A CATEGORY-THEORETIC APPROACH TO THE SEMANTICS OF PROGRAMMING LANGUAGES

Frank J. Oles

August 1982

Frank J. Oles, Ph.D. (Frankjosepholes@aol.com (mailto:Frankjosepholes@aol.com))
February 4, 2012

A CATEGORY-THEORETIC APPROACH TO THE SEMANTICS OF PROGRAMMING LANGUAGES

by

FRANK JOSEPH OLES

B. S., Case Western Reserve University, 1968
M. S., Cornell University, 1970

ABSTRACT OF DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Science in the Graduate School of Syracuse University
August, 1982

ABSTRACT OF DISSERTATION

Here we create a framework for handling the semantics of fully typed programming languages with imperative features, higher-order ALGOL-like procedures, block structure, and implicit conversions. Our approach involves the introduction of a new family of programming languages, the coercive typed λ-calculi, denoted by L in the body of the dissertation. By appropriately choosing the linguistic constants (i.e. generators) of L, we can view phrases of variants of ALGOL as syntactically sugared phrases of L.

This dissertation breaks into three parts. In the first part, consisting of the first chapter, we supply basic definitions and motivate the idea that functor categories arise naturally in the explanation of block structure and stack discipline. The second part, consisting of the next three chapters, is dedicated to the general theory of the semantics of the coercive typed λ-calculus; the interplay between posets, algebras, and Cartesian closed categories is particularly intense here. The remaining four chapters make up the final part, in which we apply the general theory to give both direct and continuation semantics

for desugared variants of ALGOL. To do so, it is necessary to show certain functor categories are Cartesian closed and to describe a category Σ of store shapes. An interesting novelty in the presentation of continuation semantics is the view that commands form a procedural, rather than a primitive, phrase type.

A CATEGORY-THEORETIC APPROACH TO THE SEMANTICS OF PROGRAMMING LANGUAGES

by

FRANK JOSEPH OLES

B. S., Case Western Reserve University, 1968
M. S., Cornell University, 1970

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Science in the Graduate School of Syracuse University
August, 1982

PREFACE

The aim of this work has been to explain the semantics of fully typed languages that combine imperative capabilities with ALGOL-like procedures and block structure. In particular this approach is meant to handle in an intuitively satisfying way the connections among language features such as higher-order types, implicit conversions, identifier binding, and stack discipline. At the core of this dissertation are the following six doctrines.

1. The collection of phrase types for a programming language is a highly structured mathematical object whose definition is dominated by the interaction between procedural types and implicit conversions. It is a free type algebra generated by the poset of primitive phrase types. (A type algebra has a category rather than a set as its underlying object, and its structure reflects the fact that hom functors are contravariant in their first argument and covariant in their second.)

2. By the device of incorporating the identifiers into the signature, a desugared, fully-typed version of ALGOL 60 is best viewed, not as a free algebra generated by its identifiers, but rather as a free algebra generated by what one usually views as operators (e.g., true, or, 0, 1, +, :=, etc.).

3. To check a program for mismatched types is to evaluate the application of a homomorphism to the program.

4. The meaning of a program or a program fragment, in the context of an assignment of types to its free identifiers, is a morphism in a Cartesian closed category.

5. Variable declarations should be viewed as changing, not the store of the machine, but rather the set of stores with respect to which commands are interpreted; the attempt to capture this intuition precisely leads to the natural introduction of Cartesian closed categories with non-functional morphisms.

6. The definition of functions by structural induction represents a failure to discern underlying algebraic structure; in so far as possible a syntactic or semantic function should be described as the unique morphism which satisfies some universal mapping property.

The debt I owe my advisor, Prof. John C. Reynolds, is substantial. His ideas, described in part in [6], [7], and [8], have had a profound impact on this work. In particular, the original suggestion that certain Cartesian closed categories whose objects are functors have a role to play in explaining stack discipline was his. Also, he drew my attention to the distinction between data types and phrase types in the course of discussing his ideas on the relevance of posets to the proper explanation of coercion. I would like also to thank Prof. F. Lockwood Morris

and Prof. Luis Sanchis. Special thanks go to A.V. Jategaonkar.

This dissertation was typed on I.B.M. Selectric typewriters using five different kinds of type balls. Obviously, the typists, Virginia Vedder, Barbara Merchant, and Bert Fancher, deserve my eternal gratitude for putting this dissertation into a readable form.

This material is based upon work partially supported by National Science Foundation Grant MCS-8017577.

CONTENTS

Page

PREFACE iii

Chapter

I. THE UTILITY OF FUNCTOR CATEGORIES 1

II. TYPE ALGEBRAS 28

III. SYNTAX, SUBSTITUTION, AND TYPE CHECKING . . 50

IV. THE FUNDAMENTAL THEOREM OF SEMANTICS . . . 86

V. CARTESIAN CLOSED CATEGORIES 138

VI. A CATEGORY OF STORE SHAPES 162

VII. DESUGARED ALGOL: DIRECT SEMANTICS 171

VIII. DESUGARED ALGOL: CONTINUATION SEMANTICS . 211

BIBLIOGRAPHY 239

CHAPTER I

THE UTILITY OF FUNCTOR CATEGORIES

This chapter is intended to serve as a foundation upon which the rest of the dissertation rests. The foundational material is of two sorts. First, there are the definitions of terms and the descriptions of notations, familiarity with which is essential to understanding the subsequent work. Second, there is a comparatively informal explanation of the connection between functor categories and the semantics of block structure. The rest of the dissertation is meant (1) to provide the mathematical substance associated with this informal connection, (2) to incorporate at the same time intuitions about implicit conversions, type checking, and the algebraic nature of syntax and semantics, and (3) to explicitly describe the semantics of a desugared variant of an ALGOL-like language.

We begin with some notational conventions. When X and Y are simply sets, then $X \to Y$ denotes the set of functions from X to Y. We write

$$f \in X \to Y ,$$

rather than

$$f: X \to Y .$$

All the infixed arrows used in this work are right-associative. Thus $W \to X \to Y \to Z$ means $W \to (X \to (Y \to Z))$. (This also applies to the heavy arrow $\Rightarrow$, which will be defined later.)

We abhor unnecessary parentheses. Therefore for functional application we write f x rather than f(x). Functional application associates to the left, so that f g h x means (((f g) h) x). Also, application binds more tightly than anything else, so that F a × G b should be read as (F a) × (G b). However, if a few extra parentheses really enhance readability, we shall throw them in.

Suppose $f \in X \to Y$. We write dom f for X, the domain of f, and cod f for Y, the codomain of f.

The function

$$[f \mid a{:}b] \in X \cup \{a\} \to Y \cup \{b\}$$

is given by

$$[f \mid a{:}b]\ x = \begin{cases} fx & \text{if} \quad x \neq a \\ b & \text{if} \quad x = a\ , \end{cases}$$

where $x \in X \cup \{a\}$.

Depending on the context, X^* means either the set of all strings of elements of X, or the set of all functions

$$f \in F \to X$$

where F is a finite subset of some specified set. The notation X^∞ denotes the set of all functions from a specified infinite set to X; the notation X^∞ always occurs in conjunction with X^*.

Ordered tuples are enclosed in angle brackets, e.g.

$$\langle y, z \rangle \in Y \times Z.$$

Also, if $f \in X \to Y$ and $g \in X \to Z$, then we use angle brackets to denote the function

$$\langle f, g \rangle \in X \to Y \times Z$$

such that

$$\langle f, g \rangle x = \langle f\ x, g\ x \rangle \quad \text{for all} \quad x \in X.$$

In order to establish notation we define some familiar category-theoretic concepts.

An ordered 6-tuple

$$C = \langle \text{Ob } C, \text{Ar } C, \text{dom}_C, \text{cod}_C, \circ_C, 1_C \rangle$$

is a <u>category</u> if

(1) Ob C (the objects of C) and Ar C (the arrows of C) are collections,

(2) $\text{dom}_C, \text{cod}_C \in \text{Ar } C \to \text{Ob } C$,

(3) $\circ_C \in \text{Ar } C \times_{\text{Ob } C} \text{Ar } C \to \text{Ar } C$, where

$$\text{Ar } C \times_{\text{Ob } C} \text{Ar } C = \{\langle \alpha, \beta \rangle \in \text{Ar } C \times \text{Ar } C \mid \text{dom}_C \alpha = \text{cod}_C \beta\}$$

(as usual, call $\circ_C$ composition and write $\alpha \circ_C \beta$ rather than $\circ_C \langle \alpha, \beta \rangle$),

(4) $\text{dom}_C (\alpha \circ_C \beta) = \text{dom}_C \beta$, $\text{cod}_C (\alpha \circ_C \beta) = \text{cod}_C \alpha$, and composition is associative,

(5) $1_C \in \text{Ob } C \to \text{Ar } C$, and

(6) for all $X \in \text{Ob } C$, $\text{dom}_C (1_C X) = X = \text{cod}_C (1_C X)$, and $1_C X$ is a two-sided identity for composition.

The subscript $\mathcal{C}$ may be omitted if it is clear from context, and alternative notations for $1_{\mathcal{C}}\ X$ are 1_X and $1^{\mathcal{C}}_X$. The names dom and cod abbreviate domain and codomain, respectively. Arrows are also called morphisms, and the set of arrows α from X to Y, i.e., such that dom α = X and cod α = Y, is denoted $X \xrightarrow[\mathcal{C}]{} Y$.

Set denotes the category of sets and functions.

A <u>functor</u> F consists of a domain $\mathcal{C}$ and a codomain $\mathcal{D}$, both of which are categories, and functions

$\mathrm{Ob}\ F \in \mathrm{Ob}\ \mathcal{C} \longrightarrow \mathrm{Ob}\ \mathcal{D}$ and $\mathrm{Ar}\ F \in \mathrm{Ar}\ \mathcal{C} \longrightarrow \mathrm{Ar}\ \mathcal{D}$

such that

(1) if $\alpha \in X \xrightarrow[\mathcal{C}]{} Y$, then $\mathrm{Ar}\ F\ \alpha \in \mathrm{Ob}\ F\ X \xrightarrow[\mathcal{D}]{} \mathrm{Ob}\ F\ Y$,

(2) if $\alpha \in X \xrightarrow[\mathcal{C}]{} Y$ and $\beta \in Y \xrightarrow[\mathcal{C}]{} Z$, then

$\mathrm{Ar}\ F\ (\beta \circ_{\mathcal{C}} \alpha) = (\mathrm{Ar}\ F\ \beta) \circ_{\mathcal{D}} (\mathrm{Ar}\ F\ \alpha)$, and

(3) if $X \in \mathrm{Ob}\ \mathcal{C}$, then $\mathrm{Ar}\ F\ (1_{\mathcal{C}}X) = 1_{\mathcal{D}}(\mathrm{Ob}\ F\ X)$

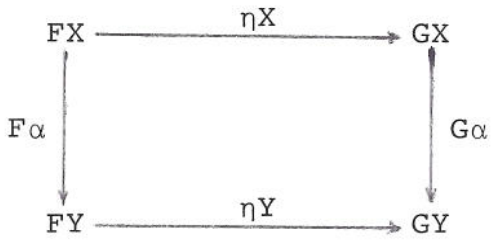
The collection of functors with domain $\mathcal{C}$ and codomain $\mathcal{D}$ is denoted $\mathcal{C} \to \mathcal{D}$. Following usual mathematical practice, we write simply F rather than Ob F or Ar F unless there is a need to be exceptionally precise. The most noteworthy functor in $\mathcal{C} \to \mathcal{C}$ is the identity functor, which is denoted $1_{\mathcal{C}}$ and which acts as the identity function on objects and arrows. Of course, functors may be composed in the usual boring way so as to obtain $G \circ F \in \mathcal{C} \to \mathcal{E}$ from $F \in \mathcal{C} \to \mathcal{D}$ and $G \in \mathcal{D} \to \mathcal{E}$.

A <u>natural transformation</u> η consists of a domain F and a codomain G, both of which are functors with identical domains

and identical codomains (say, $F,G \in C \to \mathcal{D}$), and an (Ob C)-indexed collection of arrows of $\mathcal{D}$

$$\eta X \in FX \xrightarrow[\mathcal{D}]{} GX, \quad \text{for} \quad X \in \text{Ob } C$$

such that, for all $\alpha \in X \xrightarrow[C]{} Y$, the diagram



commutes in $\mathcal{D}$.

