INFORMATION AND CONTROL 59, 127–147 (1983)

Semantics for Classical AUTOMATH and Related Systems*

HENK BARENDREGT

Mathematical Institute, University of Utrecht, Budapestlaan 6; 3508 TA Utrecht, The Netherlands

AND

ADRIAN REZUS

Furkabaan 680, 3524 ZL Utrecht, The Netherlands

INTRODUCTION

Developed from ideas of N. G. de Bruijn (1967) at the Eindhoven University of Technology (The Netherlands), the Automated Mathematics Project is a programme of formalization of actual mathematical texts in view of computer-assisted proof-checking (cf. [5, 6, 11, 21]).

Leaving aside the underlying pragmatic motivations [5, 21], the main languages in the AUT(OMATH)-family may be, roughly, viewed, as being applied typed lambda-calculi with a generalized type structure: a legal AUT-type may often depend on parameters on which its "inhabitants" also depend, such that the AUT-types cannot be characterized "beforehand" (as is the case in the First- and Second-Order Typed Lambda-Calculi [9, 8, 20]).

In [21] one of the authors complained of the lack of formal semantics for the main AUT-languages, briefly surveying the epistemological status of the problem. The present paper is intended to fill in this gap, providing a "mathematical" model-theory for Classical AUTOMATH (CA for short, otherwise called "AUT-68"; see [21] for a detailed description). The main work relies on suggestions given in [24] and consists, essentially, of a "translation" of the type-distinctions of CA into a type-free setting, viz., into specific models of the type-free lambda-calculus [2, 3].

The proposed semantics covers, obviously, the First-order Typed Lambda-Calculus of Church [2] and the analogue Theory of Functionality of Curry [9] and extends, almost trivially, to more involved type-structures as, e.g.,

* The work of the second author was partly completed during his stay at the Department of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands.

0019-9958/83 $3.00

643/59/1-3-9

those present in the Second-Order Typed Lambda-Calculus[1] of Girard [8] and Reynolds [20] or in Pure LCF ([18], etc.). However, somewhat "more structure" is necessary in order to interpret—essentially along the same lines—Zucker's AUT-Pi system of [27] or Martin-Löf's Intuitionistic Theory of Types (with one universe; see [16, 17] and [1, 4] for alternative semantics of the latter), topics which will be discussed in detail elsewhere.

1. Closure Operations in Additive Domains

Let D, D', D'',..., range over complete lattices. For D fixed arbitrarily, $\sqsubset_D$ stands for the underlying partial order and $\sup_D X$ denotes the supremum of $X \subseteq D$. A set $X \subseteq D$ is *directed* if every finite $Y \subseteq X$ has an upper bound in X. A map $f\colon D \to D'$ is *continuous* if it preserves suprema of directed sets, i.e., $f(\sup_D X) = \sup_{D'}\{f(x)\colon x \in X\}$, for directed $X \subseteq D$. (This is, in fact, topological continuity relative to the so-called *Scott-topology*; see [22; 7, Chap. II].) The *cartesian product* $D \times D'$ of two complete lattices consists of ordered pairs partially ordered "componentwise": $\langle d_1, d_1'\rangle \sqsubset \langle d_2, d_2'\rangle$ iff $d_1 \sqsubset_D d_2$ and $d_1' \sqsubset_{D'} d_2'$. The *function space* $[D \to D']$ consists of all continuous $f\colon D \to D'$, with the pointwise ordering: $f \sqsubset g$ iff $\forall x \in D.\ f(x) \sqsubset_{D'} g(x)$. Obviously, these constructions provide new complete lattices from old.

1.1. Proposition. (i) *A map $f\colon D \times D' \to D''$ is continuous iff it is continuous in each variable separately.*

(ii) *The map* $\mathrm{ev}\colon [D \to D'] \times D \to D'$, *defined by* $\mathrm{ev}(f, x) = f(x)$, *is continuous.*

(iii) *Let $f \in [D \times D' \to D'']$. Then the map $\hat{f}\colon D \to [D' \to D'']$,*

defined by $\hat{f}(x) = \lambda y \cdot f(x, y)$ is continuous. Moreover, the map $\mathrm{abs} = \lambda f \cdot \hat{f}$ *is continuous.*

Proof. Well known; cf. [22] or [7]. ∎

It follows that the category of complete lattices with continuous maps as morphisms is cartesian closed.

1.2. Proposition. *Every $f \in [D \to D]$ has a fixed point. Moreover, there is a map $fix \in [[D \to D] \to D]$ such that $fix(f)$ is, for $f \in [D \to D]$, the least fixed point of f.*

Proof. Define $fix(f) = \sup_D\{f^n(\bot)\colon n \in \mathbb{N}\}$, where $\bot\ (= \sup_D \varnothing)$ is the least element of D and $f^0(d) = d, f^{n+1}(d) = f(f^n(d))$, $\forall n \in \mathbb{N}$. ∎

[1] This has been pointed out by G. Longo, in conversation.

Let id_D be the identity on D. So, e.g., $\mathrm{id}_{[D\to D]}(f)=f$, $\forall f\in[D\to D]$.

1.3. DEFINITION. (i) A complete lattice D is (a) *reflexive* (*domain*) if $[D\to D]$ is a retract of D; i.e., there are continuous maps $F: D\to[D\to D]$ and $G: [D\to D]\to D$ such that $F\circ G=\mathrm{id}_{[D\to D]}$. (F, G) is a *retraction pair* with *retraction maps* F and G.

(ii) If, moreover, $G\circ F\sqsupset \mathrm{id}_D$ then D is an *additive domain*, while, if $G\circ F=\mathrm{id}_D$, then D is an *extensional domain*.

1.4. EXAMPLES. (i) A well-known additive domain is the *Graph Model* $P\omega=\{x: x\sqsubset\mathbb{N}\}$, partially ordered by set inclusion (see [24]). To define the retraction maps, let

$$(n, m)=\tfrac{1}{2}(n+m)(n+m+1)+m$$

be the Cantor coding of natural numbers and $(e_n)_{n\in\mathbb{N}}$ be an effective enumeration of the finite subsets of $\mathbb{N}$ via $e_n=\{k_0, k_1,\ldots, k_{m-1}\}$, with $k_0<k_1<\cdots<k_{m-1}$, iff

$$n=\sum_{i<m}2^{k_i}.$$

Then

$$F(x)(y)=\{m: \exists e_n\sqsubset y\ (n, m)\in x\}$$

and

$$G(f)=\{(n, m): m\in f(e_n)\}.$$

(ii) Scott's inverse limit construction D_∞ (see [22] or [23]) is an extensional domain. Somewhat easier, extensional domains can be "derived" from $P\omega$ [24, 25, 13] (but see [12] for a general construction).

From now on, let D be an arbitrarily fixed additive domain.

