Some Kripke Models for "one universe" Martin-Löf Type Theory

James Lipton*

TR 90-1162
October 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*after September: Department of Mathematics, University of Pennsylvania.

Some Kripke Models for "one-universe" Martin-Löf Type Theory

James Lipton
Department of Computer Science*
Cornell University
Ithaca, NY 14853

Abstract

We define several Kripke models sound for inhabited formulas of the ground-level intensional and extensional Martin-Löf Type theories with one universe. They are Kripke model versions of the realizability style semantics developed by various authors, amongst them Allen, Beeson and Aczel.

1 Introduction

There is a striking similarity between the syntax of Martin-Löf Type theory and certain Realizability interpretations for intuitionistic logic, which has been exploited by Aczel, Allen, Beeson and others to develop a type-free semantics for the Nuprl and Martin-Löf theories. Using recent work on capturing realizability semantics in Kripke models we are able to give several Kripke models for these theories, formalized in HA and in other abstract applicative systems. Before proceeding to these results, let us briefly discuss the role and utility of Kripke Models.

Kripke models were developed in 1963 by Saul Kripke, although similar interpretations were at least implicit in earlier, topological models of Tarski from the 1940's and in Beth's work in the 1950's. They were subsequently generalized by a number of researchers who strengthened some of the algebraic and topological features of the semantics. These models have proven to be a powerful tool for studying the metamathematics of intuitionistic formal systems. Much of this power lies precisely in the fact that they bring to bear on the study of formal systems an exceptionally powerful and versatile arsenal of algebraic, topological and categorical tools, yielding new consistency, conservativity and independence results. Perhaps the greatest strength of the semantics as developed in Saul Kripke's original paper, is that it supplies the means for effective systematic construction of counterexamples

*after September: Department of Mathematics, University of Pennsylvania forced by specific requirements, in the spirit of Cohen's independence results for set theory. Our effort here is to develop this kind of tool for type-theory, along the lines already begun by Mitchell and Moggi for simply typed lambda Calculus in [20], and by the author for realizability-style interpretations [17]. The models developed in this paper present a number of characteristics of interest: they require the use of covers, or *delayed satisfaction of disjunctive and existential formulas*, as well as the inhabitation of void in the semantics: *we allow inconsistent, or "fallible" nodes*. This has proven to be a critical component in all known intuitionistically valid completeness theorems (see [16,17]). As with the realizability-Kripke models in the reference just cited and in the Effective Topos (see Hyland's [13]), the truth-value structure imposed by type theory is that of the *degrees of inhabitation of definable sets* (or just the almost-negative ones[1]). That is to say, the Heyting Algebra formed by taking equivalence classes for all definable almost-negative sets (in HA or some applicative theory like APP) of their existential closures under provable equivalence, constitutes a natural domain of truth values of type theory.

2 An APP-definable model for $\mathbf{ML}_0^i$

For our purposes it will be convenient to adopt the notation and syntax for "intensional, ground-level Martin-Löf Type Theory" as presented in Troelstra and van Dalen's recent [21], as well as their formulation of the Realizability semantics for this theory. [2] We start with a brief recapitulation of the syntax and realizability semantics of $\mathbf{ML}_0^i$.

2.1 The Syntax of $\mathbf{ML}_0^i$

Language: The *terms* are built up from variables $x_1, x_2, \ldots; y_1, y_2, \ldots$; and constants (e.g. the integer 0) using *lambda-abstraction*, a *recursor* $R_{x,y}(n, t_0, t_1)$, *application* of terms $Ap(t_1, t_2)$ or simply $(t_1 t_2)$, *case-analysis* $\mathbf{D}_{x,y}(t, t_0, t_1)$, *pairing* $\mathbf{p}(x, y)$ (also written $\langle x, y\rangle$), unpairing $\mathbf{p}_0, \mathbf{p}_1$, and left and right *inclusions* (into a sum) $\mathbf{k}_i \quad (i = 0, 1)$.
Terms satisfy the following reduction rules:

$$R_{x,y}(0, t_0, t_1) \rightsquigarrow t_0 \tag{1}$$
$$R_{x,y}(St, t_0, t_1) \rightsquigarrow t_1[x, u/t, T_{x,y}(t, t_0, t_1)] \tag{2}$$
$$Ap(\lambda x \cdot t, t') \rightsquigarrow t[x/t'] \tag{3}$$
$$\mathbf{p}_i(t_0, t_1) \rightsquigarrow t_i \qquad (i = 0, 1) \tag{4}$$
$$(\mathbf{p}_0 t, \mathbf{p}_1 t) \rightsquigarrow t \tag{5}$$

[1] Realizability formulas are almost negative, as is shown in [21]. Therefore it suffices to consider just these sets in the model defined in the next section.

[2] A much stronger version of these semantics, taking computation rules into account, and extending to the whole type hierarchy and to higher-level universes was first developed by Stuart Allen [2]. This is beyond the scope of this report, and will be taken up in a sequel.

$$D_{x,y}(\mathbf{k}_i t, t_0, t_1) \quad \rightsquigarrow \quad t_i[x_i/t] \qquad (i : \{0,1\}). \tag{6}$$

Types are built up from the natural number type **nat**, and possibly other atomic, so-called "small" types, via the *join* $A + B$, *dependent sum* $\sum x : A \cdot B(x)$, *dependent product* $\prod x : A \cdot B(x)$ and, if A is a type and s and t are terms, *identity* formation $I(A, s, t)$.

The intended meaning of these notions is captured in the (copious) rules of inference, type formation, equality and computation, which are included in the appendix.

On Semantics for Intuitionistic Type Theory We assume the reader is familiar with the motivation and ideas behind the Martin-Löf style formulation of intuitionistic type theory, as found in e.g. Martin Löf's 1984 notes [19], Beeson's [4], Troesltra and van Dalen's [21] (upon which our formulation is based), or as fleshed out and extended in the Nuprl proof development system [6]. We cannot undertake a discussion of the subject here. Suffice it to say that such theories are based in a very strong way on the so-called Curry-Howard isomorphism (see e.g., [9,12]). Propositions are treated as types, and are inseparable from their proofs, which appear in the syntax as *inhabiting* or witnessing terms. Even syntactic well-formedness of formulas (types) is built into the rules of inference. This makes such theories difficult to model using "conventional" type-free semantics. In addition to the challenge of providing a sufficiently computational interpretation, one must find a way of accounting for proofs, formulas and syntactic well-formedness, all at once. For all of these reasons, interpretations in the spirit of Kleene's Realizability ([14,21,4]) are good candidates. Realizability effectively maps propositions to sets of witnesses or inhabiting terms in a way that mirrors the type-theoretic interpretation of implication and quantification. It provides a way of talking about evidence for propositions in arithmetic or abstract applicative theories without requiring us to formalize a provability predicate. It is not surprising, then, that the Kripke semantics developed here is a "forcing analogue of realizability." Both propositions (the *nodes* of our interpretation) and terms play a vital role in the construction of the model.

Since the notions used here are somewhat non-standard, before proceeding to the construction of the model, we give the briefest of summaries of Realizability and Kripke semantics, and of the **APP**-realizability interpretation of $\mathbf{ML}_0^i$. We assume the reader is familiar with some type-free formulation of partial applicative structure, such as Fefferman's or Troelstra and van Dalen's **APP**, or Beeson's **EON**. (see [21] or [4] or [16,17] for different developments). For the most part it will be sufficient to think of **APP** as a theory of partial application, with a formalized *convergence* predicate $\downarrow$, Kleene-style conditional equivalence of terms $t \simeq s$, and partial $\mathbf{s}$ and $\mathbf{k}$ combinators, pairing and unpairing operators $\mathbf{p}$, $\mathbf{p}_0$, $\mathbf{p}_1$, an **if_then_else** operator, and a recursor $\mathbf{r}(x, y, z)$. We require a *natural number predicate* (i.e. a *sort*) $\mathbf{N}$ which is true of the natural numbers. See the abovementioned references for details. The only novel feature here is the addition of a "dummy" proposition *nat* whose use will be explained below [3].

