Notes on Substitution in First–Order Logic

Marcus Kracht
kracht@humnet.ucla.edu
Department of Linguistics, UCLA, 3125 Campbell Hall, PO Box 951543, Los Angeles, California, CA 90095-1543, USA

1 Introduction

Axiomatisations of first–order logic (henceforth FOL) are given by means of a finite list of axioms or axiom schemes, each of which represents an infinite set of actual formulae. This latter set is either derived by means of a rule of substitution into an actual axiom or by properly instantiating the axiom scheme, putting actual formulae in place of the metavariables. Either way, the presentation rests on an understanding of how one properly replaces formulae (or terms) by other formulae (terms). Though any of these operations can be rigorously defined, we shall ask whether the grammar of FOL actually suggests a notion of replacement to begin with. The ideas behind this derive from [5], where we scrutinize the notion of substitution in linguistic theory and the role it plays in structural linguistics as well as modern logic. One of the underpinnings of this research was the assumption that there is no independent notion of substitution — substitution is canonically defined on the basis of the grammar and the analyses it provides for linguistic expressions. The outcome for FOL is that substitution is simply string replacement of one constituent by another. This substitution is also simpler than the standard one. This result raises the question why substitution (and not even general substitution, which allows for a change of bound variables, as in λ–terms) is chosen for FOL. For with some extra effort it can be seen that this makes no real difference for the axiomatization. While changing to string substitution seems like an unnecessary complication, the present result vindicates to a certain extent the instinctive mistrust of a novice in FOL against using a variable in the same formula both free and bound. And it may help in understanding why these difficulties exist.

2 Motivation

Consider the following sentence

```
Achilles is faster than the tortoise.
```

Suppose we replace `fast` by `wise`, `good` or `bad`. Then we expect to find the following.

```
Achilles is wiser than the tortoise.
Achilles is better than the tortoise.
Achilles is worse than the tortoise.
```

Thus, we do *not* get *`wiseer`, *`gooder`, nor *`bader`. So, in none of these cases the substitution is mere string substitution. As for the first, the fact that we have one `e` rather than two is due to a general writing convention. In the case of `good` the stem changes its form in the comparative. In the case of `bad`, it is both the stem and the comparative morpheme that change form. So, we find that sometimes the thing that we put into the hole ('filler') changes form, that sometimes only the container changes form, and sometimes both. The explanation is as follows. We analyze the comparative form as a combination of stem plus comparative suffix. We can represent this as $\sigma \bullet \gamma$, where $\bullet$ is a binary symbol of combination. Here σ and γ are not necessarily strings but they can be more complex. In this case, however, they are strings. Moreover, $\bullet$ can be something else than mere string concatenation. In the present case, the actual form of $\sigma \bullet \gamma$ sometimes is the concatenation of σ and γ (`fast`$^\frown$`er`), sometimes the concatenation with one `e` removed (`wise`$^\frown$`r`). Or σ is changed before concatenation, or it is γ that changes, or even both.

Each string has a term associated with it, a so–called **analysis**, of which the string is a **representative**. The term defines the string uniquely, but the relation between terms and strings may be many–to–one. If it is one–to–one, the language is said to be **uniquely readable**. This is a property that logical languages are required to have. While both stem and suffix are generally considered to be strings, this is in actual fact only a simplification (see [5]). On a more abstract level of analysis, however, the situation is simply as follows. The 'container' has an analysis $\mathfrak{s}(x)$ in which there is a hole x (which may contain several occurrences of x). The filler has the analysis $\mathfrak{t}$ which we put in place of x and obtain $\mathfrak{s}(\mathfrak{t})$. The term $\mathfrak{s}(\mathfrak{t})$ both determines the form of the expression (the string) and the meaning. Substitution is the replacement of $\mathfrak{t}$ by a different term $\mathfrak{u}$, giving $\mathfrak{s}(\mathfrak{u})$ in place of $\mathfrak{s}(\mathfrak{t})$.

Logical languages are man made. Therefore we do not expect the pathological examples of natural languages to exist. In particular, we expect that the objects that we manipulate are simply strings, and that the operation that forms constituents is simply concatenation. This is the case in propositional logic. For example, look at

$$\texttt{((p0}\wedge\texttt{(}\neg\texttt{p01))}\to\texttt{p0)}$$

We can think of this string as being obtained from

$$\texttt{((p0}\wedge\texttt{(}\neg\underline{\hspace{3em}}\texttt{))}\to\texttt{p0)}$$

by inserting the string `p01`. If we substitute something else for `p01`, we just replace the string occurrence of `p01` by whatever substitutes it. Similarly, replacing one or two occurrences of `p0` by another variable is simply string replacement.

We notice here right away that in addition to the official definition there exist 'dialects' of logical languages obtained by changing more or less drastically the strings that represent a formula (not to speak of the famous Begriffsschrift of [2]). For example, brackets are often dropped, variables are denoted by metavariables, and more function symbols are added. While logicians abstract from these changes and deal with the strings as imperfect representatives (in place of the correct ones), we shall treat these strings as objects in their own right. The approach is thus not normative, it is descriptive. It is not our aim to prescribe which strings constitute the language, we are interested in what happens if they are one way or another.

3 Sign Systems

Definition 1 *A **sign** is a triple $\sigma = \langle e, c, m\rangle$, where e is the **exponent** of σ, c its **category**, and m its **meaning**. A **language** is a set of signs.*

A **signature** is a function $\Omega\colon F \to \mathbb{N}$ for an arbitrary set F of function symbols. A (partial) Ω–algebra is a pair $\mathfrak{A} = \langle A, \mathcal{I}\rangle$, where A is a set and for every $f \in F$, $\mathcal{I}(f)$ is a partial $\Omega(f)$–ary function on A.

