Kripke-style models for typed lambda calculus

John C. Mitchell
Department of Computer Science
Stanford University

Eugenio Moggi
Department of Computer Science
University of Edinburgh

November 13, 1996

Abstract

The semantics of typed lambda calculus is usually described using Henkin models, consisting of functions over some collection of sets, or *concrete* cartesian closed categories, which are essentially equivalent. We describe a more general class of Kripke-style models. In categorical terms, our Kripke lambda models are cartesian closed subcategories of the presheaves over a poset. To those familiar with Kripke models of modal or intuitionistic logics, Kripke lambda models are likely to seem adequately "semantic." However, when viewed as cartesian closed categories, they do not have the property variously referred to as concreteness, well-pointed-ness, or having enough points. While the traditional lambda calculus proof system is not complete for Henkin models that may have empty types, we prove strong completeness for Kripke models. In fact, every set of equations that is closed under implication is the theory of a single Kripke model. We also develop some properties of logical relations over Kripke structures, showing that every theory is the theory of a model determined by a Kripke equivalence relation over a Henkin model. We discuss cartesian closed categories but present the main definitions and results without the use of category theory.

1 Introduction

Lambda calculus is a calculus of functions: we read the lambda term $\lambda x\colon\sigma.M$ as, "the function defined by treating the expression M as a function of the variable x," where $x\colon\sigma$ indicates that the domain of this function is type σ. Formalizing this reading, it is natural to base a mathematical semantics of typed lambda calculus on sets of functions. When terms are given functional types in the usual way, it is easy to see how each term defines a set-theoretic function with the appropriate domain and range. However, classical set theory[1] entails some subtle semantic properties which are slightly at odds with the traditional axiom system.

One way to see the influence of classical principles on the semantic properties of terms is to consider an implication presented in [MMMS87]. Suppose a and b are types and f is a function $f\colon(a\to a\to a)\to b$ mapping curried two-argument functions on a into b. Since the two functions

$$\begin{array}{rcl} \pi_1 & ::= & \lambda x\colon a.\lambda y\colon a.x \\ \pi_2 & ::= & \lambda x\colon a.\lambda y\colon a.y \end{array}$$

both have type $a\to a\to a$, we can apply f to either one. We will show, by a simple case analysis, that

$$(*) \qquad \lambda x\colon a.f\pi_1 = \lambda x\colon a.f\pi_2 \text{ implies } f\pi_1 = f\pi_2.$$

In words, we will show that if the two functions $\lambda x\colon a.f\pi_1$ and $\lambda x\colon a.f\pi_2$ are equal, then their values $f\pi_1$ and $f\pi_2$ must be equal. The first case to consider is when the type a has no elements. If this happens, then we have $\pi_1 = \pi_2$, since both expressions denote the empty function, and so it must be that $f\pi_1 = f\pi_2$. The second case is a not empty, and so there is some element u of type a. We can apply both functions in the antecedent to u, and since "equals applied to equals produce equals," we obtain $f\pi_1 = f\pi_2$. Thus regardless of whether a is empty, the implication above is semantically sound.

Although the traditional axiom system is complete for proving equations that are valid in all Henkin models, there is a slightly complicated relationship between the axiom system and semantic implication. In the traditional axiom system, there is no provision for reasoning by cases, and so the argument

[1] We use the term *set theory* to mean any *classical set theory*, such as *ZF*. This should be distinguished from set theories developed in intuitionistic logic. As will become apparent, our Kripke lambda models are essentially a semantics of typed lambda calculus developed in a form of intuitionistic set theory.

above cannot be formalized. However, it has been common to assume that no type is empty. When we make this simplifying assumption, we eliminate one case, and the inference is easily carried out within the appropriate axiom system. This leads to the completeness theorems of [Fri75, Hen50, Sta85a]. The drawback, however, is that in many computer science applications it is not appropriate to assume every type is nonempty (inhabited). This point is discussed in [MMMS87]. Related discussions of multi-sorted equational logic appear in [GM82, GM86]. When we reject the nonemptiness assumption and allow types to be empty, we are led to non-equational principles, as in [MMMS87], which formalize reasoning by cases as above. The extended axiom system of [MMMS87] is semantically complete, but it has a very different flavor from the traditional system. For example, some useful connections between equational theories and β, η-conversion fail[2]. In addition, we give up the usual "minimal model" property of lambda calculus and equational logic: with empty types allowed, there is a set of equations which is closed under semantic implication, but not the theory of any single model.

It is worth noting that these comments hold true whether we describe models using the language of set theory or category theory. As proposed in [Sco80, Section 2], one might choose *concrete* cartesian closed categories, those satisfying

$$f = g : A \to B \ \text{ iff } \ f \circ a = g \circ a \text{ all } a\colon 1 \to A,$$

as "models." (This use of the word *concrete* has been adopted in much of the computer science literature, but it is not standard in category theory. An alternate phrase is to say that 1 *is a generator* or the category *has enough points.*) Since any object A of a concrete category may be identified with the set of arrows $a\colon 1 \to A$ from the terminal object, our discussion of Henkin models above applies to concrete cartesian closed categories as well. In particular, the traditional inference system is not complete for semantic implication over concrete cartesian closed categories when "empty" objects are allowed.

The goal of the present paper is to give a natural set-like semantic account of the traditional inference system. To do this, we must find a semantics which does not support the classically valid justification of $(*)$. Since the argument assumes that either a is empty or a is not empty, the law of the

[2]Here is one example, based on the results of [Jac75]. We write $M \overset{U,V}{\longleftrightarrow} N$ if there is a term P with $PUV =_{\beta,\eta} M$ and $PVU =_{\beta,\eta} M$. Then in the traditional inference system for typed lambda calculus, we have $U = V \vdash M = N$ iff $M \overset{U,V}{\longleftrightarrow} \ldots \overset{U,V}{\longleftrightarrow} N$. This fails when the new inference rules for empty types are added.

excluded middle is used in a critical way. Thus one might expect an intuitionistic semantics to provide a completeness theorem. Since categorical logic is essentially intuitionistic, the equivalence between typed lambda theories (defined using the traditional axiom system) and arbitrary cartesian closed categories could be considered an intuitionistic completeness theorem (see, e.g. , [Fou77, Lam80, LS86]). However, we prefer the completeness theorem using only Kripke models for several reasons. For one, Kripke models are relatively easy to picture, and they seem to support a set-like intuition about the lambda terms better than arbitrary cartesian closed categories. In addition, predicate logic may be interpreted over Kripke lambda models, while there is no analogous interpretation in arbitrary cartesian closed categories (except indirectly via the Yoneda embedding). A practical advantage is that it is often easy to devise Kripke counter-models to implications like $(*)$. Finally, the useful techniques of logical relations generalize to Kripke lambda models without much difficulty and provide an easy way to construct Kripke lambda models from Henkin-like structures.

We define Kripke lambda models in Section 2 and describe the axiom system and the semantics of terms in Section 3. In Section 4, we discuss the relationship between Kripke lambda models and cartesian closed categories (CCC's): every Kripke model determines a CCC, and (as pointed out to us by Edmund Robinson and Pino Rosolini) every small CCC may be embedded in a Kripke model. We prove soundness and completeness theorems in Section 5, along with a correspondence between nonempty types and intuitionistically valid propositional formulas. Finally, in Section 6, we turn our attention to logical relations. We describe some general properties of Kripke logical relations, which are closely related to *I-relations* [Plo80], and show that a general class of Kripke models may be obtained as Kripke quotients of classical models. As an application of Kripke quotients, we construct a counter-model to the implication $(*)$ given in the second paragraph of the paper.

While we began our study of Kripke lambda models by working out a definition from first principles, our model definition and many of our results may be developed using a paradigm that is well-known to researchers in categorical logic. We are grateful to Edmund Robinson and Pino Rosolini for some helpful discussion of this point of view, and refer the reader to [Fou77, Sco80, LS86] for related discussion. In short, the usual definition of semantics of typed lambda calculus, as in [Bar84, Fri75, Hen50], may be formalized in the language of set theory: a model is a collection of sets satisfying several properties easily described by logical formulas. While we usually interpret this definition in the "standard classical model" of set theory, other interpretations are possible. In particular, our definition of Kripke lambda model may be viewed as an explicit description of the meaning of *typed lambda model* in a class of Kripke-style interpretations of an intuitionistic set theory. In categorical terms, these interpretations of set theory are functor (or presheaf) categories Set^P, where P is a poset. Since much of the development seems entirely routine from this point of view, we will use the Kripke interpretation of logical formulas to motivate part of the model definition. We should emphasize that in topos theory, a "Kripke model" is just an interpretation of a first-order signature in a presheaf category over a poset, as it will become clear from the discussion in Sections 2 and 4.

2 Kripke lambda models

2.1 Possible worlds

As with other Kripke-style semantics, a Kripke lambda model will include a partially-ordered set W of "possible worlds." Instead of having a set of elements of each type, a Kripke lambda model will have a set of elements of each type at each possible world $w \in W$. The relationship between elements of type σ at worlds w and $w' \geq w$ is that every $a\colon \sigma$ *at* w is associated with some unique $a'\colon \sigma$ *at* w'. Informally, using the common metaphor of $\leq$ as relation in time, this means that every element of σ at w will continue to be an element of σ in every possible $w' \geq w$. As we move from w to a possible future world w', two things might happen: we may acquire more elements, and distinct elements may become identified. These changes may be explained by saying that as time progresses, we may become aware of (or construct) more elements of our universe, and we may come to know more "properties" of elements. In our case, the properties of interest are equations, and so we may have more equations in future worlds. Since a type σ may be empty at some world w and then become nonempty at $w' \geq w$, some types may be neither "globally" empty nor nonempty.

