First-order Logic without bound variables: Compositional Semantics

W. W. Tait

This paper was written in honor of Dag Prawitz for the occasion, now alas long past, of his 70th birthday.

An attractive format for semantics is that in which composite expressions are built up from atomic ones by means of the operation of concatenation and the concatenation XY expresses the application of a function denoted by X to an argument denoted by Y. The use of relative pronouns presents an obstacle to this form of compositional semantics, since the reference of a relative pronoun in one component may occur in another. In the standard notation of first-order predicate logic this occurs in the form of variable-binding operations of quantification: in the sentence $\forall x\phi(x)$, the reference of x in $\forall x$ is in $\phi(x)$ and neither component has an independent meaning. Frege, in the interests of compositional semantics, was led by this to declare that the open formula $\phi(x)$ is semantically significant: it simply denotes an 'incomplete object'. We won't discuss here the many reasons for rejecting this very ugly idea, but reject it we will. So the demands of compositional semantics require that we formalize first-order predicate logic without using bound variables.

Of course the use of bound variables is very natural and the means that we use to eliminate them can result in quite complex expressions. Our purpose, therefore, is not the practical one of finding the most readable notation: it is the theoretical one of obtaining a compositional semantics. On the other hand, we shouldn't be too humble: although the notation $\phi(v)$ with free variable v and instances $\phi(t)$ where t is a term is quite intuitive, the substitutions involved in actual cases, in substituting t for the possibly multiple occurrences of v in $\phi(v)$, can create long expressions. The elimination procedure will consist in showing that $\phi(v)$ can be expressed by $\phi' v$, expressing

application of the function ϕ' to the argument v, where ϕ' itself is built up in accordance with our function-argument paradigm from the variables, other than v, and constants in $\phi(v)$. Thus for example, $\forall x\phi(x)$ is now expressed by $\forall\phi'$, which again will be seen to express application of a function to an argument. Once ϕ' is constructed, the substitutions $\phi's$, $\phi't$, etc., for v in $\phi'v$ are more easily processed than the corresponding substitutions $\phi(s), \phi(t)$, etc. The equivalence of $\phi'v$ and $\phi(v)$ consists of a sequence of reduction steps reducing the former to the latter, where each reduction step consists in replacing an expression XY for the application of a function (X) to an argument (Y) by the expression for its value.

In Part I we are going to consider only sentences expressed in an arbitrary first-order language $\mathcal{F}$ without identity and with universal quantification $\forall$ as the only variable-binding operation. It suffices to take as the remaining logical constants just $\rightarrow$ for implication and $\perp$ for the absurd proposition. (Negation $\neg\phi$ is expressed by $\phi \rightarrow\perp$.) This of course suffices only so long as we are considering just the classical conception of logic. On the constructive conception, we would need to add conjunction, disjunction, and the further variable-binding operation $\exists$ of existential quantification. The inclusion of these constants in the case of classical logic would not complicate the treatment of bound variables in formulas, only lengthen it; and so we won't bother.

It is not only the semantics of formulas that loses its modularity on account of bound variables. This also happens with deductions: for example, the deduction of $\forall x\phi(x)$ from a deduction $p(v)$ of $\phi(v)$ binds the variable in $p(v)$. For any individual term t, $p(t)$ denotes the corresponding deduction of $\phi(t)$. The other case involves $\rightarrow$-Introduction, where a deduction of $\phi \rightarrow \psi$ arises from a deduction of ψ from the assumption of ϕ. So long as we think of deductions as purely syntactical objects, the 'Deduction Theorem' shows how to eliminate $\rightarrow$-introduction. But, if we take the view that deductions denote certain objects, namely proofs, then the assumption of ϕ should be understood as a variable $v = v_\phi$ ranging over proofs of ϕ and the deduction of ψ then again has the form $p(v)$, and v again gets bound when we 'discharge' the assumption to obtain $\phi \rightarrow \psi$. For any deduction q of ϕ, $p(q)$ then denotes the result of replacing the assumptions v of ϕ by its proof q. Again, in both the case of $\forall$-introduction and $\rightarrow$-introduction, we will show how to obtain $p(v)$ from an expression $p'v$, where p' is built up by means of application of function to argument from the variables other than v and constants in $p(v)$. The reduction of $p'v$ to $p(v)$ again is a matter of replacing expressions XY

of the application of a function to an argument by the corresponding expression for the values. The proof of the Deduction Theorem turns out to be essentially the construction of p' from $p(v)$. We will discuss the elimination of variable-binding in deductions in Part II of the paper.

I tackled the same problem of eliminating bound variables for the Curry-Howard type theory in [Tait, 1998]. The attempt to relativize that paper to predicate logic doesn't quite work: but something like it does. The problem arises in connection with eliminating bound variables from deductions. The earlier method of eliminating bund variable, applied to deductions in the framework of first-order logic, leads to deductions outside that framework. In order to avoid this, we need to introduce further operations on first-order deductions than those provided by the earlier paper but which, from the higher point of view of Curry-Howard type theory, are not really new.

As we will indicate, W. V. O. Quine addressed the problem of eliminating bound variables both from first-order formulas [1960a] and from deductions [1966], but the two are not integrated: in his formalism for first-order deductions, the formulas contain bound variables. Moreover, his treatment of formulas does not provide a compositional semantics for them and his treatment of deductions provides no semantics at all. My aim is a semantics of formulas and deductions in which neither contain bound variables and in which the semantics is compositional in the sense indicated above.

PREAMBLE

The central idea of this paper was first presented in a lecture in Göttingen in 1920 by Moses Schönfinkel, a Russian member of Hilbert's group in foundations of mathematics from 1914-1924. In 1924 Heinrich Behman, another member of the group, published the lecture under the title "Über die Bausteine der mathematischen Logik", with some added paragraphs of his own.[1] The idea in question was of course the *theory of combinators* or, as we

[1]There is another paper of Schönfinkel, on special cases of the decision problem. This was prepared and published in 1929 by Paul Bernays. Not much seems to be known about Schönfinkel. Here is what Wikipedia has to say about his life after Göttingen:

> After he left Göttingen, Schönfinkel returned to Moscow. By 1927 he was reported to be mentally ill and in a sanatorium. His later life was spent in poverty, and he died in Moscow some time in 1942. His papers were burned

should now say, of *untyped* combinators. We start with an alphabet of atomic symbols, some constants and some variables. These are to include the constants K and S, called *combinators*.[2] The set $\mathcal{W}$ of formulas is the least set containing all the atomic symbols and such that, if X and Y are in $\mathcal{W}$, then so is (XY). Parentheses are necessary here, so we don't yet have our ideal, mentioned in the introduction, of composition simply by concatenation. For $n > 1$ we will write

$$X_1X_2 \cdots X_n := (\cdots (X_1X_2) \cdots X_n)$$

(association to the left). We call formulas of the form KXY and $SXYZ$ *convertible* and call X and $(XZ)(YZ)$ their *values*, respectively, and write

$$KXY \ CONV \ X \qquad\qquad SXYZ \ CONV \ (XZ)(YZ).$$

The relation

$$X \succ' Y$$

between formulas is defined to mean that Y is obtained by replacing one occurrence of a convertible part of X by its value. We say that X *reduces to* Y, written

$$X \succeq Y$$

if there is a chain

$$X = Z_0 \succ' \cdots \succ' Z_n = Y$$

with $n \geq 0$ (so that X reduces to itself). Let

$$X \equiv Y$$

mean that X and Y reduce to a common formula. Call a formula *normal* if it contains no convertible parts. If X reduces to Y and Y is normal, we say that Y is a *normal form* of X. Here is a theorem:

Church-Rosser Theorem [Uniqueness of Normal Form]. If X reduces to both Y and Z, then Y and Z reduce to a common formula U. So every formula has at most one normal form. [Church and Rosser, 1936]

---

by his neighbors for heating.

[2]Schönfinkel used C instead of K.

The second part is of course immediate from the first: If X has the two normal forms Y and Z, then they must reduce to a common formula U. Given that Y and Z are normal, we must then have $Y = U = Z$. In consequence of the Church-Rosser Theorem, $\equiv$ is an equivalence relation among formulas.

Actually Church and Rosser proved this for a related formalism, Church's *calculus of lambda conversion* or simply the *lambda calculus* [Church, 1941]. I'm not sure who first proved it for the theory of untyped combinators; but the usual proof given now, both for combinators and the lambda calculus, is mine (first presented in a seminar on the lambda calculus at Stanford in Spring of 1965). Unlike the original Church-Rosser proof, the argument is quite simple: the idea is to define a weak notion of reduction (1-reduction, below) which implies $\succeq$ and for which the theorem is trivial but such that every reduction is obtained by a sequence of weak reductions. (Once considered in this way, the requisite notion of weak reduction in various extensions of the theory of combinators or the lambda calculus is usually obvious.)