Definition 2 *A* ***grammar*** *consists of*

1. *a finite set F of* ***modes,***
2. *a signature $\Omega\colon F \to \mathbb{N}$,*
3. *for every $f \in F$, partial $\Omega(f)$–ary functions f^{ε} on E, f^{γ} on C, f^{μ} on M*

A grammar G is a ***grammar for*** *$\Delta \subseteq E \times C \times M$ if the functions $\langle f^{\varepsilon}, f^{\gamma}, f^{\mu}\rangle$, $f \in F$, generate Δ from the empty set.*

[5] introduces a few more conditions on grammars. One is that functions are not allowed to destroy material of the exponents, the second that all functions are computable. A third condition is that the categories are distribution classes. These requirements need comment. First, the requirement of computability derives from the notion of compositionality; if we understand the meaning of a complex expression by applying a certain function to the meaning of its parts then we cannot strictly speaking understand the meaning of a complex expression if it is derived using a noncomputable function. This is awkward and we cannot go into the ramifications of this, but it has certainly been a concern in the foundation of mathematics (for example in intuitionism and constructivism). We shall not insist on computability. However, the other constraints shall be met.

To give an example, let Δ be the set of pairs $\langle \vec{x}, T, n\rangle$, where $\vec{x} \in \{\mathtt{0}, \mathtt{1}\}^*$ is a binary sequence and n is the number that $\vec{x}$ represents in binary. This set can be generated from the zeroary functions $f_0 := \langle \mathtt{0}, T, 0\rangle$ and $f_1 := \langle \mathtt{1}, T, 1\rangle$ and two unary functions, f_2 and f_3.

$$\begin{array}{ll} f_2(\langle \vec{x}, T, n\rangle) & := \langle \vec{x}^\frown\mathtt{0}, T, 2n\rangle \\ f_3(\langle \vec{x}, T, n\rangle) & := \langle \vec{x}^\frown\mathtt{1}, T, 2n+1\rangle \end{array}$$

So, put $F := \{f_0, f_1, f_2, f_3\}$, $\Omega : f_0, f_1 \mapsto 0; f_2, f_3 \mapsto 1$.

4 A Grammar for Generating the Exponents

We first deal with the **morphology** of predicate logic. The exponents of signs are also called (**well–formed**) **expressions** (**wfes**). Predicate logic is given by its well–formed expressions and their meanings. We assume that the expressions are strings and that the only operation to form complex expressions is concatenation (and no blank is inserted). It turns out that there is a context free grammar generating the wfes.

For concreteness' sake, there will be a unary predicate symbol `p`, a binary function symbol `+` and a binary relation symbol `=`. The exponents must be strings over a *finite* alphabet A. Thus, it is not possible to assume an infinite supply of primitive symbols. So we put

$$A := \{\neg, \rightarrow, +, \forall, (,), \texttt{0}, \texttt{1}, \texttt{p}, \texttt{x}, =\}$$

A **variable** is a sequence $\texttt{x}\vec{\alpha}$, where $\vec{\alpha}$ is a binary sequence (a decimal representation would be just as fine, but changes nothing in principle).

Definition 3 *A* ***context free grammar*** *is a quadruple $\langle S, N, A, R\rangle$, where S is the* ***start symbol****, N the set of* ***nonterminals****, A the* ***alphabet*** *and R the set of* ***rules****.*

A is as above, $N = \{\langle\text{variable}\rangle, \langle\text{term}\rangle, \langle\text{formula}\rangle\}$, $S = \langle\text{formula}\rangle$. Here is now the set of rules.

$$\begin{array}{lll}
\langle\text{variable}\rangle & ::= & \texttt{x} \mid \langle\text{variable}\rangle{}^\frown\texttt{0} \mid \langle\text{variable}\rangle{}^\frown\texttt{1}\\
\langle\text{term}\rangle & ::= & \langle\text{variable}\rangle \mid (^\frown\langle\text{term}\rangle{}^\frown+^\frown\langle\text{term}\rangle{}^\frown)\\
\langle\text{formula}\rangle & ::= & \texttt{p}^\frown(^\frown\langle\text{term}\rangle{}^\frown) \mid (^\frown\langle\text{term}\rangle{}^\frown=^\frown\langle\text{term}\rangle{}^\frown)\\
 & & \mid (^\frown\neg^\frown\langle\text{formula}\rangle{}^\frown)\\
 & & \mid (^\frown\ \langle\text{formula}\rangle{}^\frown\rightarrow^\frown\langle\text{formula}\rangle{}^\frown)\\
 & & \mid (\forall^\frown\langle\text{variable}\rangle{}^\frown)^\frown\langle\text{formula}\rangle
\end{array}$$

A **context** is a pair $C = \langle\vec{u}_1, \vec{u}_2\rangle$ of strings. If $\vec{y}$ is a string, then $C(\vec{y}) := \vec{u}_1\vec{y}\vec{u}_2$. Let $\vec{x} = \vec{u}_1\vec{y}\vec{u}_2$ be a string. We say that $\vec{y}$ is a **substring** of $\vec{x}$, and say that $C = \langle\vec{u}_1, \vec{u}_2\rangle$ is an **occurrence** of $\vec{y}$ in $\vec{x}$. Given a CFG $G = \langle S, N, A, R\rangle$, if there is a derivation $Y \Rightarrow_G^* \vec{u}_1 X \vec{u}_2$ and $\vec{y}$ is an X–string, we say that the occurrence $\langle\vec{u}_1, \vec{u}_2\rangle$ of $\vec{y}$ in $\vec{x}$ is a **constituent occurrence** (**of category** X). The **distribution classes** are the following sets (where S is the start symbol of G).