2.2 Applicative structures and predicate logic

Kripke lambda models will be defined precisely using the subsidiary notion of applicative structure. A *Kripke applicative structure*

$$A = \langle W, \leq, \{A_w^\sigma\}, \{App_w^{\sigma,\tau}\}, \{i_{w,w'}^\sigma\}\rangle$$

consists of

- a set W of "possible worlds" partially-ordered by $\leq$,
- a family $\{A^\sigma_w\}$ of sets indexed by types σ and worlds $w \in W$,
- a family $\{App^{\sigma,\tau}_w\}$ of "application maps" $App^{\sigma,\tau}_w \colon A^{\sigma\to\tau}_w \times A^\sigma_w \to A^\tau_w$ indexed by pairs of types σ, τ and worlds $w \in W$,
- a family $\{i^\sigma_{w,w'}\}$ of "transition functions" $i^\sigma_{w,w'} \colon A^\sigma_w \to A^\sigma_{w'}$ indexed by types σ and pairs of worlds $w \leq w'$

subject to the following conditions. We want the transition from A^σ_w to A^σ_w to be the identity

$$(id) \qquad i^\sigma_{w,w} \colon A^\sigma_w \to A^\sigma_w \text{ is the identity,}$$

and other transition functions to compose

$$(comp) \qquad i^\sigma_{w',w''} \circ i^\sigma_{w,w'} = i^\sigma_{w,w''} \quad \text{all } w \leq w' \leq w''$$

so that there is exactly one mapping of A^σ_w into $A^\sigma_{w'}$ given for $w \leq w'$. We also require that application and transition commute in a natural way

$$(nat) \qquad \forall f \in A^{\sigma\to\tau}_w . \forall a \in A^\sigma_w .$$

$$i^\tau_{w,w'}(App^{\sigma,\tau}_w(f,a)) = App^{\sigma,\tau}_{w'}((i^{\sigma\to\tau}_{w,w'} f), (i^\sigma_{w,w'} a)),$$

which may be drawn

$$\begin{tikzcd}
A^{\sigma\to\tau}_{w'} \times A^\sigma_{w'} \arrow[r, "App^{\sigma,\tau}_{w'}"] & A^\tau_{w'} \\
A^{\sigma\to\tau}_{w} \times A^\sigma_{w} \arrow[u, "i^{\sigma\to\tau} \times i^\sigma"] \arrow[r, "App^{\sigma,\tau}_{w}"'] & A^\tau_{w} \arrow[u, "i^\tau"']
\end{tikzcd}$$

and will be described informally below. This completes the definition.

If $a \in A^\sigma_w$ and $w \leq w'$, then we can read $i^\sigma_{w,w'} a \in A^\sigma_{w'}$ as "a viewed at world w'." The purpose of the application map $App^{\sigma,\tau}_w$ is to associate a function $App^{\sigma,\tau}_w(f, \cdot)$ from A^σ_w to A^τ_w with each element $f \in A^{\sigma\to\tau}_w$. Since we can view $f \in A_w^{\sigma\to\tau}$ as an element at any future world $w' \geq w$, the application map at world w' also associates a function with $i_{w,w'}^{\sigma\to\tau} f$ at w'. The condition *(nat)* is intended to give a degree of coherence to the functions associated with different views of f. Basically, *(nat)* says that if we apply f to argument a at world w, and then view the result at a later world $w' \geq w$, then we see the same value as when we view f and a as elements of world w', and apply f to a there.

Kripke applicative structures can also be defined using category-theoretic concepts. The usual definition of applicative structure (also called a *prestructure*; see [Fri75, Sta85a]) may be understood in any cartesian category. The usual definition of applicative structure is a pair $\langle \{A^\sigma\}, \{App^\sigma\}\rangle$, where $\{A^\sigma\}$ is a family of sets A^σ indexed by types and $\{App^\sigma\}$ is a family of application functions

$$App^{\sigma,\tau} \colon A^{\sigma\to\tau} \times A^\sigma \to A^\tau$$

indexed by pairs of types. We may interpret this definition "inside" a category C by regarding the word "set" as meaning "object from C" and "function" as meaning "morphism from C". Thus an applicative structure in C is a collection of objects $\{C^\sigma\}$ indexed by types and a collection of morphisms $\{App^{\sigma,\tau}\}$ indexed by pairs σ, τ of types such that $App^{\sigma,\tau}$ has the domain and codomain given above. To derive our definition of Kripke applicative structure from this general idea, we regard a poset $\langle W, \leq\rangle$ as a category in the usual way (see Section 4), and consider the category $Set^{\langle W, \leq\rangle}$ of functors from $\langle W, \leq\rangle$ to sets. If we work out what *applicative structure* means in a category of the form $Set^{\langle W, \leq\rangle}$, then "sets" will be functors and "functions" natural transformations. So we end up with exactly the definition of applicative structure spelled out explicitly above. We will say a little more about functors and natural transformations in Section 4; some related details may be found in [Sco80, Section 4] and [LS86, Example 9.5 of Part II].

It is often convenient to omit the application map App, writing fx for $App_w^{\sigma,\tau}(f, x)$ when this does not seem confusing.

It is relatively easy to use Kripke applicative structures to interpret a predicate logic with quantification over all types. A brief discussion of logic at this point will make it easier to motivate the further conditions needed to define Kripke lambda models. We will use the notation

$$w \Vdash \phi\,[\eta]$$

for formula ϕ holding at world w relative to variable assignment (environment) η, and let $A \Vdash \phi$ mean $w \Vdash \phi\,[\eta]$ for every world w and environment η.

Equations between expressions without λ are easily interpreted, and we will see how to interpret equations between typed lambda terms in the next section. If we take equations as atomic formulas, then conjunction, disjunction and existential quantification are straightforward. For example,

$$w \Vdash \phi \wedge \psi\,[\eta] \quad \text{iff} \quad w \Vdash \phi\,[\eta] \text{ and } w \Vdash \psi\,[\eta].$$

Negation is interpreted by taking $\neg\phi \equiv \phi \supset \bot$, where by definition $\bot$ is a formula that does not hold at any w. As in Kripke semantics of propositional or first-order logic, implication and quantification make use of the partial ordering of worlds. For example,

$$w \Vdash \phi \supset \psi\,[\eta] \quad \text{iff} \quad \forall w' \geq w.\ w' \Vdash \phi\,[\eta] \text{ implies } w' \Vdash \psi\,[\eta].$$

For this to make sense, we must be able to regard any variable assignment at w as a variable assignment at $w' \geq w$, a technical detail we will address below. We will illustrate the interpretation of quantification by example below. More information about this interpretation of predicate logic in $Set^{\langle W, \leq \rangle}$, which is entirely standard, may be found in [Sco80, LS86].

2.3 Extensionality and combinators

A classical applicative structure may fail to be a model for two reasons, and these reasons apply to Kripke applicative structures as well. The first possibility is that we may not have enough elements. For example, $\sigma \rightarrow \sigma$ might be empty, making it impossible to give meaning to the identity function $\lambda x\colon \sigma.x$. The second problem is that application may not be extensional, i.e. , we may have two distinct elements of functional type which have the same functional behavior. Consequently, the meaning of a lambda term $\lambda x\colon \sigma.M$ may not be determined uniquely.

The usual statement of extensionality is that $f = g$ whenever $fx = gx$ for all x of the appropriate type. In Kripke applicative structures, we are concerned not only with the behavior of elements $f, g \in A_w^{\sigma \rightarrow \tau}$ as functions from A_w^σ to A_w^τ, but also as functions from $A_{w'}^\sigma$ to $A_{w'}^\tau$ for all $w' \geq w$. Therefore, we must specify that for all $f, g \in A_w^{\sigma \rightarrow \tau}$,

$$\begin{array}{ll} f = g & \text{whenever} \\ & \forall w' \geq w. \forall a \in A_{w'}^\sigma.\ (i_{w,w'}^{\sigma \rightarrow \tau} f)\, a = (i_{w,w'}^{\sigma \rightarrow \tau} g)\, a. \end{array}$$

This can be said a little more simply by appealing to the interpretation of predicate logic described above. Specifically, we will say that a Kripke applicative structure A is *extensional* if

$$(ext) \qquad A \Vdash (\forall x{:}\,\sigma.\ f\ x = g\ x) \supset f = g$$

where the variables f and g have type $\sigma \to \tau$. (We will discuss a syntactic mechanism for specifying the types of free variables in the next section.) It is a routine calculation to see that (ext) is equivalent to the more elaborate statement above with explicit quantification over possible worlds.

There are two common ways of specifying that a classical applicative structure has enough elements to interpret every lambda term. The *environment model condition* uses the inductive definition of the meanings of terms, while the *combinatory model condition* uses equationally-defined elements K and S called combinators (see [Bar84, Chapter 5] or [Mey82]). Since the two are equivalent (for both Kripke and classical applicative structures), we will define models using combinators.

For Kripke applicative structures, the description of K and S is simplified by introducing the notion of global element. A *global element* $a{:}\,\sigma$ of A is a mapping $w \mapsto a_w$ from worlds to elements such that $a_w \in A^\sigma_w$ and, whenever $w \le w'$, we have $i^\sigma_{w,w'} a_w = a_{w'}$. Constant symbols in logical formulas denote global elements; we interpret a constant $a{:}\,\sigma$ at world w as $a_w \in A^\sigma_w$. A Kripke applicative structure A *has combinators* if, for any types ρ, σ, τ there exist global elements K and S of types $\sigma \to \tau \to \sigma$ and $(\rho \to \sigma \to \tau) \to (\rho \to \sigma) \to \rho \to \tau$ such that

$$(K) \qquad A \Vdash K\ x\ y = x$$

$$(S) \qquad A \Vdash S\ x\ y\ z = x\ z\ (y\ z).$$

where we assume variables x, y, z are given the appropriate types, e.g. , $x{:}\,\sigma$ and $y{:}\,\tau$ in (K). In more detail, (K) means that for every world w and all "local" elements $a \in A^\sigma_w, b \in A^\tau_w$, we have $w \Vdash K_w\ a\ b = a$. Condition (S) may be spelled out similarly.