2.2 A Thumbnail Sketch of Kripke and Realizability semantics

The kind of Kripke model $\mathcal{K}$ we will be concerned with here is, in fact, a variant of the standard one in the literature. Strictly speaking, it is a *fallible Beth model*, because

- nodes are allowed to be inconsistent, i.e. some p may force *every* φ, provided not every node does so.
- We relax the definition of forcing of disjunction and existential quantification. Our variant, an instance of *forcing with covers over a site* (see Troelstra and van Dalen's [21], or Grayson's [11]), can be thought of as an analogue of Beth's forcing with *bars*. [4]

Therefore, we define a *Kripke/Beth structure* $\mathcal{B} = \langle B, \leq_B, \mathbf{D}, \Vdash_B, Cov\rangle$ over a language $\mathcal{L}$ (containing e.g., relation symbols, R_i, and constant symbols c_i [5]) as follows:

1. $\langle B, \leq_B\rangle$ is a pre-order. Members of B are called *nodes*.
2. $\mathbf{D}$ assigns a *domain* of individuals to nodes in a monotone way:
   $p \leq q \rightarrow \mathbf{D}(p) \subseteq \mathbf{D}(q)$.
3. $\Vdash$ is a *monotone* binary relation between *nodes* and *atomic sentences* over $\mathcal{L}$:

   $$p \leq q \quad \& \quad p \Vdash R(a) \quad \Rightarrow q \Vdash R(a).$$

   satisfying the *covering property* (if every member of a cover of p forces a sentence, then so does p):

   $$Cov(p, \mathbf{S}) \,\&\, (\forall q \in \mathbf{S})(q \Vdash \varphi) \Rightarrow p \Vdash \varphi$$

   where:

[3] At several points in our argument, when defining an *enrichment by constants* $\mathbf{APP}\underline{\mathcal{C}}$, and in the proof of theorem (2.8), we will need to think of an explicit set of axioms. To this end we have spelled out in some detail the necessary arguments, using Beeson's axioms, in the appendix.

[4] As a first approximation, we can think of forcing with covers as relaxing the condition that a node p of a Kripke model force a series of formulas by allowing instead that some set of nodes associated with p do the forcing. Such an associated set could be interpreted as a set of future states after state p: in which case we are tolerating some delay in the confirmation that p really does force something. See the references cited for other, more topological insights into this idea.

[5] since **APP** can be formulated relationally by taking *App* as a ternary predicate, this is all we need. Adding function symbols to the formalism is easy: the n-ary function symbol $\bar{f}$ is interpreted by the function f at node p if for every q above p $f : \mathbf{D}(q) \rightarrow \mathbf{D}(q)^n$ and the graph of f restricted to $\mathbf{D}(p)$ is contained in the graph of f at all higher nodes.

4. Cov is a binary relation between nodes p and *sets* of nodes $\mathbf{S} \subseteq B$, satisfying a series of *cover axioms*. Rather than give a general formulation of cover axioms here, we will limit ourselves to giving the one required for our argument, in definition 2.7 below. (See Grayson's [11] for a quite general formulation, or Troelstra and van Dalen, *op.cit.* for the original definition of forcing over a site, due to Joyal, and based on earlier ideas of Grothendieck.)

The forcing relation is extended to all sentences φ over the language $\mathcal{L}$ as follows:

1. $p \Vdash \varphi \,\&\, \psi$ iff $p \Vdash \varphi$ and $p \Vdash \psi$
2. $p \Vdash \varphi \vee \psi$ iff $(\exists \mathbf{S}) Cov(p, \mathbf{S})$ and $(\forall q \in \mathbf{S}) q \Vdash \varphi$ or $q \Vdash \psi$
3. p $\Vdash \varphi \rightarrow \psi$ iff $(\forall \geq p) q \Vdash \varphi \Rightarrow q \Vdash \psi$
4. p $\Vdash \exists x \varphi(x)$ iff $(\exists \mathbf{S}) Cov(p, \mathbf{S})$ and $(\forall q \in \mathbf{S})(\exists a \in \mathbf{D}(q)) q \Vdash \varphi(a)$
5. p $\Vdash \forall x \varphi(x)$ iff $(\forall q \geq p)(\forall a \in \mathbf{D}(q)) q \Vdash \varphi(a)$.

A more complete discussion of this and related realizability Kripke models and categories can be found in [16,15].

Interpreting Dependent Types We need to add one twist to the definitions just given. Using a syntax close to that of conventional first-order logic, we will be interpreting *dependent type expressions* of the form, e.g., $\prod x : D \cdot A(x)$. At first sight, this might seem like a natural "correlate" to the first-order predicate $(\forall x \in D)(A(x))$. However, in the type-free first-order language of arithmetic (or of **APP**) the latter formula is usually understood to be an abbreviation for $(\forall x)(D(x) \rightarrow A(x))$. This clearly violates the spirit of the Martin-Löf theory, in identifying type membership $x : D$, with what –for want of a better word – we call *parametricity*, $D(x)$. In our type free *semantics* we will remedy this by using *realizability* to distinguish between the two notions: $x : D$ will become $|(D)^r|(x)$ where the bars denote realizability, and the r- superscript denotes a translation to be defined below. However, in the *syntax* we will require an extension of the definition of well-formed formula to include *formalized bounded quantification* and some additional formulas as well:

Definition 2.1 *If D is a formula and $\theta(x)$ is a formula with x free, then $(\forall_{x \in D})\theta(x)$ and $(\exists_{x \in D})\theta(x)$ are formulas, with free variables $FV(\theta) \cup FV(D) \setminus \{x\}$. If D is a formula then τ_D is an* atomic *formula with one more variable free than D. $\tau_D(a)$ is simply a formalization of $a : D$ into our "extended first-order syntax."*[6] *We will often write $a : D$ for $\tau_D(a)$.*[7]

[6]The reader may wonder why we don't write $\tau_D(a)$ as $|D|(a)$, in keeping with the ideas expressed above. This is really what we are doing, only we are able to simplify the presentation somewhat with this device, by tacitly applying the *idempotence* of realizability (see [21]) and avoiding such nasty looking expressions as $||D|(a)|(b)$, i.e. b realizes that a realizes ...

[7]the context will make it clear whether we are referring to the syntax of the type theory or the extended first-order formulae.

All schemas of first-order logic and the axioms of **APP** are to be extended to the new formulae. Their *meaning* and role in our work should be made clear by the way they are treated in the semantics. We must now add to the definitions of Kripke forcing given above to include these new sentences.

$$p \Vdash (\exists_{x \in D})\theta(x) \stackrel{\text{def}}{\equiv} (\exists \mathbf{S})(Cov(p, \mathbf{S}))(\forall q \in \mathbf{S})(\exists a \in \mathbf{D}(q)) p \Vdash a : D \,\&\, \theta(a)$$
$$p \Vdash (\forall_{x \in D})\theta(x) \stackrel{\text{def}}{\equiv} (\forall q \geq p)(\forall a \in \mathbf{D}(q)) q \Vdash a : D \to \theta(a)$$

For a a constant in $D(p)$,

$$p \Vdash a : D \qquad (\text{ or } p \Vdash \tau_D(a))$$

is defined as follows:

$$p \Vdash a : C \vee D \stackrel{\text{def}}{\equiv} (\exists \mathbf{S})(Cov(p, \mathbf{S}))(\forall q \in \mathbf{S})$$
$$(q \Vdash \mathbf{p}_0 a = 0 \text{ and } q \Vdash \mathbf{p}_1 a : C) \vee (q \Vdash \mathbf{p}_0 a \neq 0 \text{ and } q \Vdash \mathbf{p}_1 a : D)$$
$$p \Vdash a : (\exists_{x \in D})\theta(x) \stackrel{\text{def}}{\equiv} p \Vdash \mathbf{p}_0 a : D \text{ and } p \Vdash \mathbf{p}_1 a : \theta(\mathbf{p}_0 a)$$
$$p \Vdash a : (\forall_{x \in D})\theta(x) \stackrel{\text{def}}{\equiv} (\forall q \geq p)(\forall a \in \mathbf{D}(q))(q \Vdash u : D \Rightarrow q \Vdash au : \theta(u))$$

For *atomic* D, the $p \Vdash a : D$ must also be specified by the atomic forcing assignment for the model in question. Every occurrence of an application au above is *strict*: it is understood that $au \downarrow$. Also, every ocurrence of the disjunctive flag condition $\mathbf{p}_0 a = 0$ or $\mathbf{p}_0 a \neq 0$ is preceded by a tacit $N(\mathbf{p}_0 a)$: the condition is decidable for natural numbers. For clarity, it is made explicit in definition (2.2) below.

We briefly recall the definition of **APP** realizability. The strong similarity to the definitions of the corresponding types in $\mathbf{ML}_0^i$ is what makes this semantics so natural.

The following definitions hold for **APP** as well as the enrichment by new constants **APP** $\underline{C}$ we will be considering below.