We define the notion that X 1-*reduces* to Y, written (temporarily) $X \longrightarrow Y$, by induction on the number of occurrences of symbols in X:

- $KXY \longrightarrow X$.
- $SXYZ \longrightarrow (XZ)(YZ)$.
- If $X_i = Y_i$ or $X_i \longrightarrow Y_i$ for $i = 1, \ldots, n$, then $X_1 \cdots X_n \longrightarrow Y_1 \cdots Y_n$.

By induction on the number of occurrences of symbols in X, one easily proves:

Lemma. Let X 1-reduce to Y and to Z. Then there is a U such that Y and Z 1-reduce to U:

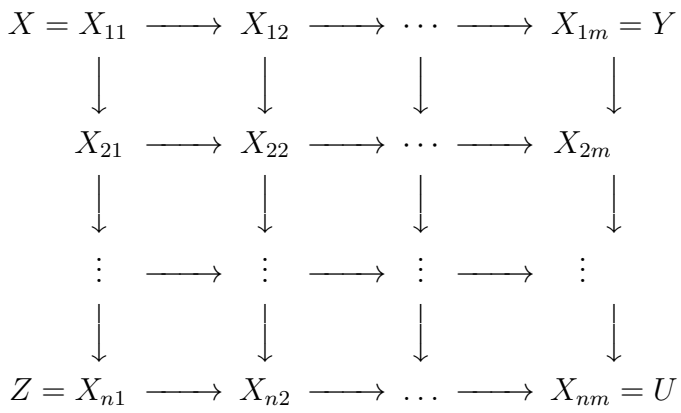
$$\begin{array}{ccc} X & \xrightarrow{1-RED} & Y \\ {\scriptstyle 1-RED}\downarrow & & \downarrow{\scriptstyle 1-RED} \\ Z & \xrightarrow[1-RED]{} & U \end{array}$$

□

Since $X \succ Y$ implies X 1-reduces to Y, it is immediate that X reduces to Y if and only if there is a chain

$$X \longrightarrow \cdots \longrightarrow Y.$$

The proof of Church-Rosser then is:



□

So we may think of a formula X as defining a partial function $\overline{X}$ on the set of normal terms in $\mathcal{W}$: for each normal $Y \in \mathcal{W}$, $\overline{X}Y$ is defined and $= Z$ if and only if Z is the (unique) normal form of the formula XY. For example, it was on the basis of this idea that Church represented all the partial recursive functions in his calculus of lambda conversion. On the other hand, just because of this representation, we know *a priori* that not every formula has a normal form. Indeed it is easy to construct an example: let

$$I := SKK$$

Then

$$IX \; CONV \; KX(KX) \succ' \succ' X$$

so that I is the 'identity function'. Now let $Y = SII$. Then

$$YY = SIIY \succ (IY)(IY) \succ \cdots \succ YY \succ \cdots .$$

represents the only reduction chains for YY, and so it has no normal form.

Two distinct normal formulas X and Y may define the same function $\overline{X} = \overline{Y}$. For example, for any normal formula X, $S(KX)I$ is normal and $\overline{S(KX)I} = \overline{X}$. We will see that this particular example is significant and has led to the extension of the class of convertible formulas to include those of the form $S(KX)I$ and taking its value to be X. So, in the definition of 1-reduction, we should add the clause

$$S(KX)I \longrightarrow X.$$

There will be no difficulty extending the proof of the Lemma and so of the Church-Rosser Theorem to admit these conversions, called *η-conversions*. (Schönfinkel did not consider η-conversion.)

The power of the theory of combinators—and why it is essentially equivalent to the calculus of lambda conversion and why it is of interest to us—lies in the following:

Explicit Definition Theorem I [Schönfinkel, 1924]. Let $X(v)$ be a formula, where v is a variable. There is a formula X' of such that

$$X'v \succeq X(v).$$

Moreover, the atomic symbols in X' exclude v and are either combinators or are in $X(v)$.

Of course it then follows by substitution that, for any formula θ, $\phi(\theta) \equiv \phi'\theta$. The proof is by induction on the complexity of $X(v)$:

Case 1. $X(v) = v$. Set $X' = I$.

$$X'v = Iv \succeq v.$$

Case 2. $X(v) = X$ is a formula not containing v. Set $X' = KX$.

$$X'v = KXv \succeq X.$$

Case 3. $X(v) = Y(v)Z(v)$ contains v. Set $X' = SY'Z'$. Then

$$X'v = SY'Z'v \succeq (Y'v)(Z'v) \succeq Y(v)Z(v) = X(v).$$

□

Notice that η-conversion is not used in this construction. Given $X(v)$, we will denote the corresponding X' given by the Explicit Definition Theorem by

$$\Lambda y.X(y)$$

Here y is a *bound variable*; we also use x, z, all with or without subscripts, as bound variables. They are distinct from the free variables that occur in

$\mathcal{W}$ and occur only bound by Λ or, later, by quantifiers. They occur only in the context of abbreviations for formulas that contain no bound variables at all: after all, that is our aim, to eliminate bound variables.

$\Lambda yX(y)$ satisfies the principle of *lambda-conversion*

$$(\Lambda yX(y))Y \equiv X(Y)$$

for any formula Y. So far, what we have said holds true whether or not we include η- conversions. Note that $S(KX)I = \Lambda yXy$ and so η-conversion yields the equation

$$\Lambda yXy \equiv X.$$

Remark 1. I don't use the more usual lower case λ here because that is used in Church's calculus of lambda conversion. There is an obvious translation of each system, combinator theory or lambda calculus, into the other; but in either direction the formula of combinator theory may be normal while the corresponding formula of the lambda calculus is not. Thus, if we replace each λ in a formula of the form $\lambda x.t(x)$ of the lambda calculus by Λ, the result is a normal formula in combinator theory, whether or not the original formula is normal. On the other hand, a formula of the theory of combinators can be translated into the lambda calculus by replacing K by $\lambda x\lambda y.x$ and S by $\lambda x\lambda y\lambda z.xz(yz)$. But again, the normal formulas Kv and Suv, for example, translate into non-normal formulas of the lambda calculus.

An important difference between λ and Λ arises in connection with η-conversion. Suppose $\overline{X} = \overline{Y}$, i.e. $Xv \equiv Yv$ for a free variable v not in either X or Y. Then $\lambda zXz \equiv \lambda zYz$ and so by η-conversion, $X \equiv Y$. Thus in the context of the lambda-calculus, distinct normal terms define distinct normal functions. But in the theory of combinators $Xv \equiv Yv$ does not imply $\Lambda zXz \equiv \Lambda zYz$; and so η-conversion in that context is not so natural. □

The Explicit Definition Theorem accomplishes part of what we want: formulas $X(v)$ are obtained from the corresponding formulas $X'v$ by successively replacing certain parts (UV) by their 'values'. But if we were to just apply this to eliminating bound variables from first=order formulas and proofs, it would be a purely formal transformation, with no semantical content. However, by introducing type structure into the theory of combinators, the combinators become semantically meaningful: XY (and now parentheses are unnecessary) denotes the application of a function to one of its arguments.

We will in fact need two such type structures: one in Part I, to eliminate bound variables from formulas, and the other in Part II, to eliminate bound variables in deductions. The types in Part I are built up from atoms by passing from types A and B to the type $A \Rightarrow B$ of functions from objects of type A to objects of type B. The types of the combinators are just the axioms for the theory of implication, when the atoms are regarded as atomic sentences and $\Rightarrow$ is understood as implication. This type structure for combinatorial logic (and the lambda calculus) seems to have been first discussed in [Curry and Feys, 1958, Chapter 9] and is generally associated with Curry. The type structure in Part II derives from Bill Howard's extension of Curry's idea of propositions as types of objects to the case of propositions expressed in first-order predicate logic. (Howard distributed some notes on this idea in the late 1960's and finally published them in [1980].)

PART I

1. First, some conventions and notation: we take 1 and 0 to be the truth-values $TRUE$ and $FALSE$, respectively, so that

$$2 = \{0, 1\}$$

is the set of truth-values. Let A and B be sets. As we just specified,

$$A \Rightarrow B$$

denotes the set of all functions from A to B. We define

$$A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B$$

for $n > 1$ by induction to be $A_1 \Rightarrow (A_2 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)$ (association to the right). When $A_1 = \cdots = A_n = A$, we denote this by

$$A \Rightarrow_n B.$$

For the case $n = 0$, we simply define $A \Rightarrow_0 B$ to be B.

