholder. This is **capture**.

Capture is the one trap of substitution, and the rules of Sections 5.3 to 5.7 substitute at almost every step. So we need exact answers to three questions: what the expressions are that can be put in; which occurrences of x get replaced; and when putting t in for x is safe.

5.2.2 Terms and formulas

The first things to build are the names of objects.

A term is a name of an object, built from variables and constants by function symbols.

Definition 5.2.1. Term

The **terms** are the strings given by the following clauses.

- (Tm1) Every variable is a term, and every constant symbol is a term.
- (Tm2) If f is a function symbol of arity n and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.
- (Tm3) **Nothing else** is a term: a string is a term **only if** it is the last entry of a finite list of strings in which every entry is a variable, a constant symbol, or $f(s_1, \dots, s_n)$ for some function symbol f of arity n and **earlier** entries $s_1, \dots, s_n$.

In words: variables and constants are terms, a function symbol applied to as many terms as it has places is a term, and nothing else is.

A **variable of** a term t is a variable that occurs in t . A term with no variables, such as 2 or $f(c)$, names one fixed object once the symbols are interpreted; a term with variables, such as $x + 1$, names an object only once its variables have values.

Shorthand. We write terms in the familiar way: $x + y$ for $+(x, y)$, x^2 , $|x|$, a_n for $a(n)$ (a sequence is a function symbol, as Section 5.1 said). School conventions about grouping say which term is meant.

Here are terms, simplest first.

- x and 2: by (Tm1). The degenerate terms, built in no steps; every argument about all terms has to handle them.
- $x + 1$, officially $+(x, 1)$: by (Tm2) from the terms x and 1.
- $f(f(c))$: by (Tm2) twice, from c , for a one-place f . Its only building blocks are constants, so it has no variables.
- $|a_n - L|$, officially $|\cdot|(-(a(n), L))$, with the two-place minus: by (Tm2) three times, from the variables n and L . Its variables are n and L . The letter a is a function symbol, not a variable.

○ **Second reading** · *The symbols of the language, strings that are not terms, and two more shorthand rules*

Section 4.1 described the propositional language by its alphabet and its formation rules. The language of predicate logic has more symbols, of more kinds.

- **Variables:** x, y, z, u , with subscripts $x_0, x_1, y_0, \dots$, and in analysis also $\varepsilon, \delta, N, n, k, L$. There are infinitely many, so that a fresh one is always available. Each is one symbol.
- **Constant symbols**, such as 0, 1, 2, a, b, c . (A letter may be a constant in one language and a function symbol in another: in a language with a sequence, a is the sequence's one-place function symbol, as below, and not a constant.)
- **Function symbols**, each with a number of places $n \geq 1$, its **arity**: f and g , the arithmetic symbols $+$, $\cdot$ and $-$ (two places), and $-$ in front of a number, halving $\cdot/2$, squaring and $|\cdot|$ (one place). The one-place and the two-place minus are different symbols written alike; the number of terms they are applied to shows which is meant.
- **Predicate symbols**, each with an arity $n \geq 1$: $P, Q, R, S, <, \leq, >, \geq$, Prime, and $\in A$ for a fixed set A (one place, as in Section 5.1).
- **Equality** $=$, which every language has.
- The constant $\perp$, the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ of Chapter 4, and the **quantifier symbols**

$\forall$ and $\exists$.

- Parentheses $(,)$, and the comma.

Which constant, function, and predicate symbols are available depends on the subject. For the argument of Section 5.1 we need the constant 2 and the one-place predicate symbols P and S . For arithmetic we need 0, 1, +, $\cdot$, and $<$. Every formula uses only finitely many symbols, and we never need to fix the whole list in advance.

Strings that are **not** terms, each by one clause:

- $x < 1$ is not a term. It does not name an object; it says something about one. It is built with a predicate symbol, and (Tm2) uses only function symbols. It will be a formula.
- $f(x, y)$, for a one-place f , is not a term: (Tm2) applies f to exactly one term.

Terms can be drawn as trees, exactly as formulas were in Section 4.1: a variable or a constant is a single node, and $f(t_1, \dots, t_n)$ is a node labeled f with the trees of $t_1, \dots, t_n$ below it, in order. And a term can be read in only one way. The proof is the proof of unique readability for formulas in Section 4.3 (Theorem 4.3.3), with the parentheses and commas of (Tm2) doing the work that the parentheses of (F3) did there. We do not repeat it.

Shorthand rules 1 and 3.

Rule 1. Two-place predicate symbols such as $<, \leq$ are written between their terms: $x < y$ is shorthand for $<(x, y)$. The one-place symbol $\in A$ is written after its term: $t \in A$ is shorthand for $(\in A)(t)$. (When sets are themselves objects of the domain, $\in$ is a two-place predicate symbol and $x \in A$ is $\in(x, A)$; in this chapter A is a fixed set, except in one example of Section 5.7 where sets are the objects, so we treat $\in A$ as one-place.) We write $t_1 \neq t_2$ for $\neg(t_1 = t_2)$. **Rule 3.** The bounded quantifiers of Definition 5.1.1, $\forall x \in A \varphi$ and the rest, stand for the formulas given there.

Next, the statements and open sentences. Atomic formulas say something about terms; everything else is built from them by connectives and quantifiers.

A first-order formula is built from predicates applied to terms, by the connectives and the two quantifiers.

Definition 5.2.2. First-Order Formula

The **formulas** (of predicate logic, or **first-order formulas**) are the strings given by the following clauses.

- (Fm1) **Atomic formulas.** If R is a predicate symbol of arity n and $t_1, \dots, t_n$ are terms, then $R(t_1, \dots, t_n)$ is a formula. If t_1 and t_2 are terms, then $t_1 = t_2$ is a formula. And $\perp$ is a formula.
- (Fm2) If φ is a formula, then $\neg\varphi$ is a formula.
- (Fm3) If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are formulas.
- (Fm4) If φ is a formula and x is a **variable**, then $\forall x \varphi$ and $\exists x \varphi$ are formulas.
- (Fm5) **Nothing else** is a formula: a string is a formula only if it is the last entry of a finite list in which every entry is an atomic formula, or is built by (Fm2), (Fm3) or (Fm4) from **earlier** entries.

- *(Fm1)*: the atomic formulas, a predicate symbol with its terms, or an equation, or $\perp$, take the place of Chapter 4’s atoms.
- *(Fm2)*, *(Fm3)*: the connectives, as before.
- *(Fm4)*: puts $\forall x$ or $\exists x$, with a **variable** x , in front of a formula, without parentheses, as $\neg$ is.

Structural induction and recursion (Theorem 4.3.1, Theorem 4.3.4) carry over to terms and first-order formulas, with the atomic formulas as base case and one new building step for each quantifier (the fold below says how).

🕒 **Second reading** • *Induction and recursion on formulas*

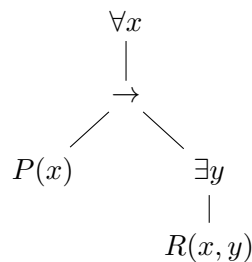
The base case covers every **atomic** formula (and $\perp$), and there are two new building steps, from φ to $\forall x \varphi$ and to $\exists x \varphi$, for every variable x , each with one part, like the step to $\neg \varphi$. So, to prove that every formula has a property, prove it for the atomic formulas and $\perp$, and prove that it passes from φ to $\neg \varphi$, $\forall x \varphi$, and $\exists x \varphi$, and from φ, ψ to $(\varphi \circ \psi)$. To define a function on formulas, give one clause for each of these cases. The proofs in Section 4.3 go through word for word with the new cases added, and unique readability holds for the same reason; we use all three without repeating them. The same two principles hold for terms, with (Tm1) as the base case and (Tm2) as the one building step.