Suppose C and $\mathcal{D}$ are categories. Of fundamental importance in the sequel is the <u>functor category</u> $C \Rightarrow \mathcal{D}$ (alternative notation: $\mathcal{D}^C$) whose objects are all functors $F \in C \to \mathcal{D}$, and whose morphisms are all natural transformations between functors in $C \to \mathcal{D}$. Composition in $C \Rightarrow \mathcal{D}$ is defined as follows: if $\eta \in F \xrightarrow[C \Rightarrow \mathcal{D}]{} G$ and $\nu \in G \xrightarrow[C \Rightarrow \mathcal{D}]{} H$, then $\nu \circ \eta \in F \xrightarrow[C \Rightarrow \mathcal{D}]{} H$ is given by

$$(\nu \circ \eta)X = (\nu X) \circ (\eta X) \quad \text{for all} \quad X \in \text{Ob } C.$$

We omit the details.

Each category C has an <u>opposite</u> category denoted C^{op} and defined by

(1) Ob C^{op} = Ob C,

(2) Ar C^{op} = Ar C,

(3) $\text{dom}_{C^{op}} = \text{cod}_C$,

(4) $\mathrm{cod}_{\mathcal{C}^{op}} = \mathrm{dom}_{\mathcal{C}}$,

(5) for all composable pairs of arrows α and β,

$\alpha \circ_{\mathcal{C}^{op}} \beta = \beta \circ_{\mathcal{C}} \alpha$,

(6) $1_{\mathcal{C}^{op}} = 1_{\mathcal{C}}$.

To each functor $F \varepsilon \mathcal{C} \to \mathcal{D}$ there corresponds a functor $F^{op} \varepsilon \mathcal{C}^{op} \to \mathcal{D}^{op}$ which agrees with F on objects and arrows but has different domain and codomain.

If $\langle \mathcal{C}, \mathcal{D} \rangle$ is an ordered pair of categories, then its <u>product</u> $\mathcal{C} \times_{Cat} \mathcal{D}$, also denoted $\mathcal{C} \times \mathcal{D}$ when the abbreviated notation causes no confusion, is defined by

(1) $\mathrm{Ob}\ (\mathcal{C} \times \mathcal{D}) = (\mathrm{Ob}\ \mathcal{C}) \times (\mathrm{Ob}\ \mathcal{D})$,

(2) $\mathrm{Ar}\ (\mathcal{C} \times \mathcal{D}) = (\mathrm{Ar}\ \mathcal{C}) \times (\mathrm{Ar}\ \mathcal{D})$,

(3) $\mathrm{dom}_{\mathcal{C} \times \mathcal{D}} \langle \alpha, \beta \rangle = \langle \mathrm{dom}_{\mathcal{C}}\ \alpha,\ \mathrm{dom}_{\mathcal{D}}\ \beta \rangle$,

(4) $\mathrm{cod}_{\mathcal{C} \times \mathcal{D}} \langle \alpha, \beta \rangle = \langle \mathrm{cod}_{\mathcal{C}}\ \alpha,\ \mathrm{cod}_{\mathcal{D}}\ \beta \rangle$,

(5) $\langle \alpha', \beta' \rangle \circ_{\mathcal{C} \times \mathcal{D}} \langle \alpha, \beta \rangle = \langle \alpha' \circ_{\mathcal{C}} \alpha,\ \beta' \circ_{\mathcal{D}} \beta \rangle$, and

(6) $1_{\mathcal{C} \times \mathcal{D}} \langle X, Y \rangle = \langle 1_{\mathcal{C}} X,\ 1_{\mathcal{D}} Y \rangle$.

Each pair of functors $F \varepsilon \mathcal{C} \to \mathcal{C}'$ and $G \varepsilon \mathcal{D} \to \mathcal{D}'$ determines a product functor

$$F \times G \varepsilon \mathcal{C} \times \mathcal{C}' \to \mathcal{D} \times \mathcal{D}'$$

which acts on objects and arrows by applying F to the first component and G to the second.

Let $\mathcal{C}$ be a category, and let I be a set. We say $\mathcal{C}$ has <u>I-ary products</u> if for each I-indexed collection

$$X \varepsilon I \to \mathrm{Ob}\ \mathcal{C}$$

of objects of $\mathcal{C}$ there is some product diagram satisfying the usual universal mapping property, i.e. there is an

object C and an I-indexed collection

$$\pi \varepsilon I \to \mathrm{Ar}\ C, \text{ where } \pi i \varepsilon C \xrightarrow[C]{} Xi \text{ for all } i \varepsilon I$$

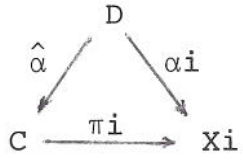
of arrows (called projections) of C such that

$$D \varepsilon \mathrm{Ob}\ C,\ \alpha \varepsilon I \to \mathrm{Ar}\ C, \text{ and } \alpha i \varepsilon D \xrightarrow[C]{} Xi \text{ for all } i \varepsilon I$$

implies the existence of a unique

$$\hat{\alpha} \varepsilon D \xrightarrow[C]{} C$$

making the diagrams



commute for all $i \varepsilon I$. Of course, a category with I-ary products may have more than one product diagram for an I-indexed collection of objects. Our interest is mainly in categories having I-ary products <u>specifically given</u>. In this situation, the object in the distinguished product diagram for $X \varepsilon I \to \mathrm{Ob}\ C$ is denoted $\Pi_I X$, and the I-indexed collection of distinguished projections arrows is written

$$\mathrm{proj}_I \varepsilon I \longrightarrow \mathrm{Ar}\ C,$$

so that for all $i \varepsilon I$

$$\mathrm{proj}_I\ i \varepsilon \Pi_I X \xrightarrow[C]{} Xi.$$

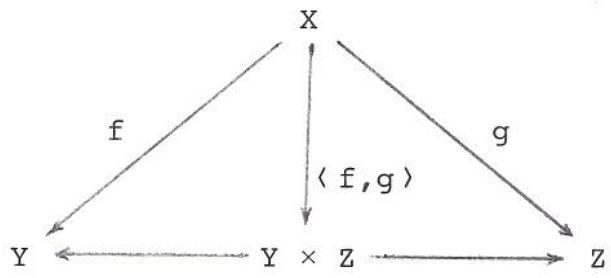
We say that C has <u>finite products specifically given</u> if C has I-ary products specifically given for each finite set I; in this instance C has a distinguished terminal object

Π_ϕ emp_C , where

$$emp_C \in \phi \to Ob\ C$$

denotes the empty collection.

Binary products are usually denoted with $\times$. We extend the use of angle brackets from *Set* to an arbitrary category C with a distinguished binary product functor by letting $\langle f, g \rangle$ denote the unique arrow such that



commutes, where the horizontal arrows are projections.

A category K with finite products specifically given is a Cartesian closed category if the following three conditions hold:

(Cart1) K has a distinguished terminal object K_{term}

(note that we do not require $K_{term} = \Pi_\phi\ emp_K$),

(Cart2) K has a distinguished binary product functor

$$K_\times \in K \times K \to K$$

(notation: $K_\times \langle X, Y \rangle = X \times_K Y = X \times Y$)

(Cart3) for each ordered pair $Y, Z \in Ob\ K$ there is $Y \Rightarrow Z \in Ob\ K$ and

$$Ap_K \langle Y, Z \rangle \in (Y \Rightarrow Z) \times Y \xrightarrow[K]{} Z$$

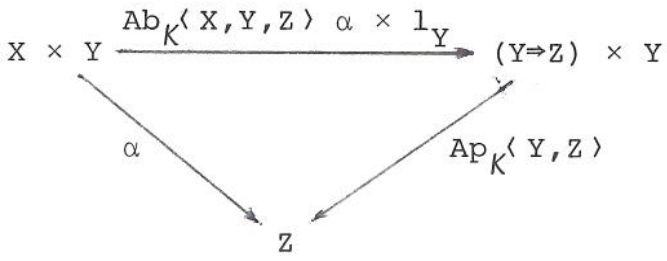
such that whenever $X \varepsilon \mathrm{Ob}\ K$ and

$$\alpha \varepsilon X \times Y \xrightarrow[K]{} Z$$

there exists a unique arrow

$$\mathrm{Ab}_K \langle X, Y, Z \rangle\ \alpha \varepsilon X \xrightarrow[K]{} (Y \Rightarrow Z)$$

making the diagram



commute (notation:

$$Y \Rightarrow Z = Z^Y = Y \xRightarrow[K]{} Z = K_{\Rightarrow} \langle Y, Z \rangle\ ).$$

It is not hard to see that, for a Cartesian closed category K, there is a unique way to extend the definition of $K_{\Rightarrow}$ to give a functor

$$K_{\Rightarrow} \varepsilon K^{\mathrm{op}} \times K \longrightarrow K\ .$$

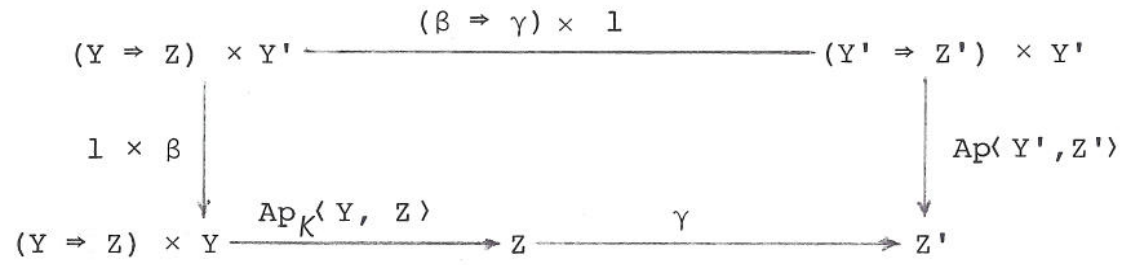
Indeed, if

$$\beta \varepsilon Y' \xrightarrow[K]{} Y \quad \text{and} \quad \nu \varepsilon Z \xrightarrow[K]{} Z'$$

then

$$K_{\Rightarrow} \langle \beta, \nu \rangle = \beta \Rightarrow \nu$$

is the unique arrow such that



commutes. The functor $K_{\Rightarrow}$ is sometimes called an "internal" hom-functor because it is reminiscent of the "external" hom-functor

$$\hom_K \varepsilon K^{op} \times K \to Set$$

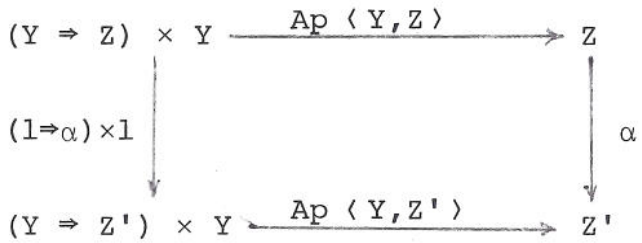
When X, Y, and Z are understood, we write

Ab or Ab_K for $Ab_K\langle X, Y, Z\rangle$ ("abstraction")

and

Ap or Ap_K for $Ap_K\langle Y, Z\rangle$ ("application").

Of course, *Set* is the most common Cartesian closed category, and for it the internal and external hom-functors coincide. Another easily verifiable fact about Cartesian closed categories is that Ap is natural in its second argument. Thus, for Y, Z, Z' ε Ob K and $\alpha \varepsilon Z \xrightarrow[K]{} Z'$, the diagram



commutes. In IV.6 of MacLane [4] an equivalent definition of Cartesian closed category in terms of specified right adjoints may be found.

The point of the discussion so far has been to fix notation for some standard concepts in category theory. Definitions of other category-theoretic terms which appear in the sequel may be found in MacLane [4] or Arbib and Manes [1]. We now move on to posets.

A <u>partial order</u> on a set X is a relation on X, i.e., a subset of X×X, which is reflexive, antisymmetric, and transitive. An ordered pair

$$P = \langle \text{Ob } P, \text{Ar } P \rangle$$

is a <u>poset</u> if Ob P is a set and Ar P is a partial order on Ob P. The assertion $\langle x, y \rangle \in \text{Ar } P$ is usually written $x \leq_P y$. In addition the expression $x \leq_P y$ is also used to denote the ordered pair $\langle x, y \rangle$ when it is true that $\langle x, y \rangle \in \text{Ar } P$. The correct reading of $x \leq_P y$ will be clear from context. The poset P becomes a category if we make the canonical definitions:

(1) $\text{dom}_P (x \leq_P y) = x$,

(2) $\text{cod}_P (x \leq_P y) = y$,

(3) $(y \leq_P z) \circ (x \leq_P y) = (x \leq_P z)$, and

(4) $1_P x = (x \leq_P x)$.