The *types* are defined by

- $\mathcal{D}$ and 2 are types.

- If A and B are types, so is $A \Rightarrow B$.

$\mathcal{D}$ is just a formal symbol and types are just syntactical objects. But we are speaking about semantics and so we are assuming that a specific model M of $\mathcal{F}$ is given. So we may think of $\mathcal{D}$ as denoting its domain D_M. In this way every type A becomes a set A_M. An n-ary function constant f of $\mathcal{F}$ denotes in M an object f_M in $\mathcal{D}_M \Rightarrow_n \mathcal{D}_M$. An n-ary relation constant R of $\mathcal{F}$ denotes in M an object $R_M \in \mathcal{D}_M \Rightarrow_n 2$. Notice that an individual constant denotes an element of the domain $\mathcal{D}_M$ of the model and a propositional constant denotes a truth-value.

2. We already are drawing on another idea from [Schönfinkel, 1924]. It is standard to interpret an n-ary function or relation constant for $n > 0$ as a function in

$$\mathcal{D}^n \Rightarrow E$$

where E is either 2 or $\mathcal{D}$ and $\mathcal{D}^n$ is the set of ordered n-tuples of elements of $\mathcal{D}$. Schönfinkle was first to note explicitly that functions of several variables (i.e. $n > 1$) can be reduced to functions of a single variable via the one-to-one correspondence

$$f \mapsto f'$$

between the $f \in D_1 \times \cdots \times D_n \Rightarrow E$ and the $f' \in D_1 \Rightarrow \cdots \Rightarrow D_n \Rightarrow E$ given by

$$f(x_1, \ldots, x_n) = (\cdots (f' x_1) \cdots x_n) = f' x_1 \cdots x_n.$$

By utilizing this correspondence in the interpretation of the function and relation constants of $\mathcal{F}$ (where E is $\mathcal{D}$ and 2, respectively), we avoid having to introduce Cartesian products $E \times F$ into the type set $\mathbf{M}$. On the standard interpretation of the function and relation constants, our compositional semantics would be complicated by the fact that we would need two forms of composition: besides the operation XY expressing the application of a function to an argument, we would also need the operation (X, Y) expressing the operation of taking pairs.

3. We introduce a new formalism

$$\mathcal{G}$$

consisting of a set of expressions, called the *formulas* of $\mathcal{G}$, and an assignment to each formula of $\mathcal{G}$ a type.

- The n-ary function constants of $\mathcal{F}$ are atomic formulas in $\mathcal{G}$ of type $\mathcal{D} \Rightarrow_n \mathcal{D}$. Each denotes an object in its type.
- The n-ary relation constants of $\mathcal{F}$ are atomic formulas in $\mathcal{G}$ of type $\mathcal{D} \Rightarrow_n 2$. Each denotes an object in its type.
- Other constants will be specified presently as formulas of $\mathcal{G}$, each with an assigned type and each denoting an object in its type.
- There is an infinite supply of free variables of type $\mathcal{D}$.
- All other formulas of $\mathcal{G}$ are composite. The rule of formation of composite formulas and the determination of their types is simply the rule of *evaluation*—application of a function to an argument: If the formula ϕ is of type $A \Rightarrow B$ and the formula ψ is of type A, then $\phi\psi$ is a formula of type B. All formulas are obtained from the constants and free variables by means of this construction.

Notice that, unlike in the theory of untyped combinators, we do not need parentheses around $\phi\psi$:

Unique Readability Lemma. Every formula ϕ of $\mathcal{G}$ is a unique concatenation

$$\phi = \phi_0\phi_1 \cdots \phi_n$$

where $n \geq 0$, ϕ_0 is a constant and each ϕ_i is a formula. □

We won't bother to prove this result. The reason why we do not need parentheses is of course the type structure. But unique readability does not imply *easy* readability and so we will sometimes use parentheses to indicate how the formula should be read, even though it is the only way in which it can be read.

The terms and atomic formulas of $\mathcal{F}$ are formulas of $\mathcal{G}$ of types $\mathcal{D}$ and 2, respectively: let f and R be n-ary function and relation constants, respectively, and let $\theta_1, \ldots, \theta_n$ be formulas of type $\mathcal{D}$, then

$$f\theta_1 \cdots \theta_n$$

is of type $\mathcal{D}$ and

$$R\theta_1 \cdots \theta_n$$

is of type 2. The remaining constants include the logical constants:

Absurdity

$$\bot$$

is a constant formula of $\mathcal{G}$ of type 2 and, specifically, is false (i.e. denotes 0).

Implication

$$\rightarrow$$

is a constant formula of $\mathcal{G}$ of type $2 \Rightarrow (2 \Rightarrow 2)$. Specifically, if ϕ and ψ denote truth-values, then $(\phi \rightarrow \psi) =\!\!\rightarrow \phi\psi$ is true (i.e. denotes 1) unless ϕ is true and ψ is false.

The *universal quantifier*

$$\forall$$

is of type

$$(\mathcal{D} \Rightarrow 2) \Rightarrow 2.$$

Specifically, if ϕ is a closed formula of $\mathcal{G}$ of type $\mathcal{D} \Rightarrow 2$, then $\forall\phi$ is true just in case $\phi\theta$ takes the value 1 for all closed formulas θ of $\mathcal{G}$ of type $\mathcal{D}$.

And now we pay the price for compositionality: for the remaining constants of $\mathcal{G}$, we must introduce the *typed combinators*:

- For all types A, the constant $K_{\mathcal{D},A}$ of type

  $$A \Rightarrow (\mathcal{D} \Rightarrow A)$$

  is a formula of $\mathcal{G}$, defined for $a : A, t : \mathcal{D}$ by

  $$K_{\mathcal{D}}, A\phi t \; CONV \; \phi.$$

- For all types A, B, the constant $S_{\mathcal{D},A,B}$ of type

  $$[\mathcal{D} \Rightarrow (A \Rightarrow B)] \Rightarrow [(\mathcal{D} \Rightarrow A) \Rightarrow (\mathcal{D} \Rightarrow B)]$$

  is a formula of $\mathcal{G}$ defined for $\phi : (\mathcal{D} \Rightarrow (A \Rightarrow B), \psi : (\mathcal{D} \Rightarrow A), t : \mathcal{D}$ by

  $$S_{\mathcal{D},A,B}\phi\psi t \; CONV \; (\phi t)(\psi t).$$

Finally, it is in connection with $\mathcal{G}$ that we need certain instances of η-conversion, but first we need a definition of the identity function on type $\mathcal{D}$: we simply give type structure to the 'identity function' I above, making it the identity function $I_{\mathcal{D}}$ on $\mathcal{D}$: Let $B = [\mathcal{D} \Rightarrow \mathcal{D}]$. Then $S_{\mathcal{D},B,\mathcal{D}}$ has type

$$[\mathcal{D} \Rightarrow (B \Rightarrow \mathcal{D})] \Rightarrow [(\mathcal{D} \Rightarrow B) \Rightarrow (\mathcal{D} \Rightarrow \mathcal{D})]$$

$K_{\mathcal{D},B}$ has type $\mathcal{D} \Rightarrow (B \Rightarrow \mathcal{D})$ and $K_{\mathcal{D},\mathcal{D}}$ has type $\mathcal{D} \Rightarrow B$. So

$$I_{\mathcal{D}} := S_{\mathcal{D},B,\mathcal{D}} K_{\mathcal{D},B} K_{\mathcal{D},\mathcal{D}}$$

has type $\mathcal{D} \Rightarrow \mathcal{D}$ and for v of type $\mathcal{D}$

$$I_{\mathcal{D}} v \succeq K_{\mathcal{D},B} v (K_{\mathcal{D},\mathcal{D}} v) \succeq v.$$

The case of η-conversion we shall need is this: Let ϕ be a formula of type $\mathcal{D} \Rightarrow B$. Then

$$S_{\mathcal{D},\mathcal{D},B}(K_{\mathcal{D},\mathcal{D}\to B}\phi) I_{\mathcal{D}} \; CONV \; \phi.$$

Thus, the typed combinators are actually defined as objects of a certain type by equations that, in untyped combinator theory, are just formal rules of conversion. The realization (in Curry-Feys) that combinator theory could be typed rested on the observation that the formulas of the two untyped conversion relations $X \; CONV \; Y$ can be 'stratified'—i.e. types can be assigned to X and Y so that the right-hand side of the conversion for KXY respects the type assignment and types can be assigned to X, Y and Z so that the right hand side of the conversion of $SXYZ$ respects the type assignment. This is trivial for K. The stratification of $XY(ZY)$ is almost as easy. (But then, remember the story of Christopher Columbus and the egg.)