We define a *Kripke lambda model* to be a Kripke applicative structure A which is extensional and has combinators.

3 Terms, Equations and Interpretation

3.1 Syntax

As usual in typed lambda calculus, we will be interested in equations between terms of the same type, but not concerned with equations between types.

Since we wish to allow empty types, we will be explicit about the types assigned to variables (see [MMMS87]). Consequently, terms and their types are defined using the subsidiary notion of type assignment. A *type assignment* Γ is a finite set of formulas $x\colon\tau$, with no x occurring twice in Γ. The formula $x\colon\tau$ may be read "the variable x has type τ." We write $\Gamma, x\colon\sigma$ for the type assignment

$$\Gamma, x\colon\sigma = \Gamma \cup \{x\colon\sigma\},$$

where, in writing this, we assume that x does not appear in Γ. Terms will be written in the form $\Gamma \triangleright M\colon\tau$, which may be read, "$M$ has type τ relative to Γ." Since open terms may define "partial," or "nonglobal elements," there may be some confusion about what it means to use a variable. In contrast to the logic of partial elements of [Fou77], for example, all of our expressions will have existential import. When we write $x\colon\sigma$ in a type assignment, we mean that x is defined, or "exists," and has type σ. The symbol "$\triangleright$" acts as implication with respect to existence, so that $x\colon\sigma \triangleright M\colon\tau$ says, "for all w, if x denotes an element of type σ at world w, then M is defined at w and denotes an element of type τ."

The well-typed terms are defined as follows.

$$(var) \qquad x\colon\tau \triangleright x\colon\tau$$

$$(\rightarrow E) \qquad \frac{\Gamma \triangleright M\colon\sigma\rightarrow\tau,\ \ \Gamma \triangleright N\colon\sigma}{\Gamma \triangleright MN\colon\tau}$$

$$(\rightarrow I) \qquad \frac{\Gamma, x\colon\sigma \triangleright M\colon\tau}{\Gamma \triangleright \lambda x\colon\sigma.M\colon\sigma\rightarrow\tau}$$

$$(add\ \ var) \qquad \frac{\Gamma \triangleright M\colon\tau}{\Gamma, x\colon\sigma \triangleright M\colon\tau}$$

An easy induction shows that if $\Gamma \triangleright M\colon\sigma$ is well-typed, then Γ must mention every free variable of M.

It is convenient to omit the empty type assignment when writing closed terms. In addition, since the type of a closed term is uniquely determined, we sometimes omit the type as well. For example, it is convenient to write $\lambda x\colon\sigma.x$ instead of $\emptyset \triangleright \lambda x\colon\sigma.x\colon\sigma\rightarrow\sigma$.

With type assignments as part of the syntactic formulation of terms, it is natural to write equations in the form

$$\Gamma \triangleright M = N : \tau$$

where we assume that $\Gamma \triangleright M\colon\tau$ and $\Gamma \triangleright N\colon\tau$ are both well-typed. For typographical reasons, it is sometimes helpful to leave off the types of the terms, writing $\Gamma \triangleright M = N$ instead of $\Gamma \triangleright M = N : \tau$. We will write $[N/x]M$ for the result of substituting N for free occurrence of x in M. In defining $[N/x]M$, we must be careful to rename bound variables in M to avoid capture, as usual.

We have the usual axioms for renaming bound variables, evaluating function application by substitution, and equating extensionally equal functions.

$$(\alpha) \qquad \Gamma \triangleright \lambda x\colon\sigma.M = \lambda y\colon\sigma[y/x]M, \mathring{f}or\ y \notin FV(M)$$

$$(\beta) \qquad \Gamma \triangleright (\lambda x\colon\sigma.M)N = [N/x]M$$

$$(\eta) \qquad \Gamma \triangleright \lambda x\colon\sigma.Mx = M, \mathring{f}or\ x \notin FV(M)$$

We also need a reflexivity axiom

$$(\mathit{ref}) \qquad \Gamma \triangleright M = M : \sigma$$

and several inference rules. The main inference rules are symmetry and transitivity

$$(\mathit{sym}) \qquad \frac{\Gamma \triangleright M = N : \sigma}{\Gamma \triangleright N = M : \sigma}$$

$$(\mathit{trans}) \qquad \frac{\Gamma \triangleright M = N : \sigma, \Gamma \triangleright N = P : \sigma}{\Gamma \triangleright M = P : \sigma}$$

as well as congruence with respect to application and lambda abstraction

$$(\mathit{cong}) \qquad \frac{\Gamma \triangleright M_1 = M_2 : \sigma\to\tau,\ \Gamma \triangleright N_1 = N_2 : \sigma}{\Gamma \triangleright M_1N_1 = M_2N_2 : \tau}$$

$$(\xi) \qquad \frac{\Gamma, x\colon\sigma \triangleright M = N : \tau}{\Gamma \triangleright \lambda x\colon\sigma.M = \lambda x\colon\sigma.N : \sigma\to\tau}.$$

Since type assignments are explicitly included in equations, we also need the rule

$$(\mathit{add\ var}) \qquad \frac{\Gamma \triangleright M = N : \sigma}{\Gamma, x\colon\tau \triangleright M = N : \sigma}$$

This lets us add additional typing hypotheses to equations. We write $\mathcal{E} \vdash \Gamma \triangleright M = N : \sigma$ if the equation $\Gamma \triangleright M = N : \sigma$ is provable from the equations in $\mathcal{E}$.

A useful fact about typing and equational reasoning is that if $\Gamma \triangleright M\colon\sigma$ is well-typed, and M β,η-reduces to N, then $\Gamma \triangleright N\colon\sigma$ is also well-typed. The converse fails, however, since when M β,η-reduces to N, the term M may have more free variables. Therefore, $\Gamma \triangleright N\colon\sigma$ does not imply $\Gamma \triangleright M\colon\sigma$.

To emphasize the difference between our proof system and the proof rules that apply when types are assumed not empty, it is worth mentioning that we do *not* have the rule

$$(\textit{nonempty}) \qquad \frac{\Gamma, x\colon\sigma \triangleright M = N : \tau}{\Gamma \triangleright M = N : \tau} \; x \notin FV(M,N)$$

since this inference is sound only if there exists a global element of type σ.

It is interesting to observe that we have a Kripke-like structure within the syntax of terms or equations. We may think of a type assignment Γ as the "possible world" in which the variables appearing in Γ "exist," in the sense of [Fou77], or "are defined." We may then read $\Gamma \triangleright M\colon\sigma$ as saying, "M exists and has type σ at world Γ." The natural ordering on type assignments is containment, and rule *(add var)* ensures that if M is defined and has type σ at world Γ, then M "continues" to be a term of type σ at every world $\Gamma' \geq \Gamma$. We can also incorporate equations, and read $\Gamma \triangleright M = N : \tau$ as, "M and N define the same element of type τ at world Γ." Since more terms can be defined when we have more variables, it is clear that any $\Gamma' \geq \Gamma$ will have at least the elements of Γ. What is perhaps less obvious is that with respect to certain lambda theories, we may have more equations at $\Gamma' \geq \Gamma$. To take a simple example, suppose we have a constant $c\colon\sigma\to\sigma$, and let $\mathcal{E}$ be the single equation $\{\lambda x\colon\tau.c = \lambda x\colon\tau.\lambda y\colon\sigma.y\}$. At world $\Gamma = \emptyset$, we cannot prove $c = \lambda y\colon\sigma.y$ from $\mathcal{E}$. (This is most easily demonstrated by a semantic argument, as in Section 6.4.) However, it is easy to see that at world $\Gamma' = \{z\colon\tau\} \geq \Gamma$, we have $\mathcal{E} \vdash \Gamma' \triangleright c = \lambda y\colon\sigma.y$. Thus the properties of the transition functions $i^{\sigma}_{w,w'}$ are well-motivated by properties of the proof system for typed lambda calculus. We will use type assignments as "possible worlds" in proving the completeness theorem.

3.2 Environments and meanings of terms

An *environment* η for a Kripke applicative structure A is a partial mapping from variables and worlds to elements of A such that

$$(\textit{env}) \qquad \text{If } \eta x w \in A^{\sigma}_{w} \text{ and } w' \geq w, \text{ then} \quad \eta x w' = i^{\sigma}_{w,w'}(\eta x w).$$

Intuitively, an environment η maps a variable x to a "partial element" ηx which may exist (or be defined) at some worlds, but not necessarily all worlds. Since a type may be empty at one world and then nonempty later, we need to have environments such that $\eta x w$ is undefined at some w, and then "becomes" defined at a later $w' \geq w$. We will return to this point after defining the meanings of terms.

If η is an environment and $a \in A_w^\sigma$, we write $\eta[a/x]$ for the environment identical to η on variables other than x, and with

$$(\eta[a/x])xw' = i^{\sigma}_{w,w'}a$$

for all $w' \geq w$. We take $(\eta[a/x])xw'$ to be undefined for w' not $\geq w$.

If η is an environment for applicative structure A, and Γ is a type assignment, we say w *satisfies* Γ *at* η, written $w \Vdash \Gamma\,[\eta]$ if

$$\eta x w \in A_w^\sigma \text{ for all } x{:}\,\sigma \in \Gamma.$$

Note that if $w \Vdash \Gamma\,[\eta]$ and $w' \geq w$, then $w' \Vdash \Gamma\,[\eta]$.