For posets P and Q, the object part of any functor

$$F \in P \to Q$$

is a monotone function, i.e. $x \leq_P y$ implies $F x \leq_Q F y$. Conversely, any monotone function $F \in \text{Ob } P \to \text{Ob } Q$ has a

unique extension to a functor $F \varepsilon P \to Q$. The adjectives "faithful" and "full" in the context of posets have the following meanings. A functor is faithful if it is injective on objects. A functor $F \varepsilon P \to Q$ is full if for all $x, y \varepsilon$ Ob P,

$$x \leq_P y \quad \text{iff} \quad F\,x \leq_Q F\,y.$$

Let R be the poset whose objects are monotone functions from P to Q and whose partial order is given by

$$f \leq_R g \quad \text{iff} \quad fx \leq_Q gx \quad \text{for each } x \varepsilon \text{ Ob } P.$$

We may compare the canonical category derived from R with the functor category $P \Rightarrow Q$ created from the canonical categories derived from P and Q. It is easy to see that they are isomorphic. Indeed, the clarity of the subsequent exposition is not harmed by writing $R = P \Rightarrow Q$ and $f \leq_{P \Rightarrow Q} g$ rather than $f \leq_R g$.

A <u>predomain</u> is a directed-complete poset, i.e. a poset all of whose directed subsets have least upper bounds, and a <u>domain</u> is a predomain that has a minimal element. The minimal element of a domain P is denoted $\perp$ (read "bottom") or $\perp_P$. For predomains P and Q, a function $f \varepsilon$ Ob $P \to$ Ob Q is said to be <u>continuous</u> if f preserves least upper bounds of directed subsets, i.e., for each directed subset D of P, $f(\sqcup_P D) = \sqcup_Q (fD)$. Of course, compositions of continuous functions are continuous, as also are identity functions. We are thus led to *Pdom*, the category whose objects are

predomains and whose arrows are continuous functions, with composition in $\mathcal{P}dom$ being functional composition. The category $\mathcal{D}om$ is defined to be the full subcategory of $\mathcal{P}dom$ whose objects are domains. A continuous function $f \in P \xrightarrow[\mathcal{D}om]{} Q$ between domains is <u>strict</u> if it is bottom-preserving, i.e. $f\,\perp_P = \perp_Q$. By taking as objects all domains and as morphisms all strict continuous functions, we obtain $\mathcal{S}dom$, a subcategory of $\mathcal{D}om$. In Chapter V we will show that both $\mathcal{P}dom$ and $\mathcal{D}om$ are Cartesian closed categories; for either of them the underlying set of $P \Rightarrow Q$ is the set of continuous functions from P to Q.

From each predomain P we may create a domain $P_\perp$ by simply adding an element not already in P as a minimal element. In most contexts the added minimal element is thought of as an "undefined element of P."

The connection between $\mathcal{S}et$ and $\mathcal{P}dom$ is worth exploring. There is an obvious forgetful functor

$$U \in \mathcal{P}dom \longrightarrow \mathcal{S}et$$

that forgets partial orders and regards continuous functions simply as functions. For each set S, let $=_S$ be the discrete partial order on S. We then get an embedding functor.

$$E \in \mathcal{S}et \longrightarrow \mathcal{P}dom$$

given by

$$ES = \langle S, =_S \rangle \text{ for } S \in \text{Ob } \mathcal{S}et,\ E\sigma = \sigma \text{ for } \sigma \in \text{Ar } \mathcal{S}et.$$

Clearly, $U \circ E = 1_{\mathcal{S}et}$. Also

$$S \xrightarrow[\mathcal{S}et]{} U\,P \quad = \quad ES \xrightarrow[\mathcal{P}dom]{} P$$

for all sets S and all predomains P. Thus E is a left adjoint

of U, and E *Set* (the image of E) is a full subcategory of *Pdom* that is isomorphic to *Set*. For completeness, we note that E is also the right adjoint of the functor in *Pdom* → *Set* which assigns to each predomain its set of equivalence classes modulo the least equivalence relation containing the partial order. The proof is easy.

This completes our review of definitions and notations. We now embark on an informal attack on the nature of block structure.

An intuitive grasp of the nature of variable declarations and of what is happening as one enters and exits from blocks is essential to programming in an ALGOL-like language. However, a precise semantic description of the constructs involved is difficult, and it is particularly difficult if one wants semantics for block structure that mesh elegantly with semantics for procedures.

Our goal is to outline in general terms how functor categories can be used to explain the semantics of ALGOL-like languages which possess

1) a rich type structure,

2) higher-order procedures, whose types may be arbitrarily complex,

3) imperative capabilities,

and

4) block structure.

The principal intuition we will try to capture is that during the execution of a program written in a language that obeys a

stack discipline not only is the store changed by commands (statements), but also the shape of the store is changed by variable declarations during block entrances and by block exits.

An extended description of the characteristics of the kinds of languages in which we are interested can be found in [6]. We have not worked out the details of the semantics of every language feature mentioned there, but we do know how to deal with the most important features, and we can tell a complete story about block structure.

For our purposes we think of a language as beginning with the specification of a set $\mathcal{D}$ of data types. Let's say int (for integer) and bool (for boolean) are elements of $\mathcal{D}$. Along with $\mathcal{D}$ we have in mind Val, an assignment of sets (of values) to data types. Say

$$\text{Val } \underline{\text{int}} = \mathbb{Z} \text{ (the set of integers)}$$

and

$$\text{Val } \underline{\text{bool}} = \{\text{true, false}\}.$$

Each data type $\delta \in \mathcal{D}$ gives rise to three primitive phrase types.

1) δ-exp (for δ-expression),

2) δ-var (for δ-variable),

3) δ-acc (for δ-acceptor).

Another primitive phrase type is comm (for command). The reader probably has an excellent intuitive feeling for the nature of phrases that are assigned the phrase types δ-exp, δ-var, and comm. A phrase has type δ-acc if it gobbles up

values of type δ and produces commands. Thus, if x and y are δ-variables, then in the assignment command x := y the δ-variable y is used as a δ-expression and the δ-variable x is used as a δ-acceptor. (Hence, even if a language turns out not actually to have any phrases of type δ-acc, it is still useful to introduce that phrase type in order to explain δ-var.) A complete exposition of the connections between δ-exp, δ-var, and δ-acc would involve us in implicit conversions, a topic that we will take up in subsequent chapters. See also Reynolds [8].

The collection of all phrase types, denoted by T, certainly contains the primitive phrase types already described. Also, for all elements τ, θ of T, there is a phrase type $\tau \Rightarrow \theta$ that is assigned to procedures accepting arguments (i.e. having formal parameters) of type τ and producing results (i.e. having calls) of type θ. The use of the same symbol for both procedural phrase types and exponentiation in a Cartesian closed category is, of course, intentional.

An integral part of the semantics of a typed language is the assignment to each phrase type τ of a meaning, denoted Mng τ. Perhaps with the plan of arranging matters so that program fragments of type τ denote elements of Mng τ, one might suppose Mng τ is a set. However, the possible existence of nonterminating programs, which lead to an "undefined" state, provides an inducement to partially order Mng τ, where the relation $x \leq y$ means x is "more undefined" than y. For instance, see [9]. Following this line of reasoning, a

command which never terminates, such as while true do noaction, denotes the minimal element in Mng comm. The need to give meanings to recursively defined expressions of type τ causes us to require that directed subsets of Mng τ have least upper bounds, i.e. that Mng τ is a predomain. (Also, we generally want Mng τ to have a minimal element, but we must tread cautiously at this point to avoid becoming overcommitted to the use of *Dom* rather than *Pdom*. As we shall see later, *Dom* is technically inadequate.)

Suppose we try only to give the semantics of programs which do not contain block entrances and exits. We start by positing the existence of a set S of possible stores. Regard sets as discretely ordered predomains. Since a command is a transformation of S that possibly may not terminate, and a function from S to $S_\perp$ is the same as a continuous function from S to $S_\perp$, we expect

$$\text{Mng comm} = S \Rightarrow S_\perp .$$

Here $\Rightarrow$ is the internal hom functor for *Pdom*.

Also, for each data type δ,

$$\text{Mng } \delta\text{-exp} = S \Rightarrow (\text{Val } \delta)_\perp ,$$
$$\text{Mng } \delta\text{-acc} = \text{Val } \delta \Rightarrow \text{Mng comm},$$
$$\text{Mng } \delta\text{-var} = \text{Mng } \delta\text{-acc} \times \text{Mng } \delta\text{-exp}$$

In other words, a δ-expression attempts to compute a value of type δ from the current store, a δ-acceptor uses a value of type δ to update the current store, and a δ-variable may be used as either a δ-expression or a δ-acceptor. Finally, for

all τ, $\theta \in T$, we expect

$$\text{Mng } (\tau \Rightarrow \theta) = \text{Mng } \tau \Rightarrow \text{Mng } \theta,$$

i.e. the predomain of meanings for the procedural phrase type $\tau \Rightarrow \theta$ is the predomain of continuous functions from Mng τ to Mng θ.

Although the approach of the preceding paragraph is attractively comprehensible, it is inadequate for the semantics of block structure because the set S is fixed throughout the exposition. The whole point of block structure is to permit S to vary during program execution. For instance, if we view stores as being functions from finite sets of locations in memory to the set of data-type values, then the domains of those functions may be regarded as store shapes. Variable declarations at the start of a block alter the shape of the store by adding locations to it, whereas block exit restores the shape of the store to its condition at block entrance. The semantics of a language obeying a stack discipline should reflect this dynamic behavior.

Therefore, let Σ be the collection of all store shapes. To each $X \in \Sigma$, there is a set St X of stores of that shape. Since the meaning of $\tau \in T$ varies with the store shape, Mng τ is not a predomain, but is rather a Σ-indexed collection of predomains. For instance we might arrange matters so that

$$\text{Mng } \underline{\text{comm}} \text{ X} = \text{St X} \Rightarrow (\text{St X})_{\perp} ,$$

$$\text{Mng } \delta\text{-}\underline{\text{exp}} \text{ X} = \text{St X} \Rightarrow (\text{Val } \delta)_{\perp} ,$$

$$\text{Mng } \delta\text{-}\underline{\text{acc}}\ X = \text{Val } \delta \Rightarrow \text{Mng } \underline{\text{comm}}\ X,$$

$$\text{Mng } \delta\text{-}\underline{\text{var}}\ X = \text{Mng } \delta\text{-}\underline{\text{exp}}\ X \times \text{Mng } \delta\text{-}\underline{\text{acc}}\ X,$$

where $X \in \Sigma$, $\delta \in \mathcal{D}$.

It is important to realize that for $\tau \in T$ and $X, Y \in \Sigma$ the predomains Mng τ X and Mng τ Y cannot be arbitrarily different. After all, we want the notion of command to have a uniform meaning for all store shapes or else the definition of operations like ; (concatenation of commands) will be bizarrely complicated. For instance, consider the program skeleton

```
begin int-var x;
        .
        .
        .
        x := 3;
        .
        .
        .
        begin bool-var y;
                .
                .
                .
                x := 3;
                .
                .
                .
        end;
        .
        .
        .
end
```

Suppose X is the store shape corresponding to the outer block and Y is the store shape corresponding to the inner block. Then Mng comm X is relevant to the first occurrence of the assignment command x := 3, while Mng comm Y is relevant to the second occurrence. However, both occurrences are meant

to alter the contents of the same location. Roughly speaking, the fact that X can be "expanded" to give Y induces a function from Mng comm X to Mng comm Y. So it becomes important to contemplate the notion of an expansion from a store shape X to a store shape Y. Certainly, expansions ought to be composable. The composition ought to be associative. For each store shape X, there ought to be an identity expansion which involves "doing nothing" to X. In short, we assert that we erred in letting Σ be the collection of store shapes. From now on, take Σ to be the category of store shapes. The morphisms of Σ are called expansions. Furthermore, for each phrase type τ we should require that Mng τ be a functor

$$\text{Mng } \tau \in \Sigma \longrightarrow Pdom;$$

this will elegantly take care of how an expansion

$$\sigma \in X \xrightarrow[\Sigma]{} Y$$

induces a function

$$\text{Mng } \tau\ \sigma \in \text{Mng } \tau\ X \xrightarrow[Pdom]{} \text{Mng } \tau\ Y.$$

Procedural phrase types are a bit tricky. Let $\tau, \theta \in T$. Recall that in the simpler setting the predomain of meanings of type $\tau \Rightarrow \theta$ was the set of continuous functions from Mng τ to Mng θ. One might hope that in the more sophisticated setting, where Mng $(\tau \Rightarrow \theta)$ is to be a functor, that the set of procedural meanings of type $\tau \Rightarrow \theta$ in the context of a store shape X would be the set of continuous functions from Mng τ X to Mng θ X, i.e.

$$\text{Mng } (\tau \Rightarrow \theta)\ X = \text{Mng } \tau\ X \Rightarrow \text{Mng } \theta\ X.$$

Alas, this does not define a functor because $\Rightarrow$ in *Pdom* is contravariant in its first argument. (For the same reason, one cannot define a functor from *Set* to *Set* by diagonalizing the hom functor, i.e. by letting X go to $X \to X$.) Another idea might be to recall that Mng τ and Mng θ are objects of a functor category and to try letting Mng $(\tau \Rightarrow \theta)$ be the set of natural transformations from Mng τ to Mng θ. That's plain nonsense, because there is no way to regard such a set of natural transformations as a functor. We are, however, getting closer to the heart of the matter. In Chapter V we will prove that the functor category $\Sigma \Rightarrow$ *Pdom* is Cartesian closed. Therefore, the appropriate equation governing meanings of procedural types is just

$$\text{Mng } (\tau \Rightarrow \theta) = \text{Mng } \tau \Rightarrow \text{Mng } \theta,$$

where the heavy arrow on the right is exponentiation in the functor category $\Sigma \Rightarrow$ *Pdom*. Further evidence of the essential rightness of this equation will be presented at the conclusion of this section.