Of course nothing in the above discussion depends upon the particular role of $\mathcal{D}$ in the system $\mathcal{G}$. Along with $K_{\mathcal{D},A}$ and $S_{\mathcal{D},A,B}$, we could introduce $K_{C,A}$ and $S_{C,A,B}$ with the corresponding conversion rules for an arbitrary type C and then define the identity function I_C on C. We don't need to do this, however, because the only variables in formulas of $\mathcal{G}$ are of type $\mathcal{D}$. (See the Explicit Definition Theorem II below.)

Now we have specified all of the constants of $\mathcal{G}$, together with their denotations, and so have completed the definition of the set of formulas of $\mathcal{G}$ with their types.

4. The notions of a reduction, normal term, normal form and equivalence $\equiv$ between formulas of $\mathcal{G}$ are defined exactly as in the case of the untyped combinators: just ignore the types. There are two well-known properties of reduction. The first of them we have already proved:

Church-Rosser Theorem [Uniqueness of Normal Form] If ϕ reduces to ψ and to χ, then ψ and χ reduce to a common formula θ. So, in particular, if ψ and χ are normal, then $\psi = \chi$.

The proof given in the Preamble applies since it has nothing to do with type structure. $\Box$

Recall that $\phi \succ' \psi$ means that ψ is obtained by converting a single part of ϕ. The other property is the

Well-foundedness Theorem Every reduction sequence

$$\phi_0 \succ \phi_1 \succ \cdots$$

is finite.

This result, which does depend on type structure, implies that every formula has a (necessarily unique) normal form: simply iterate the process of taking arbitrary simple reductions. By well-foundedness, the process will come to and end with a normal formula. In the case of untyped combinators, we had the counterexample YY, where $Y = SII$. The Well-foundedness Theorem is sometimes called the *Strong Normalization Theorem* because it states not only that every formula has a normal form, but that, no matter what choice one makes in taking successive simple reductions, one will arrive at the normal form.

A simple method of proof, found in a somewhat richer setting in [Tait, 1963, Appendix B] and [Tait, 1967], is as follows: define the notion of a *computable formula of type* A by induction on the complexity of A.

- If A is $\mathcal{D}$ or 2, then a formula ϕ of type A is computable if and only if it is well-founded (i.e. every reduction sequence starting with ϕ is finite.
- A formula ϕ of type $A \Rightarrow B$ is computable if and only if $\phi\psi$ is a

computable formula of type B for every computable formula ψ of type A.

Now prove simultaneously by induction on the complexity of A that every computable term of type A is well-founded and that every variable of type A is computable, and then prove by induction on the complexity of the formula ϕ that, if it is of type A, then it is a computable formula of type A. $\Box$

It follows from Church-Rosser and Well-foundedness that the relation $\equiv$ is a decidable equivalence relation on the set of all formulas of $\mathcal{G}$.

5. The proof of the Explicit Definition Theorem is essentially the same as in the untyped case:

Explicit Definition Theorem II Let $\phi(v)$ be a formula of $\mathcal{G}$ of type B, where v is a free variable of type $\mathcal{D}$. There is a normal formula ϕ' of $\mathcal{G}$ of type $C \Rightarrow B$ such that

$$\phi(v) \equiv \phi' v.$$

ϕ' does not contain v. It contains only variable and constants that are in ϕ and combinators. $\Box$

Again, this theorem does not depend on the special role of $\mathcal{D}$. By introducing the combinators $K_{C,A}$ and $S_{C,A,B}$ for arbitrary type C, the theorem would hold with $\mathcal{D}$ replaced throughout by C.

Given $\phi(v)$, we will again denote the corresponding ϕ' given by the Explicit Definition Theorem II by

$$\Lambda x.\phi(x).$$

Let $\phi(v_1, \ldots, v_n)$ be a formula of type B where $v_1, \ldots, v_n$ are distinct free variables of type $\mathcal{D}$. Then by n iterated applications of the Explicit Definition Theorem

$$\Lambda x_1 \cdots \Lambda x_n.\phi(x_1, \ldots, x_n)$$

is of type

$$\mathcal{D} \Rightarrow_n B$$

and

$$(\Lambda x_1 \cdots \Lambda x_n.\phi(x_1, \ldots, x_n))t_1 \cdots t_n \equiv \phi(t_1, \ldots, t_n)$$

for trms t_i of $\mathcal{F}$.

In particular let $\phi(v)$ be a normal formula of $\mathcal{G}$ of type 2, where v is a free variable of type $\mathcal{D}$. Then $\Lambda x.\phi(x)$ is of type $\mathcal{D} \Rightarrow 2$. So we may introduce the bound variable notation for the universal quantifier as an abbreviation:

$$\forall x.\phi(x) := \forall \Lambda x.\phi(x).$$

$\forall x.\phi(x)$ will be true just in case $(\Lambda x \in \mathcal{D}.\phi(x))d \equiv \phi(d)$ is true for each $d \in D$, just in case it should be.

6. With each formula ϕ of $\mathcal{F}$, we associate a normal formula ϕ^* of $\mathcal{G}$ as follows:

- If ϕ is atomic, then $\phi^* = \phi$.
- $(\phi \to \psi)^* = \ \phi^* \to \psi^*$
- $(\forall x \phi(x))^* = \forall \Lambda x[\phi(x)^*]$ where $\phi(x)^*$ is the result of substituting x for v in $\phi(v)^*$.

We will identify the formula ϕ of $\mathcal{F}$ with the corresponding ϕ^* in $\mathcal{G}$.

Interpretation Theorem [Interpretation of $\mathcal{F}$ in $\mathcal{G}$].

a. The terms of $\mathcal{F}$ are precisely the normal formulas of $\mathcal{G}$ of type $\mathcal{D}$.

b. The formulas of $\mathcal{F}$ are normal formulas of $\mathcal{G}$ of type 2 and every normal formula of $\mathcal{G}$ of type 2 is equivalent to a formula of $\mathcal{F}$.

It is immediate that the terms of $\mathcal{F}$ are normal formulas of $\mathcal{G}$ of type $\mathcal{D}$ and that the formulas of $\mathcal{F}$ are normal formulas of $\mathcal{G}$ of type 2.

Note that a normal formula ϕ of type $\mathcal{D}$ or 2 in $\mathcal{G}$ cannot begin with a combinator. For let ϕ be

$$T\phi_1 \cdots \phi_n$$

where T is a constant and $n \geq 0$. If $T = K_{A,B}$ then $n < 2$, since ϕ is normal. But then ϕ is of type $A \Rightarrow (B \Rightarrow A)$ or of type $B \Rightarrow B)$. A similar argument shows that T cannot be of the form $S_{A,B,C}$.

Proof of *a*. Let ϕ be a normal formula of $\mathcal{G}$ of type $\mathcal{D}$. We prove by induction on the number of occurrences of constants in it that it is a term of $\mathcal{F}$. Let ϕ be

$$T\phi_1 \cdots \phi_n$$

where T is a constant or variable.

- If T is a variable, then $n = 0$ and $\phi = T$ is a term of $\mathcal{F}$.
- If T is a constant, then it is an n-ary function constant ($n \geq 0$) and $\phi_1, \ldots, \phi_n$ are normal terms of type $\mathcal{D}$. So by the induction hypothesis, they are terms of $\mathcal{F}$ and hence so is ϕ.

Proof of *b*. We assume now that $\phi = T\phi_1 \cdots \phi_n$ is a normal formula of $\mathcal{G}$ of type 2, where T is a constant. We prove by induction on the number of nested occurrences of $\forall$ in ϕ and within that, by induction on the number of occurrences of symbols in it, that ϕ is a formula of $\mathcal{F}$.

- T cannot be a variable.
- T is a non-logical constant. Then it is an n-ary relation constant ($n \geq 0$) and, by the first part of the theorem, the ϕ_i are terms of $\mathcal{F}$. So $\phi = \phi^*$ is an atomic formula of $\mathcal{F}$.
- $T = \bot$. Then $n = 0$ and $\phi = \bot$.
- $T = \rightarrow$. Then $n = 2$ and by the induction hypothesis, ϕ_1 and ϕ_2 are equivalent to formulas of $\mathcal{F}$. Hence so is ϕ.
- $T = \forall$. Then $n = 1$ and ϕ_1 is normal and of type $\mathcal{D} \Rightarrow 2$. By the induction hypothesis on the number of occurrences of $\forall$, we can assume that $\phi_1 v$ is equivalent to a formula $\psi'(v)$ of $\mathcal{F}$. So $\phi_1 \equiv \Lambda x \phi_1 x \equiv \Lambda x \psi'(x)$, *where the first equivalence requires η-conversion.* It follows that ϕ is equivalent to $(\forall x \psi'(x))$. □

The restricted use of η-conversion is necessary here, as the example $\forall R$, where R is a unary relation constant of $\mathcal{F}$, makes clear.