Definition 2.2 *Let A, B be formulas in one free variable over the language of* **APP** *(with possibly a denumerable set of new constants added). Then*

$$(A \times B)(x) \stackrel{\text{def}}{\equiv} A(\mathbf{p}_0 x) \,\&\, B(\mathbf{p}_1 x)$$
$$(A + B)(x) \stackrel{\text{def}}{\equiv} N(\mathbf{p}_0 x) \,\&\, (\mathbf{p}_0 x = 0 \to |A|(\mathbf{p}_1 x)) \,\&\, (\mathbf{p}_0 x \neq 0 \to |B|(\mathbf{p}_1 x))$$
$$(A \Rightarrow B)(x) \stackrel{\text{def}}{\equiv} (\forall y)(A(y) \to xy \downarrow \,\&\, B(xy))$$

Let $A(x, y)$ be a formula in two free variables, and D a formula. Then $(\sum_{xA \in D}(x, y))$ and $(\prod_{x \in D} A(x, y))$ are formulas in one free variable given by

$$(\textstyle\sum_{x \in D} A(x, y))(z) \stackrel{\text{def}}{\equiv} D(\mathbf{p}_0 z) \,\&\, A(\mathbf{p}_0 z, \mathbf{p}_1 z)$$
$$(\textstyle\prod_{x \in D} A(x, y))(z) \stackrel{\text{def}}{\equiv} \forall y[D(y) \to (zy) \downarrow \,\&\, A(y, zy)]$$

Definition 2.3 *Let A, B be sentences over the language of* **APP**. *Then we define inductively the realizability formulas $|A|$ in one free variable as follows*[8]*:*

$$\text{If } A \text{ is prime } |A|(x) \stackrel{\text{def}}{\equiv} A \,\&\, x\downarrow$$

$$\begin{aligned}
|nat|(e) &\stackrel{\text{def}}{\equiv} \mathbf{N}(e) \\
|A \,\&\, B| &\stackrel{\text{def}}{\equiv} |A| \times |B| \\
|A \vee B| &\stackrel{\text{def}}{\equiv} |A| + |B| \\
|A \to B| &\stackrel{\text{def}}{\equiv} |A| \Rightarrow |B| \\
|\exists_{y \in D} A(y)| &\stackrel{\text{def}}{\equiv} (\textstyle\sum_{y \in |D|} |A(y)|(x)), \text{ i.e. } \{z : |D|(\mathbf{p}_0 z) \,\&\, |A(\mathbf{p}_0 z)|(\mathbf{p}_1 z)\} \\
|\forall_{y \in D} A(y)| &\stackrel{\text{def}}{\equiv} (\textstyle\prod_{y \in D} |A(y)|(x)), \text{ i.e. } \{z : forall y[|D|(y) \to zy\downarrow \,\&\, |A(y)|(zy)]\}. \\
|\neg A| &\stackrel{\text{def}}{\equiv} |A \to \perp| \equiv \forall y \neg |A|(y)
\end{aligned}$$

Realizability of unbounded universal and existential quantification is just the special case of the corresponding bounded ones with D taken to be some tautological atomic sentence, such as $0 = 0$. We will also insist that nothing realize $\perp$.

In order to state and prove our main theorem below, we will need to define the realizability of the special formulas $a : D$, (that is to say, $|\tau_D(a)|$).

$$|a : D|(x) \stackrel{\text{def}}{\equiv} |D|(a).$$

In the literature the statement $|A|(x)$, i.e. x *realizes* A, is usually written $x \,\underset{\sim}{\mathbf{r}}\, A$. Note that if A is a formula in n variables over **APP** (or $\mathbf{APP}\underline{\mathbf{C}}$) then the above clauses define an associated realizability formula $|A|(x)$ in $n+1$ variables.

2.3 The Interpretation of $\mathbf{ML}_0^i$ into APP

We now give a whirlwind sketch of the realizability interpretation of the intensional $\mathbf{ML}_0^i$ theory, similar in spirit to that developed by Allen [2], Beeson [4], Troelstra and van Dalen [21], based on earlier work of Aczel[1] and on Martin-Löf's own informal semantics. Our definitions are very similar to those in Troelstra and van Dalen's textbook [21].

We first describe a translation of $\mathbf{ML}_0^i$ *types* A into *predicates* $[\![A]\!]$ in the language $\mathcal{L}(\mathbf{APP})$ (or, with slight variations, the language of Beeson's **EON** [4]), and of $\mathbf{ML}_0^i$ *terms*

[8] We explicitly created the proposition *nat* occurring below, so as to have a sentence which is *realized precisely by the set of natural numbers*. It is our way of dealing with the "coincidence of type and data" inherent in $a : nat$ having the same meaning as $\mathbf{N}(a)$. This is not indispensible in type theory: in Curry's type theory, or the polymorphic lambda calculus, **nat** is inhabited by *proofs of the induction schema* and not by "atomic data".

t into **APP**- or **EON**- terms t^*. We then define an induced translation of **ML** *judgements* Θ or contexts, into **APP** predicates $[\![\Theta]\!]$.

The translation of terms is given by: To each variable x we assign a variable x^* of **APP**, such that $x \not\equiv y \Rightarrow x^* \not\equiv y^*$.

$$0^* \stackrel{\text{def}}{\equiv} 0, \qquad (St)^* \stackrel{\text{def}}{\equiv} S(t^*);$$

$$R_{x,y}(t, t_0, t_1[x,y])^* \stackrel{\text{def}}{\equiv} \mathbf{r}t_0^*(\lambda y^* x^* \cdot t_1^*[x^*, y^*])t^*;$$

$$(\lambda x \cdot t)^* \stackrel{\text{def}}{\equiv} \lambda x^* \cdot t^*, \qquad (t_0 t_1)^* \stackrel{\text{def}}{\equiv} t_0^* t_1^*;$$

$$(t_0, t_1)^* \stackrel{\text{def}}{\equiv} \mathbf{p}t_0^* t_1^*, \qquad (\mathbf{p}_i t)^* \stackrel{\text{def}}{\equiv} \mathbf{p}_i t^* \qquad (i \in 0,1)$$

$$(\mathbf{k}_i t)^* \stackrel{\text{def}}{\equiv} \mathbf{p}_i t^* \qquad (i \in 0,1);$$

$$D_{x,y}(t, t_0[x], t_1[y])^* \stackrel{\text{def}}{\equiv} \mathbf{d}(t_0^*[x^*/\mathbf{p}_1 t^*])(t_1^*[y^*/\mathbf{p}_1 t^*])0(\mathbf{p}_0 t^*)$$

(the λ on the right-hand side is defined in **APP**);
For types, define:

$$[\![\mathbf{nat}]\!] \stackrel{\text{def}}{\equiv} \{x : \mathbf{N}(x)\}$$

$$[\![\textstyle\prod x : A \cdot B]\!] \stackrel{\text{def}}{\equiv} \prod_{x \in [\![A]\!]} [\![B(x)]\!] \equiv \{y : \forall x([\![A]\!](x) \to [\![B(x)]\!](y)\},$$

$$[\![\textstyle\sum x : A \cdot B]\!] \stackrel{\text{def}}{\equiv} \sum_{x \in [\![A]\!]} [\![B(x)]\!] \equiv \{\mathbf{p}xy : [\![A]\!](x) \wedge [\![B(x)]\!](y)\},$$

$$[\![A + B]\!] \stackrel{\text{def}}{\equiv} \{x : \mathbf{N}(\mathbf{p}_0 x) \wedge [(\mathbf{p}_0 x = 0 \wedge [\![A]\!](\mathbf{p}_1 x)) \vee (\mathbf{p}_0 x \neq 0 \wedge [\![B]\!](\mathbf{p}_1 x))]\}$$

and if s^* and t^* have been defined,

$$[\![I(A, s, t)]\!] \stackrel{\text{def}}{\equiv} \{0 : s^* = t^*\}$$

Judgements are translated as follows. A context $\langle x_1 : A_1, \ldots, x_n : A_n \rangle$ is translated as $[\![A_1]\!](x_1^*) \wedge \ldots \wedge [\![A_n]\!](x_n^*)$.

$$[\![t : A]\!] \stackrel{\text{def}}{\equiv} [\![A]\!](t^*);$$

$$[\![A = B]\!] \stackrel{\text{def}}{\equiv} [\![A]\!](x) \leftrightarrow [\![B]\!](x);$$

$$[\![t = s : A]\!] \stackrel{\text{def}}{\equiv} t^* = s^* \wedge [\![A]\!](t^*) \wedge [\![A]\!](s^*);$$

$$[\![A \text{ type}]\!] \quad \stackrel{\text{def}}{\equiv} \quad 1 = 1 \text{ (i.e., true}^9 \text{)}.$$

Troelstra and van Dalen obtain the following soundness result for their translation. A similar result is to be found in Beeson, *op.cit.*

Theorem 2.4 *If* $\mathbf{ML}_0^i \vdash \Gamma \gg \theta$, *then* $\mathbf{APP} \vdash [\![\Gamma]\!] \to [\![\theta]\!]$.