Shorthand. The conventions of Section 4.1 for dropping parentheses stay, with three additions. Rules 1 and 3 (predicate symbols written between their terms, as in $x < y$; bounded quantifiers as abbreviations) are in the fold on the symbols of the language, above; rule 2 matters in every section:

Rule 2. A quantifier binds as tightly as $\neg$: it applies to the smallest formula that follows it. So $\forall x P(x) \rightarrow Q(c)$ stands for $(\forall x P(x) \rightarrow Q(c))$, an implication whose hypothesis is $\forall x P(x)$. For the quantifier to govern the whole implication, write $\forall x (P(x) \rightarrow Q(x))$, where the parentheses are the official ones around the implication.

Rule 2 matters in every rule section. The formula $\forall x P(x) \rightarrow Q(c)$ has main connective $\rightarrow$, while $\forall x (P(x) \rightarrow Q(x))$ has main symbol $\forall x$. The rule for using a “for every” (Section 5.3) applies only to the second.

Parse trees. A parse tree for a first-order formula is drawn as in Section 4.1, with one new kind of node: a **quantifier node**, labeled $\forall x$ or $\exists x$, with the tree of the formula it governs as its only subtree. The leaves are atomic formulas (or $\perp$); their terms have trees of their own, which we do not draw. Here is the tree of $\forall x (P(x) \rightarrow \exists y R(x, y))$.



The root is the quantifier node $\forall x$, so the formula’s main symbol is $\forall x$. Below it is the implication, with the atomic formula $P(x)$ on the left and the quantifier node $\exists y$ on the right, which

governs the atomic formula $R(x, y)$.

Quick check. Draw the parse tree of $\forall x P(x) \rightarrow \exists y \forall x R(x, y)$, written in shorthand. What is its main symbol?

Solution. By shorthand rule 2, each quantifier applies to the smallest formula after it, so the official formula is $(\forall x P(x) \rightarrow \exists y \forall x R(x, y))$ and its main symbol is $\rightarrow$. The root is $\rightarrow$. Its left subtree is the node $\forall x$ above the leaf $P(x)$. Its right subtree is the node $\exists y$, below it the node $\forall x$, and below that the leaf $R(x, y)$. The two $\forall x$ nodes are on different branches.

5.2.3 Free and bound: drawing the arrows

The parse tree shows which quantifier governs which part of a formula. That is exactly what decides whether a letter is a hole. Draw an arrow from each occurrence of a variable to the nearest quantifier on that variable above it in the parse tree. An occurrence with an arrow is bound; one with none is a hole still open.

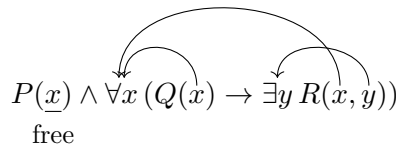


Figure B. Each bound occurrence points to its quantifier. The first x points nowhere: it is free.

An occurrence of a variable is bound if a quantifier on that variable stands above it in the parse tree; otherwise it is free.

Definition 5.2.3. Free and Bound Variables

Let φ be a formula and x a variable. An **occurrence** of x in φ is an appearance of x inside one of the terms of an atomic formula of φ . (The x written just after a quantifier symbol, as in $\forall x$, is part of the quantifier node and is not an occurrence.)

- An occurrence of x is **bound** if the path from its leaf up to the root of the parse tree passes through a quantifier node $\forall x$ or $\exists x$, with the **same** variable x . The lowest such node **binds** the occurrence.
- An occurrence that is not bound is **free**.
- The **scope** of a quantifier node is the part of the tree below it: the formula it governs.
- The variable x is a **free variable of** φ , or is **free in** φ , if at least one occurrence of x in φ is free.

A variable **appears in** φ if it is written anywhere in φ : inside a term, or just after a quantifier symbol. Every freshness condition of this chapter is stated with “appear”; in practice, a fresh letter is one written nowhere yet.

Definition 5.2.4. Sentence

A **sentence** is a formula with no free variables.

A sentence is what Chapter 1 called a statement, once the domain and the symbols are fixed: every hole is closed, so it is true or false. A formula with free variables is an open sentence, true or false only once values are given to its free variables.

Examples. The domain is irrelevant to all of them: being free or bound is a matter of the string, not of what it means.

- $\forall x (x \geq y)$: the occurrence of x is bound by $\forall x$; the occurrence of y has no quantifier on y above it, so it is free. The free variables are y alone. Not a sentence.
- $P(x) \wedge \forall x Q(x)$: the occurrence of x in $P(x)$ is **free** (no quantifier stands above it: the $\forall x$ governs only $Q(x)$), and the occurrence in $Q(x)$ is bound. So x is a free variable of this formula, and it is not a sentence.
- $\forall x \exists x P(x)$: the occurrence of x has two quantifiers on x above it, and the lower one, $\exists x$, binds it. The outer $\forall x$ binds nothing. A sentence.

🕒 **Second reading** · Occurrences, ‘appear’, and the recursive description of ‘free’

In words: an occurrence of x is bound when it lies in the scope of a quantifier on x ; quantifiers on other variables do not bind it. If several quantifiers on x stand above it, the nearest one, the lowest in the tree, is the one that binds it. A free occurrence is a hole still open. A variable is free in a formula if it has **at least one** free occurrence, even if it has bound occurrences too.

So y appears in $\forall y P(x)$, although it has no occurrence there. A variable that does not appear in φ has no occurrence in φ , and no quantifier of φ is on it; that is why the freshness conditions use “appear”.

Three more examples:

- $\perp$ and $P(c)$ are sentences: they contain no variables at all. So is $\forall x P(c)$. These are the degenerate cases.
- $\forall x (x^2 \geq 1)$: the single occurrence of x lies below the node $\forall x$, so it is bound. A sentence.
- $\forall x (P(x) \rightarrow \exists y R(x, y))$, the parse tree drawn above: both occurrences of x are bound by the root; the occurrence of y in $R(x, y)$ is bound by $\exists y$. A sentence.

The recursive description. In proofs about all formulas we need “ x is free in φ ” in the form that structural induction can use, one case per clause of [Definition 5.2.2](#).

- x is free in an atomic formula exactly when x occurs in one of its terms; x is never free in $\perp$.
- x is free in $\neg\varphi$ exactly when x is free in φ .
- x is free in $(\varphi \circ \psi)$ exactly when x is free in φ or in ψ .
- x is free in $\forall y \varphi$, and in $\exists y \varphi$, exactly when x is a different variable from y **and** x is free in φ .

Each line is read off the tree. In an atomic formula there are no quantifier nodes, so every occurrence is free. The tree of $\neg\varphi$ or $(\varphi \circ \psi)$ adds a root that is not a quantifier node, so it binds nothing. The tree of $\forall y \varphi$ adds the root $\forall y$ on top of the tree of φ : an occurrence of x inside φ keeps its path, with one more node at the top, so it is free in $\forall y \varphi$ exactly when it was free in φ .

and the new node is not a quantifier on x , that is, when x and y are different variables.