By an *n-predicate* I will mean a formula of $\mathcal{G}$ of type $\mathcal{D} \Rightarrow_n 2$ containing no free variables. Every formula of $\mathcal{F}$ is to within equivalence of the form $\phi v_1 \cdots v_n$ for some $n \geq 0$, where ϕ is an n-predicate and the v_i are distinct

variables of type 2. Namely, write the formula as $\phi(v_1, \ldots, v_n)$, where the v_i include all the variables in the formula. Then it is $\equiv (\Lambda x_1 \cdots \Lambda x_n . \phi(x_1, \ldots, x_n) v_1 \cdots v_n$. By an *n-term*, I mean a formula of $\mathcal{G}$ of type $\mathcal{D} \Rightarrow_n \mathcal{D}$ containing no free variables. Every term of $\mathcal{F}$ is to within equivalence of the form $tv_1 \cdots v_n$, where t is an n-term and the v_i include all the free variables in the term.

7. It will be useful to briefly describe Quine's method of eliminating bound variables in first-order formulas [1960a; 1981] and relate it to the present approach. Quine was somewhat more dismissive of the theory of combinators of Schönfinkel and Curry than an attentive peruser of [Curry and Feys, 1958] might have been.

Remark 2. A year prior to Quine's paper, as he himself observed, Bernays published a paper [1959] that he had delivered at a colloquium on constructive mathematics in 1957 in Amsterdam and in which he accomplishes essentially the same thing. Bernays abstracted from the von Neumann-Bernays operations for constructing the first-order definable classes of sets in axiomatic set theory to obtain operations for constructing the first-order definable subclasses of $\mathcal{D}^n$ $(n \geq 0)$ in our ambient model of $\mathcal{F}$. There is an interesting historical puzzle connected with Bernays' lecture: he refers to Curry's 'combinatory theory of functionality' in connection with his construction, but without any details or further reference to Curry. Chapter 9 of Curry-Feys is entitled "The Basic Theory of Functionality", and that is where type structure is introduced into combinator theory. If Bernays had just been referring to untyped combinatory logic, it would have been natural for him to refer also to Schönfinkel—after all, he had known him and certainly knew his paper on combinators. On the other hand, the publication date of Curry-Feys is 1958, a year after Bernays' lecture.

Quine's treatment is a bit easier to present than Bernays' and we can use bits of it. In fact, Quine makes a restriction on the first-order language $\mathcal{F}$ in that he assumes that there are no individual or function constants. Another difference from our treatment is that Quine takes $\wedge$, $\neg$, and $\exists$ as the logical constants, whereas we take $\rightarrow$, $\bot$ and $\forall$. But lets leave these differences aside and interpret Quine's term 'n-ary *predicate* simply to mean an $n-$predicate in our sense. Let P^n denote the set of n-predicates. The *atomic* n-predicates are those containing no occurrences of $\rightarrow$ or $\forall$. (In the absence of individual or function constants, this would coincide with Quine's notion of an atomic

n-ary predicate.) Quine introduces certain operations on these classes such all n-predicates can be built up from the atomic ones by means of these operations. The operations in question, which are all definable in terms of the combinators, are, for each $n \geq 0$:

- The operation
$$\rightarrow^n \in P^n \Rightarrow_2 P^n$$
defined by
$$\rightarrow^n \phi\psi := \Lambda x_1 \cdots x_n.[\phi x_1 \cdots x_n \rightarrow \psi x_1 \cdots x_n].$$

- The operation
$$\forall^n \in P^{n+1} \Rightarrow P^n$$
defined by
$$\forall^n \phi := \Lambda x_1 \cdots x_n.\forall[\phi x_1 \cdots x_n]$$
.

- The operation
$$\uparrow^n \in P^n \Rightarrow P^{n+1}$$
defined by
$$\uparrow^n \phi := \Lambda x_1 \cdots x_{n+1}.\phi x_1 \cdots x_n.$$
So $\uparrow^n$ transforms an n-predicate into an $n+1$-predicate by adding a dummy last argument to it.

- The operation
$$\downarrow^n \in P^{n+1} \Rightarrow P^n$$
defined by
$$\downarrow^n \phi := \Lambda x_1 \cdots \Lambda x_n.\phi x_1 \cdots x_n x_n.$$

- For each permutation π of the set $\{1, \ldots, n\}$, the operation
$$\pi \in P^n \Rightarrow P^n$$
defined by
$$\pi\phi := \Lambda x_1 \cdots x_n.\phi x_{\pi 1} \cdots x_{\pi n}.$$

Quine makes use of the fact that all the permutations of $\{1, \ldots, n\}$ can be generated from just two, the transposition $(1, 2)$ and the cycle $(1, \ldots, n)$, to restrict the operations π on P^n to just these two.

The superscripts n on $\rightarrow^n, \forall^n, \uparrow^n$ and $\downarrow^n$ are tedious and can be deduced from the nature of the predicates to which the operations in question are applied; so I will drop them when there can be no confusion. We will also have occasion to use the object

$$\perp^n := \Lambda x_1 \cdots \Lambda x_n \perp$$

but, again, will decently refrain from adding the superscript unnecessarily.

When ϕ is an $n+1$-predicate, r is an n-term and t is a term of $\mathcal{F}$, we define $\phi^* r$ and $\phi * t$ by

$$\phi * r := \Lambda x_1 \cdots x_n . \phi x_1 \cdots x_n (r x_1 \cdots x_n).$$

$$\phi * t := \Lambda x_1 \cdots x_n . \phi x_1 \cdots x_n t.$$

When ϕ is an n-predicate, its *universal closure* is the sentence

$$\phi^+ = \forall \cdots \forall \phi \ (= \forall^0 \cdots \forall^{n-1} \phi).$$

PART II

8. We noted that variable binding in first-order logic doesn't stop with the explicit variable binding of the quantifiers: it occurs in deductions, too.

For example, when by the rule

$$\frac{\begin{array}{c}\vdots \\ \phi(v)\end{array}}{\forall x \phi(x)}$$

of $\forall$-*Introduction* we pass from a deduction $p(v)$ of $\phi(v)$ with free individual variable v to a deduction of $\forall x \phi(x)$, we are essentially binding the variable in $p(v)$—Lets denote the result by

$$\Lambda x \in \mathcal{D} . p(x).$$

(It will become clear why this is an appropriate notation.) Remember the usual restriction: If the deduction $p(v)$ contains an hypothesis ψ and v occurs in ψ, then v is said to be *fettered* in $p(v)$. The required restriction is that v be unfettered in $p(v)$. Otherwise, for example, from the hypothesis $\phi(v)$ itself we would be able to deduce $\forall x \in \mathcal{D}.\phi(x)$.

Similarly, when by the rule

$$\frac{\begin{array}{c}[\phi]\\ \vdots \\ \psi\end{array}}{\phi \to \psi}$$

of $\to$-*Introduction*, we pass from a deduction p of ψ from the hypothesis ϕ to a deduction of $\phi \to \psi$, the hypothesis should be thought of as a free variable ranging over proofs of ϕ—so that p should really be written $p(v)$ again, where v is a free variable ranging over proofs of ϕ. In passing from the deduction of ψ from ϕ, we 'discharge zero or more occurrences of the hypothesis ϕ in the deduction: i.e. we are binding that variable v. That is the significance of the square brackets around ϕ.—Lets denote the result by

$$\Lambda x \in \phi.p(x).$$

Here we don't need the restriction that v be unfettered in $p(v)$: it is always satisfied. For in first-order logic variables ranging over proofs of other formulas (if we don't count $\mathcal{D}$ as a formula) do not occur in formulas.[3]

But before discussing this further, we need to change our attitude towards the nature of sentences and deductions. If one thinks of deductions simply as arrays of symbols with a specified structure (as for example did Hilbert in his proof theory), then no semantics is involved. But if one takes the more interesting position that deductions denote something, namely proofs, and that proofs themselves have mathematical structure (and perhaps this position was first manifested by Brouwer in his argument for the Bar Theorem), then semantics is in business.