1.5. DEFINITION. (i) The set of *λ-terms over* D (notation: $\Lambda(D)$) is defined inductively by

$$x_0, x_1,\ldots\in\Lambda(D), \qquad \text{(variables)}$$
$$d\in D\Rightarrow c_d\in\Lambda(D), \qquad \text{(constants over } D)$$
$$M, N\in\Lambda(D)\Rightarrow (MN)\in\Lambda(D),$$
$$M\in\Lambda(D)\Rightarrow(\lambda x\cdot M)\in\Lambda(D).$$

(ii) The *theory* λ consists of equations between λ-terms, axiomatized by the axiom scheme

$$(\lambda x \cdot M)N = M_{[x:=N]}$$

(where $M_{[x:=N]}$ denotes substitution) and the usual equality axioms and rules, including

$$M = N \Rightarrow \lambda x \cdot M = \lambda x \cdot N.$$

See [2] for the syntactic care needed to define substitution and to insure λ-term-disambiguation.

1.6. DEFINITION. Let ρ: Variables $\to D$ be a valuation (in D). One defines, by induction on the structure of M, the *value of* M *at* ρ *in* D (notation: $[\![M]\!]^D_\rho$), as follows:

$$\begin{aligned}
[\![x]\!]^D_\rho &= \rho(x),\\
[\![c_d]\!]^D_\rho &= d,\\
[\![MN]\!]^D_\rho &= F([\![M]\!]^D_\rho)([\![N]\!]^D_\rho)\\
[\![\lambda x \cdot M]\!]^D_\rho &= G(\lambda d \cdot [\![M]\!]^D_{\rho(x:=d)}),
\end{aligned}$$

where

$$\rho(x := d)(y) = \begin{cases} \rho(y), & \text{if } y \neq x \\ d, & \text{if } y = x. \end{cases}$$

1.7. PROPOSITION. $[\![\ ]\!]^D_\rho$ *is well defined and determines a model of* λ:

$$\lambda \vdash M = N \Rightarrow [\![M]\!]^D_\rho = [\![N]\!]^D_\rho, \qquad \text{for all } \rho.$$

Proof. See [3]. ∎

1.8. *Notation.* (i) $D \models M = N$ iff $[\![M]\!]^D_\rho = [\![N]\!]^D_\rho$, for all ρ. This notion is extended in the obvious way to first-order formulas.

(ii) We loosely use, e.g., $\lambda x \cdot xd$ to denote $[\![\lambda x \cdot xc_d]\!]^D_\rho \in D$, or $\lambda x \cdot f(x)$ to denote $G(f)$, for continuous f: $D \to D$.

Note that additivity of D implies $D \models x \subset \lambda y \cdot xy$. Moreover, the finitary sup-operation is continuous.

1.9. DEFINITION. Let $a \in D$. Then, where $f \circ g := \lambda x \cdot f(gx)$, we say that

(i) a is a *retract* if $a = a \circ a$,

(ii) a is a *closure* if a is a retract and $\mathscr{J} := \lambda x \cdot x \subset a$.

(iii) $a^{\vee} = \{ax: x \in D\}$. Notation: $x \mathsf{E} a$ iff $x \in a^{\vee}$.

A retract a $(= \lambda x \cdot ax)$ is, in fact, a retraction map from D onto $a^{\vee}$.

The following construction, due to P. Martin-Löf, P. Hancock, and D. Scott, independently, shows that the set

$$\{a \in D: a \text{ is a closure}\}$$

is itself of the form $\mathscr{V}^{\vee}$, for some closure $\mathscr{V} \in D$.
As the map $f(x, y) = x \cup y$ is continuous, the following makes sense.

1.10. DEFINITION. $\mathscr{V} := \lambda xy \cdot \mathit{fix}\,(\lambda z \cdot y \cup xz)$.

1.11. LEMMA. *For all* $x, y \in D$,

(i) $\mathscr{V}x(\mathscr{V}xy) = \mathscr{V}xy$ *and*

(ii) $\mathscr{V}x$ *is a closure.*

Proof. (i) Note that

$$\mathscr{V}xy = y \cup x(\mathscr{V}xy). \tag{1}$$

Hence

$$y \subset \mathscr{V}xy, \tag{2}$$

$$x(\mathscr{V}xy) \subset \mathscr{V}xy. \tag{3}$$

But $\mathscr{V}x(\mathscr{V}xy)$ is the least z such that

$$z = \mathscr{V}xy \cup xz. \tag{4}$$

Now $z = \mathscr{V}xy$ satisfies (4) by (3); moreover $z = \mathscr{V}xy \cup xz \Rightarrow \mathscr{V}xy \subset z$. So we have (i).

(ii) By (i) and (2). ∎

1.12. THEOREM. (i) *For all* $x \in D$, x *is a closure iff* $x \mathsf{E} \mathscr{V}$.

(ii) $\mathscr{V}$ *is a closure* (*that is*: $\mathscr{V} \mathsf{E} \mathscr{V}$).

Proof. Use Lemma 1.11, noting that if x is a retract then $x = \lambda y \cdot xy$, $\mathscr{V} = \lambda x \cdot \mathscr{V}x$ (since $\mathscr{V}$ is an abstract) and D is additive. ∎

As usual, let $\mathscr{K} := \lambda xy \cdot x \in D$. Then, for $a \in D$, $\mathscr{K}a = \lambda x \cdot a$.

1.13. DEFINITION. (i) $\mathscr{S} := \lambda uvxy \cdot v(uy)(x(uy))$.

(ii) $\lambda x: a \cdot b := (\lambda x \cdot b) \circ a$ (i.e., $= \lambda z \cdot b_{[x:=az]}$).

(iii) $\pi x: a \cdot b := \mathscr{S}a(\lambda x \cdot b)$ (i.e., $= \lambda xy \cdot b_{[x:=ay]}(x(ay))$).

(iv) $a \circ\!\!\rightarrow b := \mathscr{S}a(\mathscr{K}b)$ (i.e., $= \lambda x \cdot b \circ x \circ a$).

1.14. PROPOSITION. *Let* $a \in \mathscr{V}$. *Then*

(i) $\forall x \in a\, b_{[x]} \in \mathscr{V} \Leftrightarrow (\pi x: a \cdot b_{[x]}) \in \mathscr{V}$.

(ii) $\forall x \in a\, b_{[x]} \in c_{[x]} \Leftrightarrow (\lambda x: a \cdot b_{[x]}) \in (\pi x: a \cdot c_{[x]})$.

(iii) $f \in \pi x: a \cdot b_{[x]} \Rightarrow f = \lambda x: a \cdot fx$.

(iv) $f \in \pi x: a \cdot b_{[x]} \Leftrightarrow (\forall x \in a\, fx \in b_{[x]}) \,\&\, (f = \lambda x: a \cdot fx)$.