○ **Second reading** · *The free variables of a limit statement*

Example. The free variables of a limit statement

With the conventions of Section 5.1 (domain $\mathbb{R}$, the sequence as a function symbol a), “the sequence (a_n) converges to L ” is the formula

$$\forall \varepsilon > 0 \exists N \in \mathbb{N} \forall n \in \mathbb{N} (n \geq N \rightarrow |a_n - L| < \varepsilon).$$

Write it without bounded quantifiers, and find its free variables. Is it a sentence?

Solution. Unfolding the three bounded quantifiers by Definition 5.1.1 ($\rightarrow$ under each $\forall$, $\wedge$ under the $\exists$) gives

$$\forall \varepsilon (\varepsilon > 0 \rightarrow \exists N (N \in \mathbb{N} \wedge \forall n (n \in \mathbb{N} \rightarrow (n \geq N \rightarrow |a_n - L| < \varepsilon)))).$$

The variables that occur are ε , N , n , and L . Check each.

- Every occurrence of ε lies below the root $\forall \varepsilon$: bound.
- Every occurrence of N lies below $\exists N$: bound.
- Every occurrence of n (in $n \in \mathbb{N}$, in $n \geq N$, and in the term a_n) lies below $\forall n$: bound.
- The occurrence of L has no quantifier on L above it: free.

So L is the only free variable, and the formula is **not** a sentence. It is an open sentence in L , as it should be: until L is given a value, “ (a_n) converges to L ” is neither true nor false, and which values make it true depends on the sequence (for $a_n = 1/(n+1)$ the value $L = 0$ does and $L = 1$ does not; for $a_n = (-1)^n$ no value does). Putting $\exists L$ in front gives the sentence “ (a_n) converges”.

The other symbols in it are not variables, and “free” and “bound” do not apply to them. The letter a is a function symbol; the formula is certainly about the sequence a , but a is fixed by the language, like $+$, and cannot be quantified in this logic. The symbol $\in \mathbb{N}$ is a one-place predicate symbol, 0 is a constant, and $>$, $\geq$, $<$, $-$ and $|\cdot|$ are predicate and function symbols of the language.

Warning. The same letter can occur free and bound in one formula. In $P(x) \wedge \forall x Q(x)$, the first x is a hole and the second is the quantifier’s placeholder, and the two have nothing to do with each other. This is legal, and it is a trap for the reader: a proof that “uses x ” from such a formula has to say which x it means. The book avoids it by renaming the bound variable (the last part of this section explains how): $P(x) \wedge \forall z Q(z)$ says the same thing, and the question does not arise.

Quick check. Is x free in $\forall y (x < y) \rightarrow \exists x (x = 0)$? At which occurrences?

Solution. Yes. By shorthand rule 2 the formula is $(\forall y (x < y) \rightarrow \exists x (x = 0))$. The occurrence of x in $x < y$ lies below $\forall y$ only, which is a quantifier on y , so it is **free**. The occurrence of x in $x = 0$ lies below $\exists x$, so it is bound. One free occurrence is enough: x is a free variable of the formula. The variable y , by contrast, is not free: its only occurrence lies below $\forall y$.

5.2.4 Substitution, by example first

We now come to the operation that every quantifier rule uses. The rule of Section 5.3 will take the premise $\forall x (P(x) \rightarrow S(x))$ of Section 5.1's argument and produce $P(2) \rightarrow S(2)$: it strips the quantifier and puts 2 in for x . The rule of Section 5.5 goes the other way. Both need to say exactly what "put 2 in for x " does. The answer has one principle: only the **free** occurrences are replaced, because only they are holes. A bound occurrence belongs to its quantifier, and the quantifier is not asking for a value.

$\varphi[t/x]$ is φ with t put in for every free occurrence of x , and nothing else changed.

Example. Four substitutions

Compute each substitution by the rule in the slogan above.

- (a) $(P(x) \rightarrow S(x))[2/x]$.
- (b) $(x < y \wedge \forall x (x \geq 0))[(y + 1)/x]$.
- (c) $(\forall y R(x, y))[f(z)/x]$.
- (d) $(\forall x P(x))[c/x]$.

Solution. (a) The implication passes the substitution to both sides, and in each atomic formula the term x becomes 2: the result is $P(2) \rightarrow S(2)$. This is the step that Section 5.3's rule makes in the argument of Section 5.1.

(b) The conjunction passes it to both parts. In $x < y$ the x becomes $y + 1$, giving $y + 1 < y$. In $\forall x (x \geq 0)$ the quantifier is on x , so the substitution stops. The result is

$$y + 1 < y \wedge \forall x (x \geq 0).$$

Only the free occurrence of x changed. The bound one, in $x \geq 0$, was the quantifier's placeholder, and replacing it would have turned the sentence "every x is at least 0" into a claim about $y + 1$.

(c) The quantifier is on y , a variable different from x , so it passes the substitution on: the result is $\forall y R(f(z), y)$.

(d) The quantifier is on x , so nothing changes: the result is $\forall x P(x)$. There was no free x to replace.

Here are the clauses those computations followed.

Definition 5.2.5. Substitution of a Term for a Variable

Let t be a term and x a variable. For a term s , the term $s[t/x]$ is s with every occurrence of x replaced by t ; by recursion on terms,

$$\begin{aligned} x[t/x] &= t, & y[t/x] &= y \text{ for a variable } y \neq x, & c[t/x] &= c, \\ f(s_1, \dots, s_n)[t/x] &= f(s_1[t/x], \dots, s_n[t/x]). \end{aligned}$$

For a formula φ , the formula $\varphi[t/x]$, read “ φ with t for x ”, is defined by recursion on φ :

$$\begin{aligned} R(s_1, \dots, s_n)[t/x] &= R(s_1[t/x], \dots, s_n[t/x]), \\ (s_1 = s_2)[t/x] &= (s_1[t/x] = s_2[t/x]), & \perp[t/x] &= \perp, \\ (\neg\varphi)[t/x] &= \neg(\varphi[t/x]), \\ (\varphi \circ \psi)[t/x] &= (\varphi[t/x] \circ \psi[t/x]), \\ (\forall x \varphi)[t/x] &= \forall x \varphi, \\ (\forall y \varphi)[t/x] &= \forall y (\varphi[t/x]) \text{ for a variable } y \neq x, \end{aligned}$$

and the last two lines likewise with $\exists$ in place of $\forall$.

A compound term is written in parentheses: $\varphi[(y+1)/x]$ puts $y+1$ in for x .

Two facts follow from the definition at once, and both are used again and again.

First, $\varphi[x/x] = \varphi$: putting x in for itself changes nothing. In each clause the replaced symbol is x again.

Second, and more useful: **if x is not free in φ , then $\varphi[t/x] = \varphi$** , for every term t . Example (d) was one instance. Later sections use it; its proof, a structural induction, is in the fold below.

🕒 **Second reading** · *The clauses in words, why the definition defines, and the claim proved*

In words, clause by clause: in a term, every x is replaced, since a term has no quantifiers. In an atomic formula, the replacement happens inside its terms. The connectives pass the substitution to their parts. A quantifier on x **stops** it: below $\forall x$ there is no free occurrence of x to replace. A quantifier on another variable y passes it on.

Well defined. The clauses give one value for each case of [Definition 5.2.2](#), so structural recursion (carried over as above) says that they define exactly one formula $\varphi[t/x]$ for each φ . And the result is a formula: each clause builds its value by the clauses of [Definition 5.2.1](#) and [Definition 5.2.2](#) from values already known to be terms and formulas.

This is a different operation from the substitution σ of Section 4.5 (the one in Lemma 4.5.3). There, formulas were put in for **atoms**, at every occurrence. Here a **term** is put in for a **variable**, and bound occurrences are left alone.