$$\mathrm{Dist}_G(\vec{y}) := \{\langle\vec{u}_1, \vec{u}_2\rangle : (\exists X \in N)(S \Rightarrow_G^* \vec{u}_1 X \vec{u}_2 \text{ and } X \Rightarrow_G^* \vec{y})\}$$

If $\vec{x}$ and $\vec{y}$ are X–strings, then they define the same distribution class. The converse need not hold. If the converse holds, the grammar is called **balanced**.

The grammar above is balanced. This grammar defines in total 3 different distribution classes: that of variables, of terms and of formulae. And they correspond exactly to the nonterminals used above. This contrasts with predicate logic as defined in textbooks, where only terms and formulae are recognized. The category of a variable is motivated on purely distributional grounds: only a variable can occur right after $\forall$. Indeed, it is generally assumed that a variable has no occurrence after a quantifier, so that substitution will not touch it. The grammar speaks a slightly different language: the variable occurs as a *variable*, but not as a *term*. And term substitution targets exclusively the term occurrences. (Often, variable substitutions are also considered, known as 'replacement of bound variables' or 'α–conversion'.)

5 A Grammar for FOL

Now we shall propose a grammar for FOL in the sense of our earlier definition. Put $E := A^*$ and $C := \{\nu, \tau, \varphi\}$. A **structure** is defined as usual, except that we fix the underlying domain to be a cardinal number. This makes the class of models a set. Clearly, every ordinary structure has an isomorphic structure of that kind so that we do not loose anything. Let $\mathbb{N}$ be the set of natural numbers. A **valuation** in a structure $\mathfrak{M}$ is a function from $\mathbb{N}$ to the carrier set of $\mathfrak{M}$. A **model** is a pair $\langle \mathfrak{M}, \beta \rangle$, where $\mathfrak{M}$ is a structure and β a valuation. An **index** is a function from models to truth values. (The set of truth values is as usual $\{0, 1\}$.) A **point** is a function from models to elements of the carrier set. $\mathbb{H}$ is the set of indices over countable models, $\mathbb{P}$ the set of points over countable models. Then

$$M := \mathbb{N} \cup \mathbb{P} \cup \mathbb{H}$$

(These three sets correspond to the three categories ν, τ and φ.) For a number m, let $\pi(m)$ be the point such that $\pi(m)(\langle \mathfrak{M}, \beta \rangle) = \beta(m)$. If p and q are points, $p + q$ is defined on $\mathfrak{M} = \langle M, \mathfrak{I} \rangle$ by

$$(p + q)(\langle \mathfrak{M}, \beta \rangle) := \mathcal{I}(\mathsf{+})(p(\langle \mathfrak{M}, \beta \rangle), q(\langle \mathfrak{M}, \beta \rangle))$$

Further, $(=(p,q))(\langle\mathfrak{M},\beta\rangle) := 1$ iff $p(\langle\mathfrak{M},\beta\rangle) = q(\langle\mathfrak{M},\beta\rangle)$. We assume the following functions on truth values:

	$-$
0	1
1	0

$\to$	0	1
0	1	1
1	0	1

Then we put

$$\begin{array}{ll} (-i)(\langle\mathfrak{M},\beta\rangle) & := -(i(\langle\mathfrak{M},\beta\rangle)) \\ (i \to j)(\langle\mathfrak{M},\beta\rangle) & := i(\langle\mathfrak{M},\beta\rangle) \to j(\langle\mathfrak{M},\beta\rangle) \end{array}$$

Finally,

$$A(n,i)(\langle\mathfrak{M},\beta\rangle) := 1 \Leftrightarrow \text{ for all } \beta' \sim_n \beta : i(\langle\mathfrak{M},\beta\rangle) = 1$$

We propose a zeroary mode Z, unary modes N, E, T, Q and binary modes X, P, G, C.

$$\begin{array}{ll} \mathsf{Z} & := \langle \mathtt{x}, \nu, 1\rangle \\ \mathsf{N}(\langle\vec{x},\nu,m\rangle) & := \langle \vec{x}^\frown\mathtt{0}, \nu, 2m\rangle \\ \mathsf{E}(\langle\vec{x},\nu,m\rangle) & := \langle \vec{x}^\frown\mathtt{1}, \nu, 2m+1\rangle \\ \mathsf{T}(\langle\vec{x},\rho,m\rangle) & := \langle \vec{x}, \tau, \pi(m)\rangle \\ \mathsf{P}(\langle\vec{x},\tau,p\rangle, \langle\vec{y},\tau,q\rangle) & := \langle \mathtt{(}^\frown\vec{x}^\frown\mathtt{+}^\frown\vec{y}^\frown\mathtt{)}, \tau, p+q\rangle \\ \mathsf{G}(\langle\vec{x},\tau,p\rangle, \langle\vec{y},\tau,q\rangle) & := \langle \mathtt{(}^\frown\vec{x}^\frown\mathtt{=}^\frown\vec{y}^\frown\mathtt{)}, \varphi, =(p,q)\rangle \\ \mathsf{Q}(\langle\vec{x},\varphi,i\rangle) & := \langle \mathtt{(}^\frown\neg^\frown\vec{x}^\frown\mathtt{)}, \varphi, -i\rangle \\ \mathsf{C}(\langle\vec{x},\varphi,i\rangle, \langle\vec{y},\varphi,j\rangle) & := \langle \mathtt{(}^\frown\vec{x}^\frown\to^\frown\vec{y}^\frown\mathtt{)}, \varphi, i \to j\rangle \\ \mathsf{X}(\langle\vec{x},\nu,n\rangle, \langle\vec{y},\varphi,i\rangle) & := \langle (\forall^\frown\vec{x}^\frown\mathtt{)}^\frown\vec{y}, \varphi, A(n,i)\rangle \end{array}$$

