Lecture Notes on Substitution in FOL

Introduction to Logic (Spring 2020)

Henning Basold

In these notes, we will discuss an important operation of first-order logic: the substitution of a term for a variable. This operation will replace any occurrence of a variable in a formula by the given term. For instance, we will be able to substitute the term $f(m, x)$ for the variable y in the formula $P(y) \wedge Q(r, y)$ to obtain

$$P(f(m, x)) \wedge Q(r, f(m, x)).$$

However, the substitution operation is surprisingly complex, as we need to deal with variables that are bound by quantifiers. The aim of these notes is to give a rigorous presentation of variables and binding that allows us to safely carry out substitutions.

1 The Difficulty of Names and Variables

Let us first discuss two questions that arise in FOL:

1. Are the formulas $\forall x.\, P(x)$ and $\forall y.\, P(y)$ expressing the same?
2. What is the scope of variables?

Formula Equality and Renaming The first question can be answered by reading the formula without explicitly naming variables:

> "P holds for all objects".

Clearly, this sentence corresponds to both formulas $\forall x.\, P(x)$ and $\forall y.\, P(y)$, and we should consider both formulas to be the same, already *syntactically*! This gives us a first rule that we will have to adhere to for FOL:

> Two formulas are considered to be the same, if we can *bijectively rename* the variables of one formula to obtain the other.

For instance, consider the formulas $\varphi = \forall x.\, \forall y.\, Q(x, y)$ and $\psi = \forall w.\, \forall z.\, Q(w, z)$. Then, a bijective renaming would be to rename x to w and y to z, which allows us to transform φ into ψ. However, renaming x to r and y to r is not bijective. Thus, we do not consider the formula $\forall r.\, \forall r.\, Q(r, r)$ to be same as φ. Note, that in the latter formula, the r refers to the inner-most quantifier and the outer quantifier has no effect!

(a) The plain tree (b) Tree with indication of the variable references

Figure 1: de Bruijn-Tree Representations of $\forall x.\, \forall y.\, Q(x, f(y, z))$

Scoping The second question concerns the scope of variables. This requires us to determine which objects a variable refers to. For instance, the variable x in the subformula $P(x)$ of $\forall x.\, P(x)$ refers to what the quantifier ranges over. We say that x is *in the scope* of $\forall x$ in this formula. However, the variable y in $\forall x.\, Q(x, y)$ is in the scope of no quantifier and is thus a global reference. If we were now naively substituting x for y in this formula, then we would obtain $\forall x.\, Q(x, x)$. Here, the scope of the second argument of Q, and with it the meaning of the formula, has suddenly changed. This leads to a second rule:

> Substituting a term in a formula should not change the scoping of variables.

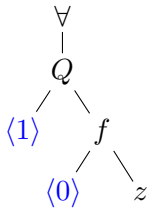
2 De Bruijn Trees

There are several ways to deal with variables, binding and substitution. An intuitively understandable way is to represent terms and formulas as *de Bruijn trees*. The idea is that bound variables are represented by a number that points to the quantifier that binds this variables. All free variables keep their name.

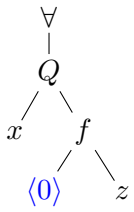
Figure 1 shows the (de Bruijn-) tree representation of the formula $\forall x.\, \forall y.\, Q(x, f(y, z))$, where the left figure shows the actual tree and the right figure indicates to which quantifier the numerical name refers. We see that the bound variables are numbered starting from the *inner-most* quantifier. Also note that the variable z is free and thus keeps its name in the tree representation.

Since the nesting depth of quantifiers is important, we also note that tree representations are *not closed under subtrees*! For instance, the tree in fig. 2a is not a valid tree representation because $\langle 1 \rangle$ is a dangling reference. Instead, if we want to remove the outer quantifier, we need to pick a name that we replace $\langle 1 \rangle$ with, say x, and then obtain the tree representation in fig. 2b.