Proof. (i) ($\Rightarrow$): Assume

$$\forall x \in a \cdot b_{[x]} \in \mathscr{V},$$

write for convenience,

$$A := \pi x: a \cdot b_{[x]}.$$

We show A is a closure. Indeed, writing $b_{[a]}$ for $b_{[x:=a]}$, etc., one has

$$\begin{aligned} A(Ax) &= \lambda y \cdot b_{[ay]}(b_{[a(ay)]}(x(ay))) \\ &= \lambda y \cdot b_{[ay]}(b_{[ay]}(x(ay))) && \text{since } a \in \mathscr{V}, \text{ hence a closure,} \\ &= \lambda y \cdot b_{[ay]}(x(ay)) && \text{since } b_{[ay]} \in \mathscr{V}, \\ &= Ax. \end{aligned}$$

Moreover,

$$\begin{aligned} x &\subset \lambda y \cdot xy && \text{since } D \text{ is additive,} \\ &\subset \lambda y \cdot x(ay) && \text{since } a \in \mathscr{V}, \\ &\subset \lambda y \cdot b_{[ay]}(x(ay)) && \text{since } b_{[ay]} \in \mathscr{V} \\ &\subset Ax. \end{aligned}$$

($\Leftarrow$): Assume

$$\pi x: a \cdot b_{[x]} \in \mathscr{V}.$$

Then, with A as above, one has $A(Ax) = Ax$ and $x \subset Ax$ ($\forall x \in D$). Hence

$$b_{[ay]}(b_{[ay]}(x(ay)) = b_{[ay]}(x(ay)).$$

So

$$b_{[ay]}(b_{[ay]}z) = b_{[ay]}z, \qquad (\text{take } x \equiv \lambda w \cdot z),$$

and therefore

$$\forall x \,\mathsf{E}\, a\; b_{[x]} \text{ is a retract.} \tag{1}$$

Moreover,

$$x \subset \lambda y \cdot b_{[ay]}(x(ay)).$$

So

$$xy \subset b_{[ay]}(x(ay)),$$

wherefrom, with $x \equiv \lambda w \cdot z$,

$$z \subset b_{[ay]}z$$

and therefore

$$\forall y \,\mathsf{E}\, a\; \mathscr{Y} \subset b_{[y]}. \tag{2}$$

By (1) and (2),

$$\forall y \,\mathsf{E}\, a\; b_{[y]} \text{ is a closure.}$$

Hence Theorem 1.12(i) applies.

(ii) ($\Rightarrow$): Assume

$$\forall x \,\mathsf{E}\, a\; b_{[x]} \,\mathsf{E}\, c_{[x]}.$$

Then

$$\begin{aligned}(\pi x{:}\, a \cdot c_{[x]})(\lambda x{:}\, a \cdot b_{[x]}) &= \mathscr{G}a(\lambda x \cdot c_{[x]})((\lambda x \cdot b_{[x]}) \circ a)\\ &= \lambda y \cdot c_{[ay]}(b_{[a(ay)]})\\ &= \lambda y \cdot c_{[ay]}(b_{[ay]}) \qquad \text{since } a \,\mathsf{E}\, \mathscr{Y},\\ &= (\lambda x \cdot b_{[x]}) \circ a \qquad \text{by assumption,}\\ &= \lambda x{:}\, a \cdot b_{[x]}.\end{aligned}$$

Therefore

$$(\lambda x{:}\, a \cdot b_{[x]}) \,\mathsf{E}\, (\pi x{:}\, a \cdot c_{[x]}).$$

($\Leftarrow$): Assume

$$(\lambda x: a \cdot b_{[x]}) \mathsf{E} (\pi x: a \cdot c_{[x]}).$$

Then

$$\begin{aligned}\lambda x \cdot b_{[ax]} &= (\pi x: a \cdot c_{[x]})(\lambda x: a \cdot b_{[x]}) \\ &= \lambda y \cdot c_{[ay]}(b_{[ay]}).\end{aligned}$$

Hence

$$b_{[ax]} = c_{[ax]}(b_{[ax]}),$$

i.e.,

$$\forall x \mathsf{E} a\, b_{[x]} \mathsf{E} c_{[x]}.$$

(iii) First note that

$$\begin{aligned}f = (\pi x: a \cdot b_{[x]}) f \Rightarrow fx &= \mathscr{G} a(\lambda x \cdot b_{[x]}) fx \\ &= b_{[ax]}(f(ax)).\end{aligned}$$

So

$$\begin{aligned}f \mathsf{E} (\pi x: a \cdot b_{[x]}) &\Rightarrow \forall x \mathsf{E} a\, fx = b_{[x]}(fx) \\ &\Rightarrow \forall x \mathsf{E} a\, fx \mathsf{E} b_{[x]}.\end{aligned}$$

Now

$$\begin{aligned}f \mathsf{E} (\pi x: a \cdot b_{[x]}) \Rightarrow f &= \mathscr{G} a(\lambda x \cdot b_{[x]}) f \\ &= \lambda x \cdot b_{[ax]}(f(ax)) \\ &= \lambda x \cdot f(ax), \qquad \text{by the above,} \\ &= \lambda x: a \cdot fx.\end{aligned}$$

(iv) Immediate, from (ii) and (iii). ∎

1.15. *Remark.* Let a be a closure. Define

$$\perp_a := a\perp \qquad \text{and} \qquad \mathscr{Y}_a := fix \circ (a \multimap a).$$

Then

(i) $a^{\vee}$ is an algebraic lattice.

(ii) $\perp_a$ is the least element of $a^{\vee}$;

(iii) $\mathscr{Y}_a \mathsf{E} (a \multimap a) \multimap a$.

(iv) For all $x \in D$, if $x \mathsf{E} (a \multimap a)$ then $\mathscr{Y}_a x$ is the least fixed point of x in $a^{\vee}$.

Proof. (i) See [7].

(ii) In D one has

$$\forall x \perp \subset x;$$

hence, by monotonicity,

$$\forall x\, a\perp \subset ax,$$

i.e.,

$$\forall x \in a^{\vee} \perp_a \subset x.$$

(iii) Easy computations show

$$\mathscr{Y}_a \circ (a \circ\!\!\rightarrow a) = \mathscr{Y}_a$$

and

$$a \circ \mathscr{Y}_a = \mathscr{Y}_a.$$

Therefore

$$a \circ \mathscr{Y}_a \circ (a \circ\!\!\rightarrow a) = \mathscr{Y}_a$$

and we are done.

(iv) Let $x \in (a \circ\!\!\rightarrow a)$. Then

$$\begin{aligned}\mathscr{Y}_a x = \mathit{fix}((a \circ\!\!\rightarrow a)\, x) &= \mathit{fix}\ x = x(\mathit{fix}\ x)\\ &= x(\mathit{fix}((a \circ\!\!\rightarrow a)\, x) = x(\mathscr{Y}_a x).\end{aligned}$$

Moreover, if $xy = y$ then

$$\mathscr{Y}_a x = \mathit{fix}\ x \subset y. \quad \blacksquare$$

2. Classical AUTOMATH: Syntax and Semantics

In first-order logic one first defines two recursive syntactic categories: terms and well-formed formulas (wff's); after that one can define the set of provable formulas as a subset of wff's.

In Classical AUTOMATH (CA) the situation is similar, but more complex (viz., roughly comparable with that encountered in Martin-Löf's type theories [16, 17]).