The reader may wonder why we didn't engineer this discussion so as to end up with the functor category $\Sigma \Rightarrow$ *Dom*. Unfortunately, contrary to the claim in [6], it does not appear that $\Sigma \Rightarrow$ *Dom* is a Cartesian closed category, in spite of the fact that *Dom* is Cartesian closed.

This all sounds nice enough, but we have lost somewhere the idea that phrases of type τ should have denotations

which are elements of Mng τ, because Mng τ doesn't seem to have any elements if it is a functor. The solution is to reject even in the simple setting where block structure is ignored the intuition that a phrase of type τ denotes an element of Mng τ. Actually, this is a rather conventional notion. In the simpler setting, complete specification of the semantics of a language requires not only a set S of stores, but also a function A ε Id $\rightarrow$ T, where Id is the set of identifiers. The function A has two uses. First, it is used to assign types to free occurrences of identifiers, enabling the type of the entire phrase to be determined. Second, A determines a predomain E of environments by taking E to be the product in *Pdom* of the Id-indexed collection {Mng (A x) | x ε Id}. Thus, each element of E associates a meaning of the right type to each identifier. Then even in this simpler setting we conceive of the denotation of a phrase of type τ as being a (continuous) function from E to Mng τ. We have thus been led to the idea that the semantics of a phrase is a morphism in a category from an environment object to a meaning object. We will give more details shortly, but at least we need not worry that Mng τ is a functor rather than a set of elements.

The idea that each identifier possesses a unique phrase type is workable in some contexts, but it is always unpalatable because it robs the programmer of the ability to bind any identifier to any phrase type. Furthermore, it is not in the spirit of ALGOL-like languages since an identifier should have

the ability to be bound to different phrase types in different blocks. The assignment of types to identifiers should be dynamic. Before program execution begins, no identifier has a type; during execution some have types, and some don't. This leads us to introduce A, the set of phrase type assignments. A phrase type assignment α is a function

$$\alpha \in I \longrightarrow T$$

where $I = \text{dom } \alpha$ is a finite set of identifiers. Thus,

$$\text{emp}_T \in \phi \to T ,$$

the phrase type assignment with empty domain, is the phrase type assignment most appropriate to the beginning of program execution.

Now let L be a desugared version of an ALGOL-like language with block structure and higher-order procedures. Without getting into the exact algebraic nature of L, we wish to be a little more precise about what phases in L denote. Keep in mind that for each phrase type $\tau \in T$, Mng τ is an object of the Cartesian closed category $\Sigma \Rightarrow Pdom$, where Σ is the category of state shapes. To avoid complicated subscripts, let $K = \Sigma \Rightarrow Pdom$. Actually in Chapters VII and VIII we use a slightly different K, but this one is good enough for the time being. We will use Type to stand for the function that assigns phrase types to phrases. However, since a phrase may contain free occurrences of identifiers, a type can be assigned to $\ell \in L$ only in the context of a phrase type assignment $\alpha \in A$. Hence, we see that

$$\text{Type} \in L \to (A \to T).$$

(Thus, it seems that we try occassionally to assign a type to a phrase $\ell \in L$ in a context $\alpha \in A$ that does not give types to all the free identifiers in L. This situation can be handled by introducing a nonsense type $\underline{\text{ns}} \in T$.)

Each phrase type assignment α determines an environment Env α, which is a functor in $\Sigma \to Pdom$, defined by taking the product in K of the (dom α) - indexed collection of functors $\{\text{Mng } (\alpha x) \mid x \in \text{dom } \alpha\}$. If X is a store shape, then Env α X is the product in $Pdom$ of the (dom α)-indexed collection of predomains $\{\text{Mng}(\alpha x)\ X \mid x \in \text{dom } \alpha\}$, which is a "conventional environment."

Let us use Semf to stand for the function which assigns denotations to phrases. A denotation can only be determined in the context of a phrase type assignment and the result will be a morphism (natural transformation) in the functor category K. Hence, we want

1) $\text{Semf} \in L \to (A \to \text{Ar } K)$

and

2) for all $\ell \in L$, $\alpha \in A$,

$$\text{Semf } \ell\ \alpha \in \text{Env } \alpha \xrightarrow[K]{} \text{Mng } (\text{Type } \ell\ \alpha).$$

In particular, the denotation of a phrase $\ell \in L$ in the context of a phrase type assignment $\alpha \in L$ and a store shape $X \in \text{Ob } \Sigma$ is a function from a "conventional environment" to a "conventional collection of meanings," i.e.

$$\text{Semf } \ell\ \alpha\ X \in \text{Env } \alpha\ X \xrightarrow[Pdom]{} \text{Mng } (\text{Type } \ell\ \alpha)\ X.$$

Our final topic in this section is a discussion of how this approach lends itself to an explanation of the interactions between procedures and block structure. However, we must preface this with the definition of the functor

$$\hom^{\Sigma} \varepsilon \Sigma^{op} \to K$$

The usual hom functor is

$$\hom_{\Sigma} \varepsilon \Sigma^{op} \times \Sigma \to Set.$$

Let

$$E \varepsilon Set \longrightarrow Pdom$$

be the embedding functor which gives a discrete partial order to each set. By composing, we get

$$E \circ \hom_{\Sigma} \varepsilon \Sigma^{op} \times \Sigma \longrightarrow Pdom.$$

Then curry to get

$$\hom^{\Sigma} \varepsilon \Sigma^{op} \longrightarrow (\Sigma \Rightarrow Pdom).$$

Thus, for store shapes X and Y, $\hom^{\Sigma}$ X Y is $\hom_{\Sigma}\langle X,Y\rangle$ = $X \xrightarrow[\Sigma]{} Y$, equipped with a discrete partial order.

Let's take a look at a phrase $\ell \varepsilon L$, which in the context of $\alpha \varepsilon A$, has a procedural type; say

$$\text{Type } \ell\ \alpha = \tau \Rightarrow \theta$$

where $\tau, \theta \varepsilon T$. Recall that we have an equality of functors:

$$\text{Mng } (\tau \Rightarrow \theta) = \text{Mng } \tau \Rightarrow \text{Mng } \theta.$$

Therefore,

$$\text{Semf } \ell\ \alpha \varepsilon \text{ Env } \alpha \xrightarrow[K]{} \text{Mng } \tau \Rightarrow \text{Mng } \theta.$$

Suppose the procedure ℓ is defined in a program in a block whose store shape is X. In this context, the denotation of the procedure is

$$\text{Semf } \ell \ \alpha \ X \ \varepsilon \ \text{Env } \alpha \ X \xrightarrow[Pdom]{} (\text{Mng } \tau \Rightarrow \text{Mng } \theta) \ X.$$

A peek ahead at the proof that K is a Cartesian closed category shows that the predomain (Mng τ ⇒ Mng θ) X is obtained by partially ordering the set of natural transformations

$$(\text{hom}^{\Sigma} \ X) \times (\text{Mng } \tau) \xrightarrow[K]{} \text{Mng } \theta.$$

Let e ε Env α X be the conventional environment existing when ℓ is defined. Then

$$\text{Semf } \ell \ \alpha \ X \ e \ \varepsilon \ (\text{hom}^{\Sigma} \ X) \times (\text{Mng } \tau) \xrightarrow[K]{} \text{Mng } \theta.$$

Now suppose the procedure ℓ is called in an interior block whose store shape is Y. Thus, there is an expansion

$$\sigma \ \varepsilon \ X \xrightarrow[\Sigma]{} Y.$$

The denotation of ℓ constructed at the time of its definition should be clearly connected with an element of the predomain Mng τ Y ⇒ Mng θ Y, which is the proper collection of conventional meanings at the time of the call. Note that

$$\text{Semf } \ell \ \alpha \ X \ e \ Y \ \varepsilon \ (X \xrightarrow[\Sigma]{} Y) \times (\text{Mng } \tau \ Y) \xrightarrow[Pdom]{} (\text{Mng } \theta \ Y).$$

Using the fact that *Pdom* is a Cartesian closed category we obtain

$$\text{Ab}_{Pdom}(\text{Semf } \ell \ \alpha \ X \ e \ Y) \ \varepsilon \ (X \xrightarrow[\Sigma]{} Y) \xrightarrow[Pdom]{} (\text{Mng } \tau \ Y \Rightarrow \text{Mng } \theta \ Y).$$

Finally, apply this function to the specific expansion

connecting the shape X of the store at the time of definition to the shape Y of the store at the time of the call, and we obtain

$$Ab_{Pdom}(\text{Semf } \ell\ \alpha\ X\ e\ Y)\ \sigma\ \varepsilon\ \text{Mng}\ \tau\ Y \Rightarrow \text{Mng}\ \theta\ Y,$$

a conventional meaning for the procedure at the time of call. We have captured here two important intuitions about the meanings of procedures:

(1) The environment used to determine the meaning of a procedure is not the environment existing at the moment of the call, but is rather the environment existing when the procedure is defined. (An intuition described by other approaches as well.)

(2) The meaning of a procedure is also dependent on the store existing at the moment of call, the store existing at the moment of definition, and the expansion connecting them.

CHAPTER II

TYPE ALGEBRAS

Our aim in this chapter is the creation of mathematical objects that define collections of phrase types assignable to meaningful program fragments. These objects are certain "type algebras" freely generated by posets whose elements are viewed as primitive phrase types. These free type algegras are adequate for the subsequent discussion, but they don't provide the final answer to the question "What is a type?" In particular we don't delve into the nature of recursively defined phrase types.

It might seem adequate to construct simply a set T of phrase types, and, in the light of Chapter I, a semantic description of a programming language would start by specifying an object Mng τ of a category (e.g. $\mathcal{Set}$ or $\Sigma \Rightarrow Pdom$) for each $\tau \varepsilon T$.

This approach has an important deficiency. We wish to study the role of implicit conversions, i.e. coercions, in semantics. To say there is an implicit conversion from phrase type τ to phrase type θ (a condition expressed succinctly and suggestively as $\tau \leq \theta$) means in part that any phrase of type τ can meaningfully replace any occurrence of a phrase of type θ in a program. This relation is surely reflexive and transitive, and it is difficult to think of

an instance in which it is not antisymmetric. Thus T is not just a set, but rather is a poset. Furthermore, in giving the semantics of a language, one must give a conversion morphism from Mng τ to Mng θ whenever $\tau \leq \theta$. These conversion morphisms must mesh properly with the partial order on T. This meshing is best described by making the poset T into a category (in the usual way that posets become categories) and then requiring that Mng be a functor from T to Set or $\Sigma \Rightarrow Pdom$ or whatever category is used.

We begin by establishing some notation and discussing some construction for posets.

We use ϕ to denote both the empty set and the poset whose underlying set of objects is empty. Similarly we use {a} to denote both the singleton set and the singleton poset whose sole object is a. We say that P is a <u>subposet</u> of Q (notation: $P \subseteq Q$) if Ob $P \subseteq$ Ob Q and Ar $P \subseteq$ Ar Q; in this case the inclusion mapping is monotone and is consequently a functor. If the inclusion functor is full, i.e. for all x, y ε Ob P

$$x \leq_P y \quad \text{iff} \quad x \leq_Q y,$$

then P is a <u>full</u> subposet of Q (notation: $P \sqsubseteq Q$). The <u>intersection</u> $P \cap Q$ of posets P and Q is defined by Ob $(P \cap Q)$ = (Ob P) $\cap$ (Ob Q) and Ar $(P \cap Q)$ = (Ar P) $\cap$ (Ar Q). Thus

$$x \leq_{P\cap Q} y \text{ iff } x \leq_P y \text{ and } x \leq_Q y.$$

Of course, $P \cap Q$ is a subposet of both P and Q. If $P \cap Q = \phi$, then the posets P and Q are <u>disjoint</u>.