Claim. Let x be a variable and t a term. For every formula φ : if x is not free in φ , then $\varphi[t/x] = \varphi$.

Idea. Substitution changes only free occurrences of x , and there are none. Formally, structural induction on φ , using the recursive description of “free” case by case. The only case with any content is a quantifier on a variable other than x , where the substitution is passed on, and the induction hypothesis applies because x is still not free inside.

Proof. For terms first: if x does not occur in a term s , then $s[t/x] = s$, by induction on s (a variable other than x and a constant are unchanged by the clauses, and $f(s_1, \dots, s_n)[t/x] = f(s_1, \dots, s_n)$ when each $s_i[t/x] = s_i$). Now we prove the claim by structural induction on φ .

Atomic formulas and $\perp$. If x is not free in $R(s_1, \dots, s_n)$, then x occurs in none of the s_i (the recursive description), so each $s_i[t/x] = s_i$, and the clause for atomic formulas returns $R(s_1, \dots, s_n)$. The same holds for $s_1 = s_2$, and $\perp[t/x] = \perp$.

Connectives. If x is not free in $\neg\varphi$, it is not free in φ , so $(\neg\varphi)[t/x] = \neg(\varphi[t/x]) = \neg\varphi$ by the induction hypothesis. If x is not free in $(\varphi \circ \psi)$, it is free in neither part, and the same argument applies to both.

Quantifiers. Consider $\forall y \varphi$. If y is x , then $(\forall x \varphi)[t/x] = \forall x \varphi$ by the definition, whatever φ is. If y is a different variable and x is not free in $\forall y \varphi$, then x is not free in φ (the recursive description), so $(\forall y \varphi)[t/x] = \forall y (\varphi[t/x]) = \forall y \varphi$ by the induction hypothesis. The case of $\exists y \varphi$ is the same.

This proves the claim for every formula φ . ■

One substitution at a time. $\varphi[t/x]$ replaces one variable. Two substitutions in a row are made one after the other, and the second acts on the result of the first. So $R(x, y)[y/x]$ is $R(y, y)$, and then $[x/y]$ turns it into $R(x, x)$, not into the swapped $R(y, x)$: the first step has already lost track of which y was which. To swap two variables, go through a letter that appears nowhere: $[z/x], [x/y], [y/z]$ give $R(z, y), R(z, x), R(y, x)$.

Quick check. Compute $((P(x) \wedge Q(y))[y/x])[x/y]$. Which formula would swapping x and y give?

Solution. The first step gives $P(y) \wedge Q(y)$, and the second $P(x) \wedge Q(x)$. The swap is $P(y) \wedge Q(x)$, reached by $[z/x], [x/y], [y/z]$ with a new letter z .

5.2.5 Capture, and the practical rule

The definition of substitution never refuses. Given any φ , t and x , it returns a formula. Sometimes that formula does not say what we meant.

Take $\varphi = \exists y (y \neq x)$, the example at the start of the section, and $t = y$. The quantifier $\exists y$ is on

a variable different from x , so the definition passes the substitution through, and

$$(\exists y (y \neq x))[y/x] = \exists y (y \neq y).$$

The definition was followed correctly. The meaning is wrong. The formula $\exists y (y \neq x)$ says something about the number x : some number differs from it, which over $\mathbb{N}$ is true whatever x is. We wanted to say the same thing about the number y , and we got “some number differs from itself”, which is false. The y we put in was supposed to name a number from outside, and it landed in the scope of $\exists y$ and was **captured**.

Compare a substitution that is safe. Over $\mathbb{R}$, $\forall z (x < z)$ says that x is smaller than every number (false, as it happens, for every x ; but what matters is what it says). Put $y + 1$ in for x : the quantifier $\forall z$ passes the substitution on, and

$$(\forall z (x < z))[(y + 1)/x] = \forall z (y + 1 < z),$$

which says exactly that $y + 1$ is smaller than every number. Nothing was captured, because the term $y + 1$ does not contain z , the only variable quantified above the occurrence of x .

The difference between the safe case and the unsafe one is exactly this: whether a variable of t is quantified somewhere above a free occurrence of x .

A term is free for x when putting it in for x lets none of its variables be captured.

Definition 5.2.6. Free For

Let t be a term, x a variable, and φ a formula. The term t is **free for x** in φ if **no** free occurrence of x in φ lies in the scope of a quantifier $\forall y$ or $\exists y$ such that y is a variable of t .

In words: look at each place where t will go, that is, at each free occurrence of x . Look at the quantifiers above it in the tree. If none of them is on a variable of t , the substitution is safe there. The term is free for x when it is safe at every such place. Informally: **no variable of t becomes bound** when t is put in.

Four special cases, each immediate from the definition, cover most uses.

- A term with **no variables** (2 , c , $f(c)$) is free for every variable in every formula: there is no y to capture.
- The variable x is free for x in every formula. A free occurrence of x never lies in the scope of a quantifier on x ; if it did, it would be bound.
- A variable z that does **not appear** in φ is free for every variable in φ : no quantifier of φ is on z .
- If x is **not free** in φ , every term is free for x in φ : there are no free occurrences to check. (And then $\varphi[t/x] = \varphi$ by the claim above.)

In our examples: y is **not** free for x in $\exists y (y \neq x)$, because the free occurrence of x lies in the scope of $\exists y$ and y is a variable of the term y . Likewise $y + 1$ is not free for x in $\exists y (y > x)$. And $y + 1$ **is** free for x in $\forall z (x < z)$: the only quantifier above the free occurrence of x is on z , which is not a variable of $y + 1$.

“Free for” is the precise condition under which $\varphi[t/x]$ says about t what φ says about x . The rules $\forall E$ (Section 5.3), $\exists I$ (Section 5.5) and $=E$ (Section 5.7) carry it as a side condition.

it needs

Here is a second example, with a compound term. Over $\mathbb{R}$, $\exists y (y > x)$ is true for every x : there is a number larger than x . Put $y + 1$ in for x :

$$(\exists y (y > x))[(y + 1)/x] = \exists y (y > y + 1).$$

The result says that some number is larger than its own successor, which is false. It should have said that some number is larger than $y + 1$, which is true.

The recursive description. For induction proofs, “free for” can be read off case by case, like “free”:

- t is free for x in every atomic formula and in $\perp$;
- t is free for x in $\neg\varphi$ exactly when it is free for x in φ , and in $(\varphi \circ \psi)$ exactly when it is free for x in both φ and ψ ;
- t is free for x in $\forall y \varphi$ (and in $\exists y \varphi$) exactly when **either** x is not free in $\forall y \varphi$, **or** y is not a variable of t and t is free for x in φ .

The first two lines hold because an atomic formula has no quantifiers and a connective node is not a quantifier. For the third: if x is not free in $\forall y \varphi$, there is nothing to check. If it is free, then y is a different variable and the free occurrences of x in $\forall y \varphi$ are those in φ ; each lies in the scope of the new root $\forall y$ and of the quantifiers inside φ above it. The root is harmless exactly when y is not a variable of t , and the quantifiers inside are harmless exactly when t is free for x in φ .

Section 5.3 shows the capture example of this section turning a true premise into a false conclusion when the side condition is ignored. The definition asks for no more than it needs: a variable of t may be quantified elsewhere in φ , as long as no free occurrence of x lies in its scope. For instance, y is free for x in $P(x) \wedge \exists y Q(y)$, and the result $P(y) \wedge \exists y Q(y)$ says of y what the original said of x .