For any Kripke model A and environment $w \Vdash \Gamma\,[\eta]$, we define the meaning $[\![\Gamma \triangleright M{:}\,\sigma]\!]\eta w$ of term $\Gamma \triangleright M{:}\,\sigma$ in environment η at world w by induction on the structure of terms.

$$\begin{array}{l}
[\![\Gamma \triangleright x{:}\,\sigma]\!]\eta w = \eta x w \\
[\![\Gamma \triangleright MN{:}\,\tau]\!]\eta w = \\
\qquad App_w^{\sigma,\tau}\ ([\![\Gamma \triangleright M{:}\,\sigma \to \tau]\!]\eta w)\ ([\![\Gamma \triangleright N{:}\,\sigma]\!]\eta w) \\
[\![\Gamma \triangleright \lambda x{:}\,\sigma.M{:}\,\sigma \to \tau]\!]\eta w = \text{the unique } d \in A_w^{\sigma\to\tau} \\
\qquad \text{such that for all } a \in A_{w'}^\sigma \text{ and } w' \geq w, \\
\qquad App_{w'}^{\sigma,\tau}\,(i^{\sigma\to\tau}_{w,w'}d)a = [\![\Gamma, x{:}\,\sigma \triangleright M{:}\,\tau]\!]\eta[a/x]\ w'
\end{array}$$

Combinators and extensionality guarantee that in the $\Gamma \triangleright \lambda x{:}\,\sigma.M{:}\,\sigma\to\tau$ case, d exists and is unique. This is proved as in the classical setting, using translation into combinators [Bar84, HS86, Mey82] for existence, and extensionality for uniqueness.

We can see the importance of partial environments by looking at the lambda abstraction case in a little more detail. The meaning of a lambda abstraction in environment η at w is determined by "patched" environments $\eta[a/x]$ for $a \in A_{w'}^\sigma$ with $w' \geq w$. If A_w^σ is empty, but there exist many $a \in A_{w'}^\sigma$, then $A_{w'}^{\sigma\to\tau}$ may be large, and so there are many possible meanings for $\lambda x{:}\,\sigma.M$. However, every $\eta[a/x]$ with $a \in A_{w'}^\sigma$ must be partial, since there is no possible value for x at w. Therefore, we need partial environments to determine the meaning of a lambda term uniquely.

We say an equation $\Gamma \triangleright M = N : \sigma$ *holds at w and η*, written

$$w \Vdash (\Gamma \triangleright M = N : \sigma)\ [\eta]$$

if, whenever $w \Vdash \Gamma\,[\eta]$, we have

$$[\![\Gamma \triangleright M \colon \sigma]\!]\eta w = [\![\Gamma \triangleright N \colon \sigma]\!]\eta w.$$

This is the base case of the inductive definition of $w \Vdash \phi\,[\eta]$ for formula ϕ of predicate logic, given earlier. It is an easy exercise, which we leave to the interested reader, to work out the complete definition of $w \Vdash \phi\,[\eta]$ for logical formulas written using type assignments (see [LS86, Sco80] for significant hints).

A model $\mathcal{A}$ *satisfies* $\Gamma \triangleright M = N : \sigma$, written $\mathcal{A} \Vdash \Gamma \triangleright M = N : \sigma$, if every w and η for $\mathcal{A}$ satisfy the equation.

4 Kripke lambda models and cartesian closed categories

It is easy to extend the definitions of Kripke applicative structure and lambda model to include cartesian product types $\sigma \times \tau$ and a terminal type 1 with one element at each world. In this section, we will see that any Kripke model A with products and a terminal type determines a cartesian closed category $\mathcal{C}_A$. As one would hope, the categorical interpretation of a term in $\Gamma \triangleright M \colon \sigma$ in $\mathcal{C}_A$ coincides with the meaning of $\Gamma \triangleright M \colon \sigma$ in A given above. We will also sketch the full and faithfull embedding of any small cartesian closed category into a cartesian closed category determined by a Kripke lambda model. This embedding preserves the cartesian closed structure, but not necessarily "on the nose." Rather than discuss all of the fine points, we will refer to the appropriate literature. The reader who is unfamiliar with category theory may skip to the next section without loss of continuity.

We regard a partially-ordered set $\langle W, \leq \rangle$ as a category in the usual way. Specifically, the objects of this category are the elements of W and there is a unique "less-than-or-equal-to" arrow $\ell_{w,w'}$ from w to w' iff $w \leq w'$. Since a category must have identities and be closed under composition, we let $\ell_{w,w}$ be the identity on w and define composition by

$$\ell_{w',w''} \circ \ell_{w,w'} = \ell_{w,w''}.$$

Given a Kripke applicative structure A, it is easy to see that each type σ determines a functor Φ_σ from $\langle W, \leq \rangle$ to sets. Specifically, we take

$$\begin{array}{lcl} \Phi_\sigma(w) & = & A^\sigma_w \\ \Phi_\sigma(\ell_{w,w'}) & = & i^\sigma_{w,w'} \end{array}$$

and use conditions *(id)* and *(comp)* in the definition of Kripke applicative structure to show that this map is functorial. While it may seem simplest to use functors Φ_σ as objects of $\mathcal{C}_A$, this may identify types in the case where $\sigma \neq \tau$ syntactically, but $A^\sigma_w = A^\tau_w$ happen to be the same set. Since we would not necessarily want to identify application functions on the two types, this could lead to unnecessary confusion. Therefore, we will use the type expressions as the objects of $\mathcal{C}_A$.

Since each type determines a functor, we will use natural transformations as the morphisms of $\mathcal{C}_A$. For every pair of types σ and τ, condition *(nat)* in the definition of Kripke applicative structure says that the map $w \mapsto App^{\sigma,\tau}_w$, which we shall write simply as $App^{\sigma,\tau}$, is a natural transformation from $\Phi_{\sigma\rightarrow\tau} \times \Phi_\sigma$ to Φ_τ. Using $App^{\sigma,\tau}$, we can see that every global element a of type $\sigma\rightarrow\tau$ induces a natural transformation ν from Φ_σ to Φ_τ, namely

$$\nu_w \quad = \quad App^{\sigma,\tau}_w(a_w, \cdot)$$

For extensional applicative structures (and hence models), it is easy to see that if two global elements a and b determine the same natural transformation, then $a_w = b_w$ at every world w. We let the morphisms from σ to τ in $\mathcal{C}_A$ be all natural transformations $\nu : \Phi_\sigma \rightarrow \Phi_\tau$ induced by global elements of A of type $\sigma\rightarrow\tau$ and let composition of morphisms be ordinary composition of natural transformations in $Set^{\langle W, \leq \rangle}$.

A routine calculation shows that if A is a Kripke lambda model, then $\mathcal{C}_A$ is a category with an object for each type, and there is a one-one correspondence between global elements of type $\sigma\rightarrow\tau$ in $\mathcal{A}$ and morphisms from σ to τ in $\mathcal{C}_A$. In addition, it is easy to show that $\mathcal{C}_A$ is cartesian closed if A has products and a terminal object. The relationship between the categorical interpretation of terms, as in [Sco80][3], and the meaning function we have given is summarized

[3] There is a minor source of confusion in [Sco80, page 413]. In assigning an arrow of a category to an open term M, we must decide which variables to consider free in M. In particular, we may want to consider some variables "vacuously" free. Scott's slightly informal discussion does not address this point. However, in the formalism of the present paper, we have explicit type assignments, and so we simply treat all variables in Γ as occurring free in $\Gamma \triangleright M{:}\sigma$.

in the following theorem. Note that with product types, any $\Gamma \triangleright M\colon \tau$ is easily transformed into a semantically equivalent $x\colon \sigma \triangleright M'\colon \tau$ with only one free variable. (Simply replace the collection of variables in Γ by a single variable of the appropriate product type.)

Theorem 4.1 *If A is a Kripke lambda model with products and terminal type, then the interpretation of $x\colon \sigma \triangleright M\colon \tau$ in $\mathcal{C}_A$, as defined in [Sco80], is the natural transformation from Φ_σ to Φ_τ induced by the global element $w \mapsto [\![\lambda x\colon \sigma.M\colon \sigma \to \tau]\!]\emptyset\, w$, where $\emptyset$ is the empty environment.*

Therefore an equation holds in A iff it holds in $\mathcal{C}_A$. It should be pointed out that the functor from $\mathcal{C}_A$ to $Set^{\langle W, \leq \rangle}$, mapping τ to Φ_τ, is faithful and preserves products, but it may not be full or may not preserve function spaces. The reason is that a Kripke lambda model is just a first order structure in a topos of presheaves. Therefore, the interpretation, $\Phi_{\sigma\to\tau}$ of $\sigma\to\tau$ need not be $\Phi_\sigma \to \Phi_\tau$; extensionality only requires that $\Phi_{\sigma\to\tau}$ be a "subfunctor" of $\Phi_\sigma \to \Phi_\tau$.

We now sketch a method for defining a Kripke lambda model from any small cartesian closed category. More specifically, we assume we are given an association of type constants to objects and term constants to arrows of a small cartesian closed category D. Such an association determines an interpretation of typed lambda calculus in D, in the sense of [Sco80]. We will show that there exists a Kripke lambda model B satisfying the same equations as D. In the special case that our categorical interpretation of typed lambda calculus is the internal language of D (see [LS86]), this construction gives us a Kripke lambda model B which is equivalent to D (in the usual categorical sense), but not necessarily isomorphic.

There are three steps from a small CCC to a Kripke lambda model. The first step transforms our categorical interpretation in D into a categorical interpretation in $Set^{D^{op}}$, the topos of presheaves over D. The second takes any categorical interpretation in a topos of presheaves $Set^{D^{op}}$ and produces an applicative structure in the same category. This applicative structure satisfies (K), (S) and extensionality, which are all first-order expressible. The third step finds an elementarily equivalent applicative structure in Set^W, where W is a poset with a least element.