We may think of the class of proofs of a sentence as its meaning. In Part I we assigned formulas in $\mathcal{G}$ types: these are the types of the objects they

[3]Of course, we could consider a richer system of propositions in which this is no longer the case. This would lead to the Curry-Howard theory of dependent types, which is more complicated. But there is no—or at least, insufficient—reason to go there now. Again, see [Tait, 1998] for a formalization without bound variables of Curry-Howard type theory.

denote, and sentences all have type 2. But when we are talking about proof, we are not talking about truth-values: 1 doesn't need a proof and 0 doesn't have one. A sentence not only has a truth-value, it has a *sense*: it expresses a 'thought' as Frege put it, or a proposition,as I would prefer to put it. We may regard the sense of a mathematical sentence to be given by the class of objects that count as proofs of it. In what follows we shall use the sentence to denote the class of its proofs; in Frege's words, we will use it with *oblique reference*—to denote what, in non-oblique contexts, is its sense.

If one accepts this 'proposition-as-type' ideology, then eliminating bound variables from deductions is an essential part of providing a compositional semantics for sentences. But this ideology aside, just assuming one takes seriously the idea of proofs as objects, the problem of providing deductions with a compositional semantics remains.

An n-predicate then denotes a Cartesian product

$$\Pi_{x_1 \ldots x_n} Q(x_1, \ldots, x_n)$$

where the x_i range over $\mathcal{D}$. We call this its *type*. So the type of a sentence in the present sense is a set Q—which we have agreed to simply denote by ϕ. In what follows, we will identify an n-predicate or n-term with its normal form. The connection between the type (viz. 2) of the sentence as an element of $\mathcal{G}$ in Part I and its type Q in the present sense is simply that it denotes 1 in the earlier sense just in case Q is non-empty.[4]

- We assume that the atomic n-ary predicates have already been assigned denotations.

  This is of course more of a demand on the given semantics of $\mathcal{F}$ than we assumed in Part I, where it was only assumed that a denotation in $\mathcal{D} \Rightarrow_n \mathcal{D}$ is assigned to each n-ary function constants and a denotation in $\mathcal{D} \Rightarrow_n 2$ is assigned to each n-ary relation constant.

  Notice that the n-predicates include not just atomic relation symbols, but all expressions $\Lambda x_1 \cdots \Lambda x_n \phi(x_1, \ldots x_n)$, where $\phi(v_1, \ldots v_n)$ is an atomic formula of $\mathcal{F}$, possibly containing function symbols.

[4]From the point of view of abandoning denotation in favor of sense, however, here is a retro element in our interpretation of $n+1$-predicates: $Q(x_1, \ldots, x_{n+1})$ doesn't depend on the sense of (the assignments of values to) the x_i, only on their denotations. I will not go into this.

- Of course, we know what class of proofs $\bot$ denotes: it is the null class.

- What about $\phi \to \psi$ or, in official notation, $\to \phi\psi$? Its proofs should should enable us to move from a proof of ϕ to a proof of ψ: That is what the rule

$$\frac{\phi \qquad \phi \to \psi}{\psi}$$

of $\to$-*Elimination* or *Modus Ponens* tells us. This suggests that a proof of $\phi \to \psi$ should be a function from proofs of ϕ to proofs of ψ—i.e $\phi \to \psi$ denotes $\phi \Rightarrow \psi$. We shall accept that suggestion. In view of this, we can formulate $\to$-Elimination more fully as the rule

$$\frac{p \in \phi \qquad f \in \phi \to \psi}{fp \in \psi}$$

of application of a function to an argument, in accordance with our aim of compositional semantics.

In particular, consider the proof of $\phi \to \psi$ that we denoted by $\Lambda x \in \phi.p(x)$ and that we obtain by $\to$-Introduction from the deduction $p(v)$ of ψ from the hypothesis v of ϕ. The notation is justified in that the proof is a function, namely the one defined by the lambda-conversion

$$[\Lambda x \in \phi.p(x)]r \equiv p(r)$$

when r is a proof of ϕ.

- $\forall x \in \mathcal{D}.\phi(x)$ is very like an implication: a proof should yield us, for every $d \in \mathcal{D}$, a proof of $\phi(d)$. That is what the rule

$$\frac{\forall x \in \mathcal{D}.\phi(x)}{\phi(t)}$$

of $\forall$-*Elimination* or of *Instantiation* tells us. Thus, $\forall x \in \mathcal{D}.\phi(x)$ should express the type of all functions f defined on $\mathcal{D}$ such that, for each $d \in \mathcal{D}$, $fd \in \phi(d)$. In other words, $\forall x \in \mathcal{D}.\phi(x)$ just expresses the Cartesian product $\Pi_{d \in \mathcal{D}}.\phi(d)$. In view of this, $\forall$-Elimination is more fully expressed as the rule

$$\frac{t \in \mathcal{D} \qquad f \in \forall x.\phi(x)}{ft \in \phi(t)}$$

of application of a function to an argument.

Again, consider the proof $\Lambda v.p(v)$ that we obtain by $\forall$-Introduction from the deduction $p(v)$ of $\phi(v)$ from the 'hypothesis' v of $\mathcal{D}$. That too denotes a function, namely the one defined by the lambda conversion

$$[\Lambda x.p(x)]r = p(r)$$

when $r \in \mathcal{D}$, i.e. when r is a 'proof' of $\mathcal{D}$.

In the richer environment of the Curry-Howard type theory, one can understand $\phi \to \psi$ as just a special case of $\forall x \in \phi.\psi(x)$, namely the case in which v does not occur in $\psi(v)$. But that unification isn't available to us, since we are not admitting quantification over types other than $\mathcal{D}$. (We may regard $\mathcal{D}$ as a type in the present sense, as opposed to the type structure associated with $\mathcal{G}$: it is the type of the objects in $\mathcal{D}$.

Remark 3. The extension of the Curry-Feys analogy between the theory of functionality expressed by typed combinators and positive implicational logic to a conception of the formulas of first-order predicate logic as types was, as I mentioned above, due to Howard [1980].

9. Now we have made explicit the problem, namely of eliminating the variable-binding operation Λ from deductions. Our deductions are going to be built up by means of evaluation ultimately from given constant deductions of closed formulas of $\mathcal{F}$ and variable deductions of formulas of $\mathcal{F}$. So, again, every deduction will be *uniquely* of the form

$$p_0 p_1 \cdots p_n$$

where p_0 is either a constant or a variable and the remaining p_i are either deductions or terms of $\mathcal{F}$ ('deductions' of $\mathcal{D}$). The types of the constant deductions then are the *axioms* and we must, in each case[5], give a definition of the constant by means of conversion rules that is in keeping with its type, the axiom. I will write

$$p \vdash \phi$$

[5]With the exception of the axioms for $\bot$, which we will discuss below.

to mean that p is a deduction of ϕ.

Before proceeding, it is convenient to add a condition on $\mathcal{F}$, namely that it contain at least one individual constant, witnessing the fact that $\mathcal{D}$ is non-empty. (It is not really a restriction: we could set aside a variable to play the special role of the constant. Then, where we speak of a variable-free deductions, i.e. without assumptions. it should be understood to mean one in which there are no variables other than the distinguished one.)

The obvious way to eliminate bound variables is, of course, to consult Schönfinkel again: we need to define $\rightarrow$-Introduction $\Lambda x \in \phi.p(x)$ and $\forall$-Introduction $\Lambda x \in \mathcal{D}.p(x)$ in terms of suitable combinators. These will be constant deductions whose types then are axioms.

- For all n-predicates ϕ, ψ and χ ($n \geq 0$), we introduce the constant deductions

  $$K_{\chi,\phi} \vdash [\phi \rightarrow (\chi \rightarrow \phi)]^+ \qquad K_\phi \vdash [\phi \rightarrow \forall \uparrow \phi]^+$$

  Let $\bar{t}$ be the string of n terms of $\mathcal{F}$. Then

  $$K_{\chi,\phi}\bar{t} \vdash \phi\bar{t} \rightarrow (\chi\bar{t} \rightarrow \phi\bar{t}) \qquad K_\phi\bar{t} \vdash \phi\bar{t} \rightarrow \forall \uparrow (\phi\bar{t}).$$

  In each case, K is defined by

  $$K\bar{t}pc\ CONV\ p.$$

  In both cases, p is a deduction of $\phi\bar{t}$. In the first case, c is a deduction of $\chi\bar{t}$. In the second case, c is a term of $\mathcal{F}$ (i.e. of type $\mathcal{D}$). $(\forall \uparrow (\phi t))c \equiv \phi\bar{t}$ and so this makes sense.