First, one defines (this is a "correctness-free" description stage) the following recursive *syntactic categories*:

—*terms*,

—*sentences* (*E*- resp. *Q*-*sentences*),

—*contexts*,

—*lines* (*primitive* and *defining lines*), while a *book* is a finite set of lines.

Then one defines (the "correctness" description stage) the r.e. set of *provable formulas* (or *statements*) *of* CA of the form

$$\Delta \vdash_B \varphi$$

(where B is a book, Δ is a context and φ is an E- or a Q-sentence of CA; accurately, $\vdash$, *the classical de Bruijn type-assignment*, is a ternary relation and "$\Delta \vdash_B \varphi$" is shorthand for "$\langle B, \Delta, \varphi\rangle \in \vdash$").

The intuition behind this is as follows. If a, b are terms then

$$a : b$$

is an *E-sentence*, with the intended meaning "a is of type b." So terms denote (as in [16, 17]) both types and objects. Moreover,

$$a = b$$

is a Q-sentence and can be read as "a is convertible to b" or as "a is definitionally equal to b." A *context* is a finite sequence (*not* just a finite set, not a Curry basis, say; cf. [9])

$$\Delta := \varphi_1, \ldots, \varphi_n$$

of E-sentences $\varphi_i := v_i : a_i$, where the $\vec{v} := v_1, \ldots, v_n$ are pairwise distinct variables and $\vec{a} := a_1, \ldots, a_n$ are terms. Each φ_i of this form is called an *assumption in* Δ and each assumption φ_i "declares" a variable v_i of type a_i.

The *terms* are formed from variables, a "universe constant" τ (denoting the type of all types), closed under application, typed abstraction, cartesian products and explicit function-definition (so, if for v of type a, the term $b_{[v]}$ is of type $b'_{[v]}$ then $\lambda v{:}\, a \cdot b_{[v]}$ is a function of type $\pi v{:}\, a \cdot b'_{[v]}$, while if c is an n-ary function constant and $\vec{a} := a_1, \ldots, a_n$ are terms of appropriate types then $c(\vec{a}) \equiv c(a_1, \ldots, a_n)$ is the value of a function having as type the appropriate "generalized" cartesian product).

The *lines* of CA (also called "constructions" in [21]) serve to specify the behaviour of function constants in CA. They are of the form

$$\{c = a : b\},$$

where c is either a *primitive* or a *defined constant*, a is the *definiens* of c and b is its *type*. In general, if c is an n-ary defined constant its definiens is of the form $a :\equiv \lambda\vec{v}{:}\vec{f}\cdot a'$, whereas, if c is primitive, one would want to specify its *pseudo-definiens* (just to signal that c is primitive) as $a :\equiv \lambda\vec{v}{:}\vec{f}\cdot c(\vec{v})$. Correspondingly, the type of an n-ary constant c is of the form $b :\equiv \pi\vec{v}{:}\vec{f}\cdot b'$.

Terms, E-sentences, Q-sentences and contexts that occur in some provable statement

$$\Delta \vdash_B \varphi \qquad (*)$$

of CA (but not in B) are called (CA-) *correct*, while a book B is (CA-) *correct* if $(*)$ is provable for some context Δ and some sentence φ.

In fact, CA-correct books (also called "compatible sites" in [21]) play exclusively the role of a book-keeping device and are used to "store" information concerning the behaviour of the function constants in CA (cf. with the "theories" in first-order logic).

The formal description of the CA-syntax is now as follows.

2.1. DEFINITION (CA "correctness-free" syntax). (i) The *alphabet* of CA consists of

1° a set $\mathrm{Var} = \{e_i : i \in \mathbb{N}\}$ of *variables*;

2° for each $n \in \mathbb{N}$, sets

$$\mathrm{Pcons}_n = \{p_i^n : i \in \mathbb{N}\} \text{ and } \mathrm{Dcons}_n = \{d_i^n : i \in \mathbb{N}\}$$

of *primitive* resp. *defined constants of arity n* (*p*- resp. *d-constants*, for short);

3° a "*universe symbol*": τ;

4° *abstractors*: λ, Π;

5° (binary) *predicates*: : (" $\cdots$ has type $\cdots$ "),

$=$ (" $\cdots$ equals $\cdots$ ");

6° *auxiliary symbols* ("punctuation"): . , ( ) { } [ ].

Syntactic Variables

$v, v', v'', \ldots$	range over Var,
$p, p', \ldots$	range over p-constants,
$d, d', \ldots$	range over d-constants,
$c, c', \ldots$	range over Cons
	$(\mathrm{Cons} = \bigcup_{n\in\mathbb{N}} (\mathrm{Pcons}_n \cup \mathrm{Dcons}_n)$.

(ii) The set Term of (CA-)*terms* is defined inductively by

1° $\mathrm{Var} \subseteq \mathrm{Term}$, $\tau \in \mathrm{Term}$.

2° If $c \in \mathrm{Pcons}_n \cup \mathrm{Dcons}_n$ $(n \in \mathbb{N})$ and $a_1,\ldots, a_n \in \mathrm{Term}$ then $c(a_1,\ldots, a_n) \in \mathrm{Term}$.

3° If $a, b \in \mathrm{Term}$ then (ab), $(\lambda v{:}\, a \cdot b)$, $(\Pi v{:}\, a \cdot b) \in \mathrm{Term}$.

Syntactic Variables

$a, b,\ldots, f, g, h\ldots$ (with sub- and/or superscripts) range over Term.

(iii) The *sentences* of CA, ranged over by $\varphi, \varphi',\ldots$, are

1° E-sentences, of the form $a : b$,

2° *Q-sentences*, of the form $a = b$.

(iv) A (CA-) *context* is a sequence $[v_1 : a_1,\ldots, v_n : a_n]$, where $v_1 : a_1,\ldots, v_n : a_n$ are E-sentences with the $v_1,\ldots, v_n$ pairwise distinct. In particular, $[\ ]$ denotes the *empty context.*

Syntactic Variables

$\Delta, \Delta',\ldots$, range over contexts.

Notation

$$\text{If } \Delta := [v_1 : a_1,\ldots, v_n : a_n] \text{ then we set}$$

$$\lambda\Delta \cdot b := \lambda v_1 : a_1 \cdots \lambda v_n : a_n \cdot b \text{ and}$$

$$\Pi\Delta \cdot v := \Pi v_1 : a_1 \cdots \Pi v_n : a_n \cdot b \text{ resp.}$$

(v) Let $\Delta := [v_1 : a_1,\ldots, v_n : a_n]$, $n \in \mathbb{N}$ and $g, h \in \mathrm{Term}$.

1° If $p \in \mathrm{Pcons}_n$ then $\{p = \lambda\Delta \cdot p(\vec{v}) : \Pi\Delta \cdot h\}$ is a *p-line* ("primitive line") in CA.

2° If $d \in \mathrm{Dcons}_n$ then $\{d = \lambda\Delta \cdot g : \Pi\Delta \cdot h\}$ is a *d-line* ("defining line") in CA.

3° A (CA-) *line* is either a p-line or a d-line in CA.