The first step uses the Yoneda embedding Y_D of D into the topos of presheaves over D. This produces a categorical interpretation in the topos of presheaves over D, as spelled out in [Sco80]. For instance, if d is the interpretation of the base type σ in D, then we use $Y_D(d)$ as the interpretation of σ in the topos of presheaves, and similarly for the interpretations of constants.

This extends uniquely to all type expressions and terms. Since Y_D preserves the cartesian closed structure, the interpretation of any type and term in the topos of presheaves is the image (via Y_D) of its interpretation in D.

The categorical interpretation in the topos of presheaves over D gives us an applicative structure A in the same topos. Specifically, A is the applicative structure with the type σ interpreted as a functor Φ_σ from D^{op} to Set, and $App^{\sigma,\tau}$ as the evaluation morphism $eval_{\Phi_\sigma,\Phi_\tau}$ from $(\Phi_\sigma \to \Phi_\tau) \times \Phi_\sigma$ to Φ_τ. (Here we have $\Phi_{\sigma\to\tau} \cong \Phi_\sigma \to \Phi_\tau$, as sets.) Moreover, the applicative structure A satisfies the axioms (K), (S) and the extensionality condition *(ext)*, according to the Kripke-Joyal semantics of formulas in $Set^{D^{op}}$ (see [LS86]).

We now have a lambda model $\mathcal{A}$ in $Set^{D^{op}}$, but not necessarily a *Kripke* lambda model since D^{op} may not be a poset. The third step of the construction uses the Diaconescu cover. The general construction given in Example 2.8 and Corollary 3.3 of [Joh80] produces a poset W and a functor $d\colon W \to D^{op}$ such that any applicative structure A in $Set^{D^{op}}$ is elementarily equivalent to an applicative structure B in Set^W obtained by composing each functor and natural transformation in A with $d\colon W \to D^{op}$. In particular, A and B satisfy the same equations between typed lambda terms. However, it should be pointed out that, even when A is induced by a categorical interpretation of lambda terms (i.e. , $\Phi_{\sigma\to\tau} \cong \Phi_\sigma \to \Phi_\tau$), it does not follow that B is also induced by such an interpretation.

In order to prove that D and $\mathcal{C}_B$ are equivalent, we show that D is equivalent to $\mathcal{C}_A$ and $\mathcal{C}_A$ is isomorphic to $\mathcal{C}_B$. However, the latter isomorphism requires a modification to the Diaconescu cover construction using the terminal object of D. To produce $\mathcal{C}_B$ isomorphic to $\mathcal{C}_A$, we take W in the definition of B to be the poset of finite composable sequences of morphisms in D^{op}, including the empty sequence $\perp$. This set is partially ordered by $w_1 \leq w_2$ iff the sequence w_1 is an initial segment of w_2. This poset W is P^{op} of Example 2.8 in [Joh80], except that we have added the empty sequence. The functor d is defined as in [Joh80], extended to $\perp$ by mapping the empty sequence to the terminal object 1 of D. Note that there is exactly one way to extend d to morphisms from $\perp$ to any $w \in W$, because 1 is the initial object in D^{op}. More explicitly, d is the functor from W to D^{op} such that for any sequence $f_1, \ldots, f_n$ of composable maps (object of W), we let

$$d(f_1, \ldots, f_n) = \begin{cases} 1 \text{ if } n = 0 \\ \text{the codomain of } f_n \text{ in } D^{op}\text{, otherwise.} \end{cases}$$

If the sequence w' is w followed by $f_1, \ldots, f_n$, then d maps the unique arrow

$\ell_{w,w'}\colon w \to w'$ of W to

$$d(\ell_{w,w'}) = \begin{cases} \text{the only morphism from 1 to } d(w'), \text{ if } w = \bot \\ \text{the identity on } d(w), \text{ if } n = 0 \\ \text{the composition of } f_1, \ldots, f_n, \text{ otherwise.} \end{cases}$$

Since the modified functor $d\colon W \to D^{op}$ still satisfies the conditions of Corollary 3.3 in [Joh80], the modified B remains elementarily equivalent to A. Moreover, the global element of Φ^A_σ in $Set^{D^{op}}$ are in natural correspondence with the global elements of Φ^B_σ in Set^W. In fact, this construction has the following properties:

$$\begin{array}{rl} Set^W(1, \Phi^B_\sigma) \cong \Phi^B_\sigma(\bot) & \text{by the Yoneda Lemma,} \\ \Phi^B_\sigma(\bot) = \Phi^A_\sigma(1) & \text{because } \Phi^B_\sigma = \Phi^A_\sigma \circ d \text{ and } d(\bot) = 1, \\ Set^{D^{op}}(1, \Phi^A_\sigma) \cong \Phi^A_\sigma(1) & \text{again by the Yoneda Lemma.} \end{array}$$

Therefore the global elements of any type Φ^A_σ in A are in one-to-one correspondence with the global elements of the corresponding type Φ^B_σ in B. Since the two categories $\mathcal{C}_A$ and $\mathcal{C}_B$ have the same set of objects, we may conclude that $\mathcal{C}_A$ and $\mathcal{C}_B$ are isomorphic. Specifically, the morphisms from σ to τ in either category are in one-one correspondence with the global elements of $\Phi_{\sigma\to\tau}$, and we know that the corresponding sets of global elements are in one-to-one correspondence, by the argument above.

The two cartesian closed categories $\mathcal{C}_A$ and D are equivalent via the correspondence that maps an object σ of $\mathcal{C}_A$ to the interpretation $[\![\sigma]\!]$ of σ in D. It is easy to see that $\Phi_\sigma = Y_D([\![\sigma]\!])$, since this is clearly true for base types and preserved at higher types because $\Phi_{\sigma\to\tau} = \Phi_\sigma \to \Phi_\tau = Y_D([\![\sigma\to\tau]\!])$. By definition of $\mathcal{C}_A$ and $\Phi_{\sigma\to\tau} = \Phi_\sigma \to \Phi_\tau$, we have $\mathcal{C}_A(\sigma, \tau) \cong Set^{D^{op}}(\Phi_\sigma, \Phi_\tau)$. Therefore $\mathcal{C}_A(\sigma, \tau)$ is isomorphic to $D([\![\sigma]\!], [\![\tau]\!])$ by

$$\mathcal{C}_A(\sigma, \tau) \cong Set^{D^{op}}(\Phi_\sigma, \Phi_\tau) = Set^{D^{op}}(Y_D([\![\sigma]\!]), Y_D([\![\tau]\!])) \cong D([\![\sigma]\!], [\![\tau]\!]).$$

Thus the cartesian closed categories $\mathcal{C}_A$ and D are equivalent. Since $\mathcal{C}_A$ is isomorphic to $\mathcal{C}_B$, this completes the proof that the original category D and the category $\mathcal{C}_B$ determined by Kripke lambda model B are equivalent.

5 Soundness, Completeness and Inhabitation

Using the relationships between Kripke lambda models and cartesian closed categories described in the last section, the soundness and completeness theorems for Kripke lambda models may be derived from well-known theorems about lambda calculus and cartesian closed categories (see [LS86, Part I]). However, we will give a direct completeness proof since it is quite straightforward and the construction has other uses.

The following lemmas are easily proved by induction on typed lambda terms.

Lemma 5.1 *(Transition Lemma) Let A be a Kripke lambda model and η an environment satisfying Γ at w. Then for every $w' \geq w$, we have*

$$[\![\Gamma \triangleright M\colon \sigma]\!]\eta w' = i^{\sigma}_{w,w'}([\![\Gamma \triangleright M\colon \sigma]\!]\eta w).$$

Lemma 5.2 *(Substitution Lemma) Let A be a Kripke lambda model and η an environment satisfying Γ at w. For any well-typed terms $\Gamma \triangleright N\colon \sigma$ and $\Gamma, x\colon \sigma \triangleright M\colon \tau$, we have*

$$[\![\Gamma \triangleright [N/x]M\colon \tau]\!]\eta w = [\![\Gamma, x\colon \sigma \triangleright M\colon \tau]\!](\eta[[\![\Gamma \triangleright N\colon \sigma]\!]\eta w/x])w$$

It is now easy to prove soundness by induction on equational proofs.

Lemma 5.3 *(Soundness) Let $\mathcal{E}$ be a set of well-typed equations. If $\mathcal{E} \vdash \Gamma \triangleright M = N : \sigma$, then every model satisfying $\mathcal{E}$ also satisfies $\Gamma \triangleright M = N : \sigma$.*

For Kripke lambda models, we prove deductive completeness by showing the stronger property that every theory has a model.

Theorem 5.4 *(Completeness) Let $\mathcal{E}$ be any set of equations closed under $\vdash$. There is a Kripke lambda model A with $A \Vdash \Gamma \triangleright M = N : \sigma$ iff $\Gamma \triangleright M = N : \sigma \in \mathcal{E}$.*

Proof Sketch: The completeness theorem is proved by constructing a term model $A = \langle W, \leq, \{A^{\sigma}_{w}\}, \{App^{\sigma,\tau}_{w}\}, \{i^{\sigma}_{w,w'}\}\rangle$ in the following way.

- W is the poset of finite type assignments Γ ordered by inclusion. In what follows, we will write Γ for an arbitrary element of W.

- A^{σ}_{Γ} is the set of all $[\Gamma \triangleright M\colon \sigma]$, where $\Gamma \triangleright M\colon \sigma$ is well-typed, and

  $$[\Gamma \triangleright M\colon \sigma] = \{\Gamma \triangleright N\colon \sigma \mid \mathcal{E} \vdash \Gamma \triangleright M = N : \sigma\}$$

  is the equivalence class of $\Gamma \triangleright M\colon \sigma$ with respect to $\mathcal{E}$.