An immediate corollary:

Corollary 2.5 *Suppose the type A is provably inhabited in $\mathbf{ML}_0^i$, that is to say, there is a proof in $\mathbf{ML}_0^i$ ending with the sequent*

$$\gg t : A$$

for some term t. Then $\mathbf{APP} \vdash [\![A]\!](t^*)$

In order to mediate between type theory and the first-order language of partial application, we will need to make the following "reverse translation" conventions:

$$(\mathbf{nat})^r \stackrel{\text{def}}{\equiv} nat \tag{7}$$

$$(A+B)^r \stackrel{\text{def}}{\equiv} (A)^r \vee (B)^r \tag{8}$$

$$(\textstyle\prod x : A \cdot B(x))^r \stackrel{\text{def}}{\equiv} (\forall_{x \in (A)^r})[(B(x))^r] \tag{9}$$

$$(\textstyle\sum x : A \cdot B(x))^r \stackrel{\text{def}}{\equiv} (\exists_{x \in (A)^r})[(B(x))^r] \tag{10}$$

Lemma 2.6 (Reverse-Translation lemma) *Let A be a type in $\mathbf{ML}_0^i$. Then the following equivalence is provable in* $\mathbf{APP}$ *:*

$$[\![A]\!](e) \equiv |(A)^r|(e)$$

proof: A straightforward induction argument. $[\![\mathbf{nat}]\!](e)$ is *defined* to be $\mathbf{N}(e)$, whereas $(\mathbf{nat})^r$ is nat, and $|nat|(e) \equiv \mathbf{N}(e)$. Proceeding inductively, we have:

$$[\![A+B]\!](e) \equiv (\mathbf{p}_0 e = 0 \,\&\, [\![A]\!](\mathbf{p}_1 e)) \vee (\mathbf{p}_0 e = 1 \,\&\, [\![B]\!](\mathbf{p}_1 e))$$

which, by the inductive hypothesis, gives:

$$[\![A+B]\!](e) \equiv (\mathbf{p}_0 e = 0 \,\&\, |(A)^r|(\mathbf{p}_1 e)) \vee (\mathbf{p}_0 e = 1 \,\&\, |(B)^r|(\mathbf{p}_1 e))$$

the right-hand side of which is $|(A+B)^r|$.

The existential case:

$$\begin{array}{rcl} [\![\textstyle\sum x : A \cdot B]\!](e) & \equiv & [\![A]\!](\mathbf{p}_0 e) \,\&\, [\![B(\mathbf{p}_0 e)]\!](\mathbf{p}_1 e) \\ & & |(A)^r|(\mathbf{p}_0 e) \,\&\, |(B)^r(\mathbf{p}_0 e)|(\mathbf{p}_1 e) \\ & & |(\exists_{x \in A})(B(x))^r|(e) \\ & & |\textstyle\sum x : A \cdot B|(e) \end{array}$$

We leave the proof of the $\prod$ - case to the reader.

Now we are in a position to define a Kripke model, $\mathcal{K}$, sound for $\mathbf{ML}_0^i$ in the sense that

$$A \text{ is provably inhabited in } \mathbf{ML}_0^i \Rightarrow \mathcal{K} \models (A)^r$$

The Model $\mathcal{K}$

Let $\mathbf{APP}\underline{C}$ stand for the theory $\mathbf{APP}$ enriched with a denumerable set $\underline{C}$ of fresh constants $\{c_i : i \in \omega\}$ together with the extension of all relevant axiom schemas to include these constants (e.g. $c \downarrow$ for every such c, and the equality schemas, such as (A1)-(A8) in the appendix). $\underline{C}$ will play a role similar to the *set of witnesses* from Henkin's famous proof of the completeness theorem (see Chang-Keisler's treatment in [5]). Let Δ be the set of all closed (variable free) terms of $\mathbf{APP}\underline{C}$.

Definition 2.7 *The* **nodes** *of $\mathcal{K}$ are formulas $A(x)$ of $\mathbf{APP}\underline{C}$ in one free variable. The* **order relation** *$\leq$ is given by*

$$A \geq B \iff (\exists t \in \Delta)\ \mathbf{APP}\underline{C}\ \vdash \forall x[A(x) \rightarrow tx \downarrow \ \& \ B(tx)]. \tag{11}$$

When (11) obtains we may write $A \overset{t}{\geq} B$ or $t : A \rightarrow B$ if we want to make reference to the witnessing term t.

The **domain** *function* $\mathbf{D}$ *is constant and always equal to Δ. The* **atomic forcing relations** *are given by*

$$A \Vdash \varphi \quad \iff \quad A \geq |\varphi| \tag{12}$$

and, in particular,

$$A \Vdash a : D \quad \iff \quad A \geq |a : D| \tag{13}$$

where φ and D are atomic in (12). $\mathcal{K}$ is equipped with the following **notion of cover***: For $A \in |\mathcal{K}|$ and $\mathbf{S}$ a set of nodes of $\mathcal{K}$ we have $Cov(A, \mathbf{S})$ whenever*

1. $(\forall B \in \mathbf{S})(\exists t \in \Delta)(B \overset{t}{\geq} A)$
2. *whenever for some $D \in \mathcal{K}$ we have $(\forall B \in \mathbf{S})(\exists t_B \in \Delta)(B \overset{t}{\geq} D)$ then there is an $f \in \Delta$ such that $A \overset{f}{\geq} D$*

i.e., $Cov(A, \mathbf{S}) \iff A$ is the **greatest lower bound** *of* $\mathbf{S}$.

Note that for atomic φ, $|\varphi|(x) \overset{\text{def}}{\equiv} \varphi$, so (12) reduces to

$$A \Vdash \varphi \iff \mathbf{APP}\underline{C}\ \vdash \forall x(A(x) \rightarrow \varphi).$$

Theorem 2.8 *Equivalence (12) obtains for* all *sentences φ That is to say, let φ be any (possibly nonatomic) sentence over the language of $\mathbf{APP}\underline{C}$. Then*

$$A \Vdash \varphi \iff A \geq |\varphi|.$$

proof: The atomic case follows from the specification of our Kripke model. For any formulas free of *special subformulas* $a : D$, $(\exists_{x \in D})\theta(x)$, or $(\forall_{x \in D})\theta(x)$ this is the same result as in [16]. So we will consider only the disjunctive and special formulas (which are the only ones required to interpret $\mathbf{ML}_0^i$). First, however, we will need the following lemma, which essentially asserts that for the trivial special case of a formula which *already supplies the inhabiting witness* our Kripke model $\mathcal{K}$ satisfies the conclusion of the preceding theorem in the obvious required way: it respects the choice of witness. The lemma is an immediate consequence of the definition of forcing of special formulas $a : D$.

Lemma 2.9 *For any node A, any term $a \in \Delta$ and any formula D, we have:*

$$A \Vdash a : D \iff A \overset{\lambda x \cdot a}{\geq} |a : D|,$$

that is to say:

$$\mathbf{APP}\underline{C} \vdash \forall x(A(x) \rightarrow |D|(a)).$$

proof: The atomic case follows by definition.

Disjunction: suppose $A \Vdash a : C \vee D$. Then

$$A \Vdash \mathbf{p}_0 a = 0 \,\&\, \mathbf{p}_1 a : C \quad \text{or} \quad A \Vdash \mathbf{p}_0 a \neq 0 \,\&\, \mathbf{p}_1 a : D.$$

Recall that by definition of forcing of ground atomic formulae,

$$A \Vdash \mathbf{p}_0 a = 0 \quad \equiv \quad \mathbf{APP}\underline{C} \vdash \forall x(A(x) \rightarrow \mathbf{p}_0 a = 0).$$

Therefore, by inductive hypothesis,

$$\forall x(A(x) \rightarrow \mathbf{p}_0 a = 0 \,\&\, |C|(\mathbf{p}_1 a)) \text{ or } \forall x(A(x) \rightarrow \mathbf{p}_0 a \neq 0 \,\&\, |D|(\mathbf{p}_1 a)),$$

provably in $\mathbf{APP}\underline{C}$. But in either case, we have $\forall x(A(x) \rightarrow |C \vee D|(a)$

Suppose $A \Vdash a : (\exists_{x \in D})\theta(x)$. Then

$$A \Vdash \mathbf{p}_0 a : D \text{ and } A \Vdash \mathbf{p}_1 a : \theta(\mathbf{p}_0 a).$$

By inductive hypothesis, we have:

$$\mathbf{APP}\underline{C} \vdash \forall x(A(x) \rightarrow |D(\mathbf{p}_0 a)|$$

and

$$\mathbf{APP}\underline{C} \vdash \forall x(A(x) \rightarrow |\theta(\mathbf{p}_0 a)|(\mathbf{p}_1 a))$$

from which

$$\mathbf{APP}\underline{C} \vdash \forall x(A(x) \rightarrow |D(\mathbf{p}_0 a)| \,\&\, |\theta(\mathbf{p}_0 a)|(\mathbf{p}_1 a))$$

which is the conclusion $A \overset{\lambda x \cdot a}{\geq} |a : (\exists_{x \in D})\theta(x)|$. The converse is straightforward and left to the reader.