Before you substitute. Look at each free occurrence of x and at the quantifiers above it. If none of them is on a letter of t , go ahead. If one is, rename that quantifier’s letter to a brand-new letter first (next subsection), and then substitute. Nothing else can go wrong.

Quick check. Is y free for x in $\exists y (x < y) \wedge P(y)$? Is $x + 1$?

Solution. By shorthand rule 2 the formula is $(\exists y (x < y) \wedge P(y))$. The term y is not free for x : the free x of $x < y$ lies under $\exists y$, and y is a letter of the term. Rename first, $\exists z (x < z) \wedge P(y)$, and then substitute to get $\exists z (y < z) \wedge P(y)$. The term $x + 1$ is free for x : the only quantifier above the free x is on y , which is not a letter of $x + 1$.

Warning. Capture changes meaning, and it gives no warning: the definition of $\varphi[t/x]$ returns a formula either way. Over $\mathbb{R}$, $\exists y (y > x)$ is true for every x , and $(\exists y (y > x))[(y + 1)/x]$ is $\exists y (y > y + 1)$, which is false. Before putting t in for x , check that no variable of t is quantified above a free occurrence of x .

Quick check. Is $y + 1$ free for x in $\forall z (x < z)$? Is it free for x in $\forall y (x < y)$? Compute both substitutions and say what each result claims.

Solution. In $\forall z (x < z)$: yes. The only quantifier above the occurrence of x is on z , and z is not a variable of $y + 1$. The result is $\forall z (y + 1 < z)$, “ $y + 1$ is smaller than every number”, which is what the original said about x .

In $\forall y (x < y)$: no. The occurrence of x lies in the scope of $\forall y$, and y is a variable of $y + 1$. The definition gives $\forall y (y + 1 < y)$: “every number is larger than its own successor”. That is a sentence about no particular number at all, and a false one; the original said something about x , and the result should have said the same about $y + 1$. The y of $y + 1$ was captured.

5.2.6 Renaming bound variables

What should we do when t is not free for x in φ ? A bound variable is a placeholder, and the name of a placeholder does not matter. “Some number differs from x ” can be written $\exists y (y \neq x)$ or $\exists z (z \neq x)$, just as $\int_0^1 s^2 ds$ and $\int_0^1 u^2 du$ are the same integral. So the way out is: **rename the bound variable first, then substitute**.

Renaming. Let φ be a formula and y, z variables, with z **not appearing** in φ . The formula $\forall z \varphi[z/y]$ is obtained from $\forall y \varphi$ by **renaming** the bound variable y to z ; likewise $\exists z \varphi[z/y]$ from $\exists y \varphi$. The same may be done to a quantifier anywhere inside a larger formula.

In words: change the variable after the quantifier, and change the occurrences it binds, to a letter that appears nowhere in the formula. The condition that z does not appear in φ does two jobs. It makes z free for y in φ (the third special case above), so no new z is captured by a quantifier inside φ . And it ensures that no free z already in φ is captured by the new quantifier $\forall z$. Without it, renaming can change the meaning: $\forall y (y \geq z)$ says something about z (over $\mathbb{N}$, that $z = 0$), and “renaming” y to z would give $\forall z (z \geq z)$, a sentence that is simply true.

Renaming a bound letter is treated as never changing what a formula says; Sections 5.4 and 5.6 derive it at the front of a formula, and the fold says where the rest stands.

○ **Second reading** · *The status of renaming*

Its status. Renaming is not a rule of the calculus, and we have not proved that it keeps meaning. Here is where it stands. For a renaming at the front of a formula, $\forall y \varphi$ and $\forall z \varphi[z/y]$ can each be derived from the other in three lines, once the rules of Section 5.4 are available; that section does it, and Section 5.6 does the same for $\exists$. A renaming **inside** a larger formula we take as a convention of this book: formulas that differ only by renaming bound variables are treated as saying the same thing. Its justification in terms of meaning is the one the Looking-ahead box at the end of this section points to.

The recipe. To put t in for x in φ when t is not free for x : rename each quantifier of φ that is on a variable of t and has a free occurrence of x in its scope, each to a different variable that appears neither in φ nor in t (each such step is a renaming in the sense above); then substitute.

After the renaming, t is free for x .

- $\exists y (y \neq x)$ with y for x : rename y to z , giving $\exists z (z \neq x)$; then substitute, giving $\exists z (z \neq y)$, “some number differs from y ”. True over $\mathbb{N}$ for every y , as intended.
- $\exists y (y > x)$ with $y + 1$ for x : rename to $\exists z (z > x)$; substitute, giving $\exists z (z > y + 1)$. True over $\mathbb{R}$, as intended.

Which fresh letter we choose does not matter: two choices give formulas that differ only by a renaming. From now on, whenever a rule writes $\varphi[t/x]$, it will require t to be free for x in φ , and if we want a t that is not, we rename first.

5.2.7 How substitution looks in a mathematical proof

Mathematicians substitute constantly and never write $[t/x]$. Two examples show the operation, and its trap, in their natural habitat.

“Apply the hypothesis with $\varepsilon/2$ in place of ε .” Suppose we know that (a_n) converges to L , and we have fixed an $\varepsilon > 0$ for some other purpose. A proof says: “applying the hypothesis with $\varepsilon/2$, there is an $N \in \mathbb{N}$ with $|a_n - L| < \varepsilon/2$ for all $n \geq N$ ”. The hypothesis is $\forall \varepsilon \varphi$, with

$$\varphi = (\varepsilon > 0 \rightarrow \exists N \in \mathbb{N} \forall n \in \mathbb{N} (n \geq N \rightarrow |a_n - L| < \varepsilon)),$$

and the proof uses the instance $\varphi[t/\varepsilon]$ with the term $t = \varepsilon/2$:

$$\varepsilon/2 > 0 \rightarrow \exists N \in \mathbb{N} \forall n \in \mathbb{N} (n \geq N \rightarrow |a_n - L| < \varepsilon/2).$$

The term $\varepsilon/2$ contains the variable ε : the fixed ε of the proof, not the bound one of the hypothesis. It is free for ε in φ , because the only quantifiers above the free occurrences of ε in φ are on N and n . Section 5.3’s rule makes this step, and “ $\varepsilon/2 > 0$ ” then comes from $\varepsilon > 0$ by arithmetic.

“Take $N = n$.” Now suppose the proof has chosen N , so it knows

$$\psi = \forall n \in \mathbb{N} (n \geq N \rightarrow |a_n - L| < \varepsilon),$$

with N free. A tempting step is to “apply this with n in place of N ”. No rule allows this: N has been chosen, not quantified, so no term may be put in for it. Writing the step out shows how badly it goes wrong, because n is not even free for N in ψ : the occurrence of N lies in the scope of $\forall n$. The substitution produces

$$\forall n \in \mathbb{N} (n \geq n \rightarrow |a_n - L| < \varepsilon),$$

which says that **every** term of the sequence is within ε of L , a far stronger claim that does not follow. In words, the error is to let the threshold N be “the n we are looking at”, when n is the placeholder of “for all $n \geq N$ ”. A proof that needs a statement about a particular index names it with a new letter, say $m \geq N$, and substitutes m , which is free for n .