- For all n-predicates ϕ, ψ and χ ($n \geq 0$), we introduce the constant deduction

  $$S_{\chi,\phi,\psi} \vdash [\chi \rightarrow (\phi \rightarrow \psi)] \rightarrow [(\chi \rightarrow \phi) \rightarrow (\chi \rightarrow \psi))]^+$$

  and for each n and all $n+1$-predicates ϕ and ψ, the constant deduction

  $$S_{\phi,\psi} \vdash [\forall(\phi \rightarrow \psi) \rightarrow (\forall\phi \rightarrow \forall\psi)]^+.$$

  Again, for $\bar{t}$ a string of n terms of $\mathcal{F}$,

  $$S_{\chi,\phi,\psi} \in [\chi t \rightarrow (\phi\bar{t} \rightarrow \psi\bar{t})] \rightarrow [(\chi\bar{t} \rightarrow \phi\bar{t}) \rightarrow (\chi\bar{t} \rightarrow \psi\bar{t}))]$$

and

$$S_{\phi,\psi}\bar{t} \in [\forall(\phi\bar{t} \to \psi\bar{t}) \to (\forall\phi\bar{t} \to \forall\psi\bar{t})].$$

In both cases then S is defined by

$$Stpqc\ CONV\ (pc)(qc).$$

In the first case, $p \in \chi\bar{t} \to (\phi\bar{t} \to \psi t)$, $q \in \chi\bar{t} \to \phi\bar{t}$ and $c \in \chi\bar{t}$. In the second case, $p \in \forall(\phi\bar{t} \to \psi\bar{t})$, $q \in \forall\phi\bar{t}$ and c is a term of $\mathcal{F}$.

As we mentioned above, the two forms of the K combinators and the two forms of the S combinators collapse to one when we regard $\phi \to \psi$ as the vacuous quantification $\forall x\phi.\psi$.

We need three more kinds of constants: for each $n+1$-predicate ϕ, n-predicate χ and n-term r, we introduce the constants

$$C_{\phi,r} \vdash [\forall\phi \to \phi * r]^+$$

and

$$D_\phi \vdash [\forall\phi \to \forall x(\phi * x)]^+$$

and for ϕ an $n+2$-predicate we introduce the constant

$$E_\phi \vdash [\forall\forall\phi \to \forall \downarrow \phi]^+$$

$C_{\phi,r}$ is defined as follows: if $\bar{t}$ is a string of n terms of $\mathcal{F}$ and $p \vdash \forall\phi\bar{t})$, then $C_{\phi,r}$ is defined by

$$C_{\phi,r}\bar{t}p\ CONV\ p(r\bar{t}).$$

Of course, when $\mathcal{F}$ contains no individual or function constants, there are no n-terms r and so the constants $C_{\phi,r}$ are not in play. $\mathcal{D}_\phi$ is defined by

$$D_\phi\bar{t}ps\ CONV\ ps.$$

and E_ϕ is defined by

$$E_\phi\bar{t}ps\ CONV\ pss.$$

We define the notion of one deduction p reducing to another q, $p \succeq q$, as usual; however in this case we do not require η-conversion. Both the Church-Rosser Theorem and the Well-foundedness Theorem are proved essentially

as they are in Part I, so that the relation of two deductions being equivalent, $p \equiv q$, is a well-defined and decidable equivalence relation.

We complete the definition of $\vdash$ by specifying that

$$p \equiv q,\ \phi \equiv \psi,\ p \vdash \phi \Rightarrow\ q \vdash \psi.$$

10. Before proving that $\rightarrow$-introduction and $\forall$-introduction can be eliminated (Explicit Definition Theorem III), we need some lemmas. The first lemma establishes propositional logic under the closure operation $^+$ and derives from [Quine, 1966, p. 90].

Lemma 1. Let ϕ and ψ be n-predicates. If $p \vdash (\phi \rightarrow \psi)^+$ and $q \vdash \phi^+ \ \Rightarrow$ $\vdash \psi^+$., then there is an $F \vdash \psi^+$ such that for t a string of variables of type $\mathcal{D}$,

$$Ft \succeq pt(qt).$$

Proof by induction on n. When $n = 0$, $F = pq$. Assume $n > 0$. We have

$$p \vdash (\forall(\phi \rightarrow \psi))^+$$

and (omitting type subscripts) the combinator

$$S \vdash [\forall(\phi \rightarrow \psi) \rightarrow (\forall\phi \rightarrow \forall\psi)]^+.$$

So by the induction hypothesis, we have $G \vdash (\forall\phi \rightarrow \forall\psi)^+$, where $G\overline{u} \succeq Su(p\overline{u})$, for a string $\overline{u}$ of $n-1$ variables of type $\mathcal{D}$. $q \vdash (\forall\phi)^+$ and so by the inductive hypothesis again, there is an $F \vdash \psi^+$ with $F\overline{u} \succeq G\overline{u}(q\overline{u}) \succeq S\overline{u}(p\overline{u})(q\overline{u})$ and so with v a variable of type $\mathcal{D}$

$$F\overline{u}v \succeq S\overline{u}(pu\overline{u})(q\overline{u})v \succeq p\overline{u}v(q\overline{u}v).$$

□

Lemma 2. Let χ and ϕ be n-predicates and $\bar{t}$ a string of n variables of type $\mathcal{D}$.

i) If $p \vdash \phi^+$, then there is an $F \vdash (\chi \rightarrow \phi)^+$ such that $F\bar{t}q \succeq p\bar{t}$ for $q \vdash \chi\bar{t}$.

ii) If $p \vdash (\chi \to (\phi \to \psi)^+$, then there is an $F \vdash [(\chi \to \phi) \to (\chi \to \psi)]^+$ such that $F\bar{t}qr \succeq p\bar{t}r(qr)$ for $q \vdash \chi\bar{t} \to \phi t$ and $r \vdash \chi\bar{t}$.

By Lemma 1 using the combinators $K_{\chi,\phi}$ and $S_{\chi,\phi,\psi}$, resp. □

Lemma 3. Let χ be an n-predicate, ϕ an $n+1$-predicate, ψ an $n+2$-predicate and r an n-term. Then there are variable-free

$$C'_{\chi,\phi,r} \vdash [(\chi \to \forall\phi) \to (\chi \to \phi * r)]^+$$

$$D'_{\chi,\phi} \vdash [(\chi \to \forall\phi) \to (\chi \to \forall x\phi * x)]^+$$

such that, for $\bar{t}$ a string of n variables of type $\mathcal{D}$

$$C'_{\chi,\phi,r}\bar{t}pq \succeq C_{\phi,r}\bar{t}(pq), \qquad D'_{\chi,\phi}\bar{t}pq \succeq D_{\phi}\bar{t}(pq).$$

By Lemma 2 i) we have $F \vdash [\chi \to (\forall\phi \to \phi * r)]^+$ with $F\bar{t}r \succeq C_{\phi,r}\bar{t}$, where $r \vdash \chi\bar{t}$. So by Lemma 2 ii), we have $C'_{\chi,\phi,r}$ such that

$$C'_{\chi,\phi,r}\bar{t}pq \succeq F\bar{t}q(pq) \succeq C_{\phi.r}\bar{t}(pq).$$

Similarly for $D'_{\chi,\phi}$. □

Lemma 4. Let $p \vdash \phi$, where ϕ is normal, and let v be a variable of type $\mathcal{D}$. If p does not contain v, then neither does ϕ.

This is easily proved by induction on the complexity of p. Note that, if ϕ is a formula of $\mathcal{G}$ that does not contain v, then its normal form does not contain v either. □

For ϕ an n-predicate and $\bar{t}$ a list of n variables of type $\mathcal{D}$, the identity function $I_{\phi\bar{t}} \vdash \phi\bar{t} \to \phi\bar{t}$ is defined as usual:

$$I_{\phi\bar{t}} = S_{\phi,\phi\to\phi,\phi}\bar{t}(K_{\phi,\phi\to\phi}\bar{t})(K_{\phi,\phi}\bar{t}).$$

(The notation $I_{\phi,\bar{t}}$ may be misleading, since it looks like $I_{\phi\bar{t}}$ is a constant. It is constant only when $n = 0$.) So $I_{\phi,\bar{t}}p \equiv p$ for $p \vdash \phi\bar{t}$.