- $App^{\sigma,\tau}_{\Gamma}([\Gamma \triangleright M\colon \sigma \to \tau], [\Gamma \triangleright N\colon \sigma]) = [\Gamma \triangleright MN\colon \tau]$

- $i^{\sigma}_{\Gamma,\Gamma'}([\Gamma \triangleright M\colon\sigma]) = [\Gamma' \triangleright M\sigma]$ for $\Gamma \subseteq \Gamma'$

It is easy to check that the definition make sense, and that we have global elements K and S at all appropriate types. For example,

$$K = [\lambda x\colon\sigma.\lambda y\colon\tau.x]$$

The proof of extensionality is a little more interesting. Suppose that $[\Gamma \triangleright M\colon\sigma\to\tau]$ and $[\Gamma \triangleright N\colon\sigma\to\tau]$ have the same functional behavior, i.e. , for all $\Gamma' \geq \Gamma$ and $\Gamma' \triangleright P\colon\sigma$, we have

$$[\Gamma' \triangleright MP\tau] = [\Gamma' \triangleright NP\colon\tau]$$

Then, in particular, for $\Gamma' \equiv \Gamma, x\colon\sigma$ with x not in Γ, we have

$$[\Gamma, x\colon\sigma \triangleright Mx\colon\tau] = [\Gamma, x\colon\sigma \triangleright Nx\colon\tau]$$

and so by rule (ξ) and axiom (η), we have $[\Gamma \triangleright M\colon\sigma\to\tau] = [\Gamma \triangleright N\colon\sigma\to\tau]$. Thus $\mathcal{A}$ is a Kripke lambda model.

It remains to show that $\mathcal{A}$ satisfies precisely the equations belonging to $\mathcal{E}$. We begin by relating the interpretation of a term to its equivalence class. If Γ is any type assignment, we may define an environment η by

$$\eta x\Gamma' = \begin{cases} [\Gamma' \triangleright x\colon\sigma] & \text{if } x\colon\sigma \in \Gamma \subseteq \Gamma' \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

A straightforward induction on terms shows that for any $\Gamma'' \supseteq \Gamma' \supseteq \Gamma$, we have

$$[\![\Gamma' \triangleright M\colon\sigma]\!]\eta\Gamma'' = [\Gamma'' \triangleright M\colon\sigma]$$

In particular, whenever $\mathcal{A}$ satisfies an equation $\Gamma \triangleright M = N : \sigma$, we have $\Gamma \Vdash \Gamma[\eta]$ by construction of η, and so

$$[\Gamma \triangleright M\colon\sigma] = [\Gamma \triangleright N\colon\sigma]$$

Since this reasoning applies to every Γ, every equation satisfied by $\mathcal{A}$ must be provable from $\mathcal{E}$.

While it is possible to show that $\mathcal{A}$ satisfies every equation in $\mathcal{E}$ directly, by similar reasoning, certain complications may be avoided by restricting our attention to closed terms. There is no loss of generality in doing so, since it is easy to prove the closed equation

$$\emptyset \triangleright \lambda x_1\colon\sigma_1.\ldots\lambda x_k\colon\sigma_k.M = \lambda x_1\colon\sigma_1.\ldots\lambda x_k\colon\sigma_k.N$$

from the equation

$$x_1{:}\,\sigma_1, \ldots, x_k{:}\,\sigma_k \triangleright M = N$$

between open terms, and vice versa. For any closed equation $\emptyset \triangleright M = N : \tau \in \mathcal{E}$, we have

$$\mathcal{E} \vdash \Gamma \triangleright M = N : \tau$$

for any Γ, by rule *(add var)*. Therefore, for every world Γ of $\mathcal{A}$, the two equivalence classes $[\Gamma \triangleright M{:}\,\tau]$ and $[\Gamma \triangleright N{:}\,\tau]$ will be identical. Since the meaning of $\emptyset \triangleright M{:}\,\tau$ in any environment η at world Γ will be $[\Gamma \triangleright M{:}\,\tau]$, and similarly for $\emptyset \triangleright N{:}\,\tau$, it follows that $\mathcal{A}$ satisfies $\emptyset \triangleright M = N : \tau$. This proves the theorem. ■

One important property of the Kripke term model we construct in the completeness proof is that A_w^σ is nonempty for all $w \in W$ iff σ is an intuitionistically provable proposition. Our interest in this property stems from a well-known syntactic correspondence between typed lambda calculus and intuitionistic logic, called the *formulas-as-types principle,* or *Curry–Howard isomorphism* [How80]. In this analogy, types correspond to logical formulas and terms correspond to proofs. We read basic types as atomic propositions and read the type $\sigma \rightarrow \tau$ of functions from σ to τ as the formula "σ implies τ." The crucial part of this analogy is that since lambda terms are a notational variant of intuitionistic natural deduction proofs, there is a closed term of type σ iff σ is an intuitionistically provable formula. Based on this syntactic correspondence between terms and proofs, we might expect there to be a semantic interpretation in which the nonempty types correspond to the intuitionistically provable formulas. The term model construction may be used to prove the following correspondence between provability and type inhabitation.

Theorem 5.5 *(Inhabitation) Let Σ be a set of typed constants and $\mathcal{E}$ an equational theory over Σ. There is a Kripke lambda model A for $\mathcal{E}$ with the following property: A_w^σ is nonempty for all $w \in W$ iff the type σ, when viewed as an implicational formula, is intuitionistically provable from the types of constants in Σ.*

This theorem stands in sharp contrast to the correspondence we achieve with classical models. To construct a classical model with only the provable types nonempty, we must begin with each base type nonempty, since no atomic proposition is provable. It is easy to see that if σ and τ each have at most one element, then $\sigma \rightarrow \tau$ has at most one element, and so a straightforward induction shows that our model must have at most one element of each type. Consequently, every well-typed equation will be satisfied.

Another way to connect nonemptiness with provability is to consider classes of models. If we consider the class of full classical type hierarchies, with some base types empty and others not, then the types which are nonempty in every model are the *classical* propositional tautologies [Con85].

6 Kripke Logical Relations

6.1 Relations over applicative structures

Logical relations have proven useful in the study of Henkin lambda models. For example, we may prove the completeness of pure β, η-conversion (without equational hypotheses) for many specific classical models, and characterize the lambda definable elements of certain models using logical relations [Plo80, Sta82, Sta85b, Sta85a]. In [Plo80], Plotkin introduced *I-relations*, which are families of typed relations over a Henkin model, indexed by possible worlds. In this section, we will consider Kripke logical relations, which are the straightforward generalization of I-relations to Kripke lambda models.

In the classical model theory of typed lambda calculus, a logical relation is a family of relations indexed by types which satisfies a condition implying closure under application and lambda abstraction. The generalization to Kripke applicative structures involves indexing relations by both types and possible worlds. We will simplify our presentation by assuming a fixed structure $\langle W, \leq \rangle$ throughout Section 6.

A *Kripke logical relation* over Kripke applicative structures A and B (using the same $\langle W, \leq \rangle$) is a family $R = \{R_w^\sigma\}$ of relations $R_w^\sigma \subseteq A_w^\sigma \times B_w^\sigma$ indexed by types σ and worlds $w \in W$ satisfying the following two conditions. The first is a monotonicity condition for any base type c

$$(mon) \qquad R_w^c(a, b) \text{ implies}$$

$$R_{w'}^c(i_{w,w'}^c a, i_{w,w'}^c b) \text{ for all } w \leq w',$$

which says that when $w \leq w'$, the relation R_w^c is contained in $R_{w'}^c$, modulo the transition functions. The second condition

$$(cmpre) \qquad R_w^{\sigma \rightarrow \tau}(f, g) \text{ iff } \forall w' \geq w. \forall a, b \in A_{w'}^\sigma.$$

$$R_{w'}^\sigma(a, b) \text{ implies } R_{w'}^\tau((i_{w,w'}^{\sigma \rightarrow \tau} f)a, (i_{w,w'}^{\sigma \rightarrow \tau} g)b),$$

called "comprehension", says that relative to the functions available from A and B, the relation $R_w^{\sigma \to \tau}$ contains all functions mapping related arguments to related results. The two lemmas below are proved using the essentially the same arguments as outlined in [Plo80].

Lemma 6.1 *(Monotonicity) Let $R \subseteq A \times B$ be a Kripke logical relation. Then for every type σ and pair of worlds $w \leq w'$, if $R_w^\sigma(a, b)$ then $R_{w'}^\sigma(i_{w,w'}^\sigma a, i_{w,w'}^\sigma b)$.*

We say environments η_a, η_b are *related by R on Γ at w* if $R_w^\tau(\eta_a x w, \eta_b x w)$ for all $x{:}\,\tau$ in Γ.