Let $\langle P,Q\rangle$ be an ordered pair of posets. Form $P \times_{Cat} Q$ and consider the ordered pair whose first component is the set of objects and whose second component is the set of arrows of $P \times_{Cat} Q$. Alas, it is not in general a poset because

$$\text{Ar } (P \times_{Cat} Q) \subseteq ((\text{Ob } P)\times(\text{Ob } P)) \times ((\text{Ob } Q)\times(\text{Ob } Q))$$

which is not the same as being a subset of

$$(\text{Ob } (P \times_{Cat} Q)) \times (\text{Ob } (P \times_{Cat} Q)) = ((\text{Ob } P)\times(\text{Ob } Q))\times((\text{Ob } P)\times(\text{Ob } Q)).$$

For posets we have an additional, and perhaps more natural, product $P \underline{\times} Q$ given by

(1) $\text{Ob } (P \underline{\times} Q) = (\text{Ob } P) \times (\text{Ob } Q)$, and

(2) $\langle x, y\rangle \leq_{P\underline{\times}Q} \langle x', y'\rangle$ iff $x \leq_P x'$ and $y \leq_Q y'$.

The definition of $\underline{\times}$ has an obvious generalization to more than two factors. As categories, $P \times_{Cat} Q$ and $P \underline{\times} Q$ are isomorphic with the isomorphism being

$$I \in P \times_{Cat} Q \rightarrow P \underline{\times} Q,$$

the functor which is the identity function on objects, and which on arrows satisfies $I\langle x \leq_P x', y \leq_Q y'\rangle = (\langle x,y\rangle \leq_{P\times Q} \langle x',y\rangle)$.

Given two disjoint posets, the utility of connecting them together by providing a common top, i.e., maximal element,

will soon be seen. So suppose P and Q are disjoint posets and $t \notin (\text{Ob } P) \cup (\text{Ob } Q)$. The poset $P*t*Q$ is defined by

$$\text{Ob } (P*t*Q) = (\text{Ob } P) \cup \{t\} \cup (\text{Ob } Q)$$

and

$$x \leq_{P*t*Q} y \quad \text{iff} \quad x \leq_P y \text{ or } y = t \text{ or } x \leq_Q y.$$

Observe that one way to completely specify a functor $F \in P*t*Q \to C$ is to let $F\ t$ = term where term is a terminal object of C and to define functors in $P \to C$ and $Q \to C$ that are the restrictions of F to P and Q.

The first proposition summarizes some evident properties of full subposets.

Proposition 2.1: Suppose $P \sqsubseteq Q$ and $P' \sqsubseteq Q'$.

(1) $P^{op} \sqsubseteq Q^{op}$.

(2) $P \times P' \sqsubseteq Q \times Q'$

(3) If Q and Q' are disjoint, and $t \notin (\text{Ob } Q) \cup (\text{Ob } Q')$. then Q, Q', and $P*t*P'$ are all full subposets of $Q*t*Q'$.

Proof: Trivial. □

A moment's thought shows that the union of two posets P and Q cannot be defined by mimicking the definition of their intersection because $(\text{Ar } P) \cup (\text{Ar } Q)$ is not in general a partial order on $(\text{Ob } P) \cup (\text{Ob } Q)$. However, this simple-minded approach is entirely adequate for dealing with unions of ascending chains of posets, a matter we now take up.

Consider an ascending chain

$$P_0 \subseteq P_1 \subseteq P_2 \subseteq \ldots$$

of posets, which gives rise to a diagram

$$(*) \qquad P_0 \xrightarrow{J_0} P_1 \xrightarrow{J_1} P_2 \xrightarrow{J_2} \cdots$$

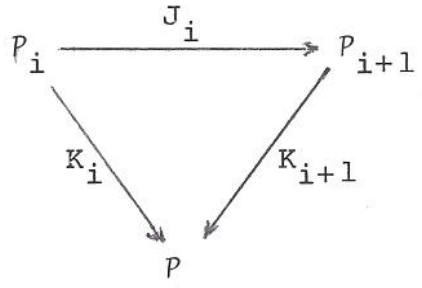
where each J_i is an inclusion mapping. We can form the poset

$$P = \cup \{P_i \mid i \in \mathbb{N}\}$$

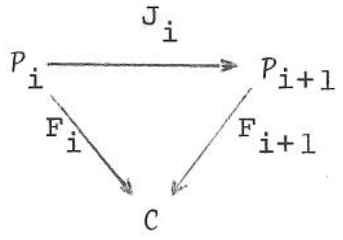
by letting Ob $P = \cup \{\text{Ob } P_i \mid i \in \mathbb{N}\}$ and Ar $P = \cup \{\text{Ar } P_i \mid i \in \mathbb{N}\}$. Thus

$$x \leq_P y \quad \text{iff} \quad x \leq_{P_i} y \quad \text{for some } i \in \mathbb{N}.$$

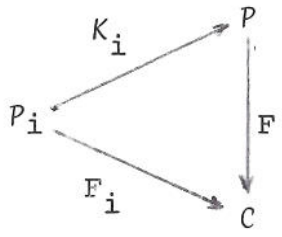
Of course, $P_i \subseteq P$ for each i. Let $K_i \in P_i \longrightarrow P$ be the inclusion mapping, so that



commutes for all i. Then $\langle P, \{K_i\}\rangle$ is a direct limit of the diagram (*) above, i.e., for each category C and each $\mathbb{N}$-indexed collection $F_i \in P_i \to C$ of functors such that



commutes for all i, there exists a unique functor $F \varepsilon P \to C$ such that



commutes for all i. Note that if each P_i is a full subposet of P_{i+1}, then each P_i is a full subposet of P. Also, if

$$Q_0 \subseteq Q_1 \subseteq Q_2 \subseteq \cdots$$

is another ascending chain of posets and

$$Q = \cup \{Q_i \mid i \varepsilon \mathbb{N}\}$$

then

$$P_0 \underline{\times} Q_0 \subseteq P_1 \underline{\times} Q_1 \subseteq P_2 \underline{\times} Q_2 \subseteq \ldots$$

is an ascending chain and

$$P \underline{\times} Q = \cup \{P_i \underline{\times} Q_i \mid i \varepsilon \mathbb{N}\}.$$

We believe that the object T containing the types for a language having coercion among its features should be a poset. If the language is to have a general procedure - definition facility, then for any phrase types τ and θ there should be a type $\tau \Rightarrow \theta$ for procedures which accept arguments (i.e. parameters) of type τ and produce results (i.e. have calls) of type θ. How should $\Rightarrow$

interact with the partial order on T? A procedure of type $\tau \Rightarrow \theta$ should also accept arguments of type τ' if $\tau' \leq \tau$; the result of such a procedure can be used in a context calling for a result of type θ' if $\theta \leq \theta'$. Thus if $\tau' \leq \tau$ and $\theta \leq \theta'$, then we want $\tau \Rightarrow \theta \leq \tau' \Rightarrow \theta'$. So $\Rightarrow$ should be antimonotone in its first argument and montone in its second, or in category-theoretic language $\Rightarrow$ is like a hom functor in that it is contravariant in its first argument and covariant in its second.

We have argued that Mng should be a functor:

$$\text{Mng} \in T \to K,$$

where K is a category with appropriate structure. One thing that K should have is a functor contravariant in its first argument and covariant in its second; we can call this functor $\Rightarrow$, too. We want Mng to preserve $\Rightarrow$, i.e.

$$\text{Mng } (\tau \Rightarrow \theta) = \text{Mng } \tau \Rightarrow \text{Mng } \theta$$

for all phrase types τ and θ. To see how plausible this is, consider the case $K = Set$ with $\Rightarrow$ equal to the ordinary hom functor; then the set of meanings for phrases of type $\tau \Rightarrow \theta$ should be the set of functions from Mng τ to Mng θ.

In order to talk about type-checking we must be able to assign a nonsense type T_{ns} to parseable expressions which contain errors due to mismatched types. Since any expression can be used in a context for which nonsense

suffices, the nonsense type should be the top, i.e. the terminal object, of T. What should be the nature of Mng T_{ns}? Knowing that a phrase has a nonsense meaning amounts to asserting that its meaning contains no information. Thus Mng T_{ns} should be a singleton set when $K = \mathcal{S}et$, and it should be the analogue of a singleton set, i.e. a terminal object, for other possible K.

To make everything elegant, both T and K should be the same sort of objects, and Mng should be a homomorphism from T to K. This serves to motivate the following definitions.

An ordered triple

$$A = \langle |A|, A_{ns}, A_{\Rightarrow} \rangle$$

is a <u>type algebra</u> if

(1) $|A|$ is a category, called the <u>carrier</u> of A,

(2) A_{ns} is a terminal object of $|A|$, called the <u>nonsense</u> object of the algebra, and

(3) $A_{\Rightarrow} \in |A|^{op} \times |A| \to |A|$.

Alternative notations for $A_{\Rightarrow} \langle a,b \rangle$ are $a \overset{A}{\Longrightarrow} b$ and, when A is readily determined from context, $a \Rightarrow b$. Cartesian closed categories give the most accessible examples of type algebras. Thus, from a Cartesian closed category K we obtain a type algebra A by letting $|A| = K$, $A_{ns} = K_{term}$, and $A_{\Rightarrow} = K_{\Rightarrow}$.

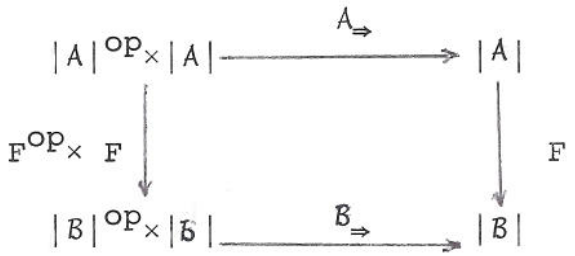
A <u>type algebra homomorphism</u> consists of a domain A

and a codomain $\mathcal{B}$, both of which are type algebras, and a functor F (notation: $F \in A \xrightarrow[Type\ Alg]{} \mathcal{B}$) such that

(1) $F \in |A| \to |\mathcal{B}|$,

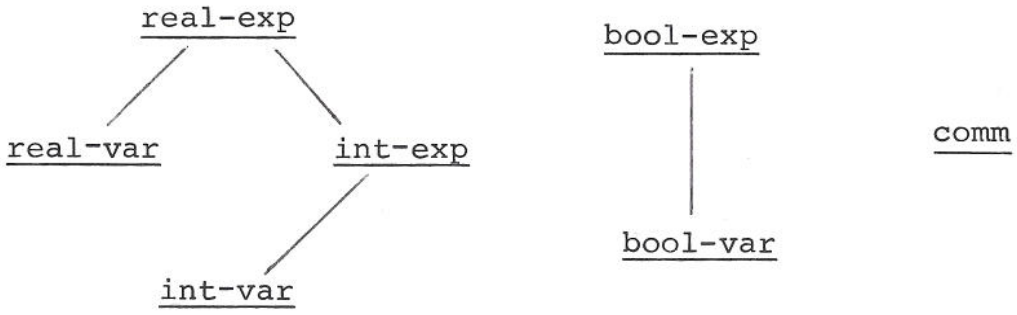
(2) $F\ A_{ns} = \mathcal{B}_{ns}$, and

(3) the diagram



commutes.

For the purpose of assigning phrase types to program fragments we need a type algebra whose carrier is a poset. Generally we expect to start with a poset P of primitive phrase types. For instance, we might take P to be the poset described by the following Hasse diagram.



Here exp, var, int, bool, and comm are abbreviations for expression, variable, integer, boolean, and command, respectively. A complete discussion of posets such as these

including motivation for the structure of the Hasse diagram above may be found in Reynolds [8]. However, the central ideas are that variables may be coerced into expressions, integers may be coerced into reals, boolean phrase types cannot be coerced into real or integer phrase types, and commands cannot be coerced into other phrase types. Of course, P is not a type algebra since it has no procedural types and no nonsense type. However, the following theorem asserts that each poset P generates a free type algebra T whose carrier is a poset. It is the free type algebra generated by the poset of primitive phrase types which provides the phrase types for program fragments.

A few comments on this theorem are in order. In the theorem Q is the poset of procedural phrase types. The theorem says that the set underlying the carrier of T is the disjoint union of the set of primitive phrase types, the singleton consisting of the nonsense type, and the procedural types. If τ is a procedural phrase type, then there are uniquely determined phrase types θ and θ' such that $\tau = \theta \Rightarrow \theta'$ because $T_{\Rightarrow}$ is injective on objects. Therefore, if we ignore the partial order, T is a free (universal) algebra generated by Ob P where the signature (i.e.ranked set of operators) consists of a single operator ns of rank 0 and a single operator $\Rightarrow$ of rank 2. In a different vein, notice how easy it will be to define the type algebra homomorphism

$$\text{Mng} \in T \xrightarrow[\textit{Type Alg}]{} K \ ;$$

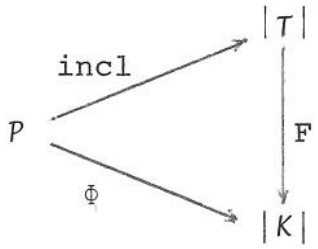
we need only give the functor which is the restriction of Mng to P.