(vi) A (CA-) *book* is a finite set of CA-lines.

Syntactic Variables

$B, B',\ldots$, range over books.

2.2. *Notation. Conventions.* (i) Terms are identified modulo uniform reletterings of their bound variables (where the bound/free variables in a term are defined in the usual way; note, however, that one has here two distinct abstractors: λ and Π).

(ii) Where $\vec{v} := v_1,\ldots, v_n$ (with the v_i's pairwise distinct) and b, $\vec{a} := a_1,\ldots, a_n$ are terms, the notation

$$b_{[\vec{v}:=\vec{a}]}$$

stands for *simultaneous substitution* (for $n = 1$, this becomes usual substitution).

(iii) For $\vec{a}$ as above and c an n-ary function constant we write

$$c(\vec{a}) := c(a_1,\ldots, a_n),$$

while, if $\Delta := [v_1 : a_1,\ldots, v_n : a_n]$ and a is a term, then

$$\Delta[v : a] := [v_1 : a_1,\ldots, v_n : a_n, v : a].$$

(iv) Finally, in the next definition,

1° $\Delta \vdash_B a$ is shorthand for $\Delta \vdash_B a = a$,

2° $\Delta \vdash_B a, b$ is shorthand for $(\Delta \vdash_B a)$ & $(\Delta \vdash_B b)$, and

3° $\Delta \vdash_B f : g : h$ stands for $(\Delta \vdash_B f : g)$ & $(\Delta \vdash_B g : h)$.

2.3. DEFINITION. (i) *A statement of* CA is of the form

$$\Delta \vdash_B \varphi$$

where B is a book, Δ is a context and φ is an (E- or Q-) sentence.

(ii) The *provable statements of* CA are inductively defined by the following set of (correctness) rules.

Correctness Rules of CA

1° *Structural rules.* Let $\Delta := [v_1 : f_1,\ldots, v_n : f_n]$.

1°1 Initialization:

$$[\ ] \vdash_\phi \tau. \qquad (r_\tau)$$

1°2 Book-recursion ($n \geqslant 0$):

$$\Delta \vdash_B \tau \Rightarrow [\ ] \vdash_{B'} \tau, \qquad (p1)$$

where

$$p \in \mathrm{Pcons}_n, \text{ fresh for } \vec{f}, B,$$

$$B' = B \cup \{\!\{p = \lambda\Delta \cdot p(\vec{v}) : \Pi\Delta \cdot \tau\}\!\}.$$

$$\Delta \vdash_B g : \tau \Rightarrow [\ ] \vdash_{B'} \tau \qquad (p2)$$

where

$$p \in \mathrm{Pcons}_n, \text{ fresh for } \vec{f}, g, B,$$
$$B' = B \cup \{\langle p = \lambda\Delta \cdot p(\vec{v}) : \Pi\Delta \cdot g\rangle\}.$$

$$\Delta \vdash_B g : h \Rightarrow [\ ] \vdash_{B'} \tau \tag{d}$$

where

$$d \in \mathrm{Dcons}_n, \text{ fresh for } \vec{f}, g, h, B,$$
$$B' = B \cup \{\langle d = \lambda\Delta \cdot g : \Pi\Delta \cdot h\rangle\}.$$

1°3 Context-recursion ($n \geqslant 0$):

$$\Delta \vdash_B \tau \Rightarrow \Delta[v : \tau] \vdash_B \tau \tag{c1}$$

provided v is fresh for Δ.

$$\Delta \vdash_B g : \tau \Rightarrow \Delta[v : g] \vdash_B \tau \tag{c2}$$

provided v is fresh for Δ, g.

1°4 Projection rules:

1°41 Book-projection ($n \geqslant 0$):

$$\left.\begin{array}{l} \Delta \vdash_B \tau; \Delta' \vdash_B \tau \\ \Delta' \vdash_B a_i : f_{i[\vec{v}:=\vec{a}]}\ (1 \leqslant i \leqslant n) \\ \langle d = \lambda\Delta \cdot g : \Pi\Delta \cdot h\rangle \in B. \end{array}\right\} \Rightarrow \Delta' \vdash_B d(\vec{a}) = g_{[\vec{v}:=\vec{a}]} \tag{bp}$$

1°42 Context-projection ($n \geqslant 1$):

$$\Delta \vdash_B \tau \Rightarrow \Delta \vdash_B v_i : f_i \qquad (1 \leqslant i \leqslant n). \tag{cp}$$

1°43 E-sentence-projection:

$$\Delta \vdash_B a : b \Rightarrow \Delta \vdash_B a \tag{r_s}$$
$$\Delta \vdash_B a : b \Rightarrow \Delta \vdash_B b. \tag{r_p}$$

1°5 Substitution ($n \geqslant 0$):

$$\left.\begin{array}{l} \Delta \vdash_B \tau; \Delta' \vdash_B \tau \\ \Delta' \vdash_B a_i : f_{i[\vec{v}:=\vec{a}]}\ (1 \leqslant i \leqslant n) \\ \langle c = \lambda\Delta \cdot g : \Pi\Delta \cdot h\rangle \in B. \end{array}\right\} \Rightarrow \Delta' \vdash_B c(\vec{a}) : h_{[\vec{v}:=\vec{a}]} \tag{sub}$$

2° *Assignment rules*:

$$\left.\begin{array}{l}\Delta \vdash_B g : \tau \\ \Delta[v : g] \vdash_B h_{[v]} : \tau\end{array}\right\} \Rightarrow \Delta \vdash_B \Pi v : g \cdot h_{[v]} : \tau. \qquad (\Pi i)$$

$$\left.\begin{array}{l}\Delta \vdash_B a : g : \tau \\ \Delta \vdash_B \Pi v : g \cdot h_{[v]} : \tau\end{array}\right\} \Rightarrow \Delta \vdash_B h_{[v:=a]} : \tau. \qquad (\Pi e)$$

$$\left.\begin{array}{l}\Delta \vdash_B g : \tau \\ \Delta[v : g] \vdash_B f_{[v]} : h_{[v]}\end{array}\right\} \Rightarrow \Delta \vdash_B \lambda v : g \cdot f_{[v]} : \Pi v : g \cdot h_{[v]}. \qquad (\text{abs})$$

$$\left.\begin{array}{l}\Delta \vdash_B a : g : \tau \\ \Delta \vdash_B f : \Pi v : g \cdot h_{[v]}\end{array}\right\} \Rightarrow \Delta \vdash_B fa : h_{[v:=a]}. \qquad (\text{app})$$

3° *Conversion rules*:

3°1 Equivalence:

$$\Delta \vdash_B a = b \Rightarrow \Delta \vdash_B b = a \qquad (\text{s})$$

$$\Delta \vdash_B f = g; \Delta \vdash_B g = h \Rightarrow \Delta \vdash_B f = h. \qquad (\text{t})$$