Lemma 6.2 *(Fundamental Lemma) If $R \subseteq A \times B$ is a Kripke logical relation over models A and B, and environments η_a, η_b are related by R on Γ at w, then for every term $\Gamma \triangleright M{:}\,\sigma$, we have $R_w^\sigma(A[\![\Gamma \triangleright M{:}\,\sigma]\!]\eta_a w, B[\![\Gamma \triangleright M{:}\,\sigma]\!]\eta_b w)$*

As with many of our other definitions, the definition of Kripke logical relation may be derived by interpreting the usual definition in the topos $Set^{\langle W, \leq \rangle}$. The usual definition of logical relation $R \subseteq A \times B$ is a family of relations $R^\sigma \subseteq A^\sigma \times B^\sigma$ such that

$$\text{(usual)} \qquad R^{\sigma \to \tau}(f, g) \text{ iff}$$

$$\forall x \in A^\sigma . \forall y \in B^\sigma .\ R^\sigma(x, y) \supset R^\tau(fx, gy).$$

To re-interpret this condition, we must first say what Kripke relations are. As we have seen, a type, or "Kripke set," is a family $A = \{A_w\}$ of sets indexed by worlds, with transition functions $i_{w,w'}^A$ for each $w \leq w'$. (Or, equivalently, a functor from $\langle W, \leq \rangle$ to sets.) If we have two such Kripke sets $A = \{A_w\}$ and $B = \{B_w\}$, then a *Kripke relation* $R \subseteq A \times B$ is a subset of $A \times B = \{A_w \times B_w\}$, i.e. , a family $R = \{R_w\}$ of relations $R_w \subseteq A_w \times B_w$ such that

$$R_w(a, b) \ \text{ implies } R_{w'}(i_{w,w'}^A a, i_{w,w'}^B b) \text{ for all } w \leq w'.$$

Thus condition *(mon)* is built into the definition of $Set^{\langle W, \leq \rangle}$. If we now say that a Kripke logical relation over A and B should be a family of Kripke relations $R^\sigma \subseteq A^\sigma \times B^\sigma$ satisfying *(usual)* above, then it only remains to check *(compre)*. This is obtained by working out the Kripke (i.e. , $Set^{\langle W, \leq \rangle}$) interpretation of the standard comprehension condition *(usual)*.

6.2 Partial equivalence relations and quotients

A *Kripke logical partial equivalence relation* $R \subseteq A \times A$ is a Kripke logical relation such that each R^σ_w is symmetric and transitive. The name partial equivalence relation comes from the fact that if R^σ_w is symmetric and transitive, then R^σ_w is an equivalence relation on the set of a with $R^\sigma_w(a, a)$. We will abbreviate the cumbersome phrase "Kripke logical partial equivalence relation" to *klper* . The following lemma shows that *klper* 's may be constructed by choosing relations at base types.

Lemma 6.3 *(Partial Equivalence) Let $R \subseteq A \times A$ be a Kripke logical relation. If each R^c_w is symmetric and transitive, for each base type c and world $w \in W$, then every R^σ_w is symmetric and transitive.*

One use of partial equivalence relations is in forming quotient structures. With *partial* equivalence relations, reflexivity fails, and so an element need not have an equivalence class. Therefore, it might be more accurate to call these "partial quotients." If

$$A = \langle W, \leq, \{A^\sigma_w\}, \{App^{\sigma,\tau}_w\}, \{i^\sigma_{w,w'}\}\rangle$$

is a Kripke applicative structure and $R \subseteq A \times A$ is a *klper* , then we define the *quotient applicative structure*

$$A/R = \langle W, \leq, \{A^\sigma_w/R^\sigma_w\}, \{App^{\sigma,\tau}_w/R\}, \{i^\sigma_{w,w'}/R\}\rangle$$

as follows.

- $A^\sigma_w/R^\sigma_w = \{\, [a]_R \mid R^\sigma_w(a, a) \,\}$, where $[a]_R$ is the *equivalence* class $[a]_R = \{\, a' \in A^\sigma_w \mid R^\sigma_w(a, a') \,\}$
- $(App^{\sigma,\tau}_w/R)\ [a]_R\ [b]_R = [App^{\sigma,\tau}_w\ a\ b]_R$
- $(i^\sigma_{w,w'}/R)\ [a]_R = [i^\sigma_{w,w'}\ a]_R$

It is a simple exercise to verify that the quotient structure is well-defined, and a Kripke applicative structure.

Lemma 6.4 *If $R \subseteq A \times A$ is a Kripke partial equivalence relation over Kripke applicative structure A, then A/R satisfies the Kripke extensionality condition (ext).*

An straightforward induction on terms may be used to prove the following "quotient model" theorem.

Lemma 6.5 *If $R \subseteq A \times A$ is a* klper *over Kripke lambda model A, then A/R is a Kripke lambda model such that for every environment η with $R^\tau_w(\eta x w, \eta x w)$ for all $x{:}\,\tau$ in Γ, we have*

$$(A/R)[\![\Gamma \triangleright M{:}\,\sigma]\!]\eta_R w = [\, A[\![\Gamma \triangleright M{:}\,\sigma]\!]\eta w\,]_R,$$

where the environment η_R for A/R is defined by taking $\eta_R x w = [\eta x w]_R$ for all x and w.

In short, the meaning of a term M in the quotient model A/R is the equivalence class of the meaning of M in A. This theorem is an adaptation of the "characterization theorem" of [Mit86], which appears to be the first use of the idea.

6.3 Kripke logical relations over classical applicative structures

We now consider Kripke logical relations over classical applicative structures. The simplest way to do this is to regard classical structures as a special case of Kripke structures. To be specific, let $A = \langle \{A^\sigma\}, \{App^{\sigma,\tau}\} \rangle$ be a classical applicative structure, i.e. , $\{A^\sigma\}$ is a collection of sets indexed by types and $\{App^{\sigma,\tau}\}$ is a collection of application functions. We define the Kripke structure

$$A^W = \langle W, \leq, \{A^\sigma_w\}, \{App^{\sigma,\tau}_w\}, \{i^\sigma_{w,w'}\} \rangle$$

by taking sets $A^\sigma_w = A^\sigma$, application functions $App^{\sigma,\tau}_w = App^{\sigma,\tau}$ and transition functions $i^\sigma_{w,w'} = identity$. It is easy to check that A^W is a Kripke lambda model whenever A is a classical lambda model, and that the meaning of a term M in A^W is essentially the same as the meaning of M in A. In categorical terms, A^W is an applicative structure of constant presheaves.

We say $R = \{R^\sigma_w\}$ is a *Kripke logical relation over classical applicative structures* A and B if R is a Kripke logical relation over A^W and B^W. This amounts to the same thing as Plotkin's definition of *I-relation*, except for the minor difference that we have used applicative structures instead of models (cif. [Plo80]). By Lemma 6.5, we can produce Kripke models by taking Kripke quotients of classical models.

Corollary 6.6 *If A is a classical typed lambda model and R is a* klper *over A, then A/R is a Kripke lambda model.*

This gives us a fairly simple class of models with intuitionistic properties. In fact, we can show that every lambda theory is the theory of a model of this form.

Theorem 6.7 *Let $\mathcal{E}$ be any set of equations closed under $\vdash$. There exists a classical lambda model A and a Kripke partial equivalence relation R such that A/R satisfies precisely those equations that belong to $\mathcal{E}$.*

Proof Sketch: Let $\mathcal{A}$ be the Kripke term model for $\mathcal{E}$, as in the proof of Theorem 5.4. We will show that $\mathcal{A}$ is isomorphic to a quotient of the open term model $\mathcal{B}$ of β, η-conversion.

We will review the classical term model construction briefly. Let Γ_∞ be an infinite type assignment that provides infinitely many variables of each type. For each type σ, let B^σ be the collection of all equivalence classes $\{M\}$ with $\Gamma \triangleright M\!:\sigma$ for some finite $\Gamma \subseteq \Gamma_\infty$, and

$$\{M\} = \{N \mid \ \vdash \Gamma \triangleright M = N : \sigma\}$$

for some $\Gamma \subseteq \Gamma_\infty$. Let $\mathcal{B} = \langle \{B^\sigma\}, \{App^{\sigma,\tau}\}\rangle$ be the applicative structure with

$$App^{\sigma,\tau}\{M\}\{N\} = \{MN\}$$

This is easily shown to be an ordinary typed lambda model. More details of the term model construction may be found in [Fri75], for example.

Using $\mathcal{E}$, we can define a *klper* $\mathcal{R}$ over $\mathcal{B}$. Since the possible worlds of $\mathcal{A}$ are type assignments, we will use type assignments as the worlds of $\mathcal{R}$. For each σ and Γ, let

$$R^\sigma_\Gamma(\{M\}, \{N\}) \text{ iff } \Gamma \triangleright M^0 = N^0 : \sigma \in \mathcal{E},$$

where M^0 is the β, η-normal form of M, and similarly for N^0. This is well defined since each $\{M\}$ has a unique β, η-normal form, by the Church-Rosser theorem. Since provable equality is symmetric and transitive with respect to any Γ, $\mathcal{R}$ is clearly a partial equivalence relation. (In general, R^σ_Γ will not be reflexive on B^σ, since some M^0 may require variables not in in Γ.) By rule *(add var)*, $\mathcal{R}$ satisfies the monotonicity condition *(mon)*. The proof that $\mathcal{R}$ satisfies *(compre)* is similar to the proof that Kripke term models are extensional.

It remains easy to show that $\mathcal{A}$ is isomorphic to $\mathcal{B}/\mathcal{R}$. Every B^σ/R^σ_Γ equivalence class is characterized by a collection of normal forms that are all well-typed in Γ and provably equal using $\mathcal{E}$. Thus for each B^σ/R^σ_Γ equivalence class $\{M\}/R^\sigma_\Gamma$, there is a unique $[\Gamma \triangleright M^0 \colon \sigma] \in A^\sigma_\Gamma$. Conversely, all of the β,η-normal forms in any $[\Gamma \triangleright M^0 \colon \sigma]$ will be equivalent modulo R^σ_Γ, and so we have a straightforward bijection between A^σ and B^σ/R^σ_Γ. It is easy to show that application behaves appropriately, and so we have an isomorphism between $\mathcal{A}$ and $\mathcal{B}/\mathcal{R}$. This proves the theorem. ∎

However, we can show that some Kripke lambda models are not isomorphic to any Kripke quotient of any classical applicative structure.