○ **Second reading** • *Induction, as a formula*

Induction, as a formula. The display after Axiom 4.2.1 in Section 4.2 used a property $P(n)$. With substitution it can be written for any formula φ in which the variable n is free, as the formula

$$(\varphi[0/n] \wedge \forall n \in \mathbb{N} (\varphi \rightarrow \varphi[(n+1)/n])) \rightarrow \forall n \in \mathbb{N} \varphi,$$

where $\varphi[0/n]$ plays the part of $P(0)$ and $\varphi[(n+1)/n]$ the part of $P(n+1)$. The term $n+1$ is free for n in φ : its only variable is n , and a free occurrence of n never lies in the scope of a quantifier on n . This is how the principle of induction reads in the language of this chapter; its status (a basic property of $\mathbb{N}$) is unchanged.

🔄 **Second reading** · *Partial terms*

Mathematics writes $1/x$ and $\lim a_n$, which name nothing for some values: $x = 0$, or a sequence with no limit. In the language of this chapter every term names an object, because every function symbol is defined everywhere on the domain. Mathematics keeps partial operations out of trouble with hypotheses: $1/x$ is written only under $x \neq 0$, as in $\forall x (x \neq 0 \rightarrow x \cdot (1/x) = 1)$, where the value at 0 is never used; and “the limit” is written only after it has been proved that there is exactly one (Section 5.8). Some logics allow terms that name nothing; this book does not need them.

Remark (Looking ahead). Chapter 7 proves, in an optional section, that renaming a bound letter never changes the meaning of a formula.

5.2.8 Exercises

5.2.8.1 A. Check your understanding

Exercise. A1

- For each formula, mark each occurrence of each variable as free or bound, list the free variables, and say whether it is a sentence: (i) $\forall x (P(x) \rightarrow Q(y))$; (ii) $\exists x \forall y R(x, y)$; (iii) $\forall x (x < y) \wedge \exists y (y < x)$; (iv) $\forall \varepsilon (\varepsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \delta < \varepsilon))$.
- Which of the following are terms, and which are formulas, for one-place f , two-place g and two-place predicate R ? (i) $g(f(x), c)$; (ii) $R(f(x), c)$; (iii) $f(R(x, c))$; (iv) $\forall c R(c, c)$.
- True or false, with a reason: if t is a variable, then t is free for x in every formula.
- True or false, with a reason: $(\forall x P(x) \wedge Q(x))[c/x]$ is $\forall x P(x) \wedge Q(c)$.

Solution. (a) (i) The single occurrence of x , in $P(x)$, is bound by $\forall x$ (the x after $\forall$ belongs to the quantifier node and is not an occurrence); the occurrence of y in $Q(y)$ is free (no quantifier on y). Free variables: y . Not a sentence.

(ii) The occurrence of x is bound by $\exists x$, that of y by $\forall y$. No free variables: a sentence.

(iii) By shorthand rule 2 this is $(\forall x (x < y) \wedge \exists y (y < x))$. In the left part, x is bound by $\forall x$ and y is free. In the right part, y is bound by $\exists y$ and x is free (the $\forall x$ governs only the left part). Free variables: x and y , each with one free and one bound occurrence. Not a sentence.

(iv) Every occurrence of ε lies below $\forall \varepsilon$, and both occurrences of δ lie below $\exists \delta$; 0 is a constant. A sentence.

(b) (i) A term, by (Tm2) twice. (ii) A formula, atomic by (Fm1): the predicate R applied to the terms $f(x)$ and c . (iii) Neither: $R(x, c)$ is a formula, not a term, and (Tm2) applies f only to a term. (iv) Neither: (Fm4) puts a **variable** after $\forall$, and c is a constant symbol.

(c) False. The variable y is not free for x in $\exists y (y \neq x)$, since the free occurrence of x lies in the scope of $\exists y$. (A variable t is free for x when t is x itself, or when t does not appear in the formula.)

(d) True. The formula is $(\forall x P(x) \wedge Q(x))$ by shorthand rule 2. The $\wedge$ passes the substitution to both parts; $\forall x P(x)$ is unchanged because its quantifier is on x , and $Q(x)$ becomes $Q(c)$.

Exercise. A2: Drill

Answer each part in a line or two.

- In $\forall y (x < y) \wedge \exists x (x = y)$, which occurrences are free, and what are the free variables?
- Is $\forall x \exists y (x + y = 0)$ a sentence?
- Someone writes $\forall x P(x) \rightarrow Q(x)$ for “every P is a Q ”. Which occurrence of x is free, and what is the repair?
- Repair $\forall \varepsilon > 0 \exists \delta > 0 (|x - a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon)$, meant as “ f is continuous at a ”, with a a variable here (not a constant symbol), so that a is its only free variable.
- Compute $(x \leq y \wedge \exists y (y < x))[3/x]$.
- Compute $(\forall x (x \geq y))[0/y]$.
- Compute $(\forall x P(x) \rightarrow P(x))[c/x]$.
- Compute $((x + y = y + x)[y/x])[x/y]$. Is the result the formula with x and y swapped?
- Compute $(P(x, y)[f(y)/x])[c/y]$.
- Is x free for y in $\exists x (x > y)$?
- Is $z + 1$ free for y in $\exists x (x > y)$? Compute.
- Put x in for y in $\exists x (x > y)$, renaming first.

Solution.

- The x of $x < y$ and the y of $x = y$ are free; the y of $x < y$ is bound by $\forall y$ and the x of $x = y$ by $\exists x$. Free variables: x, y .
- Yes: every occurrence is bound.
- By shorthand rule 2 the $\forall x$ governs only $P(x)$, so the formula is $(\forall x P(x)) \rightarrow Q(x)$ and the x of $Q(x)$ is free. Repair: $\forall x (P(x) \rightarrow Q(x))$.
- As written, x is free, so the formula speaks of one fixed x . Repair: $\forall \varepsilon > 0 \exists \delta > 0 \forall x (|x - a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon)$; f is a function symbol, not a variable.
- $3 \leq y \wedge \exists y (y < 3)$.
- $\forall x (x \geq 0)$.
- $\forall x P(x) \rightarrow P(c)$ (shorthand rule 2: the $\forall x$ governs only the first $P(x)$).
- $y + y = y + y$, then $x + x = x + x$. No: the swap is $y + x = x + y$.
- $P(f(y), y)$, then $P(f(c), c)$: the second substitution also reaches the y inside the term put in by the first.

- (j) No: the free y lies in the scope of $\exists x$, and x is the variable of the term.
- (k) Yes: $\exists x (x > z + 1)$.
- (l) Rename x to z : $\exists z (z > y)$; then $\exists z (z > x)$.

5.2.8.2 B. Practice

Exercise. B1: Computing substitutions

Compute each substitution. Where the term is not free for the variable, say which quantifier would capture it, rename first, and then substitute.

- (a) $(P(x) \wedge R(x, y))[f(y)/x]$.
- (b) $(\forall x R(x, y) \rightarrow P(x))[c/x]$.
- (c) $(\forall y (x < y))[(y + 1)/x]$.
- (d) $(\exists x (x = 2 \cdot y))[x/y]$.
- (e) $(\exists y (y > x) \wedge Q(y))[(x + 1)/y]$.

Solution. (a) No quantifiers, so $f(y)$ is free for x . Every occurrence of x is replaced: $P(f(y)) \wedge R(f(y), y)$. The y inside $f(y)$ and the y already in $R(x, y)$ now stand side by side; with no quantifier around, both are the same free y , as intended.

(b) By shorthand rule 2 the formula is $(\forall x R(x, y) \rightarrow P(x))$. The constant c has no variables, so it is free for x . The left part is unchanged (quantifier on x); the free x in $P(x)$ becomes c : $\forall x R(x, y) \rightarrow P(c)$.