Explicit Definition Theorem III.

a. Let v be a variable of type χ and $p(v) \vdash \phi$. There is a deduction $p' \vdash \chi \to \phi$ built up from the combinators and constants or variables in $p(v)$ other than v, such that $p'v \equiv p(v)$.

b. Let v be a variable of type $\mathcal{D}$ and $p(v) \vdash \phi(v)$, where v is unfettered $p(v)$. Then there is a $p' \vdash \forall\phi' = \forall x\phi(x)$ built up from the combinators and constants or variables in $p(v)$ other than v, such that $p'v \equiv p(v)$.

Proof of *a*). For all sufficiently large n, we have $\phi \equiv \phi_0\bar{t}$ and $\chi \equiv \chi_0\bar{t}$, where ϕ_0 and χ_0 are n-predicates and $\bar{t}$ is a list of n variables of type $\mathcal{D}$.

i) $p(v) = v$. Then $p' = I_{\phi_0,\bar{t}}$.

ii) $p(v) = p$ does not contain v. Then $p' = K_{\chi_0,\phi_0}\bar{t}p$.

iii) $p(v) = q(v)r(v)$, where $q(v) \vdash \psi \to \phi$ and $r(v) \vdash \psi$. By taking n large enough, we can assume that $\psi \equiv \psi_0\bar{t}$. Then $p' = S_{\chi_0,\psi_0,\phi_0}\bar{t}q'r'$.

iv) $p(v) = q(v)r$, where $q(v) \vdash \forall\psi$ and r is a term of $\mathcal{F}$. Then $q(v) \vdash \forall\psi$ and $q' \vdash \chi \to \forall\psi$. There are two subcases:

- r is a variable. Set $p' = D'_{\chi,\psi}\bar{t}q'r$.
- Otherwise we can assume that $r \equiv r_0\bar{t}$ where r_0 is an n-term. Set $p' = C'_{\chi_0,\psi_0,r_0}\bar{t}q'$.

Proof of *b*). For sufficiently large n, $\phi \equiv \phi_0\bar{t}$, where ϕ_0 is an $n+1$-predicate and $\bar{t}$ is a string of distinct variables of type $\mathcal{D}$

i) $p(v) \neq v$.

ii) $p(v) = p$ does not contain v. Then by Lemma 4, the normal form of ϕv doesn't either. The normal form is $\equiv \phi_1\bar{t}$ for some n predicate ϕ_1. Set $p' = K_{\phi_1}\bar{t}p$.

iii) $p(v) = q(v)r(v)$ where $q(v) \vdash \chi(v) \to \phi(v)$. Set $p' = S_{\chi_0,\phi_0}\bar{t}q'r'$.

iv) $p(v) = q(v)r(v)$ where $q(v) \vdash \forall\psi(v)$ and $r(v)$ is a term of $\mathcal{F}$. We can assume that $\psi(v) \equiv \psi_0\bar{t}v$ and that $r(v)$ is either a variable or is $\equiv r_0\bar{t}v$, where ψ_0 is an $n+2$-predicate and r_0 is an n-term. $q' \vdash \forall\forall\psi_0\bar{t}$. There are three subcases.

– $r(v) \equiv r_0\bar{t}v$, where r_0 is an n-term. $C\bar{t} = C_{\forall\psi_0,r_0}\bar{t} \vdash \forall\forall\psi_0\bar{t} \to \forall\psi\bar{t} * (r_0\bar{t})$ and $q' \vdash \forall\forall\psi_0\bar{t}$. So $p' = Cu(q_0\bar{t}) \vdash \forall\psi_0\bar{t} * (r_0\bar{t})$. Thus $p'v \vdash \psi_0\bar{t}v(r_0\bar{t}v) \equiv \phi(v)$.

– $r(v) = r$ is a variable other than v. Set $p' = D_{\forall\psi_0}\bar{t}q'r$.

– $r = v$. Then set $p' = E_{\psi_0}\bar{t}q'r$.

□

Note that in part b), $ii) - iv)$ are the only possible cases only because v is unfettered in $p(v)$. For example , if $p(v)$ were a variable of type ϕv, it wouldn't fall under either of these cases.

11. To obtain a complete notion of logical deduction, we have only to introduce the axioms for negation $\perp$; $\perp$-*elimination*

$$N_\phi \vdash [\perp^n \to \phi]^+$$

and (for classical logic) *Double* $\perp$-*elimination*

$$DN_\phi \vdash [((\phi \to \perp^n) \to \perp^n) \to \phi]^+$$

for each n-predicate ϕ. Unlike the other constant deductions, these have no definitions. We can think of these as theological axioms. 1) If there is a benevolent deity, there are no variable-free deductions of $\perp$ to which N_ϕ can be applied, and 2) the classical doctrine expressed by DN_ϕ is irreducible. God has made the rational universe simple: if we have established $\neg\neg\phi$, then ϕ is indeed true, although we may not have a direct proof of it. Indeed, when ϕ is a sentence and $p \vdash (\phi \to \perp) \to \perp$, that DNp cannot in general be computed, yielding a 'real' deduction of ϕ, is precisely what nonbelievers (i.e. constructive mathematicians) complain about.

So here are the axioms, i.e. the types of the constants, from which all the logical first-order truths in the language $\mathcal{F}$ can be derived using modus ponens:

i) $[\phi \to (\chi \to \phi)]^+$

ii) $[\chi \to (\phi \to \psi)] \to [(\chi \to \phi) \to (\chi \to \psi))]^+$

iii) $[\phi \to \forall x\phi]^+$

iv) $\forall(\phi \to \psi) \to (\forall\phi \to \forall\psi)]^+$

v) $[\forall\forall\phi \to \phi * r]^+$ Here ϕ is an $n+1$-predicate and r an n-term.

vi) $[\forall\phi x \to \forall x\phi * x]^+$

vii) $[\forall\forall\phi \to \forall \downarrow \phi]^+$

viii) $[\bot\to \phi]^+$

ix) $[(\phi \to\bot) \to\bot]^+$

For $n = 0$, i) and ii) are the standard axioms for the theory of implication. Added to viii) and ix) for $n = 0$, they are complete for propositional logic. If we replace i), ii), vii) and viii) by the axiom schema

$$\phi^+ \qquad \text{(for every tautology } \phi \text{ in } \mathcal{F})$$

and drop v), then this axiom system is easily seen (using Lemma 1) to be equivalent to that of Quine's *Mathematical Logic* [1951]. Axiom v) is rendered inoperative in Quine's system because the only terms are variables: there are no n-terms.

References

Bernays, P. [1959]. Über eine naturliche Erweiterung des Relationkalkuls, *in* A. Heyting (ed.), *Constructivity in Mathematics: Proceedings of the Colloquium Held in Amsterdam, 1957*, Amsterdam: North-Holland, pp. 1–14.

Church, A. [1941]. *The Calculi of Lambda-Conversion*, Princeton: Princeton University Press.

Church, A. and Rosser, J. B. [1936]. Some properties of conversion, *Transactions of the American Mathematical Society* **39**(3): 472482.

Curry, H. and Feys, R. [1958]. *Combinatory Logic I: Studies in Logic and the Foundations of Mathematics*, Amsterdam: North-Holland. 2nd edition 1968.

Howard, W. [1980]. The formula-as-types notion of construction, *in* J. Hindley and J. Seldon (eds), *To H.B. Curry: Essays on Combinatorial Logic, Lambda Calculus and Formalism*, London: Academic Press, pp. 479–490.

Quine, W. [1951]. *Mathematical Logic: Revised Edition*, Cambridge: Harvard University press.

Quine, W. [1960a]. Variables explained away, *Proceedings of the Americal Philosophical Society* . Reprinted in [Quine, 1966a, 227–235].

Quine, W. [1966]. On Carnap's views on ontology, pp. 127–134.

Quine, W. [1966a]. *Selected Logic Papers*, New York: Random House.

Quine, W. [1981]. Predicates, terms and classes, pp. 164–172.

Schönfinkel, M. [1924]. Über die Bausteine der mathematischen Logik, *Mathematische Annalen* **92**: 305–316.

Tait, W. [1963]. A second order theory of functionals of higher type, with two appendices. Appendix A: Intensional functionals. Appendix B. An interpretation of functionals by convertible terms, *Stanford Seminar Report* pp. 171–206. Unpublished. A published version is [Tait, 1967].

Tait, W. [1967]. Intensional interpretations of functionals of finite type I, *Journal of Symbolic Logic* **32**: 198–212.

Tait, W. [1998]. Variable-free formalization of the Curry-Howard type theory, *in* G. Sambin and J. Smith (eds), *Twenty-Five Years of Constructive Type Theory*, Oxford: Oxford University Press, pp. 265–274.