Now, suppose $A \mathrel{||\!-} a : (\forall_{x\in D})\theta(x)$, i.e. for every term u in Δ and every $B \geq A$.

$$B \mathrel{||\!-} u : D \qquad \Rightarrow \qquad B \mathrel{||\!-} au : \theta(u). \tag{14}$$

Then we have: $A \times |u : D| \geq A$ and $A \times |u : D| \geq |u : D|$ by the canonical projections, so, by the inductive hypothesis $A \times |u : D| \mathrel{||\!-} u : D$ hence, by (14) and the inductive hypothesis again,

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x \forall y (A(x) \,\&\, |u : D|(y) \to |\theta(u)|(au)), \tag{15}$$

where $|u : D|(y)$ means $|D|(u)$. Now, since u may be taken to be *any* term in Δ, we pick a constant c in $\underline{\mathrm{C}}$ not present elsewhere in (15). We would now like to argue that, since c is a "fresh" constant, we can quantify over it in (20). This is the whole point of adding the fresh constants $\underline{\mathrm{C}}$. But we cannot quite yet do so: *c may occur in the premisses* $\mathbf{APP}\underline{C}$. Nonetheless, the occurrence is only in finitely many axioms $\Gamma_0(x)$ from the the *logical part* of $\mathbf{APP}\underline{\mathbf{C}}$, in the axiom schemas for equality, t substitutivity, etc. Picking a specific formulation of $\mathbf{APP}$, we easily show, (see appendix II) that we can effectively quantify over c obtaining:

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(A(x) \to \forall z(|D|(z) \to |\theta(z)|(az))). \tag{16}$$

which is the conclusion of the lemma.

Conversely, suppose

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(A(x) \to \forall u(|D|(u) \to |\theta(u)|(au))). \tag{17}$$

Pick u in Δ and assume that some function f witnesses $B \geq A$, and that $B \mathrel{||\!-} u : D$. Then, by inductive hypothesis,

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \to |D|(u))$$

and

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \to fx \downarrow \,\&\, A(fx))$$

so by (17) we must have

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \to |\theta(u)|(au))$$

so, by, inductive hypothesis again, $B \mathrel{||\!-} au : \theta(u)$ so $A \mathrel{||\!-} (\forall_{x\in D})\theta(x)$.

The remaining cases for the lemma are straightforward.

proof of theorem 2.8: We begin with disjunction. Suppose $A \Vdash \varphi \vee \theta$. Then, for some cover $\mathbf{S}$ of A we have

$$(\forall B \in \mathbf{S}) B \Vdash \varphi \text{ or } B \Vdash \theta.$$

By the induction hypothesis, for each $B \in \mathbf{S}$ there is a term t_B in Δ such that

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \rightarrow t_B x \downarrow \ \& \ \varphi(t_B x)) \tag{18}$$

or

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \rightarrow t_B x \downarrow \ \& \ \theta(t_B x)). \tag{19}$$

Suppose (18) holds. Then $\lambda x \cdot \langle 0, t_B x\rangle : B \rightarrow |\varphi \vee \theta|$. If (19) holds, then $\lambda x \cdot \langle 1, t_B x\rangle : B \rightarrow |\varphi \vee \theta|$. So, in all cases, $B \geq |\varphi \vee \theta|$. Now, by definition of cover in this model, we have $A \geq |\varphi \vee \theta|$, as we wanted to show.

Conversely, if for some term f in Δ we have, provably in $\mathbf{APP}\underline{\mathbf{C}}$, $f : A \rightarrow |\varphi \vee \theta|$, then, letting

$$\begin{aligned} A_0(x) &\stackrel{\text{def}}{\equiv} A(x) \ \& \ fx = 0 \\ A_1(x) &\stackrel{\text{def}}{\equiv} A(x) \ \& \ fx = 1 \end{aligned}$$

We have f witnessing both $A_0 \geq |\varphi|$ and $A_1 \geq |\theta|$. Now all that remains to be shown is that the pair of formulas $\{A_0, A_1\}$ is a cover for A. This means showing that, for some Δ - terms s and t, and some node D, if $A_0 \stackrel{s}{\geq} D$ and $A_1 \stackrel{t}{\geq} D$ then there is an h in Δ with $A \stackrel{h}{\geq} D$. Clearly we need only set

$$h \stackrel{\text{def}}{\equiv} \lambda x \cdot \mathbf{if}\ fx = 0\ \mathbf{then}\ sx\ \mathbf{else}\ tx.$$

existential case: Suppose $A \Vdash (\exists_{x \in D})\theta(x)$, i.e., for some cover $\mathbf{S}$ of A, and every member B of $\mathbf{S}$, there is a term a in Δ with

$$B \Vdash a : D \quad \& \quad B \Vdash \theta(a).$$

By the preceding lemma, we know that

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \rightarrow |D|(a))$$

and by the inductive hypothesis, for some term t

$$\mathbf{APP}\underline{\mathbf{C}} \ \vdash \forall x(B(x) \rightarrow tx \downarrow \ \& \ |\theta(a)|(tx)).$$

Letting h be defined by $\lambda x \cdot \langle a, tx\rangle$, we have $B \stackrel{h}{\geq} |(\exists_{x \in D})\theta(x)|$. By the assumption that $\mathbf{S}$ covers A, we must have $A \geq |(\exists_{x \in D})\theta(x)|$.

Conversely, suppose $f \in \Delta$ and

$$\mathbf{APP\underline{C}} \ \vdash \forall x(A(x) \to fx\downarrow \ \& \ |D|(\mathbf{p}_0(fx)) \ \& \ |\theta(\mathbf{p}_0(fx))|(\mathbf{p}_1(fx))).$$

Proceeding somewhat like the disjunctive case, we split A into a cover by pulling back along f. Let S be the formula in two free variables given by

$$S(z,x) \stackrel{\text{def}}{\equiv} A(x) \ \& \ \mathbf{p}_0(fx) = z.^{10}$$

Then, for each a in Δ we have, trivially,

$$\mathbf{APP\underline{C}} \ \vdash \forall x(S(a,x) \to |D|(a) \ \& \ |\theta(a)|(\mathbf{p}_1(fx))).$$

Thus, by the inductive hypothesis, we have, for each a in Δ,

$$S(a,x) \Vdash a : D \quad \& \ S(a,x) \Vdash \theta(a).$$

All that's left to show is that the family of nodes $\mathbf{S} \stackrel{\text{def}}{\equiv} \{S(a,x) : a \in \Delta\}$ is a cover of A. But suppose, for some node R and each a in Δ, there is some term t_a with

$$\mathbf{APP\underline{C}} \ \vdash \forall x(S(a,x) \to t_a x\downarrow \ \& \ R(t_a x)).$$

Then pick $a \notin \mathcal{L}(\mathbf{APP} \cup \{R(x), A(x)\})$. By combinatory completeness, there is a term h in Δ in which a does not occur, such that

$$\mathbf{APP\underline{C}} \ \vdash h\downarrow \ \& \ ha \simeq t_a.$$

Thus

$$\mathbf{APP\underline{C}} \ \vdash \forall x(S(a,x) \to (ha)x\downarrow \ \& \ R((ha)x)). \tag{20}$$

By the same argument given to justify (16) we quantify over the "fresh" constant a, obtaining:

$$\mathbf{APP} \vdash \forall x \forall z(S(z,x) \to (hz)x\downarrow \ \& \ R((hz)x))$$

and hence, by the definition of S,

$$\mathbf{APP\underline{C}} \ \vdash \forall x[A(x) \ \& \ (h\mathbf{p}_0(fx))x\downarrow \ \& \ R((h\mathbf{p}_0(fx))x)],$$

so $A \geq R$. This establishes $Cov(A, \mathbf{S})$. Therefore

$$A \Vdash (\exists_{x \in D})\theta(x).$$

[10]equality between terms is always strict: $fx\downarrow$ is built-in to this formula.

universal case: Suppose $A \Vdash (\forall_{x \in D})\theta(x)$. Then for every $B \geq A$ and $a \in \Delta$,

$$B \Vdash a : D \quad \Rightarrow \quad B \Vdash \theta(a).$$

As in the proof of the preceding lemma, we argue that $A \times |a : D| \geq A$, and $A \times |a : D| \geq |a : D|$ via the canonical projections, hence, applying the inductive hypothesis, we have $A \times |a : D| \geq |\theta(a)|$, i.e. , for some f

$$\mathbf{APP\underline{C}} \;\vdash \forall x \forall y[A(x) \mathbin{\&} |a : D|(y) \rightarrow f\langle x, y\rangle\downarrow \;\&\; |\theta(a)|(f\langle x, y\rangle)].$$

But $|a : D|(y)$ means $|D|(a)$, so, substituting a for y and putting $h \stackrel{\text{def}}{\equiv} \lambda x \lambda y \cdot f\langle x, y\rangle$ we have, with a little currying:

$$\mathbf{APP\underline{C}} \;\vdash \forall x[A(x) \rightarrow (|D|(a) \rightarrow (hx)a\downarrow \;\&\; |\theta(a)|((hx)a))].$$

Now, choose a to be a fresh constant in $\underline{C}$, and universally quantify, obtaining

$$\mathbf{APP\underline{C}} \;\vdash \forall x[A(x) \rightarrow \forall z(|D|(z) \rightarrow (hx)z\downarrow \;\&\; |\theta(z)|((hx)z))],$$

in other words, $A \stackrel{h}{\geq} |(\forall_{x \in D})\theta(x)|$.