The grammar just defined is called PRED. Notice that the functions so defined are not computable. However, this is a deficit of predicate logic in general.

6 Substitution

We shall consider three types of substitution. String substitution is standard in propositional logic. We decompose the string $\vec{x} = C(\vec{y})$, then replacing the occurrence C of $\vec{y}$ by $\vec{z}$ is $C(\vec{z})$. This substitution however is a substitution that replaces proper occurrences of a subexpression only. It is not indiscriminate string substitution. We denote it by $\ulcorner t/x \urcorner$ (for the case of replacing a variable by a term, but the notation is analogously used in other cases).

The second type we find for example in [8]. We call it **standard substitution**, since it is the most widely used.

$$\begin{array}{ll}
[t/x]y & := \begin{cases} t & \text{if } y = x, \\ y & \text{else.} \end{cases} \\
[t/x]f(\vec{s}) & := f([t/x]s_0, \dots, [t/x]s_{n-1}) \\
[t/x]r(\vec{s}) & := r([t/x]s_0, \dots, [t/x]s_{n-1}) \\
[t/x](\neg\varphi) & := (\neg[t/x]\varphi) \\
[t/x](\chi\rightarrow\chi') & := ([t/x]\chi\rightarrow[t/x]\chi') \\
[t/x](\forall y)\chi & := \begin{cases} (\forall y)\chi & \text{if } y = x, \\ (\forall y)[t/x]\chi & \text{if } y \text{ not in } t \text{ or } x \text{ not free in } \chi, \\ (\forall y)\chi & \text{else.} \end{cases}
\end{array}$$

This is basically a substitution that replaces free occurrences of x by t. The third type, called **general substitution** in [7], executes a renaming of bound variables in last clause of the definition above whenever y is free in t and t actually occurs free in χ. We denote by $\{t/x\}\varphi$ the result of applying the generalized substitution to φ. (To make this into a function, y has to be chosen according to a fixed procedure, for example, choosing the smallest binary sequence possible.) Notice right away that these operations are only substitutions of variables by terms. Occasionally, however, authors do look at substitutions of predicates by predicates (see for example [4], 155–162).

Given a grammar, substitution is defined as follows. Call a **structure term** a well–formed term over the signature. Given a structure term, we can compute the sign that it denotes — if it denotes a sign at all. For, as some operations may be partial, some structure terms fail to denote signs; when they do denote a sign, however, it is unique. Structure terms can contain variables. (If they don't they are called **constant**.) Structure terms shall be written in Polish Notation. (This is an arbitrary choice of no significance.) For example, PTNZTEZ is a structure term, which defines the sign $\langle \texttt{x0+x1}, \tau, \pi(2) + \pi(3)\rangle$. Not all structure terms denote a sign. If they do, they are called **definite**. PTNZEZ is a structure term but not definite.

Definition 4 *Let $\mathfrak{s}$, $\mathfrak{t}$, $\mathfrak{t}'$ be structure terms, and let $\mathfrak{s}$ contain a single occurrence of x. Denote by $[\mathfrak{t}/x]\mathfrak{s}$ the substitution of $\mathfrak{t}$ for the occurrence of x in $\mathfrak{s}$. $[\mathfrak{t}'/x]\mathfrak{s}$ is said to be the result of substituting the occurrence of $\mathfrak{t}$ named by x by $\mathfrak{t}'$ in $\mathfrak{s}$.*

Definition 5 *Let σ and τ be signs. We say that τ is a **part of** σ **under the analysis** $\mathfrak{s}$ if there is a constant term $\mathfrak{t}$ and a term $\mathfrak{u}$ with a single* *occurrence of a free variable, x, such that $[\mathfrak{t}/x]\mathfrak{u} = \mathfrak{s}$, and $\mathfrak{s}$ unfolds to σ and $\mathfrak{t}$ unfolds to τ.*

In predicate logic as defined above, the notion of part is straightforward. A part always is a certain subexpression. Substitution is the replacement of such subexpressions by others. Moreover, it is substitution of variables by variables, of terms by terms, and of formulae by formulae.