In what follows, we will not work explicitly with tree representations, but will use another approach, see section 3 below. However, the tree representation shows how we can solve the initial problems:



(a) Invalid de Bruijn-Tree



(b) Correct tree representing $\forall y.\, Q(x, f(y, z))$

Figure 2: Attempts of finding trees that represent the subformula $\forall y.\, Q(x, f(y, z))$ of $\forall x.\, \forall y.\, Q(x, f(y, z))$

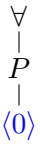
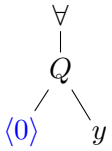


Figure 3: De Bruijn-Tree representing both formulas $\forall x.\, P(x)$ and $\forall y.\, P(y)$

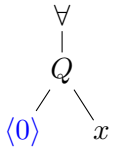
1. The formulas $\forall x.\, P(x)$ and $\forall y.\, P(y)$ have the same tree representation, see fig. 3.
2. Substitutions can be carried out without changing the binding of variables, see fig. 4. Note that the *bound* variable x is represented by $\langle 0 \rangle$ and thus there is no danger of substituting the *unbound* variable x for the *unbound* variable y. Hence, the problem of accidentally binding a variable does not exist for the tree representation.

3 Axiomatising Terms and Substitutions

The tree representations of formulas that we saw in section 1 solves our problems but it is difficult to work with. In fact, these trees are good way of *implementing* FOL on a computer, but not for humans to work with. The aim of this section is to introduce a bunch of axioms that formulas and substitution have to fulfil. These axioms are fulfilled if we were to represent terms by de Bruijn trees, but for the remainder of the course, we will leave unspecified how formulas are implemented.



(a) Representation before substitution



(b) Representation after substitution

Figure 4: Substitution of the free variable x for the free variable y by using the tree representation of $\forall x.\, Q(x, y)$

We begin with the definition of substitutions and how they can be applied to terms.

Definition 1. A *substitution* is a map $\sigma\colon \mathrm{Var} \to \mathrm{Term}$. Given $t \in \mathrm{Term}$ and $x \in \mathrm{Var}$, we write $\sigma[x := t]$ for the *updated* substitution defined by

$$\sigma[x := t](y) = \begin{cases} t, & x = y \\ \sigma(y), & x \neq y \end{cases}.$$

Similarly, we write $[x := t]$ for the substitution given by

$$[x := t](y) = \begin{cases} t, & x = y \\ y, & x \neq y \end{cases}.$$

Given a term t, we write $t[\sigma]$ for the *application* of the substitution σ to t, defined by iteration on terms as follows.

$$\begin{aligned} x[\sigma] &= \sigma(x) \\ c[\sigma] &= c \\ f(t_1, \ldots, t_n)[\sigma] &= f(t_1[\sigma], \ldots, t_n[\sigma]) \end{aligned}$$

The notation $\sigma[x := t]$ to update a substitution σ should be read like an assignment in an imperative programming language: the term that σ assigned to x will be overwritten by t. Similarly, the notation $[x := t]$ starts with a storage in which all variables y have the default value y, except for x which gets t assigned as initial value. It should also be noted that there is a substitution $\eta\colon \mathrm{Var} \to \mathrm{Term}$ that assigns to each variable y the *term* y, that is, we have $\eta(y) = y$. The notation $[x := t]$ is then a shorthand for $\eta[x := t]$.

The following example illustrates these notations.

Example 2. Let $g(x, y)$ be a term with two free variables x and y, and one binary function symbol g. Moreover, let $\sigma = [x := y][y := x]$.

1. The substitution σ exchanges the two variables:

$$g(x, y)[\sigma] = g(x[\sigma], y[\sigma]) = g(y, x)$$

2. Let now $\tau = \sigma[y := f(x)]$. In τ, the assignment of x to y is overwritten with the assignment of $f(x)$ to y. We thus have the following.

$$g(x, y)[\tau] = g(x[\tau], y[\tau]) = g(y, f(x))$$

3. It is sometimes convenient to exploit the notation and leave out square brackets. For example, instead of $g(x, y)[[y := f(x)]]$ we could write $g(x, y)[y := f(x)]$ and obtain $g(x, y)[y := f(x)] = g(x, f(x))$.