3°2 Congruence:

$$\Delta \vdash_B a = b; \Delta \vdash_B \Phi_{[a]}, \Phi_{[b]} \Rightarrow \Delta \vdash_B \Phi_{[a]} = \Phi_{[b]}. \qquad (\text{mon})$$

where $\Phi_{[v]}$ is of one of the forms

$$vg, fv, \lambda v' : v \cdot h, \Pi v' : v \cdot h, c(a_1, \ldots, v, \ldots, a_n)$$

with v occurring in $\Phi_{[v]}$ only at the indicated place (just once) and $\Phi_{[a]} := \Phi_{[v:=a]}$, etc.

$$\left.\begin{array}{l}\Delta \vdash_B g : \tau \\ \Delta[v : g] \vdash_B a = b\end{array}\right\} \Rightarrow \Delta \vdash_B \lambda v : g \cdot a = \lambda v : g \cdot b. \qquad (\xi_\lambda)$$

$$\left.\begin{array}{l}\Delta \vdash_B g : \tau \\ \Delta[v : g] \vdash_B a = b\end{array}\right\} \Rightarrow \Delta \vdash_B \Pi v : g \cdot a = \Pi v : g \cdot b. \qquad (\xi_\Pi)$$

$$\Delta \vdash_B a_1 : b; \Delta \vdash_B a_1 = a_2 \Rightarrow \Delta \vdash_B a_2 : b. \qquad (\text{eqs})$$

$$\Delta \vdash_B a : b_1; \Delta \vdash_B b_1 = b_2 \Rightarrow \Delta \vdash_B a : b_2. \qquad (\text{eqp})$$

3°3 Evaluation:

$$\left.\begin{array}{l}\Delta \vdash_B a : g : \tau \\ \Delta[v : g] \vdash_B f_{[v]} : h_{[v]}\end{array}\right\} \Rightarrow \Delta \vdash_B (\lambda v : g \cdot f_{[v]})\, a = f_{[v:=a]} \qquad (\beta)$$

3°4 Functionality:

$$\Delta \vdash_B f: (\Pi v : g \cdot h_{[v]}): \tau \Rightarrow \Delta \vdash_B \lambda v : g \cdot fv = f \qquad (\eta)$$

provided v is not free in f.
Now one can give the semantics of CA (in any additive domain D).

2.4. DEFINITION. (i) A *v-valuation in D* is a map $\rho: \mathrm{Var} \to D$.

(ii) A *c-valuation in D* is a map $\xi: \mathrm{Cons} \to D$.

(iii) The *value* (= *interpretation*) if a (CA-) term f *relative to* ρ *and* ξ *in* D (notation: $[\![f]\!]^D_{\rho,\xi}$) is defined by induction on the structure of f as follows:

1° $[\![v]\!]^D_{\rho,\xi} = \rho(v)$,
2° $[\![c]\!]^D_{\rho,\xi} = \xi(c)$,
3° $[\![\tau]\!]^D_{\rho,\xi} = \mathscr{T}$,
4° $[\![ab]\!]^D_{\rho,\xi} = F([\![a]\!]^D_{\rho,\xi})([\![b]\!]^D_{\rho,\xi})$,
5° $[\![\lambda v : a \cdot b]\!]^D_{\rho,\xi} = (G(\lambda d \cdot [\![b]\!]^D_{\rho(v:=d),\xi})) \circ ([\![a]\!]^D_{\rho,\xi})$,
6° $[\![\Pi v : a \cdot b]\!]^D_{\rho,\xi} = \mathscr{G}(G(\lambda d \cdot [\![b]\!]^D_{\rho(v:=d),\xi}))([\![a]\!]^D_{\rho,\xi})$.

(iv) An E-sentence $\varphi := a : b$ is *true at* ρ, ξ *in* D (notation: $D, \rho, \xi \models a : b$) if

$$[\![a]\!]^D_{\rho,\xi} \mathsf{E} [\![b]\!]^D_{\rho,\xi}.$$

Similarly, a Q-sentence $\varphi' := a = b$ is *true at* ρ, ξ *in* D (notation: $D, \rho, \xi \models a = b$) if

$$[\![a]\!]^D_{\rho,\xi} = [\![b]\!]^D_{\rho,\xi}.$$

(v) For all contexts $\Delta := [v_1 : f_1, \ldots, v_n : f_n]$ in CA, all E-sentences $a : b$ and all Q-sentences $a = b$, one defines

$$D, \rho, \xi \models^\Delta a : b \Leftrightarrow [\![\lambda \Delta \cdot a]\!]^D_{\rho,\xi} \mathsf{E} [\![\Pi \Delta \cdot b]\!]^D_{\rho,\xi}$$

resp.

$$D, \rho, \xi \models^\Delta a = b \Leftrightarrow [\![\lambda \Delta \cdot a]\!]^D_{\rho,\xi} = [\![\lambda \Delta \cdot b]\!]^D_{\rho,\xi}.$$

(vi) Let B be a CA-book such that, for $0 \leqslant i \leqslant n$, $0 \leqslant j \leqslant m$,

$$\{p_i = \lambda \Delta \cdot p_i(\vec{v}): \Pi \Delta \cdot f_i\} \text{ are the } p\text{-lines of } B$$

and

$$\{d_j = \lambda \Delta \cdot g_j: \Pi \Delta \cdot h_j\} \text{ are the } d\text{-lines of } B.$$

Then B is *satisfied at* ρ, ξ *in* D (notation: $D, \rho, \xi \vDash B$) if

1° $\forall i: 0 \leqslant i \leqslant n \Rightarrow (D, \rho, \xi \vDash p_i : \Pi\Delta \cdot f_i)$

2° $\forall j: 0 \leqslant j \leqslant m \Rightarrow (D, \rho, \xi \vDash d_j = \lambda\Delta \cdot g_j)$ &
$(D, \rho, \xi \vDash d_j : \Pi\Delta \cdot h_j)$.

(vii) Finally, one defines *validity for* CA-*statements* by

$$D \vDash_B^\Delta a : b \Leftrightarrow \forall \rho, \xi (D, \rho, \xi \vDash B \Rightarrow D, \rho, \xi \vDash^\Delta a : b)$$

resp.

$$D \vDash_B^\Delta a = b \Leftrightarrow \forall \rho, \xi (D, \rho, \xi \vDash B \Rightarrow D, \rho, \xi \vDash^\Delta a = b).$$

The main result can be now stated as follows.

2.5. THEOREM (Soundness for CA). *Let D be an arbitrary additive domain. Then for all* CA-*books B, all* CA-*contexts Δ and all* CA-*terms a, b,*

(i) $\Delta \vdash_B a : b \Rightarrow D \vDash_B^\Delta a : b$,

(ii) $\Delta \vdash_B a = b \Rightarrow D \vDash_B^\Delta a = b$.

Proof. Induction on the generation of $\Delta \vdash_B a : b$ and $\Delta \vdash_B a = b$, in CA, using Proposition 1.14. ∎

3. DISCUSSION: RELATED SYSTEMS

As described in Definitions 2.1 and 2.3, the syntax of CA diverges unessentially from standard presentations of AUT-68 (= the "reference" version of CA; cf. [6, 21], etc.).