Theorem 6.8 *There is a Kripke lambda model B which is not elementarily equivalent to A/R, for any classical applicative structure A and Kripke partial equivalence relation R.*

Proof Sketch: We give a formula ϕ in the predicate logic for applicative structures with base types p and q which is valid in all quotients A/R but is not valid in every Kripke lambda model B. Intuitively, ϕ is the formula that says:

- if $empty(p)$ and $\neg\neg inhabited(p \to q)$, then $inhabited(p \to q)$,

where $inhabited(\tau) \equiv (\exists x \colon \tau . x = x)$ and $empty(\tau) \equiv \neg inhabited(\tau)$ It is easy to check that this holds in any quotient of a classical applicative structure.

To show that ϕ is not intuitionistically valid, we consider the following Kripke lambda model B:

- W is the poset with two elements $0 < 1$
- B at 1 is the full type structure with the base types $p = \emptyset$ and $q = \omega$
- B at 0 is the interior of B at 1, i.e. , the applicative sub-structure whose elements are interpretations of closed λ-terms
- the transition function $i_{0,1}$ is the inclusion

Then $A_1^{p \to q}$ contains exactly one element (the empty function), and therefore B satisfies $\neg\neg inhabited(p \to q)$; but $A_0^{p \to q}$ is empty (because there are no closed terms of that type), so B does not satisfy $inhabited(p \to q)$. ∎

6.4 A counter-model to implication $(*)$

As an application of Kripke quotients, we will show how to construct a counter-model to the implication $(*)$ given in the second paragraph of the paper. We will construct a Henkin model A with base types a and b and give a Kripke logical partial equivalence relation $R \subseteq A \times A$ such that in the quotient model A/R, we will have

$$\lambda x\colon a.f\pi_1 = \lambda x\colon a.f\pi_2 \quad \text{but not} \quad f\pi_1 = f\pi_2.$$

We let A be a classical term model of β, η-conversion, as described in the proof of Theorem 6.7. Since f appears in the equations above, we include terms with constant $f\colon (a \to a \to a) \to b$ in constructing A. (The interpretation of f in $\mathcal{A}$ is its equivalence class, modulo $\vdash$.)

It remains to define the relation R at base types, since this will determine R at higher types. Since the justification of $(*)$ depends on type a being either globally empty or globally nonempty, we will make a empty at one world and nonempty at another. We let $W = \{0, 1\}$ with $0 \leq 1$ and take $R_0^a = \emptyset$ and R_1^a the identity relation A^a. Now, we want to satisfy equation

$$(E) \qquad \lambda x\colon a.f\pi_1 = \lambda x\colon a.f\pi_2$$

at both worlds. This is easy at world 0, since a is empty. We can take R_0^b to be the identity relation on A^b. To satisfy (E) at world 1 where a is not empty, we must equate $f\pi_1$ and $f\pi_2$. An easy way to do this is just to take $R_1^b = A^b \times A^b$ so that A^b/R_1^b has only one element. It is easy to verify that

$$R_w^{a \to b}(\lambda x\colon a.f\pi_1,\ \lambda x\colon a.f\pi_2)$$

at both worlds $w = 0, 1$, and so these terms are equal in the quotient model. However, since $f\pi_1$ and $f\pi_2$ are not β, η-equivalent, they are not related by R_0^b, and so the equation $f\pi_1 = f\pi_2$ does not hold at world 0 in the quotient model. Consequently, A/R satisfies (E), but not $f\pi_1 = f\pi_2$.

7 Conclusion and directions for further investigation

While the traditional axiom system is not complete for semantic implication over Henkin models, we have completeness for Kripke models. Since Kripke models satisfy intuitionistic principles, but not the law of the excluded middle

$(\phi \vee \neg\phi)$, this may be interpreted as evidence that typed lambda calculus is more an intuitionistic system than a classical one. In addition, we have a straightforward correspondence between provable propositions and nonempty types, which suggests that Kripke models may be useful for studying systems like Martin-Löf's type theory (cif. [Bee82]). It is easy to see that Kripke lambda models are more general than classical lambda models, since any classical lambda model may be be regarded as a Kripke lambda model over a set W consisting of a single possible world. Kripke models with products $\sigma \times \tau$ and a terminal type 1 are also a special kind of cartesian closed category and, conversely, any cartesian closed category may be embedded in a Kripke model.

Although we defined Kripke models without using much category theory, one way to view our development is as a "worked example" in the use of the internal language of a topos. Specifically, our Kripke lambda models result from interpreting the standard classical definition of typed lambda model in the logic of a topos of presheaves over a poset. In addition, as pointed out to us by Edmund Robinson and Pino Rosolini, our completeness theorem may be derived using connections between cartesian closed categories and presheaf toposes. Our study of Kripke logical relations may also be viewed this way using the standard notion of relation in the internal logic.

Our brief investigation of Kripke logical relations suggests that many classical model-theoretic techniques may be adapted to Kripke lambda models, and demonstrates that Plotkin's I-relations provide a useful class of "intuitionistic" lambda models. We have shown that every typed lambda theory is the theory of some Kripke quotient of a classical lambda models, but that this does not carry over to quantified formulas. Specifically, we found a formula ϕ which is valid in Kripke quotient models, but not in all Kripke lambda models.

In general, our main focus has been on theoretical aspects of Kripke lambda models. Having found Kripke models relatively natural and easy to work with, it is worth asking whether Kripke lambda models are appropriate to computer science applications. For example, do Kripke-like models arise naturally in the semantics of programming languages? One suggestion that they do comes from the study of storage allocation in Algol-like languages. John Reynolds and Frank Oles have proposed functors over "store-shapes" as a mathematical semantics for languages which admit stack-structured storage allocation [Rey81, Ole85]. (Some related discussion appears in [Ten85].) In addition to taking "storage maps" as possible worlds, some other possibilities might be sets of declarations (as in our completeness proof), program contexts, or their meanings. Given the differences between Henkin models and functor categories, it seems worthwhile to reconsider carefully which are more natural for the semantics of programs.

Acknowledgments: We are grateful to Edmund Robinson and Pino Rosolini for very helpful suggestions regarding the relationship between cartesian closed categories and Kripke lambda models. Thanks also to Peter Freyd, Andre Scedrov, Philip Scott and an anonymous referee.

References

[Bar84] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North Holland, 1984.

[Bee82] M. Beeson. Recursive models for constructive set theories. *Ann. Mathematical Logic*, 23:127–178, 1982.

[Con85] R.C. Constable. The semantics of evidence. Unpublished, 1985.

[Fou77] M.P. Fourman. The logic of topoi. In *Handbook of Mathematical Logic*, pages 1053–1090, North-Holland, 1977.

[Fri75] H. Friedman. Equality between functionals. In R. Parikh, editor, *Logic Colloquium*, pages 22–37, Springer-Verlag, 1975.

[GM82] J. Goguen and J. Meseguer. Completeness of many-sorted equational logic. *SIGPLAN Notices*, 17:9–17, 1982.

[GM86] J. Goguen and J. Meseguer. Remarks on remarks on many-sorted equational logic. *Bulletin EATCS*, 30:66–73, 1986.

[Hen50] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2), June 1950. pages 81-91.

[How80] W. Howard. The formulas-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490, Academic Press, 1980.

[HS86] J.R. Hindley and J.P. Seldin. *Introduction or Combinators and Lambda Calculus.* London Mathematical Society, 1986.

[Jac75] G. Jacopini. A condition for identifying two elements of whatever model of combinatory logic. In C. Bohm, editor, *Lambda Calculus and Computer Science Theory*, pages 213–219, Springer-Verlag, 1975.

[Joh80] P. Johnstone. Open maps of toposes. *Manuscripta Mathematica*, 31:217–247, 1980.

[Lam80] J. Lambek. From lambda calculus to cartesian closed categories. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 375–402, Academic Press, 1980.

[Lau65] H. Läuchli. Intuitionistic propositional calculus and definably non-empty terms (abstract). *Journal of Symbolic Logic*, 30, 1965. pages 263.

[Lau70] H. Läuchli. An abstract notion of realizability for which intuitionistic predicate calculus is complete. In Kino Myhill and Vesley, editors, *Intuitionism and Proof Theory: Proc. of the Summer Conference at Buffalo N.Y.*, North Holland, 1970. pages 227-234.

[LS86] J. Lambek and P.J. Scott. *Introduction to Higher-Order Categorical Logic.* Cambridge studies in advanced mathematics 7, 1986.

[Mey82] A.R. Meyer. What is a model of the lambda calculus ? *Information and Control*, 52(1), 1982. pages 87-122.

[Mit86] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In *ACM Conference on LISP and Functional Programming*, pages 308–319, August 1986.

[MMMS87] A.R. Meyer, J.C. Mitchell, E. Moggi, and R. Statman. Empty types in polymorphic lambda calculus. In *Proc. 14-th ACM Symp. on Principles of Programming Languages*, pages 253–262, 1987.

[Ole85] F.J. Oles. Type algebras, functor categories and block structure. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, Cambridge University Press, 1985.

[Plo80] G.D. Plotkin. Lambda definability in the full type hierarchy. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–373, Academic Press, 1980.

[Rey81] J.C. Reynolds. The essence of algol. In de Bakker and van Vliet, editors, *Algorithmic Languages*, pages 345–372, IFIP, North Holland, 1981.

[Sco80] D. Scott. Relating theories of the lambda calculus. In P. Seldin R. Hindley, editor, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, 1980.

[Sta82] R. Statman. Completeness, invariance and lambda-definability. *J. Symbolic Logic*, 47:17–26, 1982.

[Sta85a] R. Statman. Equality between functionals, revisited. In *Harvey Friedman's Research on the Foundations of Mathematics*, North-Holland, 1985.

[Sta85b] R. Statman. Logical relations and the typed lambda calculus. *Information and Control*, 65:85–97, 1985.

[Ten85] R. D. Tennant. Semantical analysis of specification logic. In *Logics of Programs*, pages 373–386, Springer Lecture Notes Computer Science 193, Brooklyn, NY, June 1985.