Next, we need to be able to apply substitutions to formulas. To circumvent the difficulties described in section 1, we *assume* that there is a set of terms on which we can carry out substitutions. This set of terms can *in principle* be defined by appealing to the tree representation. However, this is tedious and instead we just assume that formulas and the application of substitutions fulfil certain *axioms*.

Assumption 3. Given a formula φ, a variable x and a substitution σ, we say that x is *fresh* for σ in φ, if we have for all $y \in \mathrm{fv}(\varphi) \setminus \{x\}$ that $x \notin \mathrm{fv}(\sigma(y))$. We assume that there is an operation $\varphi[\sigma]$ that applies a substitution σ to an FOL formula φ. Further, we assume for all $\Diamond \in \{\forall, \exists\}$ and $\Box \in \{\wedge, \vee, \to\}$ that the equality on formulas fulfils the following six axioms.

$$\varphi \Box \psi = \varphi' \Box \psi' \text{ iff } \varphi = \varphi' \text{ and } \psi = \psi' \tag{EC}$$

$$\Diamond x.\, \varphi = \Diamond y.\, \psi \text{ iff } y \notin \mathrm{fv}(\varphi) \text{ and } \psi = \varphi[x := y] \tag{EQ}$$

$$\bot[\sigma] = \bot \tag{SB}$$

$$P(t_1, \ldots, t_n)[\sigma] = P(t_1[\sigma], \ldots, t_n[\sigma]) \tag{SP}$$

$$(\varphi \Box \psi)[\sigma] = \varphi[\sigma] \Box \psi[\sigma] \tag{SC}$$

$$(\Diamond x.\, \varphi)[\sigma] = \Diamond x.\, \varphi[\sigma[x := x]] \text{ if } x \text{ is fresh for } \sigma \text{ in } \varphi \tag{SQ}$$

Let us explain the intuition behind these axioms. First of all, there are two groups of axioms: those that define the equality on formulas (starting with E) and those that define the action of substitution on formulas (starting with S). The axiom (EQ) allows us to bijectively rename bound variables without illegally binding other variables. For instance, we have

$$\forall x.\, Q(x, y) = \forall z.\, Q(z, y)$$

because $z \notin \mathrm{fv}(Q(x, y))$. However, we have

$$\forall x.\, Q(x, y) \neq \forall y.\, Q(y, y)$$

because $y \in \mathrm{fv}(Q(x, y))$. The axiom (EC) allows us to carry out this renaming also in complex formulas that involve other connectives than quantifiers. Implicitly, we also use equality on terms and atoms, in the sense that

$$P(t_1, \ldots, t_n) = P'(t'_1, \ldots, t'_n) \text{ iff } P = P' \text{ and } t_k = t'_k \text{ for all } 1 \leq k \leq n$$

and that equality is an equivalence relation (reflexive, symmetric and transitive).

The second group of axioms describes how substitution can be computed iteratively. Axioms (SB), (SP) and (SC) are doing what we would expect: no action on the atom $\bot$, reduce subsitution on predicates to substitution in terms, and distribute substitution over propositional connectives. Complications arise only in the axiom (SQ), which has to make sure that the *use of a bound variable is not changed* and that *variables are not accidentally bound*. On the one hand, that the use of a bound variable is not changed is achieved by updating the substitution σ to $\sigma[x := x]$. For instance, if $\sigma(x) = g(y)$, then

naively carrying out this substitution on $\forall x.\, P(x)$ would lead to $\forall x.\, P(g(y))$, which is certainly not what we want! Instead, we have by (SQ) and (SP)

$$(\forall x.\, P(x))[\sigma] = \forall x.\, P(x)[\sigma[x := x]] = \forall x.\, P(x).$$

Accidental binding is preventing, on the other hand, by the precondition that x must be fresh for σ in φ. This condition ensures that none of the terms we want to substitute for the free variables in φ contains the variable x, which would become then bound by the quantifier.