In fact, the main differences from [21], say, are notational in nature and are justified by model-theoretic as well as readability considerations: abstraction terms, denoted by $[v : a]\, b$ in AUT-68, are *disambiguated* here according to their intended meaning, application terms (fa) have function-part on the l.h.s. (as it is usual in lambda-calculus), while the official AUT-68 lines have here somewhat a more "portable" format, fitting straightforwardly the interpretation in Definition 2.4. Finally, in the "reference" AUT-68 version, a book is a *sequence* of lines; so the present concept would rather correspond to the "sites" of [21].

As regards CA-correctness (Definition 2.3), the main novelty over [21].2 consists of the elimination of the statements

$$\Delta \vdash_B a$$

from the primitive CA-syntax. So the reflexivity rule

$$\Delta \vdash_B a \Rightarrow \Delta \vdash_B a = a \qquad \text{(r)}$$

follows trivially here, by mere notational conventions. In particular, the hypotheses of (η) are, obviously, necessary in the present setting. Finally, (eqs) is, apparently, derivable from the remaining correctness rules of CA (as in [6, 21]), but (r_p) seems necessary, due to the fact that "$\Delta \vdash_B a$" is a defined notion.

Let CA_0 be the "pure" part of CA; i.e., CA without function constants (so CA_0 has no lines and books). Then a possible set of primitive correctness rules for CA_0 is (cf. Definition 2.3 above)

1° (r_τ), (c1), (c2), (cp), (r_s), (r_p), (sub_0),

2° (Πi), (Πe), (abs), (app),

3° (s), (t), (mon), (ξ_λ), (ξ_Π), (eqs), (eqp), (β), (η),

where $B = \varnothing$ (or just omitted) and (sub_0) is the following analogue of (sub): for $\Delta := [v_1 : f_1, \ldots, v_n : a_n]$, $n \geqslant 1$,

$$\left.\begin{array}{l} \Delta \vdash g : h \\ \Delta' \vdash \tau \\ \Delta' \vdash a_i : f_{i[v:=a]} \ (1 \leqslant i \leqslant n) \end{array}\right\} \Rightarrow \Delta' \vdash g_{[v:=a]} : h_{[v:=a]}.$$

This system is very useful, for most of the known typed lamba-calculi can be obtained from it, by trivial modifications in the set of its correctness rules.

First note that (sub_0) is derivable in CA (cf. [6, 21]), so CA_0 is actually a subsystem of CA. For a mild combinatory variant of CA_0, see, e.g., [26].

Consider now the following axioms:

$$[\ ] \vdash \tau : \tau \qquad (\tau)$$

$$[\ ] \vdash \tau : (\tau \to \tau) \qquad (\tau\tau)$$

where $a \to b := \Pi v : a \cdot b$, provided v is not free in b. Both (τ) and $(\tau\tau)$ are valid in additive domains: for (τ) this follows from Theorem 1.12(ii), whereas, for $(\tau\tau)$, one checks easily that

$$\mathscr{V} = (\mathscr{V} \multimap \mathscr{V})\mathscr{V}$$

in any additive domain D. This shows that CA_0 is "classically" consistent with any one of (τ), $(\tau\tau)$; that is, one cannot derive in the resulting extensions

$$[\ ] \vdash a : b$$

for all (closed) terms a, b (and similarly for Q-sentences $a = b$). For further reference, let $CA_\tau := CA_0 + (\tau)$ and $CA_{\tau\tau} := CA_0 + (\tau\tau)$. CA_τ (which is, in fact, Martin-Löf's system in [14]) is known to be "intuitionistically" inconsistent, in the sense that it allows proving

$$\forall g \in \text{Term}\ \exists f \in \text{Term}\ [\ ] \vdash g : \tau \Rightarrow [\ ] \vdash f : g \qquad \text{(GP)}$$

(in other words: all closed CA_τ-correct types are inhabited). This is the so-called *Girard Paradox* (cf. [8, 15]) and shows, essentially, that CA_τ is inconsistent with the "formulae-as-types"-interpretation [10, 16], preferred in intuitionistic type theory and some version of AUTOMATH (e.g., in AUT-QE; see [11]).

Starting from CA_0 one can obtain easily "AUT-like"-formalizations of the First- and Second-Order Typed Lambda-Calculi $\lambda_1^\tau, \lambda_2^\tau$ (cf. [9] for λ_1^τ and [8, 20] for λ_2^τ).

Note first that the following weaker assignment rules (labelled collectively (ass-1), say) are derivable in CA_0:

$$\Delta \vdash g : \tau; \Delta \vdash h : \tau \Rightarrow \Delta \vdash g \to h : \tau \qquad (\Pi\text{i-1})$$

$$\Delta\ V g : \tau; \Delta \vdash g \to h : \tau \Rightarrow \Delta \vdash h : \tau \qquad (\Pi\text{e-1})$$

$$\Delta \vdash g : \tau; \Delta[v : g] \vdash f : h : \tau \Rightarrow \Delta \vdash \lambda v : g \cdot f : g \to h \qquad \text{(abs-1)}$$

$$\Delta \vdash a : g : \tau; \Delta \vdash f : g \to h \Rightarrow \Delta \vdash fa : h. \qquad \text{(app)-1)}$$

Similarly, the rules (ass-1), together with the following rules (labelled, for convenience, (ass-2), say), are easily seen to be derivable in CA_τ:

$$\Delta[v : \tau] \vdash h : \tau \Rightarrow \Delta \vdash \Pi v : \tau \cdot h : \tau \qquad (\Pi\text{i-2})$$

$$\Delta \vdash a : \tau; \Delta \vdash \Pi v : \tau \cdot h : \tau \Rightarrow \Delta \vdash h_{[v:=a]} : \tau \qquad (\Pi\text{e-2})$$

$$\Delta[v : \tau] \vdash f : h : t \Rightarrow \Delta \vdash \lambda v : \tau \cdot f : \Pi v : \tau \cdot h \qquad \text{(abs-2)}$$

$$\Delta \vdash a : \tau; \Delta \vdash f : \Pi v : \tau \cdot h \Rightarrow \Delta \vdash fa : h_{[v:=a]}. \qquad \text{(app-2)}$$

Let LT_1, LT_2, resp., be the systems obtained from CA_0 by replacing its assignment rules (Πi), (Πe), (abs), (app) by (ass-1) and {(ass-1), (ass-2)}, resp. So LT_1 is a subsystem of CA_0, while LT_2 is a subsystem of Martin-Löf's CA_τ. One may guess LT_1, LT_2 are conservative extensions (under the obvious translation) over λ_1^τ, λ_2^τ, resp. (though a formal proof of this may be somewhat involved). For present purposes it is enough to notice that the First-Order Typed Lambda-Calculus λ_1^τ can be interpreted in CA_0 (and hence in CA_τ), whereas the Second-Order analogue (= the so-called "parametric" typed lambda-calculus) can be interpreted trivially in CA_τ. That is, both calculi LT_1 and LT_2 (and therefore λ_1^τ and λ_2^τ, resp.) admit of a

completely similar "fixed-point closure semantics" according to Definition 2.4 above.