Conversely, suppose for some Δ-term h,

$$\mathbf{APP\underline{C}} \;\vdash \forall x[A(x) \rightarrow hx\downarrow \;\&\; \forall u(|D|(u) \rightarrow (hx)u\downarrow \;\&\; |\theta(u)|((hx)u))]. \tag{21}$$

Pick a in Δ, and suppose $B \stackrel{f}{\geq} A$ and $B \Vdash a : D$. By the lemma $B \geq |a : D|$, so we have, using (21),

$$\mathbf{APP\underline{C}} \;\vdash \forall x[B(x) \rightarrow h(fx)\downarrow \;\&\; (h(fx))a\downarrow |\theta(a)|(h(fx))a].$$

So, for $w = \lambda x \cdot (h(fx))a$, $A \stackrel{w}{\geq} |\theta(a)|$, whence, by the induction hypothesis, $A \Vdash \theta(a)$. But then, we have shown $A \Vdash (\forall_{x \in D})\theta(x)$, which proves the theorem. ∎

From this we can immediately conclude:

Corollary 2.10 *φ is true in the Kripke/Beth model $\mathcal{K}$ $\iff$ it is provably realizable in* **APP**.

proof: If φ is forced by every A then $\{0\} \Vdash \varphi$, where $\{0\}$, the root node of $\mathcal{K}$, is the predicate $x = 0$. But

$$\begin{aligned}
&\{0\} \geq |\varphi| \\
&\qquad\iff (\exists f \in \Delta)\mathbf{APP} \vdash (\forall x)(x = 0 \rightarrow |\varphi|(fx)) \\
&\qquad\iff (\exists f \in \Delta)\mathbf{APP} \vdash |\varphi|(f(0)) \\
&\qquad\iff (\exists e \in \Delta)\mathbf{APP} \vdash |\varphi|(e)
\end{aligned}$$

so being forced by every node is equivalent to being **APP**-realizable.[11]

Corollary 2.11 (Soundness of the interpretation) *Let A be a type in* $\mathbf{ML}_0^i$, *provably inhabited in that theory. In other words, for some term u, the sequent* $\gg u : A$ *is provable in* $\mathbf{ML}_0^i$. *Then* $\mathcal{K} \models (A)^r$

proof: By the corollary (2.5) to Troelstra and van Dalen's translation theorem we have $\mathbf{APP} \vdash [\![A]\!](u^*)$, from which, by the "reverse translation" lemma (2.6) we obtain $\mathbf{APP} \vdash |(A)^r|(u^*)$. By corollary (2.10), we have $\mathcal{K} \models (A)^r$ ▌

Variants of the model

Our construction provides a countable collection of non-equivalent models for $\mathbf{ML}_0^i$ since the model $\mathcal{K}_A$ taken by restricting attention to all nodes above a given node A is itself a Kripke model. It is not hard to see that $\mathcal{K}$ has countably many nonequivalent nodes. Pick any node A, and let B be a *sentence* over the language of $\mathbf{APP}\underline{\mathbf{C}}$ independent of $\mathbf{APP}\underline{\mathbf{C}} \cup \{\exists x A(x)\}$ We cannot have $A \geq B$, since this means that for some term f

$$\mathbf{APP} \vdash \forall x(A(x) \to fx\downarrow \;\&\; B).$$

But then, by existential elimination and arrow introduction, we have

$$\mathbf{APP} \vdash (\exists x A(x)) \to B,$$

contradicting independence. (Similarly we cannot have $A \geq \neg B$). We can, of course iterate this (tacitly using Gödel's incompleteness theorem to supply new sentences), obtaining a B_1 not below A or B, and then a B_2, etc. In effect our Kripke model *lifts independence results in* **APP** *to independence results in* $\mathbf{ML}_0^i$. If the realizability of $(\theta)^r$ is independent of **APP** then θ cannot be forced by the root node of $\mathcal{K}$, nor can its negation, hence neither is provably inhabited in $\mathbf{ML}_0^i$.

Conclusion, further directions

The preceding results are a first step in defining a model theory for dependent types. The models described herein are, in many respects, similar to those induced by realizability interpretations: their truth-value structure is that of the *degrees of inhabitation* of definable sets. It is perhaps not entirely surprising that models of type theory depend on how we model inhabitation itself. It would be interesting to see if that is the whole story. No

[11] This is true because we can insist on e being a closed term in the pure language of **APP**, and add $\mathbf{APP} \vdash \exists x|\varphi|(x)$ to the list of equivalences. Just abstract any "fresh" constants from e (by the *a-lemma* in the appendix) to get $e \simeq e'a_1 \cdots a_n$ where the a_i are in Δ and e' is in $\mathcal{L}(\mathbf{APP})$. Then generalize on the a_i and instantiate them to your favorite constant in the language, say 0.

complete semantics has been found as yet. This would provide an algebraic characterization of validity in the Martin-Löf type theory and is clearly the next topic to investigate. Also unexplored are ways to extend the semantics described here to higher universes, as well as how to incorporate computation rules in an explicit way, as in Nuprl [6] and in Allen's type-free model in [2].

Our model is reasonably constructive: validity in $\mathcal{K}$ is r.e. by corollary (2.10). Perhaps an even more constructive semantics would be achieved by harnessing the proof of the constructive completeness theorem for Beth models due to Friedman, Veldman, de Swart, Lopez-Escobar, Troelstra and van Dalen (see [21]), along the lines of the realizability model in [17]. This is tantamount to generating models by a tableau procedure, which is of obvious interest for automating deduction and the search for countermodels in such systems.

What can we hope to learn from developing this model theory? One of the main features of systems like that of Martin Löf is the incorporation of term-extraction into the theory itself at all levels of the hierarchy. The challenge of providing a complete semantics here is really to characterize the process of term extraction itself, which is of central importance to computer science.

Appendix

A The Syntax and Rules of $\mathbf{ML}_0^i$

Structural Rules

$$\frac{\Gamma \gg A \text{ type}}{\Gamma,\ x : A \gg x : A} \qquad x \notin FV(A) \tag{22}$$

$$\frac{\Gamma \gg A\ type \qquad \Gamma,\ \Gamma'}{\Gamma\, x : A,\ \Gamma \gg \theta} \qquad x \notin FV(\Gamma \cup \Gamma') \tag{23}$$

General Equality Rules

$$\frac{t : A}{t = t : A} \qquad \frac{t = t : A}{t : A} \qquad \frac{A \text{ type}}{A = A} \tag{24}$$

$$\frac{t = t' : A}{t' = t : A} \qquad \frac{A = B}{B = A} \tag{25}$$

$$\frac{t = t' : A \qquad t' = t'' : A}{t = t'' : A} \qquad \frac{A = B \qquad B = C}{A = C} \tag{26}$$

$$\frac{\Gamma,\ x : A,\ \Gamma' \gg \Theta \qquad \Gamma \gg t : A}{\Gamma,\ \Gamma'[x/t] \gg \Theta[x/t]} \tag{27}$$

$$\frac{\Gamma,\ x : A,\ \Gamma' \gg B \,\text{type} \qquad \Gamma \gg t = t' : A}{\Gamma,\ \Gamma'[x/t] \gg B[x/t] = B[x/t']} \tag{28}$$

$$\frac{\Gamma,\ x : A,\ \Gamma' \gg s : B \qquad \Gamma \gg t = t' : A}{\Gamma,\ \Gamma'[x/t] \gg s[x/t] = s[x/t'] : B[x/t]} \tag{29}$$

$$\frac{t : A \qquad A = B}{t : B} \tag{30}$$

Type-induction rules of $\mathbf{ML}_0^i$

$$\textbf{nat type} \qquad\qquad \frac{x : A \gg B\,\text{type}}{\prod x : A \cdot B\,\text{type}} \tag{31}$$

$$\frac{x : A \gg B\,\text{type}}{\sum x : A \cdot B\,\text{type}} \tag{32}$$

$$\frac{A\,\text{type} \qquad B\,\text{type}}{A + B\,\text{type}} \tag{33}$$

$$\frac{t : A \qquad s : A \qquad A\,\text{type}}{I(A, t, s,)\,\text{type}} \tag{34}$$