These rules and their interaction are best illustrated through some examples.

Example 4. In the following, we use the substitution σ given by

$$\sigma = [y := x][z := m].$$

1. Let $\varphi = \forall z.\, Q(z, y)$. First, we note that $\mathrm{fv}(Q(z, y))\backslash\{z\} = \{y\}$ and $\mathrm{fv}(\sigma(y)) = \{x\}$. Thus, $z \notin \mathrm{fv}(\sigma(y))$ and z is fresh. This gives us

$$\begin{aligned} \varphi[\sigma] &= \forall z.\, Q(z, y)[\sigma[z := z]] && \text{(SQ)} \\ &= \forall z.\, Q(z[\sigma[z := z]], y[\sigma[z := z]]) && \text{(SP)} \\ &= \forall z.\, Q(z, x) \end{aligned}$$

   Note that $\sigma[z := z] = [y := x]$ and the substitution of m for z in σ was "forgotten" when we applied the substitution under the quantifier. This is intuitively expected, as the bound variable z in φ is a local reference, while the z in σ refers to a global variable z that has the same name but is distinct from the local variable.

2. Let $\psi = \forall x.\, Q(x, y)$. First, we note that $\mathrm{fv}(Q(x, y))\backslash\{x\} = \{y\}$ and $\mathrm{fv}(\sigma(y)) = \{x\}$. Thus, $x \in \mathrm{fv}(\sigma(y))$ and x is *not* fresh. However, we have $z \notin \mathrm{fv}(Q(x, y))$ and $Q(x, y)[x := z] = Q(z, y)$. This allows us to rename the bound variable x in ψ to z and then carry out the substitution as above:

$$\begin{aligned} \psi[\sigma] &= (\forall z.\, Q(z, y))[\sigma] && \text{(EQ)} \\ &= \forall z.\, Q(z, x) && \text{by 1.} \end{aligned}$$

   Note that we cannot safely rename z back to x, as we would otherwise illegally bind x.

In the remainder of the course, we will not make explicit use of the axioms provided in assumption 3. Instead, we will rename *bound* variables whenever necessary before carrying out substitutions. For instance, we would just write

$$(\forall x.\, Q(x, y))[y := x] = \forall z.\, Q(z, x)$$

without explicitly referring to the axioms. However, we know that in cause of doubt, we can always go back to the axioms and formally carry out the renaming and substitution.

4 Exercises

The following exercises allow you to practise the material of these lecture notes. These exercises are not meant as homework but are strongly advised to be done before the next lecture.

Exercise 1

Give the de Bruijn-tree representations of the following formulas.

a) $Q(x, g(m))$

b) $\forall x.\, Q(x, g(m))$

c) $(\exists x.\, Q(x, g(m))) \wedge \forall y.\, Q(y, z)$

d) $\exists y.\, Q(y, g(z)) \wedge \forall x.\, Q(y, f(x, z))$

e) $\forall x.\, \exists x.\, P(x)$

Exercise 2

Let σ be given by

$$\sigma = [x := g(x)][y := x][z := y].$$

Evaluate the following applications of σ to terms.

a) $x[\sigma]$

b) $f(x, y)[\sigma]$

c) $(y[\sigma])[\sigma]$

d) $x[\sigma[x := x]]$

Exercise 3

Let σ be given by

$$\sigma = [x := g(x)][y := x][z := y]$$

and φ by

$$\varphi = \forall x.\, Q(y, z).$$

Determine whether

a) x is fresh for σ in φ.

b) y is fresh for σ in φ.

c) z is fresh for σ in φ.