The proof of the theorem is an exercise in the exploitation of the properties of unions of ascending chains of posets.

Theorem 2.2: Let P be a poset. There is a type algebra T such that

(1) $|T|$ is a poset, $T_{\Rightarrow} \in |T|^{op} \times |T| \to |T|$ is a full and faithful functor, $|T| = P^* \, T_{ns} \, * \, Q$, where Q is the image of $T_{\Rightarrow}$, and

(2) T satisfies the following universal mapping property: let K be a type algebra, let $\Phi \in P \to |K|$, and let incl $\in P \to |T|$ be the inclusion functor; then there exists a unique type algebra homomorphism $F \in T \xrightarrow[\textit{Type Alg}]{} K$ such that



commutes.

Proof: Let t be such that $t \notin$ Ob P, and let a be such that no element of (Ob P) $\cup \{t\}$ is an ordered triple <a, x, y> whose first component is a.

We start by constructing inductively a sequence

$$Q_0, T_0, Q_1, T_1, Q_2, T_2, \ldots$$

of posets such that, for all i, $P \cap Q_i = \phi$. Let

$$Q_0 = \phi.$$

Assuming that the sequence is constructed through Q_i, let

$$T_i = P * t * Q_i.$$

Assuming that the sequence is constructed through T_i, let

$$Q_{i+1} = \{a\} \times T_i^{op} \times T_i.$$

We claim that

$$Q_0 \sqsubseteq Q_1 \sqsubseteq Q_2 \sqsubseteq \ldots$$

and

$$T_0 \sqsubseteq T_1 \sqsubseteq T_2 \sqsubseteq \ldots .$$

Obviously $Q_0 \sqsubseteq Q_1$. Suppose

$$Q_0 \sqsubseteq \cdots \sqsubseteq Q_{i-1} \sqsubseteq Q_i$$

and

$$T_0 \sqsubseteq \cdots \sqsubseteq T_{i-1} .$$

It follows from Proposition 2.1 that

$$T_{i-1} = P * t * Q_{i-1} \sqsubseteq P * t * Q_i = T_i .$$

Suppose

$$Q_0 \sqsubseteq \cdots \sqsubseteq Q_i$$

and

$$T_0 \sqsubseteq \ldots \sqsubseteq T_i .$$

Use Proposition 2.1 again to see

$$Q_i = \{a\} \times T_{i-1}^{op} \times T_{i-1} \sqsubseteq \{a\} \times T_i^{op} \times T_i = Q_{i+1} .$$

This proves the claim.

Let $|T| = \cup \{T_i \mid i \in \mathbb{N}\}$ and $Q = \cup \{Q_i \mid i \in \mathbb{N}\}$.

Clearly $Q = \{a\} \times |T|^{op} \times |T|$. By the choice of a, $P \cap Q = \phi$, and, by the choice of t, $t \notin (\text{Ob } P) \cup (\text{Ob } Q)$. It is now apparent that Ob $|T|$ is a disjoint union:

$$\text{Ob } |T| = (\text{Ob } P) \cup \{t\} \cup (\text{Ob } Q) .$$

Also,

$$\begin{aligned} \tau \leq_{|T|} \theta \quad &\text{iff} \quad \tau \leq_{T_i} \theta \quad \text{for some i} \\ &\text{iff} \quad \tau \leq_P \theta \text{ or } t=\theta \text{ or } \tau <_{Q_i} \theta \text{ for some i} \\ &\text{iff} \quad \tau \leq_P \theta \text{ or } t=\theta \text{ or } \tau \leq_Q \theta \end{aligned}$$

Therefore, $|T| = P * t * Q$.

For each i, let $H_i \in T_i^{op} \times T_i \to Q_{i+1}$ be the obvious poset isomorphism given by

$$H_i \langle \tau, \theta \rangle = \langle a, \tau, \theta \rangle .$$

Consider the commutative diagram

$$\begin{array}{ccccccc} T_0^{op} \times T_0 & \longrightarrow & T_1^{op} \times T_1 & \longrightarrow & T_2^{op} \times T_2 & \longrightarrow & \ldots \\ \downarrow H_0 & & \downarrow H_1 & & \downarrow H_2 & & \\ Q_0 & \longrightarrow & Q_1 & \longrightarrow & Q_2 & \longrightarrow & \ldots , \end{array}$$

where the horizontal maps are inclusions. After noting that

$$|T|^{op}\underline{\times}\,|T| = \cup\{T_i^{op}\underline{\times}\,T_i \mid i\in\mathbb{N}\},$$

we assert that the poset isomorphism $H \in |T|^{op}\times |T| \to Q$ given by $H\langle x, y\rangle = \langle a, x, y\rangle$ is the only functor such that

$$\begin{array}{ccc} T_i^{op}\underline{\times}T_i & \xrightarrow{\text{incl}} & |T|^{op}\underline{\times}\,|T| \\ \downarrow{\scriptstyle H_i} & & \downarrow{\scriptstyle H} \\ Q_{i+1} & \xrightarrow{\text{incl}} & Q \end{array}$$

commutes for all i. Since H is an isomorphism it is full and faithful.

For $\tau, \theta \in \text{Ob}\ |T|$, we want the equation

$$T_{\Rightarrow}\langle\tau, \theta\rangle = \langle a, \tau, \theta\rangle$$

to hold; so define $T_{\Rightarrow} \in |T|^{op}\times |T| \to |T|$ so that

$$\begin{array}{ccc} |T|^{op}\times |T| & \xrightarrow{T_{\Rightarrow}} & |T| \\ \downarrow{\scriptstyle I} & & \uparrow{\scriptstyle \text{incl}} \\ |T|^{op}\underline{\times}\,|T| & \xrightarrow{H} & Q \end{array}$$

commutes, where I is the canonical isomorphism and incl is the inclusion functor. Clearly, the image of $T_{\Rightarrow}$ is Q, and $T_{\Rightarrow}$, being a composition of full, faithful functors, is itself full and faithful.

Since t is a terminal object of $|T|$, we may let $T_{ns} = t$. Therefore $|T| = P*T_{ns}*Q$. This proves the first part of the theorem.

For the second part, let K be a type algebra, and let $\Phi \in P \to |K|$. We will define a sequence of functors $G_0, F_0, G_1, F_1, G_2, F_2, \ldots$ such that

$$G_i \in Q_i \to |K| \text{ and } F_i = T_i \to |K|.$$

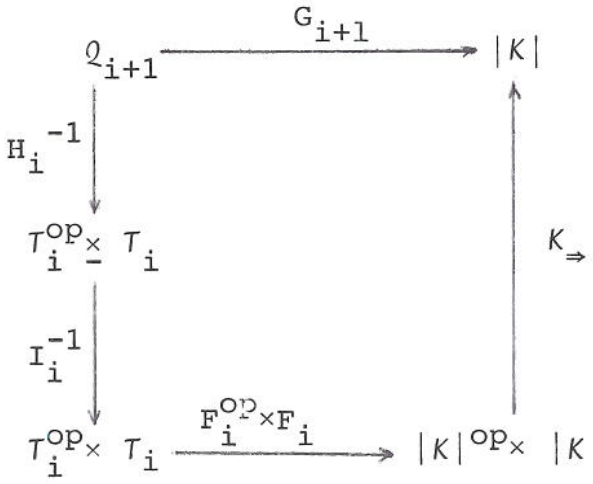
Since $Q_0 = \phi$, the functor G_0 is uniquely determined. Suppose we have constructed all functors through G_i. Recall $T_i = P*t*Q_i$, and let F_i be the unique functor satisfying

(1) $F_i t = K_{ns}$,

(2) the restriction of F_i to P is Φ, and

(3) the restriction of F_i to Q_i is G_i.

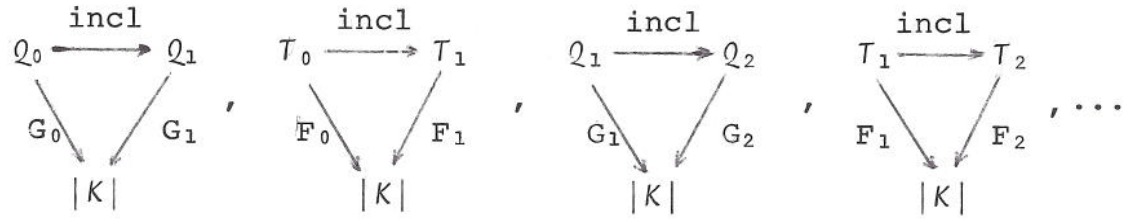
Suppose we have constructed all functors through F_i. By $I_i \in T_i^{op} \times T_i \to T_i^{op} \underline{\times} T_i$ we mean the canonical isomorphism. Let G_{i+1} be such that



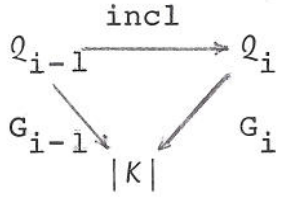
commutes. Thus for τ, $\theta \in \text{Ob } T_i$,

$$G_{i+1}\langle a,\tau,\theta\rangle = K_{\Rightarrow}\langle F_i^{op}\,\tau,\ F_i\theta\rangle.$$

Next, we claim that the diagrams in the sequence



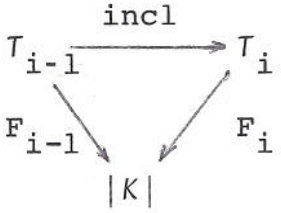
are all commutative. The first diagram commutes because $Q_0 = \phi$. Suppose all diagrams commute through



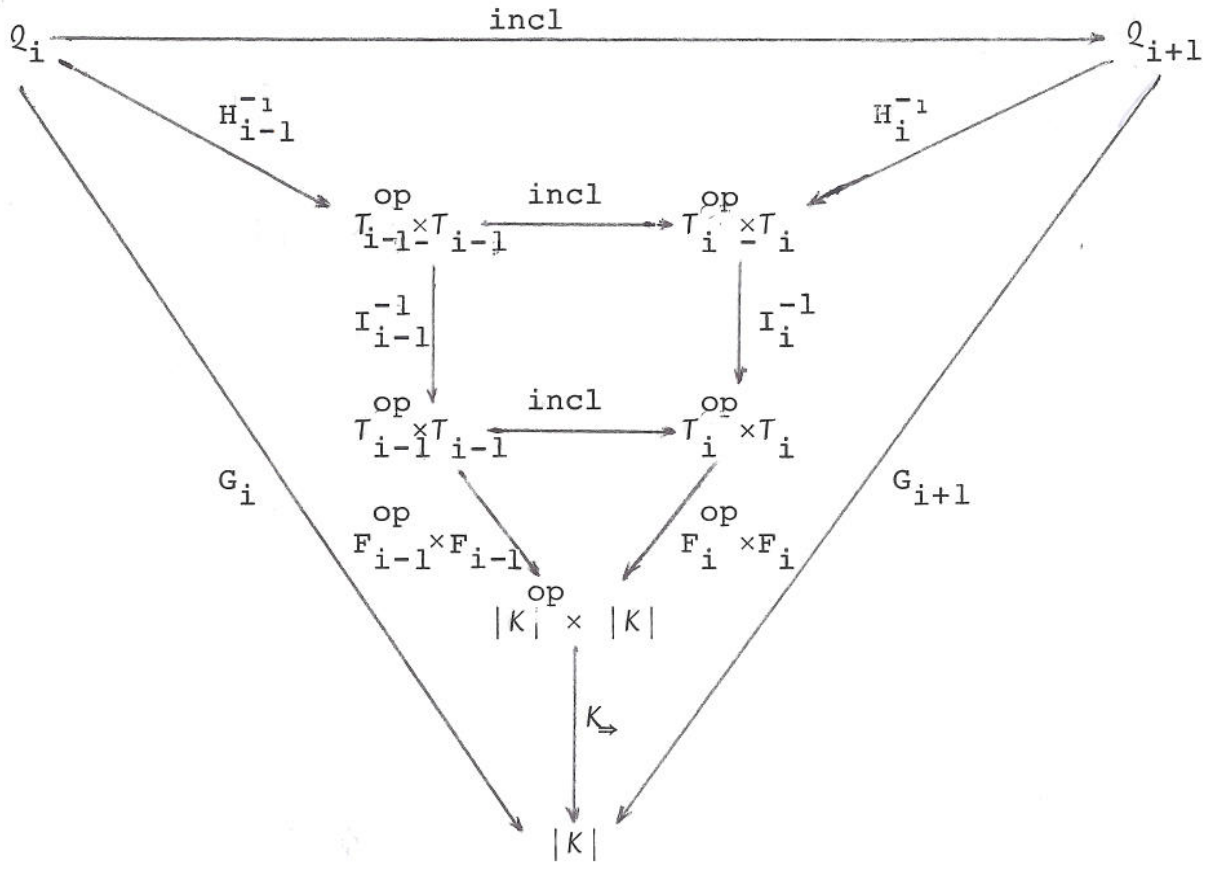
;

then the commutativity of

(**)