Remark 1.15(ii)–(iv) shows that this is also the case for the "AUT-like"-formalization of Pure LCF (cf. [18, 19]), i.e., LT_1 extended by fixed-point recursion (and even Scott-induction). Note that the resulting "closure semantics" is different from Milner's [19].

Completeness fails, in additive domains, for all systems named above: indeed (τ) is not derivable in CA_0, $CA_{\tau\tau}$, LT_1, LT_2, etc., while $(\tau\tau)$ is not derivable in CA_τ. It is also worthwhile noting that *unicity of types*

$$\Delta \vdash a : b_1; \Delta \vdash a : b_2 \Rightarrow \Delta \vdash b_1 = b_2 \qquad \text{(UT)}$$

(otherwise derivable in CA_0, CA, etc.) fails for the above semantics (one checks that $\mathscr{V} \neq (\mathscr{V} \multimap \mathscr{V})$, in $P\omega$, say).

Finally, the present semantics does *not* work for the AUT-QE system of [11] (this is just QA in [21]; reason: the presence of the rule of Type Inclusion and (app-2) in [21, p. 103]), nor for AUT-SL and LAMBDA-AUTOMATH (cf. [6] for a survey of the syntax), but can be easily extended to Zucker's AUT-Pi [27; 6, Chap. VIII], working directly in $P\omega$, say.

RECEIVED: December 24, 1983; ACCEPTED: February 29, 1984

REFERENCES

1. ACZEL, P. (1977), The strength of Martin-Löf's intuitionistic type theory with one universe, *in* "Proceedings, Symp. Math. Logic, Oulu, 1974" (S. Miettinen and S. Väänänen, Eds.); Report No. 2, Department of Philosophy, University of Helsinki, pp. 1–32.
2. BARENDREGT, H. P. (1981), "The Lambda Calculus, Its Syntax and Semantics," North-Holland, Amsterdam.
3. BARENDREGT, H. P., Lambda calculus and its models, *in* "Logic Colloquium '82" (G. Lolli, G. Longo, and A. Marcja, Eds.), North-Holland, Amsterdam, in press.
4. BEESON, M., "Foundations of Constructive Mathematics: Metamathematical Studies," Springer-Verlag, Berlin/Heidelberg/New York, in press. (See also: Recursive models for constructive set theories, *Ann. of Math. Logic* **23** (1982), 127–178.)
5. DE BRUIJN, N. G. (1980), A survey of the project AUTOMATH, *in* "To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism" (J. P. Seldin and R. Hindley, Eds.), pp. 579–606, Academic Press, New York.
6. VAN DAALEN, D. T. (1980), "The Language Theory of AUTOMATH," Ph. D. dissertation, Eindhoven University of Technology, 1980.
7. GIERZ, G., *et al.* (1980), "A Compendium of Continuous Lattices," Springer-Verlag, Berlin/Heidelberg/New York.
8. GIRARD, J.-Y. (1972), "Interpretation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur," Thèse de doctorat d'état, Université de Paris VII.
9. HINDLEY, R., *et al.* (1972), "Introduction to Combinatory Logic," Cambridge Univ. Press, London.

10. HOWARD, W. A. (1980), The formulae-as-types notion of construction, *in* "To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism" (J. P. Seldin and R. Hindley, Eds.), pp. 479–490, Academic Press, New York (written in 1969).
11. JUTTING, VAN BENTHEM L.S. (1979), "Checking Landau's "Grundlagen" in the AUTOMATH System," Mathematical Centre Tracts No. 83, Mathematical Centre, Amsterdam.
12. KOYMANS, C. P. J. (1983), Derived λ-calculus models and the construction of D_∞ inside P_ω, Preprint No. 283, Department of Mathematics, University of Utrecht. (See also: "Models of the Lambda-Calculus," Mathematical Centre Tracts, Mathematical Centre, Amsterdam, 1984.)
13. LAARHOVEN, M. A. M. (1975), "Typed and untyped extensional λ-calculus models in P_ω," Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology (mimeographed). [In Dutch]
14. MARTIN-LÖF, P. (1971), "A Theory of Types," Report 71-3, Department of Mathematics, University of Stockholm.
15. MARTIN–LÖF, P. (1972), "An Intuitionistic Theory of Types," Report, Department of Mathematics, University of Stockholm.
16. MARTIN-LÖF, P. (1973), An intuitionistic theory of types: Predicative part, *in* "Logic Colloquium '73" (H. Rose and J. Sheperdson, Eds.), pp. 73–118, North-Holland, Amsterdam.
17. MARTIN-LÖF, P. (1982), Constructive mathematics and computer programming, *in* "Logic, Methodology and Philosophy of Science VI" (L. J. Cohen, J. Łoś, H. Pfeiffer, and K.-P. Podewski, Eds.), pp. 153–175, North-Holland, Amsterdam; Polish Scientific Publishers, Warsaw.
18. MILNER, R. (1972), "Logic for Computable Functions; Description of a Machine Implementation," AI Memo 169, Stanford University.
19. MILNER, R. (1976), Models for LCF, *in* "Foundations of Computer Science II, Part 2" (K. R. Apt and J. W. de Bakker, Eds.), Mathematical Centre Tracts No. 82, Mathematical Centre, Amsterdam.
20. REYNOLDS, J. (1974), Towards a theory of type structure, *in* "Proceedings, Programming Symposium, Paris, April 9–11, 1974" (B. Robinet, Ed.), pp. 408–425, Springer-Verlag, Berlin/Heidelberg/New York.
21. REZUS, A. (1983), "Abstract AUTOMATH," Mathematical Centre Tracts No. 160, Mathematical Centre, Amsterdam.
22. SCOTT, D. S. (1972), Continuous lattices, *in* "Toposes, Algebraic Geometry and Logic" (F. W. Lawvere, Ed.), pp. 97–136, Lecture Notes in Mathematics No. 274, Springer-Verlag, Berlin/Heidelberg/New York.
23. SCOTT, D. S. (1973), Models for various type-free calculi, *in* "Logic, Methodology and Philosophy of Science IV" (P. Suppes, A. Joja, and Gr. C. Moisil, Eds.), pp. 157–187, North-Holland, Amsterdam.
24. SCOTT, D. S. (1976), Data types as lattices, *SIAM J. Comput.* **5**, 522–587.
25. SCOTT, D. S. (1980), Lambda-calculus: Some models, some philosophy, *in* "Kleene Symposium" (J. Barwise, H. J. Keisler, and K. Kunen, Eds.), pp. 223–265, North-Holland, Amsterdam.
26. SELDIN, J. P. (1979), Progress report on generalized functionality, *Ann. Math. Logic* **17**, 29–59.
27. ZUCKER, J. (1975), Formalization of classical mathematics in AUTOMATH, *in* "Colloque International de Logique, Clermont Ferrand, July 18–25, 1975" (M. Guillaume, Ed.), pp. 135–145, Editions du CNRS, Paris.