Exercise 4

Let σ be given by

$$\sigma = [x := g(x)][y := x][z := y].$$

Carry out the following substitutions.

a) $Q(x, g(m))[\sigma]$

b) $(\forall x.\, Q(x, g(m)))[\sigma]$

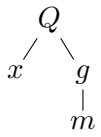
c) $((\exists x.\, Q(x, g(m))) \wedge \forall y.\, Q(y, z))[\sigma]$

d) $(\exists y.\, Q(y, g(z)) \wedge \forall x.\, Q(y, f(x, z)))[\sigma]$

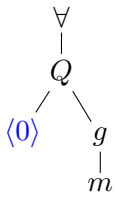
Solutions to the Exercises

Solution 1

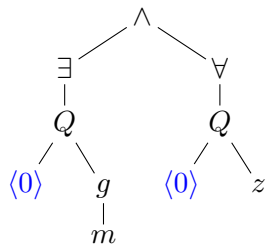
a)



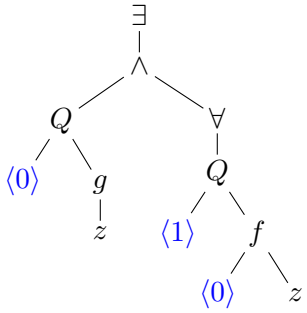
b)



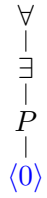
c)



d)



e)



Solution 2

a) $x[\sigma] = \sigma(x) = g(x)$

b) $f(x, y)[\sigma] = f(\sigma(x), \sigma(y)) = f(g(x), x)$

c) $(y[\sigma])[\sigma] = x[\sigma] = g(x)$

d) $x[\sigma[x := x]] = \sigma[x := x](x) = x$

Solution 3

We have $\mathrm{fv}(\varphi) = \{y, z\}$.

a) x is not fresh, since $x \in \mathrm{fv}(\sigma(y))$.

b) y is not fresh, since $y \in \mathrm{fv}(\sigma(z))$.

c) z is fresh, because $\mathrm{fv}(\varphi) \setminus \{z\} = \{y\}$ and $z \notin \mathrm{fv}(\sigma(y))$.

Solution 4

a) $Q(x, g(m))[\sigma] = Q(g(x), g(m))$

b) Since x is fresh for σ in this formula, we have

$$(\forall x.\, Q(x, g(m)))[\sigma] = \forall x.\, Q(x, g(m))[\sigma[[x := x]] = \forall x.\, Q(x, g(m)).$$

c)

$$\begin{aligned}((\exists x.\, Q(x, g(m))) \wedge \forall y.\, Q(y, z))[\sigma] &= ((\exists x.\, Q(x, g(m))))[\sigma] \wedge (\forall y.\, Q(y, z))[\sigma] \\ &= (\exists x.\, Q(x, g(m))) \wedge (\forall y.\, Q(y, z))[\sigma] \\ &= (\exists x.\, Q(x, g(m))) \wedge (\forall y'.\, Q(y', z))[\sigma] \\ &= (\exists x.\, Q(x, g(m))) \wedge (\forall y'.\, Q(y', z)[\sigma]) \\ &= (\exists x.\, Q(x, g(m))) \wedge (\forall y'.\, Q(y', y))\end{aligned}$$

d)

$$\begin{aligned}(\exists y.\, Q(y, g(z)) \wedge \forall x.\, Q(y, f(x, z)))[\sigma] &= (\exists y'.\, Q(y', g(z)) \wedge \forall x.\, Q(y', f(x, z)))[\sigma] \\ &= \exists y'.\, (Q(y', g(z)) \wedge \forall x.\, Q(y', f(x, z)))[\sigma] \\ &= \exists y'.\, Q(y', g(z))[\sigma] \wedge (\forall x.\, Q(y', f(x, z)))[\sigma] \\ &= \exists y'.\, Q(y', g(y)) \wedge (\forall x.\, Q(y', f(x, z)))[\sigma] \\ &= \exists y'.\, Q(y', g(y)) \wedge \forall x.\, Q(y', f(x, z))[\sigma] \\ &= \exists y'.\, Q(y', g(y)) \wedge \forall x.\, Q(y', f(x, y))\end{aligned}$$