Introduction and elimination rules of $\mathbf{ML}_0^i$.

$$0 : \mathbf{nat} \qquad\qquad \frac{t : \mathbf{nat}}{St : \mathbf{nat}} \tag{35}$$

$$\frac{t : \mathbf{nat} \qquad t_0 : A[x/0] \qquad x : \mathbf{nat},\ y : A \gg t_1 : A[x/Sx] \qquad x : \mathbf{nat} \gg A\,\text{type}}{R_{x,y}(t, t_0, t_1) : A[x/t]} \tag{36}$$

($R_{x,y}$ binds x and y in t_1),

$$\frac{x : A \gg t : B \qquad x : A \gg B\,\text{type}}{\lambda x \cdot t : (\prod x : A \cdot B)} \tag{37}$$

$$\frac{t : (\prod x : A \cdot B) \qquad t' : A \qquad x : A \gg B\,\text{type}}{tt' : B[x/t']} \tag{38}$$

$$\frac{t : A \qquad t' : B[x/t] \qquad x : A \gg B\,\text{type}}{\mathbf{p}(t, t') : (\sum x : A \cdot B)} \tag{39}$$

$$\frac{t : (\sum x : A \cdot B) \qquad A \text{ type}}{\mathbf{p}_0 t : A} \qquad\qquad \frac{t : (\sum x : A \cdot B) \qquad x : A \gg B \text{ type}}{\mathbf{p}_1 t : B[x/\mathbf{p}_0 t]} \tag{40}$$

$$\frac{t : A \qquad A \text{ type} \qquad B \text{ type}}{\mathbf{k}_0 t : A + B} \qquad\qquad \frac{t : B \qquad A \text{ type} \qquad B \text{ type}}{\mathbf{k}_1 t : A + B} \tag{41}$$

$$\frac{t : A + B \qquad x : A \gg t_0 : C[z/\mathbf{k}_0 x] \qquad y : B \gg t_1 : C[z/\mathbf{k}_1 y] \qquad z : A + B \gg C \text{ type}}{D_{x,y}(t, t_0, t_1) : C[z/t]} \tag{42}$$

($D_{x,y}$ binds x in t_0 and y in t_1, $x \notin FV(t_1)$, $y \notin FV(t_0)$).

$$\frac{t = t' : A \qquad A \text{ type}}{\mathbf{e} : I(A, t, t')} \tag{43}$$

$$\frac{t'' : I(A, t, t') \qquad A \text{ type}}{t = t' : A} \tag{44}$$

Reduction rules (CONV-rules)

$$\begin{aligned}
R_{x,y}(0, t_0, t_1) &\rightsquigarrow t_0 && (45)\\
R_{x,y}(St, t_0, t_1) &\rightsquigarrow t_1[x, u/t, T_{x,y}(t, t_0, t_1)] && (46)\\
Ap(\lambda x \cdot t, t') &\rightsquigarrow t[x/t'] && (47)\\
\mathbf{p}_i(t_0, t_1) &\rightsquigarrow t_i \qquad (i = 0, 1) && (48)\\
(\mathbf{p}_0 t, \mathbf{p}_1 t) &\rightsquigarrow t && (49)\\
D_{x,y}(\mathbf{k}_i t, t_0, t_1) &\rightsquigarrow t_i[x_i/t] \qquad (i : \{0, 1\}). && (50)
\end{aligned}$$

Computation Rules

$$\frac{t_0 : A[0] \qquad x : N \cdot y : A \gg t_1 : A[x/Sx]}{R_{x,y}(0, t_0, t_1) = t_0 : A[0]} \tag{51}$$

$$\frac{t : N \qquad t_0 : A[0] \qquad x : N, y : A \gg t_1 : A[x/Sx] \qquad x : N \gg A\,\text{type}}{R_{x,y}(St, t_0, t_1) = t_1[x, y/t, R_{x,y}(t, t_0, t_1)] : A[St]} \tag{52}$$

$$\frac{\lambda x \cdot t : (\prod x : A \cdot B) \qquad t' : A \qquad x : A \gg B\,\text{type}}{(\lambda x \cdot t)(t') = t[x/t'] : B[x/t']} \tag{53}$$

$$\frac{(t_0, t_1) : (\sum x : A \cdot B) \qquad A\,\text{type}}{\mathbf{p}_0(t_0, t_1) = t_0 : A} \qquad \frac{(t_0, t_1) : (\sum x : A \cdot B) \qquad x : A \gg B\,\text{type}}{\mathbf{p}_1(t_0, t_1) = t_1 : B[x/t_0]} \tag{54}$$

$$\frac{t : (\sum x : A \cdot B)}{(\mathbf{p}_0 t, \mathbf{p}_1 t) = t : (\sum x : A \cdot B)} \tag{55}$$

$$\frac{t : A \qquad \begin{array}{lcl} x_0 : A_0 & \gg & t_0 : C[z/\mathbf{k}_0 x_0] \\ x_1 : A_1 & \gg & t_1 : C[z/\mathbf{k}_1 x_1] \end{array} \qquad z : A_0 + A_1 \gg C\,\text{type}}{D_{x_0,x_1}(\mathbf{k}_i t, t_0, t_1) = t_i[x_i/t] : C[z/\mathbf{z}_i t]} \tag{56}$$

For I-types:

$$\frac{t : I(A, s, s')}{t = \mathbf{e} : I(A, s, s')} \tag{57}$$

We may also extend our theory to include new "small types" A_1, $A_2, \ldots,$ provided we introduce axioms A_1 type , A_2 type , etc. Also, we may extend $\mathbf{ML}_0^i$ by definitions to introduce primitive recursive function symbols, e.g., $+$ and $\times$ with reduction rules given by the standard interpretation of these operations.

B Some useful facts about applicative theories

We spell out a particular axiomatization of the theory of partial application, using Beeson's **EON** because of its somewhat compact presentation. Our aim is to state those properties that were used in the proofs above, and establish a few technical lemmas.

The logic of partial terms (LPT): This logic includes the usual rules for propositional logic plus the following rules of inference:

$$R\forall \frac{B \to A}{B \to \forall x A} \qquad R\exists \frac{A \to B}{\exists x A \to B} \quad (x \text{ not free in } B)$$

and the following axioms (note that A1, A2, A4, A5, A6 are axiom schemas using metavariables t, s, t_i for arbitrary terms and A7, A8 schemas for special terms.)
(A1) $\forall x A \;\&\; t \downarrow \to A[t/x]$
(A2) $A[t/x] \;\&\; t \downarrow \to \exists x A$
$\downarrow$ is a unary (post-fix) relation symbol in the language
$\simeq$ is a defined binary relation symbol

$$t \simeq s \equiv t \downarrow \vee s \downarrow \to \quad t = s.$$

We have the following axioms governing $\downarrow$, $\simeq$ *and* $=$

(A3) $x = x \;\&\; (x = y \to y = x)$
(A4) $t \simeq s \;\&\; \varphi(t) \to \varphi(s)$
(A5) $t = s \to t \downarrow \;\&\; s \downarrow$
(A6) $R(t_1, ..., t_n) \to t_1 \downarrow \;\&\; ... \;\&\; tn \downarrow$ for any atomic formula $R(t_1, ..., t_n)$ and any terms $t_1, ..., t_n$.
(A7) (i) For each constant symbol $c : c \downarrow$
(A8) (ii) For each variable $x : x \downarrow$
We note that (A5) is a special case of (A6). Another important special case of (A6) is
(A6') $f(t_1, t_2, ..., t_n) \downarrow \to t_1 \downarrow \;\&\; t_2 \downarrow \;\&\; ... \;\&\; t_n \downarrow$
N.B. (A6) does NOT imply that for any formula $\varphi, \varphi(t_1, ..., t_n) \to \quad t_1 \downarrow \;\&\; ... \;\&\; t_n \downarrow$. Consider,e.g. $\neg t \downarrow \to t \downarrow$.