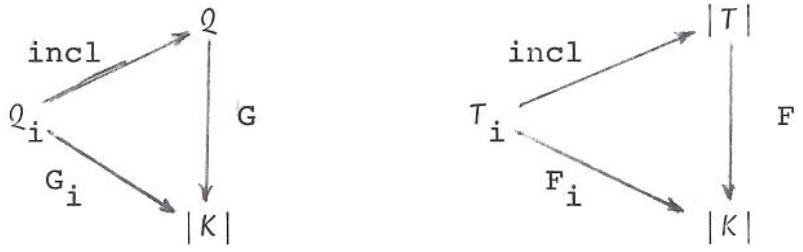


is immediate from the definitions of F_{i-1} and F_i. Suppose all the diagrams through (**) commute; then from the commutativity of all the inner diagrams in



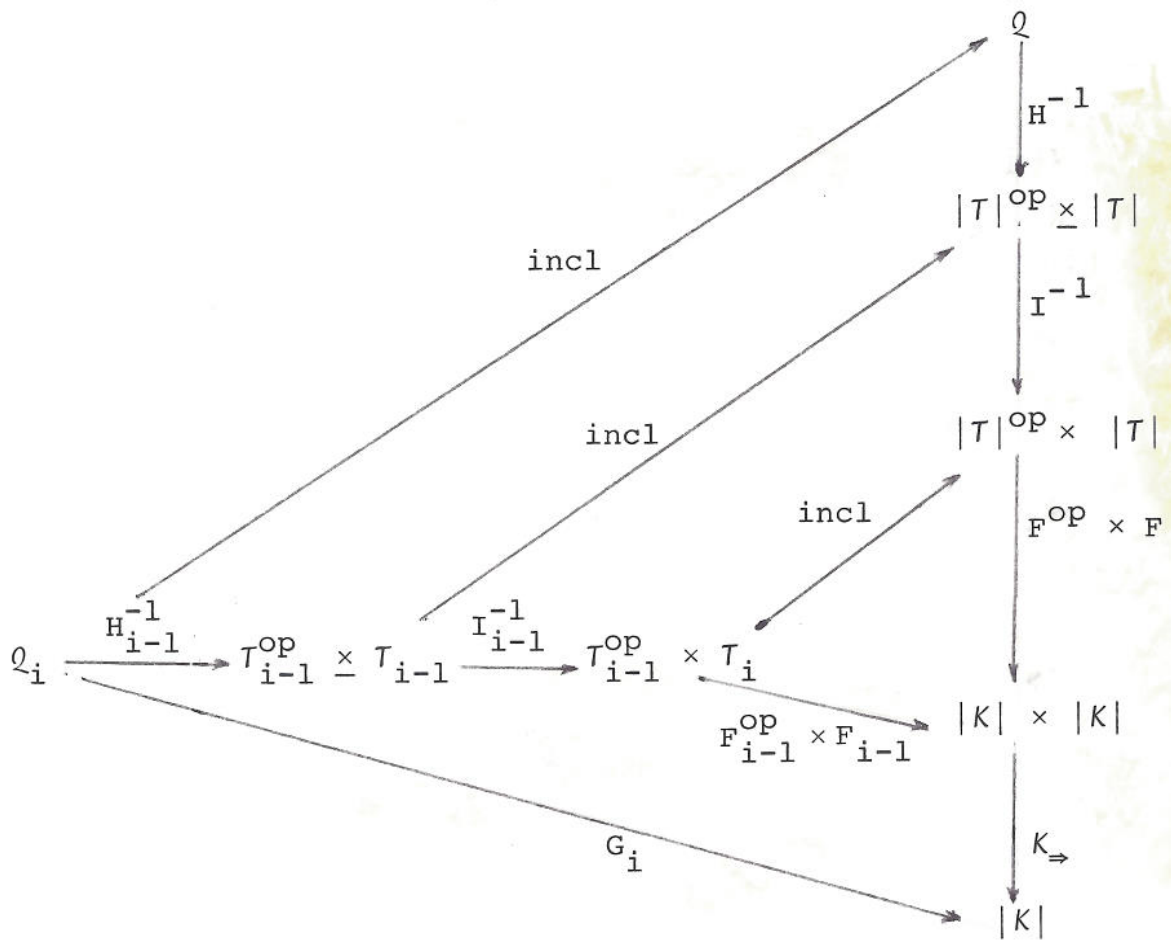
we get commutativity of the outer triangle.

By applying the earlier remarks on direct limits of ascending chains of posets, we see there exist unique functors G and F such that for all i the diagrams

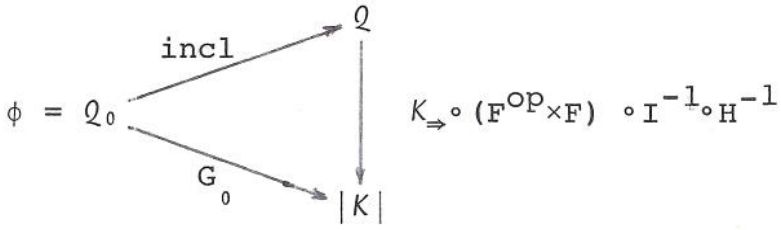


commute. The functors G and F are related by two commutative

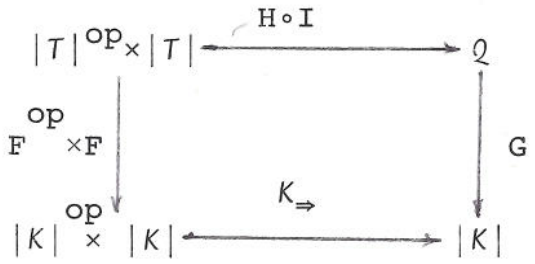
diagrams. To obtain the first diagram, note that for each i>0 the diagram



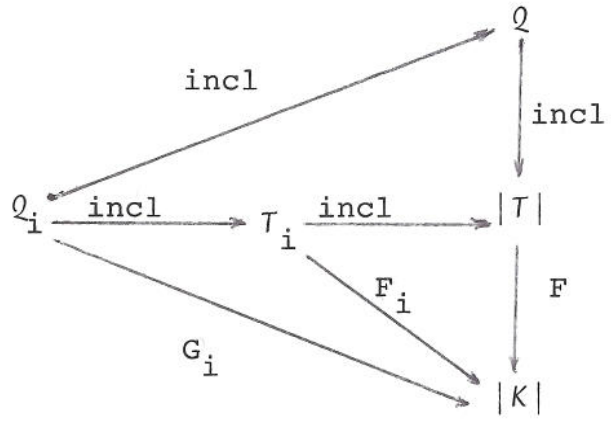
commutes, and



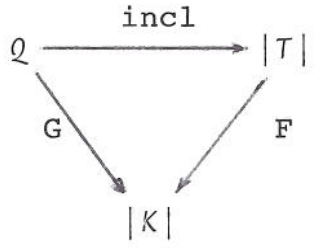
is trivially commutative. Therefore the composite vertical arrow must equal G, i.e.



commutes. Also, for each i the diagram



commutes. Again, the composite vertical arrow must equal G, and we get a commutative diagram



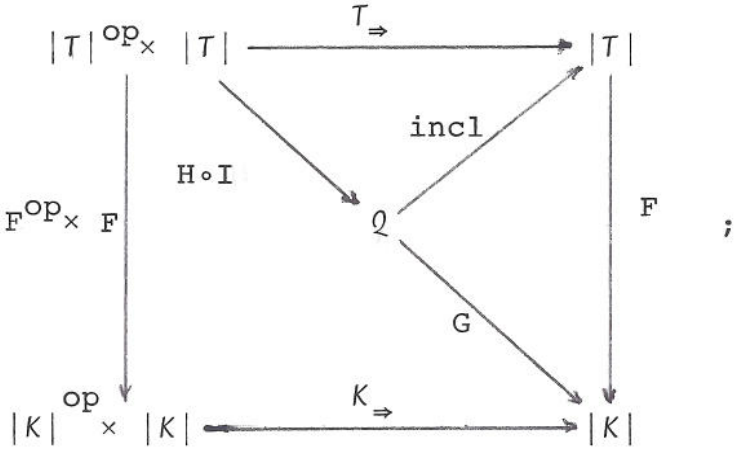
.

To conclude that F is a type algebra homomorphism, there are three conditions that must be checked. First, from the definition of F it is clear that F ε |T| → |K|. Second,

since F extends $F_0 \in T_0 \to |K|$ we easily compute

$$F\ T_{ns} = F_0\ t = K_{ns}.$$

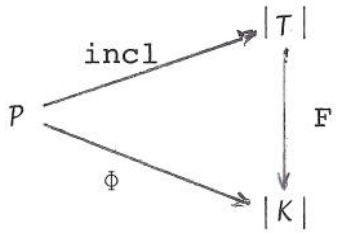
Finally, we must verify that the outer square is commutative in the diagram



and this follows because all the inner diagrams commute. Thus

$$F \in T \xrightarrow[Type\ Alg]{} K\ .$$

Notice that



commutes because F extends F_0 which in turn extends Φ.

Finally we must verify the uniqueness of F. Let

$$F' \in T \xrightarrow[Type\ Alg]{} K$$

be an arbitrary type algebra homomorphism whose restriction to P is Φ, and let

$$G'_0, F'_0, G'_1, F'_1, G'_2, F'_2, \ldots$$

be the restrictions of F'to

$$Q_0, T_0, Q_1, T_1, Q_2, T_2, \ldots,$$

respectively. We will show by induction that

$$G'_0 = G_0, F'_0 = F_0, G'_1 = G_1, F'_1 = F_1, G'_2 = G_2, \ldots.$$

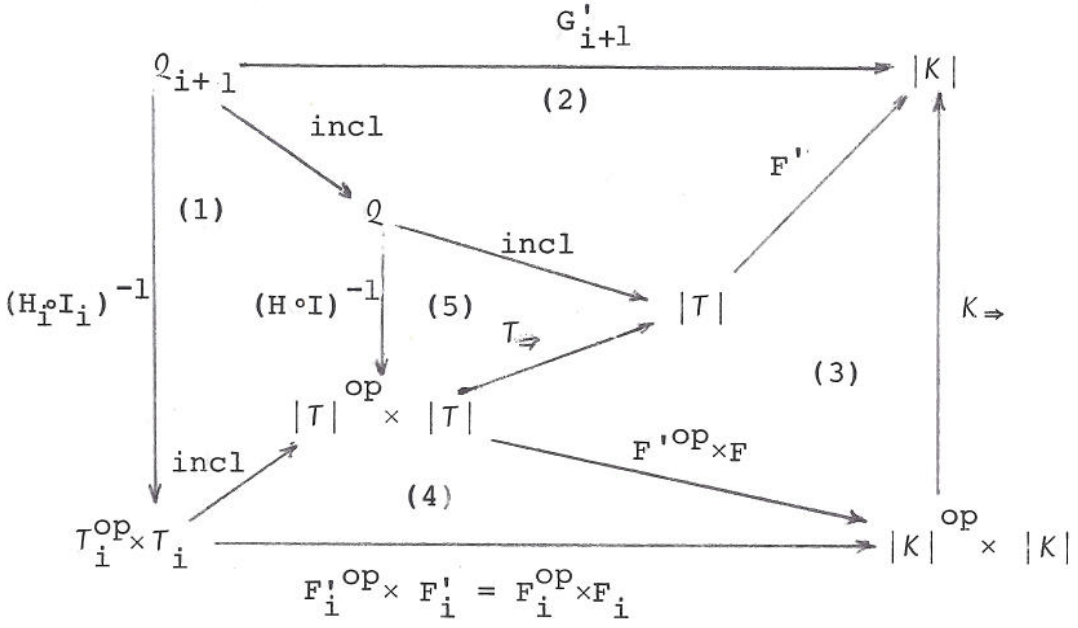
Since $Q_0 = \phi$, $G'_0 = G_0$. Suppose all the equalities through $G'_i = G_i$ are true. We make three observations.

(1) $F'_i T_{ns} = F'T_{ns} = K_{ns}$ since F' is a homomorphism.

(2) The restriction of F'_1 to P is the restriction of F' to P, which is Φ.

(3) The restriction of F'_i to Q_i is the restriction of F' to Q_i, which is $G'_i = G_i$.