Theorem 1 *The substitution defined by the grammar* Pred *is string substitution of a variable by a variable, of a term by a term, and of a formula by a formula.*

The operation might perhaps better be called *replacement* to avoid collision with the ordinary substitution of predicate logic, but we wish to contend that what we call here substitution is *the* linguistically appropriate one. To wit, the substitution we obtain on the level of exponents goes as follows. (We omit some obvious clauses.)

$$\begin{array}{lll}
\ulcorner t/x\urcorner\vec{u} & := [t/x]\vec{u}, & \text{if } \vec{u} \text{ is a term} \\
\ulcorner t/x\urcorner(\mathtt{(}^\frown\neg^\frown\vec{y}^\frown\mathtt{)}) & := \mathtt{(}^\frown\neg^\frown\ulcorner t/x\urcorner\vec{y}^\frown\mathtt{)} & \\
\ulcorner t/x\urcorner(\mathtt{(}^\frown\forall^\frown\vec{z}^\frown\mathtt{)}^\frown\vec{y}) & := \mathtt{(}^\frown\forall^\frown\vec{z}^\frown\mathtt{)}^\frown\ulcorner t/x\urcorner\vec{y} &
\end{array}$$

(The first line means that if $\vec{u}$ is a string denoting a term, then the substitution is simply ordinary substitution on terms. We shall sometimes use $\vec{u}$ to remind ourselves that the object in question is a string. t and x are also strings but we did not write $\vec{t}$ or $\vec{x}$, for example.)

In particular, the operation $\ulcorner t/x\urcorner$ does not care about the distinction between free and bound variables. There is a similar substitution of formulae by formulae, which once again is simple string replacement, and an operation of variable replacement. The latter changes all occurrences of a variable (including the ones in the quantifier prefixes), again disregarding free and bound occurrences.

Substitution also figures in what is known as **Leibniz' Principle**. Let L be a logic, here identified with its set of tautologies.

Definition 6 *A logic L is **Leibnizian** with respect to a sign grammar iff for any two structure terms $\mathfrak{t}$ and $\mathfrak{u}$, $\mathfrak{t}^\mu = \mathfrak{u}^\mu$ iff for all $\mathfrak{s}$ containing at most x free:*

1. *$[\mathfrak{t}/x]\mathfrak{s}$ is a sign iff $[\mathfrak{u}/x]\mathfrak{s}$ is.*

2. *If* $[\mathfrak{t}/x]\mathfrak{s}$ *is a formula,* $[\mathfrak{t}/x]\mathfrak{s}^\varepsilon \in L$ *iff* $[\mathfrak{u}/x]\mathfrak{s}^\varepsilon \in L$.

This is a formalization of the informal statement saying that two things have equal meaning iff they can be substituted for each other in all contexts. Notice that this definition is relative to the grammar and not just the language. This is so since it relies on the notion of substitution, which is not available in a language, only in a grammar.

Proposition 1 *Every predicate logic with identity is Leibnizian with respect to* Pred.

Proof. Suppose that φ and ψ are formulae with different meaning. Then without loss of generality there is a model in which φ is true and ψ is false. Therefore $(\varphi\to\psi) \notin L$, but $(\varphi\to\varphi) \in L$. Consider the terms $\mathfrak{t}$, $\mathfrak{u}$ such that $\mathfrak{t}^\varepsilon = \varphi$ and $\mathfrak{u}^\varepsilon = \psi$. Then put $\mathfrak{s} = \mathsf{C}\mathfrak{t}x$. We have

$$\begin{aligned}[\mathfrak{t}/x]\mathfrak{s} &:= (\varphi\to\varphi)\\ [\mathfrak{u}/x]\mathfrak{s} &:= (\varphi\to\psi)\end{aligned}$$

This shows the claim for formulae. Let terms t and u be given. If their meaning is different, then $(t{=}u) \notin L$ but $(t{=}t) \in L$. Let $\mathfrak{t}$ and $\mathfrak{u}$ be structure terms with exponents t and u. Then put $\mathfrak{s} := \mathsf{G}\mathfrak{t}x$. Similarly with variables.

The above proof works in the presence of a predicate. If the signature is empty, the theorem holds since there are no formulae anyway. Consider the following strengthening of Leibniz' Principle: *For all structure terms* $\mathfrak{t}$ *and* $\mathfrak{u}$, *and for all* $\mathfrak{s}$ *such that* $[\mathfrak{t}/x]\mathfrak{s}$ *and* $[\mathfrak{u}/x]\mathfrak{s}$ *are formulae:*

$$\mathsf{CG}\mathfrak{tu}\mathsf{C}[\mathfrak{t}/x]\mathfrak{s}[\mathfrak{u}/x]\mathfrak{s}^\varepsilon \in L$$

This means that for terms s, t (where $\varphi(s)$ results from replacing some appropriate variable z in φ by s — whatever substitution is actually used) the following holds.

$$((s{=}t)\to(\varphi(s)\to\varphi(t))) \in L$$

However, string substitution does not satisfy this property. Neither does standard substitution. Put $\varphi(z) := \texttt{(}\forall\texttt{x0)(}\neg\texttt{(}z\texttt{=x0))}$. The following is not valid in predicate logic.

$$\texttt{(}\forall\texttt{x)(}\forall\texttt{x0)((x=x0)}\to\texttt{((}\neg\texttt{(}\forall\texttt{x0)(x=x0))}\to\texttt{(}\neg\texttt{(}\forall\texttt{x0)(x0=x0)))}$$

Notice that generalized substitution satisfies the stronger version. However, Leibniz' Principle as defined above is about synonymy. It is clear that if $\mathfrak{t}$ and

$\mathfrak{u}$ are synonymous, then so are $[\mathfrak{t}/x]\mathfrak{s}$ and $[\mathfrak{u}/x]\mathfrak{s}$. It follows from Proposition 1 that the rules

$$\frac{(s{=}t)}{(\varphi(s/x)\rightarrow\varphi(t/x))} \qquad \frac{(\chi\rightarrow\chi'),\ (\chi'\rightarrow\chi)}{(\varphi(\chi/p)\rightarrow\varphi(\chi'/p))}$$

are admissible, where $\varphi(s/x)$ and $\varphi(\chi/p)$ are short for substitution of s for every occurrence of x and substitution of χ for the proposition (meta)variable p. For all three substitutions, predicate logics admit the above rules.