(c) Not free for: the free occurrence of x lies in the scope of $\forall y$, and y is a variable of $y + 1$. Rename y to z , which appears nowhere: $\forall z (x < z)$. Then substitute: $\forall z (y + 1 < z)$. (Without renaming, the definition would give $\forall y (y + 1 < y)$, which says nothing about $y + 1$.)

(d) Not free for: the free occurrence of y lies in the scope of $\exists x$, and the term is x . Rename x to z : $\exists z (z = 2 \cdot y)$, “ y doubled is something”, then substitute: $\exists z (z = 2 \cdot x)$. (Without renaming, the result would be $\exists x (x = 2 \cdot x)$, “some number is its own double”, a claim about no particular number.)

(e) By shorthand rule 2 the formula is $(\exists y (y > x) \wedge Q(y))$. The occurrence of y in $y > x$ is bound, and the one in $Q(y)$ is free and not in the scope of any quantifier; so $x + 1$ is free for y (the $\exists y$ above the bound y is irrelevant, since that occurrence is not replaced). The result is $\exists y (y > x) \wedge Q(x + 1)$.

Exercise. B2: The set of free variables

For a term t , let $\text{Var}(t)$ be the set of variables of t , and for a formula φ , let $\text{FV}(\varphi)$ be the set of its free variables. (Here $\cup$ is the union of two sets, $\setminus$ removes elements, and $\emptyset$ is the empty set, all in the everyday sense.)

- (a) Write down clauses that define $\text{Var}(t)$ by recursion on terms and $\text{FV}(\varphi)$ by recursion on formulas, one clause for each case of [Definition 5.2.1](#) and of [Definition 5.2.2](#).
- (b) Use your clauses to compute FV of $\forall x (x < y) \wedge \exists y (y < x)$ and of $\forall x \exists y (x + y = z)$,

showing each step.

Solution. (a) For terms:

$$\begin{aligned}\text{Var}(x) &= \{x\} \text{ for a variable } x, & \text{Var}(c) &= \emptyset, \\ \text{Var}(f(t_1, \dots, t_n)) &= \text{Var}(t_1) \cup \dots \cup \text{Var}(t_n).\end{aligned}$$

For formulas:

$$\begin{aligned}\text{FV}(R(t_1, \dots, t_n)) &= \text{Var}(t_1) \cup \dots \cup \text{Var}(t_n), \\ \text{FV}(t_1 = t_2) &= \text{Var}(t_1) \cup \text{Var}(t_2), & \text{FV}(\perp) &= \emptyset, \\ \text{FV}(\neg\varphi) &= \text{FV}(\varphi), & \text{FV}((\varphi \circ \psi)) &= \text{FV}(\varphi) \cup \text{FV}(\psi), \\ \text{FV}(\forall x \varphi) &= \text{FV}(\exists x \varphi) = \text{FV}(\varphi) \setminus \{x\}.\end{aligned}$$

These are the four lines of the recursive description of “free” (in the fold after the examples of free and bound occurrences), rewritten as sets: $x \in \text{FV}(\varphi)$ exactly when x is free in φ . By structural recursion (carried over to terms and to first-order formulas), each set of clauses defines exactly one function.

(b) First formula. $\text{FV}(x < y) = \{x, y\}$, so $\text{FV}(\forall x (x < y)) = \{x, y\} \setminus \{x\} = \{y\}$. $\text{FV}(y < x) = \{x, y\}$, so $\text{FV}(\exists y (y < x)) = \{x\}$. The conjunction gives $\{y\} \cup \{x\} = \{x, y\}$, which agrees with A1(a)(iii).

Second formula. $\text{FV}(x + y = z) = \text{Var}(x + y) \cup \text{Var}(z) = \{x, y, z\}$. Then $\text{FV}(\exists y (x + y = z)) = \{x, z\}$, and $\text{FV}(\forall x \exists y (x + y = z)) = \{z\}$. So the formula is an open sentence in z alone (over $\mathbb{R}$ it is true for every z : given x , take $y = z - x$).

Exercise. B3: Swapping two letters

Let $\varphi = \exists z (R(x, z) \wedge R(z, y))$.

- (a) Compute $(\varphi[y/x])[x/y]$, and say whether it is φ with x and y swapped.
- (b) Swap x and y in φ with three substitutions through a letter u that appears nowhere, checking at each step that the term is free for the variable.
- (c) Why would z not do as the go-between?

Solution. (a) $\varphi[y/x] = \exists z (R(y, z) \wedge R(z, y))$, and then $[x/y]$ gives $\exists z (R(x, z) \wedge R(z, x))$. The swap is $\exists z (R(y, z) \wedge R(z, x))$, a different formula, and it says something different: over $\mathbb{N}$ with R read as $<$ and x, y naming 2, 0, the swap is true ($z = 1$), while the computed formula, “some z is above 2 and below 2”, is false.

(b) $\varphi[u/x] = \exists z (R(u, z) \wedge R(z, y))$; then $[x/y]$ gives $\exists z (R(u, z) \wedge R(z, x))$; then $[y/u]$ gives $\exists z (R(y, z) \wedge R(z, x))$, the swap. At each step the only quantifier above the replaced occurrence is $\exists z$, and the term (u, x, y) is not z , so it is free.

(c) z is not free for x in φ : the x of $R(x, z)$ lies inside $\exists z$, so $\varphi[z/x]$ would be captured and say $\exists z (R(z, z) \wedge R(z, y))$. The go-between must appear nowhere in φ , bound letters included (“appear”, in “Free and bound: drawing the arrows”).

Worked examples of these types: “One substitution at a time”; the practical rule “Before

you substitute”.

Marking scheme. (10 points)

- 3: (a), both steps computed and the swap written, with the difference shown (by the strings, or by one reading in which they differ).
- 4: (b), the three substitutions in the right order, each result correct, and the free-for check stated once for each step.
- 3: (c), capture named, with the captured formula written.

Fatal error (0 for the part): treating $[y/x]$ and $[x/y]$ as happening at the same time.

5.2.8.3 C. Going deeper

Exercise. C1: Challenge: What substitution does to the free variables

Let φ be a formula, x a variable and t a term, with t free for x in φ , and with x free in φ .

- (a) Prove that the free variables of $\varphi[t/x]$ are the free variables of φ other than x , together with the variables of t :

$$\text{FV}(\varphi[t/x]) = (\text{FV}(\varphi) \setminus \{x\}) \cup \text{FV}(t),$$

with the notation of B2. Prove it in the following form, by structural induction on φ : for every φ in which t is free for x , if $x \in \text{FV}(\varphi)$, then $\text{FV}(\varphi[t/x]) = (\text{FV}(\varphi) \setminus \{x\}) \cup \text{FV}(t)$; if $x \notin \text{FV}(\varphi)$, then $\varphi[t/x] = \varphi$. The term version, the atomic case and the connective cases are done below; prove the case of $\forall y \varphi$ (the case of $\exists y \varphi$ is the same).

- (b) Show, with the capture example $\exists y (y > x)$, $t = y + 1$, that the conclusion can fail when t is not free for x . Show also, with an example of your own, that it can fail when x is not free in φ .