PCA : We now introduce the theory PCA over the logic of partial terms.
Language: Two constants, **k** and **s**.
A binary function symbol **Ap**
We will never explicitly write **Ap**. We use juxtaposition, (st) , or just st , to denote $Ap(s, t)$.
Axioms of PCA:
Those of LPT together with
(PCA1) $\mathbf{k}xy = x$
(PCA2) $\mathbf{s}xyz \simeq xz(yz) \;\&\; \mathbf{s}xy \downarrow$
(PCA3) $\mathbf{k} \neq \mathbf{s}$
We now define **EON**, Beeson's *Elementary Theory of Operations and Numbers* ([4]). It is **PCA** together with
constants $\pi_0, \pi_1, \mathbf{p}, \mathbf{d}, S_N, P_N$, 0,and a predicate letter N, with axioms

EON1 $\mathbf{pxy} \downarrow \;\&\; \pi_0(\mathbf{p}xy) = x \;\&\; \pi_1(\mathbf{p}xy) = y$

EON2 $N(0)\ \&\ \forall x(N(x) \to [N(S_N(x))\ \&\ P_N(S_N x) = x\ \&\ S_N x \neq 0])$

EON3 $\forall x(N(x)\ \&\ x \neq 0 \to N(P_N x)\ \&\ S_N(P_N x) = x)$

EON4 Definition by integer cases

$$N(a)\ \&\ N(b)\ \&\ a = b \to \quad d(a,b,x,y) = x$$
$$N(a)\ \&\ N(b)\ \&\ a \neq b \to \quad d(a,b,x,y) = y$$

EON5 Induction schema: for each formula φ

$$\varphi(0)\ \&\ \forall x[N(x)\ \&\ \varphi(x) \to \quad \varphi(S_N x)] \to \forall x(N(x) \to \quad \varphi(x)).$$

We will often write
if $a = b$ **then** x **else** y for $d(a,b,x,y)$
$\langle x, y\rangle$ *for* pxy.

Proofs of the following results about **EON** (and related systems) can be found in Beeson's book [4] or [21].

Theorem B.1 (The Recursion Theorem) *There is a term R such that PCA proves*

$$Rf\downarrow\ \&\ [g = Rf \to \forall x(gx \simeq fgx)]$$

Theorem B.2 *Let M be a model of* **EON**. *Then every partial recursive function is numerically representable in M.*

Theorem B.3 (Numerical and Term Existence properties) *If*

$$\mathbf{EON} \vdash \exists x A$$

then there is a term t such that

$$\mathbf{EON} \vdash t\downarrow\ \&\ A(t).$$

If **EON** $\vdash \exists n(N(n)\ \&\ A(n))$ *there is a numeral* $\bar{m} = \underbrace{s(s...s(}_{m}0)...)$ *such that* **EON** $\vdash A(\bar{m})$.

Let $\underline{C} = \{c_i | i \in \omega\}$ be a denumerable set of fresh constants. **EON**$\underline{C}$ is the theory **EON**, together with the constants in $\underline{C}$, the axioms $c\downarrow$ for each $c \in \underline{C}$ and all schemas extended to the new language. This means, e.g. we have the following new instances of the axioms:

$$\forall x A\ \&\ c\downarrow \to \quad A[c/x]$$
$$t \simeq c\ \&\ \varphi(t) \to \quad \varphi(c)$$

etc. However, no $c \in \underline{C}$ occurs in a non-logical axion of **EON**$\underline{C}$.

The following lemma is essentially a restatement of the so-called *combinatory completeness* property of the lambda calculus:

Lemma B.4 (The a-lemma) *Let a be a constant in $\underline{C}$ and define Δ/a to be the set of all terms of Δ in which a does not occur.*

Then $e \in \Delta \Rightarrow \exists h \in \Delta/a$ such that

$$\mathbf{EON}\underline{C} \vdash ha \simeq e.$$

Proof: By induction on the structure of terms in Δ. First we distinguish a special case:

if a does not occur in e, let $h = \mathbf{k}e$

if a does occur in e :

case 1: e is a. Then h is $\lambda x \cdot x \equiv \mathbf{skk}$.

case 2: e is (tu) and by the induction hypothesis, there are terms $t', u' \in \Delta/a$ such that $t \simeq t'a$ and $u \simeq u'a$ are provable in $\mathbf{EON}\underline{C}$. Hence $(tu) \simeq (t'a)(u'a)$ so $h \simeq (\mathbf{s}t'u')$ ∎

Note: The notation $\Delta/\{c_1, ..., c_n\}$ will mean all terms $t \in \Delta$ in which none of $c_1, ..., c_n$ occur.

Lemma B.5 (Generalization on Constants) *Suppose T is any theory extending LPT, the Logic of Partial Terms, and a is a constant only occurring in the logical part of T. Then if*

$$T \vdash \varphi(a) \tag{58}$$

we can conclude

$$T \vdash \forall x \varphi(x).$$

This is an elementary result in standard logic texts, and is included here only because of the slight technical distinction that a necessarily occurs in some of the premisses of T as part of the requirement for extending LPT. All we have to show is that the presence of extra logical axioms does not interfere.

Let T_0 be the finitely many axioms of T occurring in the proof of (58) in which the constant a does not occur, and $\Gamma_0(a)$ the finite conjunction of axioms involving a. By definition of extensions of LPT, the conjuncts $\gamma(a)$ of $\Gamma(a)$ must be of the form (A1) - (A7), e.g. $a \simeq x \rightarrow (\theta(a) \rightarrow \theta(x))$. It suffices to observe that *each one of these conjuncts, $\gamma(a)$ remains a theorem of* $\mathbf{EON}\underline{C}$ *when a variable is substituted for the constant a.* Therefore, we can write (58) as

$$T_0 \,\&\, \Gamma_0(a) \vdash \varphi(a)$$

which implies that

$$T_0 \vdash \Gamma_0(a) \rightarrow \varphi(a)$$

with a no longer present to the left of the turmstile, hence:

$$T_0 \vdash \forall x[\Gamma_0(x) \rightarrow \varphi(x)].$$

But since $T_0 \vdash \forall x(\Gamma_0(x))$ by the observation just made, above, we have

$$T \vdash \forall x \varphi(x).$$

References

[1] Aczel, P. [1977] The Strength of Martin-Löf's Type Theory with One Universe, in: Mietissen, S. and Väänänen, J., (eds), *The proceedings of the symposiums on Mathematical Logic iulu 1974 and Helsinki 1975*, University of Helsinki, 1977.

[2] Allen, Stuart [1988] Ph. D. Dissertation, Cornell University, Ithaca, N.Y.

[3] Allen, Stuart [1987] "A Non-Type Theoretic Definition of Martin-Löf's Types", Cornell University Department of Computer Science Technical Report 87-832, Ithaca, N.Y.

[4] Beeson, M. J. [1985a], *Foundations of Constructive Mathematics*, Springer-Verlag, Berlin.

[5] Chang, C.C., and Keisler, J. [1977], *Model Theory*, North Holland, Amsterdam.

[6] Constable, R. L., et al [1986], *Implementing Mathematics with the NUPRL Development System*, Prentice-Hall, N.J.

[7] Constable, R. L. [1986], "The Semantics of Evidence", Cornell CS Technical Report,.Ithaca, N.Y.

[8] Constable, R. L. [1983], "Programs as proofs", Information Processing Letters, 16(3), 105-112.

[9] Coquand, T. [1990], "On the analogy between propositions and types", in: *Logic Foundations of Functional Programming*, Addison-Wesley, Reading, MA.

[10] Feferman, S. [1975], "A language and axioms for explicit mathematics", in: *Algebra and Logic*, Lecture Notes in Mathematics No. 450, pp. 87-139, Springer, Berlin.

[11] Grayson, R. J. [1983], "Forcing in intuitionistic systems without power set", Journal of Symbolic Logic 48, 670-682.

[12] Howard, W. A. [1980], "The Formulae-as-types notion of construction", in: Seldin, J.P. and J. R. Hindley (eds.), *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, 479-490.

[13] Hyland, J. M. E. [1982], "The effective topos". in: Troelstra, A. S. and D. S. van Dalen (eds.), *L.E.J. Brouwer Centenary Symposium*, North-Holland, Amsterdam.

[14] Kleene, S. C. [1952], *Introduction to Metamathematics*, North-Holland (1971 edition), Amsterdam.

[15] Lipton, [1990], "Relating Kripke Models and Realizability", Ph. D. dissertation, Cornell University.

[16] Lipton, [1990] "Realizability and Kripke Forcing", to appear in the *Annals of Mathematics and Artificial Intelligence*, Vol.4, North-Holland, Amsterdam.

[17] Lipton, [1990], "Constructive Kripke Semantics and Realizability", to appear in the proceedings of the *Logic for Computer Science* conference held at the Math. Sci. Research Institute, Berkeley, Nov. 1989.

[18] Martin-Lōf, P. [1982], "Constructive Mathematics and Computer Programming", in Logic, Methodology and Philosophy of Science IV, North Holland, Amsterdam.

[19] Martin-Lōf, P. [1984], *Intuitionistic Type Theory*, Studies in Proof Theory Lecture Notes, BIBLIOPOLIS, Napoli, Italy.

[20] Mitchell, J. and E. Moggi [1987], "Kripke-style models for typed lambda calculus", *Proceedings from Symposium on Logic in Computer Science*, Cornell University, June 1987, IEEE, Washington, D.C..

[21] Troelstra, A. S. and D. van Dalen [1988], *Constructivism in Mathematics: An Introduction*, Vol. II, Studies in Logic and the Foundations of Mathematics, Vol. 123, North-Holland, Amsterdam.