7 Uniqueness of the Analysis

We have presented a grammar which makes any predicate logic Leibnizian. The substitution it defines is string substitution. The question is whether there are grammars for which the corresponding substitution on the level of terms is either standard or generalized substitution. First, notice that the meaning of a formula depends on the meaning of those well–formed formulae which have been used to build it. It follows that if χ is a subformula on whose meaning the meaning of φ depends, then χ or an equivalent formula must actually be part of φ. Unfortunately, the converse need not hold. φ may be built using formulae that do not appear in it, and these might be formulae on whose meaning its meaning does not even depend. This is not an absurd statement. [1] makes that point clearly: the meaning of the formula $(\forall x)\varphi(x)$, x free, depends on more than just its subformulae. Its truth cannot be assessed by looking at the truth of the subformulae alone as in classical logic (though its meaning can be so found). Instead the entire model must be inspected. We may see this as a reflex of the fact that this formula actually contains all substitution instances as subformulae to begin with. Here is how. Let us add a ternary mode Y. For terms $\vec{u}$, variables $\vec{x}$, and formulae $\vec{z}$ let

$$\mathsf{Y}^{\varepsilon}(\vec{u},\vec{x},\vec{z}) := \vec{z}\,[\vec{u}/\vec{x}]$$

where $\vec{z}\,[\vec{y}/\vec{x}]$ denotes the result of replacing all free occurrences of $\vec{x}$ by $\vec{y}$. The interpretation of that function is also straightforward. Let m be a number, i an index and p a point.

$$\mathsf{Y}^{\mu}(p,m,i)(\langle\mathfrak{M},\beta\rangle) := i(\langle\mathfrak{M},\beta[p(\langle\mathfrak{M},\beta\rangle)/m]\rangle)$$

where $\beta[p(\langle\mathfrak{M},\beta\rangle)/m]$ is different from β only in assigning the value of p on $\langle\mathfrak{M},\beta\rangle$ to the number m. (If p is the meaning of the term t, $p(\langle\mathfrak{M},\beta\rangle)$ is nothing but the denotation of t in the model.) For completeness' sake we remark that

$$\mathsf{Y}^{\gamma}(a,b,c) = \begin{cases} \varphi & \text{if } a = \tau, b = \nu, c = \varphi \\ \text{undefined} & \text{else.} \end{cases}$$

This fully defines the operation of the mode Y on signs. Notice that $\mathsf{Y}xx\mathfrak{u}$ denotes the same sign as $\mathfrak{u}$. So, structure terms are no longer unique for formulae. Now, let $\mathfrak{x}$ be the structure term for x, $\mathfrak{t}$ the structure term for t and $\mathfrak{u}$ some (!) structure term for φ. Then

$$\mathsf{Y}\mathfrak{t}\mathfrak{x}\mathfrak{u}^{\varepsilon} = [t/x]\varphi, \qquad \mathsf{Y}\mathfrak{x}\mathfrak{x}\mathfrak{u}^{\varepsilon} = \varphi$$

Thus, with $\mathfrak{s} := \mathsf{Y}x\mathfrak{x}\mathfrak{u}$ we have a structure term such that standard substitution is nothing but substitution in the grammar sense. Yet, the proposed solution has a defect: it makes formulae into subformulae that have no string occurrence in it. For example, $\varphi(y)$ is a subformula of $\varphi(x)$ and vice versa. This is unacceptable.

It is clear that there are many grammars that generate predicate logic. Even if we exclude pathological cases and require, say, that the grammar is context free, there are infinitely many solutions. Let us therefore try to elaborate which further assumptions on the grammar make it unique. The sign based grammar PRED is unique on the following assumptions. The datum is pairs of well–formed expressions combined with their meanings.

[a] We assume that the exponents are strings over the alphabet, and that the modes can only concatenate them, possibly adding syncategorematic symbols.

[b] We assume that the system is **monotectonic** (or **unambigous**): every well–formed expression has a unique structure term.

[c] We assume that for every substring that is a well–formed expression of some type, that occurrence is the exponent of some subterm of the structure term. This is called **transparency**.

[d] Any two well–formed expressions have the same category iff they have the same distribution.

By [a] and [c] we get that every well–formed expression is composed from its immediate subexpressions. For example, `(∀x0)(x=x0)` has as its immediate subexpressions `x0` and `(x=x0)`. Hence, the additional brackets as well as `∀` are syncategorematic. Moreover, it makes `x` a subexpression of `x0`, which is a subexpression of `x01`, and so on. Notice that by transparency substitution can be formulated as mere string replacement, since 'accidental' occurrences of subterms cannot be mistaken for proper ones (as would be `p0→p1` in `p∧p0→p1`, when brackets are dropped). [b] is put in to make sure that there are no two modes that operate in the same way on signs. [d] ensures that we do not create more categories than absolutely necessary. By distributional analysis we get in fact three categories: variables (only they appear right after `∀`), terms and formulae.