Given for (a). *Terms.* Let s be a term in which x occurs. Then $\text{Var}(s[t/x]) = (\text{Var}(s) \setminus \{x\}) \cup \text{FV}(t)$. By induction on s : if s is the variable x , then $s[t/x] = t$ and both sides are $\text{FV}(t)$. (A constant, or a variable other than x , does not contain x .) If $s = f(s_1, \dots, s_n)$, then $s[t/x] = f(s_1[t/x], \dots, s_n[t/x])$; for each s_i containing x , the induction hypothesis gives $\text{Var}(s_i[t/x]) = (\text{Var}(s_i) \setminus \{x\}) \cup \text{FV}(t)$, and for each s_i not containing x , $s_i[t/x] = s_i$ (as in the proof of the claim in the text). At least one s_i contains x , so the union over i is $(\text{Var}(s) \setminus \{x\}) \cup \text{FV}(t)$.

Formulas. We prove, by structural induction on φ , the statement $S(\varphi)$: if t is free for x in φ , then $\text{FV}(\varphi[t/x]) = (\text{FV}(\varphi) \setminus \{x\}) \cup \text{FV}(t)$ when $x \in \text{FV}(\varphi)$, and $\varphi[t/x] = \varphi$ when $x \notin \text{FV}(\varphi)$. The second half is the claim proved in the text, for every φ ; so in each case below we assume $x \in \text{FV}(\varphi)$ and prove the equation. Write $E(\chi)$ for the set $(\text{FV}(\chi) \setminus \{x\}) \cup \text{FV}(t)$.

Atomic. Let $\varphi = R(s_1, \dots, s_n)$ with x occurring in some s_i . By the clause for atomic formulas, $\text{FV}(\varphi[t/x])$ is the union over i of the sets $\text{Var}(s_i[t/x])$. Exactly as in the term case for $f(s_1, \dots, s_n)$ (the result for terms for each s_i containing x , $s_i[t/x] = s_i$ for the others, and at least one s_i containing x), this union is $E(\varphi)$. The same for $s_1 = s_2$. ($\perp$ has no free variables.)

Negation. $\text{FV}((\neg\varphi)[t/x]) = \text{FV}(\varphi[t/x]) = E(\varphi) = E(\neg\varphi)$, by the induction hypothesis (t is free for x in φ , and x is free in φ).

Binary connective. Let x be free in $(\varphi \circ \psi)$; then t is free for x in both parts, and x is free in

at least one. If it is free in both, the induction hypothesis gives $E(\varphi) \cup E(\psi) = E((\varphi \circ \psi))$. If it is free in φ only, then $\psi[t/x] = \psi$, and $E(\varphi) \cup \text{FV}(\psi) = E((\varphi \circ \psi))$, because $x \notin \text{FV}(\psi)$. Symmetrically if it is free in ψ only.

Solution (Hint 1). When x is free in $\forall y \varphi$, what does “free for” tell you about y and t ?

Solution (Hint 2). Start from $\text{FV}((\forall y \varphi)[t/x]) = \text{FV}(\varphi[t/x]) \setminus \{y\}$ and apply the induction hypothesis to φ . Then see where removing y from the resulting set needs something about y and $\text{Var}(t)$.

Solution. (a) The term version and the atomic, negation and connective cases are given in the statement. It remains to treat the quantifiers.

Quantifier. If x is not free in $\forall y \varphi$, then $(\forall y \varphi)[t/x] = \forall y \varphi$ by the claim of the text. Now let x be free in $\forall y \varphi$. Then y is not x and x is free in φ ; and since t is free for x in $\forall y \varphi$ while x is free there, $y \notin \text{Var}(t)$ and t is free for x in φ (the recursive description of “free for”). So

$$\begin{aligned} \text{FV}((\forall y \varphi)[t/x]) &= \text{FV}(\varphi[t/x]) \setminus \{y\} = ((\text{FV}(\varphi) \setminus \{x\}) \cup \text{Var}(t)) \setminus \{y\} \\ &= (\text{FV}(\varphi) \setminus \{x, y\}) \cup \text{Var}(t) = E(\forall y \varphi), \end{aligned}$$

where the third equality uses $y \notin \text{Var}(t)$, and the last uses $\text{FV}(\forall y \varphi) = \text{FV}(\varphi) \setminus \{y\}$. The case of $\exists y$ is the same.

This proves $S(\varphi)$ for every formula, and (a) is its first half.

(b) With $\varphi = \exists y(y > x)$ and $t = y + 1$: $\text{FV}(\varphi) = \{x\}$, so the equation would give $\text{FV}(\varphi[t/x]) = \{y\}$. But $\varphi[t/x] = \exists y(y > y + 1)$, whose set of free variables is $\emptyset$. The y of the term was captured, so it is not free in the result; the quantifier case of the proof fails because $y \in \text{Var}(t)$.

For the second hypothesis, take $\varphi = P(z)$ and $t = y$. Then x is not free in φ , and t is free for x (there are no quantifiers). Now $\varphi[t/x] = P(z)$, with free variables $\{z\}$, while the equation would give $\{z\} \cup \{y\} = \{y, z\}$. Without a free x , nothing is put in, so the variables of t do not appear.

Marking scheme. (10 points)

- 3: the quantifier case reduced to $\text{FV}(\varphi[t/x]) \setminus \{y\}$ by the definitions.
- 3: the use of “free for” to get $y \notin \text{Var}(t)$ and t free for x in φ , naming where it is used.
- 1: the case “ x not free in $\forall y \varphi$ ” handled by the claim of the text.
- 3: both counterexamples of (b), each checked by computing both sides.

Fatal errors (0 for the step): using the equation $\text{FV}(\chi[t/x]) = (\text{FV}(\chi) \setminus \{x\}) \cup \text{Var}(t)$ for a part χ in which x is not free (it fails there; see the second example in (b)); removing y from $\text{Var}(t)$ without justification.

Exercise. C2: Renaming is not swapping

- (a) Show that $\forall x \exists y (x < y)$ and $\forall y \exists x (y < x)$ are the same sentence up to renaming bound variables: give a sequence of renamings, each allowed by the condition of the text, leading from the first to the second.
- (b) Show that $\forall x \exists y (x < y)$ and $\exists y \forall x (x < y)$ are **not** the same up to renaming. *Hint: what does a renaming leave unchanged?*

Solution. (a) A renaming needs a variable that does not appear in the formula, so the two letters cannot be exchanged in one step; go through a third letter z .

1. In $\forall x \exists y (x < y)$, rename the bound x to z , which does not appear: $\forall z \exists y (z < y)$.
2. Rename the bound y to x , which no longer appears: $\forall z \exists x (z < x)$.
3. Rename the bound z to y , which no longer appears: $\forall y \exists x (y < x)$.

Each step renames one quantifier and the occurrences it binds, to a letter absent from the formula at that moment. The result is the second sentence. (Over $\mathbb{N}$ both say “there is no largest natural number”, in two spellings.)

(b) A renaming changes only which **letters** are used. It does not change the parse tree’s shape, which quantifier symbol ($\forall$ or $\exists$) sits at which node, or which quantifier node binds which occurrence. In $\forall x \exists y (x < y)$ the root is a $\forall$ node; in $\exists y \forall x (x < y)$ the root is an $\exists$ node. No sequence of renamings turns a $\forall$ node into an $\exists$ node, so no sequence of renamings leads from one sentence to the other.

The two sentences also say different things: over $\mathbb{N}$, the first is true (take $y = x + 1$) and the second false (for any y , take $x = y$). This is Example (Swapping two quantifiers) again. Renaming changes the **names** of the placeholders; swapping changes the **order** in which they are chosen, and the order is what carries the meaning.