It is interesting to note that different results obtain if the morphology is different. Suppose we drop brackets in conjunctive expressions as we do with additive terms, writing φ`∧`χ`∧`ψ in place of `(`φ`∧(`χ`∧`ψ`))`. Then the resulting language is no longer transparent. The string `(x=x0)∧(x0=x1)∧(x1=10)` has two different overlapping well–formed substrings, `(x=x0)∧(x0=x1)` and `(x0=x1)∧(x1=x10)`. But not both of them can be part of one and the same analysis, because of [a]. Of course, the ambiguity is spurious: we may choose either analysis. On either analysis we get the same result. (As a note of clarification: here we do *not* view the omission of brackets as an abbreviatory convention, but we take strings without brackets as genuine, well–formed expressions in their own right. We are analyzing, so to speak, the actual usage of predicate logic rather than the official norm.)

Now, there are other conventions as well. The brackets around an equation are always omitted. One also assumes that `¬` is the strongest symbol, and brackets are dropped from `(¬`φ`)`. Outer brackets are always dropped. However, notice that while `x=x0∧x0=x1` is unambigous, its negation is to be written `¬(x=x0∧x0=x1)` rather than `¬x=x0∧x0=x1`. All this only concerns the manipulation of the exponents of the grammar that generates this language. Substitution, however, remains the same operation on the level of structure terms, only that it projects differently onto the exponents (= wfes).

8 Axiomatization

The relevance of substitution is seen when we turn to axiomatization. There are two kinds of axiomatizations. The first uses an inbuilt rule of substitution, the second specifies axiom schemes, using metavariables for formulae. In propositional logics both approaches are possible, while FOL can use internal substitution only with respect to terms, since there are no variables for formulae. However, we should be aware of the fact that the employment of axiom schemes relies on a correct understanding of what is substituted and how. This includes a proper understanding of the morphology of the actual language. Let us give an example. In propositional logics substitution is taken to be replacement of occurrences of a (meta)variable by a string. Yet, this works only in the transparent case. If we do not write brackets matters are different. For example, replacing `p0` in `p0∧p1` by the disjunction `p01∨p11` will make it necessary to insert brackets, so that we get `(p0∨p11)∧p1` and not `p01∨p11∧p1`. On the other hand, if we have to substitute `p01∧p11` for `p0`, no insertion of brackets is needed. This shows clearly that substitution may require syntactic analysis. Instead, one should think of the strings as representatives of structure terms, and metavariables as proxy for structure term variables. For example, $\texttt{(¬((∀}x\texttt{)}\varphi\texttt{→(¬}\varphi\texttt{)))}$ is proxy for $\mathsf{QCX}xz\mathsf{Q}z$, where x is a variable for a structure term for variables, z a variable for a structure term for formulae. Given the grammar there is no question of what function substitution actually is, since it is universally and unequivocally determined at the level of structure terms. If, say, we instantiate x to ENZ and z to GENZEZ then the structure term becomes

$$\mathsf{QCXENZGENZEZQGENZEZ}$$

Its exponent is `(¬((∀x01)(x01=x1)→(¬(x01=x1))))`.

The first task we set ourselves is to axiomatize predicate logic using string substitution, denoted here by $\ulcorner t/x\urcorner$. Recall that predicate logic has three kinds of axioms. The first set is the propositional axioms, for example

$$\texttt{((}\varphi\texttt{→(}\psi\texttt{→}\chi\texttt{))→((}\varphi\texttt{→}\psi\texttt{)→(}\varphi\texttt{→}\chi\texttt{)))}$$

These rules are unproblematic. The given axioms can be instantiated by substituting any given structure term for formulae for the variables for terms x, y and z in

$$\mathsf{CC}x\mathsf{C}yz\mathsf{CC}xy\mathsf{C}xz$$

So these axioms are completely schematic. Likewise the rule

$$(\mathrm{mp})\quad \frac{\texttt{(}\varphi\texttt{→}\chi\texttt{)}\quad \varphi}{\chi}$$

The next set are the axioms for equality: reflexivity, transitivity and symmetry, and the rule of replacement of equals:

$$\texttt{(∀x)(∀x0)(x=x0→(p(x)→p(x0)))}$$

These axioms are the exponents of concrete structure terms. (This is enough given [2] and (gen) below.)

Finally, the following need to be added.

[1] $\texttt{((}\forall x\texttt{)(}\varphi\texttt{→}\psi\texttt{)→((}\forall x\texttt{)}\varphi\texttt{→(}\forall x\texttt{)}\psi\texttt{))}$

[2] $\texttt{((}\forall x\texttt{)}\varphi\texttt{→}[t/x]\varphi\texttt{)}$

[3] $\texttt{(}\varphi\texttt{→(}\forall x\texttt{)}\varphi\texttt{)}$ (x not free in φ)

and the rule

$$\text{(gen)}\quad \frac{\varphi}{\texttt{(}\forall x\texttt{)}\varphi}$$

Notice that [1] and the rule (gen) present no problem. [2] and [3] present two problems: they have side side conditions and they involve explicit substitutions.

Suppose that we change to the substitution $\ulcorner t/x\urcorner$.

Definition 7 *Call a formula* ***regimented*** *if there is no subformula which contains a variable both free and bound.*

Lemma 1 *Suppose that* $\texttt{(}\forall x\texttt{)}\varphi$ *is regimented. Then* $\ulcorner t/x\urcorner\varphi = [t/x]\varphi$.

The following is relatively straightforward to prove.

Proposition 2 *Let* φ *be regimented. If* φ *is provable in* FOL*, it has a proof using only regimented formulae.*

We replace the set of axioms by the following:

[i] $\texttt{((}\forall x\texttt{)(}\varphi\texttt{→}\psi\texttt{)→((}\forall x\texttt{)}\varphi\texttt{→(}\forall x\texttt{)}\psi\texttt{))}$

[ii] $\texttt{((}\forall x\texttt{)}\varphi\texttt{→}\ulcorner t/x\urcorner\varphi\texttt{)}$ ($\texttt{(}\forall x\texttt{)}\varphi$ regimented)

[iii] $\texttt{(}\varphi\texttt{→(}\forall x\texttt{)}\varphi\texttt{)}$ (x not free in φ)

[iv] $\texttt{((}\forall x\texttt{)}\varphi\texttt{→(}\forall y\texttt{)}\ulcorner y/x\urcorner\varphi\texttt{)}$ (y not free in φ, x not bound in φ)

Thus, we have effectively restricted only the second axiom. We are guaranteed only to derive the regimented formulae. That is why we have added the formula [iv]. The axiomatisation is complete. For suppose that φ contains a subformula containing a variable bound which occurs free outside of it. Then we can do a suitable replacement of that bound variable and get a formula φ'. The two are equivalent in predicate logic. φ' is regimented, and φ can be derived from it using [iv]. It is certainly valid, since under the given conditions it is identical to

$$\mathtt{((}\forall x\mathtt{)}\varphi\mathtt{\rightarrow(}\forall y\mathtt{)}[y/x]\varphi\mathtt{)}$$

Notice that the axiomatization is not entirely algebraic. There are side conditions on the formulae and we have made use of explicit substitutions. The question arises whether using a different notion of substitution can make a difference here. This would mean to present a schematic axiomatization in which we use metavariables for variables, for terms and for formulae. Clearly, this is feasible, by simply listing them. This is the way in which the axiomatization of first–order logic is standardly taken. The set is decidable but infinite. The question is whether a finite subset is enough. Suppose for simplicity that we have no function symbols. Then we only need to worry about variables and formulae. It is known that substitution can be defined by quantification. In view of a result of [6] it is impossible to reduce the list to a finite one. Interestingly, [9] show that adding explicit substitution functions (of variables by variables) does not improve the situation.

We remark here only that if we use general substitution, the axiom [2] becomes

$$\mathtt{((}\forall x\mathtt{)}\varphi\mathtt{\rightarrow}\{t/x\}\varphi\mathtt{)}$$

without side conditions. Yet, the axiom [3] still needs the side condition x is not free in φ. Thus, none of the substitutions actually substantially simplifies the task of axiomatizing FOL.

9 Conclusion

The point of this paper was to argue that the grammar of FOL virtually forces us to assume a particular kind of substitution. The general question concerning this is: why is this a concern? And why should the logician care? To answer the second question first: we have shown that string substitution is actually no more and no less suited for the purpose than is standard substitution, but it is easier to use. The real complications do not arise from substitution, they arise from FOL itself. To answer the first question: in an artifical language we can make arbitrary decisions, but in a natural language we cannot. Still, in reasoning within natural language we wish to do the very same as we did for predicate logic: formulate the rules of reasoning within language. The Stoics used expressions like *the first* or *the second* as variables for sentences, something which does not require great care in formulation. But when it comes to syllogisms, matters are less straightforward. Consider *modus barbara* (see [3]):

$$\begin{array}{l}\texttt{Every man is mortal.}\\ \texttt{Every priest is a man.}\\ \hline \texttt{Every priest is mortal.}\end{array}$$

It arises in the same way from insertion into a schematic expression:

$$\begin{array}{l}\texttt{Every } P \texttt{ is } Q\texttt{.}\\ \texttt{Every } R \texttt{ is } P\texttt{.}\\ \hline \texttt{Every } R \texttt{ is } Q\texttt{.}\end{array}$$

Yet, when we turn to other languages, the substitution will necessitate changes. In Latin and French, the predicative adjective agrees in gender with the subject. Although we wish to consider the agreement patterns to be inessential, still it is important to set up the system in such a way that they really are taken care of. In other words, we wish to set up a grammar for French and Latin in such a way that it provides a schematic expression of the kind that does the agreement automatically. Otherwise the logical schemata need to make up for that (for example by devising different schemata for different genders).

10 Bibliography

[1] José Ferreirós. The road to modern logic — an interpretation. *The Bulletin of Symbolic Logic*, 7:441 – 484, 2001.

[2] Gottlob Frege. *Begriffsschrift.* Nebert, Halle, 1879.

[3] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik.* Springer, Berlin, Heidelberg, 5 edition, 1967.

[4] Stephen C. Kleene. *Introduction to Metamathematics.* North–Holland, Amsterdam, 1964.

[5] Marcus Kracht. *The Mathematics of Language.* Mouton de Gruyter, Berlin, 2003.

[6] Donald J. Monk. Nonfinitizability of classes of representable cylindric algebras. *Journal of Symbolic Logic*, 34:331 – 343, 1969.

[7] Arnold Oberschelp. *Elementare Logik und Mengenlehre.* Bibliographisches Institut, Mannheim, 1974.

[8] Wolfgang Rautenberg. *Einführung in die mathematische Logik.* Vieweg, Wiesbaden, 1996.

[9] Ildikó Sain and Richard S. Thompson. Finite Schema Axiomatization of Quasi–Polyadic Algebras. In Hajnal Andréka, Donald Monk, and István Németi, editors, *Algebraic Logic*, number 54 in Colloquia Mathematika Societatis János Bolyai, pages 539 – 571. János Bolyai Matematikai Társulat and North–Holland, Budapest/Amsterdam, 1991.