By the uniqueness of the construction of F_i, we get $F'_i = F_i$. Now suppose all the equalities through $F'_i = F_i$ are true. Observe that in the following diagram all inner diagrams commute.



The reasons for commutativity of various diagrams is as follows:

(1) definition of H,

(2) G'_{i+1} is the restriction of F' to Q_{i+1},

(3) F' is a type algebra homomorphism,

(4) F'_i is the restriction of F' to T_i, and

(5) definition of $T_\Rightarrow$.

Thus the outer square commutes. By the definition of G_{i+1}, we see that $G'_{i+1} = G_{i+1}$. Therefore all the equalities in the sequence are valid. Since $|T| = \cup \{T_i \mid i \in \mathbb{N}\}$ and the restriction of F' to T_i in all cases equals the restriction of F to T_i, we conclude $F' = F$. □

CHAPTER III

SYNTAX, SUBSTITUTION, AND TYPE-CHECKING

In this chapter we will attempt to deal with the essentially syntactic problems of language definition, substitution, and type-checking.

First we must properly define the class of languages that are the subject of this work. On an intuitive level, a language consists of all combinations of symbols which can be parsed. Just as a natural language has many nonsensical or ill-defined sentences, so also a programming language should contain nonexecutable or meaningless expressions. This enables us to illuminate the nature of meaningful expressions by comparing them with similar nonsensical ones. Thus we will be able to address in general terms the question of why the assignment command

x := 3

is meaningful in some contexts, whereas

3 := x

is not. It is difficult to even pose this question if 3 := x is not an expression of the language. Of course, a desirable feature of an implementation of a language is a mechanism, generally a type-checking facility, which identifies nonsensical programs at compile time. The final portion of this chapter is devoted to an exposition of the view that type-checking is nothing more than the application of a

simply defined homomorphism to a phrase.

For a programming language we consider an expression to be parsable if its applicative structure can be determined. Thus, we are led to study typed λ-calculi. However, the languages are nonstandard in the sense that in each there is a single set of identifiers rather than a set of identifiers for each phrase type. This serves to intensify their similarity to implemented languages, particularly those of the ALGOL family, which permit different occurrences of an identifier in a program to have different types. Each language is syntactically determined by three parameters:

(1) P, the poset of primitive phrase types,

(2) Id, an infinite set of identifiers, and

(3) G, a set called the generating set of the language, whose elements are termed linguistic constants.

These three parameters suffice for language definition, but other parameters must be specified in order to deal with substitution and type-checking in this chapter and with semantics in the next. The resulting language is called a coercive typed λ-calculus.

In the remainder of this chapter we assume that the poset P, the infinite set Id, and the set G are all fixed. We now turn to the details of how a language L arises from P, Id, and G.

As shown in the previous chapter, the poset P of primitive phrase types generates a free type algebra

$T = \langle |T|, T_{ns}, T_{\Rightarrow} \rangle$. We will use Ob $|T|$ as the collection of all phrase types for L. For ease of reading, we usually write $\tau \varepsilon T$ rather than $\tau \varepsilon$ Ob $|T|$, $\tau \leq \theta$ rather than $\tau \leq_{|T|} \theta$, ns rather than T_{ns}, and $\tau \Rightarrow \theta$ rather than $T_{\Rightarrow}\langle \tau, \theta \rangle$.

At this point we should say a word about our notation for one-sorted algebras. A signature Σ is just a ranked set; thus each $\sigma \varepsilon \Sigma$ is assigned a rank or arity which is non-negative integer. The elements of Σ are called operators, and the set of operators of arity n is denoted Σ_n. In this setting, a Σ-algebra

$$A = \langle |A|, \{A_\sigma \mid \sigma \varepsilon \Sigma\} \rangle$$

is formally a pair consisting of a set $|A|$, the carrier of the algebra, and a collection $\{A_\sigma \mid \sigma \varepsilon \Sigma\}$ of interpretations of the operators in Σ such that

$$A_\sigma \varepsilon |A|^n \to |A|,$$

if $\sigma \varepsilon \Sigma_n$. We usually write $x \varepsilon A$ rather than $x \varepsilon |A|$. Homomorphisms of Σ-algebras are defined in the obvious way. For details, see Cohn [2].

From T and Id we get a signature Λ for ordinary one-sorted algebras. The collection of 0-ary operators of Λ is Id. For each $\tau \varepsilon T$ and $x \varepsilon$Id there is a single unary operator denoted $\lambda x{:}\tau$. Finally there is a single binary operator denoted *ap*. Thus

$$\Lambda_0 = \text{Id}, \quad \Lambda_1 = \{\lambda x{:}\tau \mid x \varepsilon \text{Id}, \tau \varepsilon T\}, \quad \Lambda_2 = \{ap\}.$$

The role of the 0-ary operators is to insure that identifiers

are expressions. The unary operator $\lambda x{:}\tau$ creates procedures (λ-expressions) designed to be meaningfully applied to arguments which can be coerced into type τ. The operator *ap* enables us to freely apply any expression to any other.

Let

$$L = \langle |L|, \{L_\omega \mid \omega \in \Lambda\} \rangle$$

be a free Λ-algebra generated by G, the generating set of linguistic constants. For example, suppose int-exp is a primitive phrase type, $x \in Id$, and 0,1,+ are all in G. An expression of L is

$$L_{ap} < L_{\lambda x:\underline{int}\text{-}\underline{exp}} \ (L_{ap} < L_{ap} <+,1>, L_x>) \ , \ 0 >,$$

which is a desugared version of

$$(\lambda x:\underline{int}\text{-}\underline{exp} \ (1+x)) \ 0 \ .$$

(We are viewing + as a curried operator that applies to an integer and returns a function from integers to integers.)

In the interest of clarity we shall use a conventional λ-calculus notation for elements of L, and we will call attention to occurrences of this conventional notation with emphatic brackets. Thus, if $x \in Id$, $\tau \in T$, $\ell \in L$ and $m \in L$, then

$$[\![x]\!] \text{ stands for } L_x \ ,$$
$$[\![\lambda x{:}\tau.\ell]\!] \text{ stands for } L_{\lambda x:\tau} \ \ell \ , \text{ and}$$
$$[\![\ell m]\!] \text{ stands for } L_{ap} <\ell, m> \ .$$

An important notion in any treatment of the λ-calculus is the set of identifiers that occur free in an expression.

The first use we make of the algebraic structure of L is to cast the assignment of sets of free identifiers to expressions as a Λ-algebra homomorphism. Define the Λ-algebra F by letting

$$|F| = \text{set of finite subsets of Id},$$

and defining

$$\begin{aligned} F_x &= \{x\} && \text{, where } x \varepsilon \mathrm{Id}, \\ F_{ap}\langle F, H\rangle &= F \cup H && \text{, where } F, H \varepsilon F, \text{ and} \\ F_{\lambda x:\tau}\ F &= F - \{x\} && \text{, where } x \varepsilon \mathrm{Id},\ \tau \varepsilon T,\ F \varepsilon F. \end{aligned}$$

There are no free occurrences of any identifiers in any linguistic constant. This leads us to define the Λ-algebra homomorphism

$$\text{Free} \varepsilon L \xrightarrow[\Lambda\text{-Alg}]{} F$$

to be the unique homomorphism such that, for all $g \varepsilon G$,

$$\text{Free } g = \emptyset .$$

The following theorem then shows that Free behaves exactly as it should.

Theorem 3.1: Free is the unique function such that for all $g \varepsilon G$, $x \varepsilon \mathrm{Id}$, $\tau \varepsilon T$, $\ell \varepsilon L$, $m \varepsilon L$ the following four equations hold:

$$\begin{aligned} &\text{(i)} && \text{Free } g = \emptyset , \\ &\text{(ii)} && \text{Free } [\![x]\!] = \{x\} , \\ &\text{(iii)} && \text{Free } [\![\ell\ m]\!] = (\text{Free } \ell) \cup (\text{Free } m), \\ &\text{(iv)} && \text{Free } [\![\lambda x{:}\tau.\ell]\!] = (\text{Free } \ell) - \{x\} . \end{aligned}$$

Proof: According to the preceding description of Free, (i) holds by definition and (ii)-(iv) follow because Free is a homomorphism. Conversely, any function satisfying (ii)-(iv) is a homomorphism, and Free is the only homomorphism satisfying (i). □

The next topic to be explored is the effect on the treatment of substitution caused by the view that a language is a free Λ-algebra. Basically substitution involves the creation of a new phrase from an assignment of phrases to identifiers. It is complicated by the fact that bound occurrences of identifiers are insulated from the effects of the substitution unless a name change is necessitated to avoid binding an occurrence which should be free.

A phrase assignment is a function $f \in F \to |L|$ whose domain is a finite set of identifiers, i.e., $F \in \mathcal{F}$. The collection of all phrase assignments is denoted $|L|^*$. A particularly noteworthy phrase assignment is the empty phrase assignment whose domain is the empty set; it is denoted by

$$emp_L \in \emptyset \to |L| .$$

Use of the empty phrase assignment to determine a function from $|L|$ to $|L|$ by substitution should yield the identity function.

A function $f \in Id \to |L|$, whose domain is the set of all identifiers, is called a comprehensive phrase assignment. The set of all comprehensive phrase assignments is denoted $|L|^\infty$. Each $f \in |L|^*$ determines a canonical $f^\infty \in |L|^\infty$ via the equation

$$f^{\infty}x \quad = \begin{cases} fx & \text{if} \quad x \in \text{dom } f \\ [\![x]\!] & \text{if} \quad x \notin \text{dom } f \ . \end{cases}$$

Thus, emp_L^{∞} resembles the identity function because, for all $x \varepsilon Id$,

$$emp_L^{\infty}\, x = [\![x]\!] \ .$$

In order to describe the condition which determines when a substitution involves a change of bound identifiers, we introduce

$$\text{clashset} \ \varepsilon \ |L|^{*} \times Id \times |L| \to |F|$$

which is defined by

$$\text{clashset} \langle f,x,\ell \rangle = \cup \{\text{Free}\ (f^{\infty}y) \mid y \in \text{Free}\ \ell \text{ and } y \neq x\}$$

The idea is this: our intuition tells us that the substitution associated with a phrase assignment f will involve changing the bound identifier x in $[\![\lambda x{:}\tau.\ell]\!]$ if $x \in \text{clashset} \langle f, x, \ell \rangle$. Note that

$$\text{clashset} \langle f,x,\ell \rangle = \cup\{\text{Free}\ (f^{\infty}y) \mid y \in \text{Free}[\![\lambda x{:}\tau.\ell]\!]\}$$

In the remainder of this chapter, and in the next chapter, we assume that the set Id of identifiers comes equipped with a function

$$\text{newid} \ \varepsilon \ |F| \to Id$$

such that, for any finite set F of identifiers,

$$\text{newid}\ F \notin F \ .$$

The function newid provides a mechanism for changing bound identifiers when required.

Our goal is to define and illuminate a function

$$\text{Sub} \in |L|^* \to |L| \to |L|$$

that accepts a phrase assignment and produces a substitution function which maps phrases to phrases. Certainly, three properties we expect of substitution are that it does not change constants, it replaces phrases consisting of a single identifier with an appropriate expression, and it acts on an application by acting on the operator and the operand separately. The action of a substitution function on λ-expressions is more delicate. As is well known, sometimes renaming of bound identifiers is necessary to avoid identifier collisions. For instance, if a phrase assignment maps y to a phrase which has x among its free identifiers, then the result of applying the associated substitution function to $[\![\lambda x{:}\tau.\ [\![y]\!]]\!]$ should involve changing x to another identifier. Furthermore, in our approach to substitution we opt never to do unnecessary renaming. Thus, we expect Sub to be found among the members of the set of functions $S \in |L|^* \to |L| \to |L|$ such that, for all $f \in |L|^*$, the following conditions hold:

(Sub1) For all $g \in G$, $S\ f\ g = g$.

(Sub2) For all $x \in Id$, $S\ f\ [\![x]\!] = f^{\infty} x$.

(Sub3) For all $\ell, m \in L$

$$S\ f\ [\![\ell m]\!] = [\![(S\ f\ \ell)\ (S\ f\ m)]\!] .$$

(Sub4) For all $x \varepsilon Id$, $\tau \varepsilon T$, $\ell \varepsilon L$,

$$S\ f\ [\![\lambda x{:}\tau.\ell]\!] = \begin{cases} [\![\lambda x{:}\tau.\ (S\ [f \mid x{:}[\![x]\!]]\ \ell)\ ]\!] \\ \qquad \text{if } x \notin \text{clashset } \langle f, x, \ell \rangle \\ [\![\lambda z{:}\tau.\ (S\ [f \mid x{:}[\![z]\!]]\ \ell)\ ]\!] \\ \qquad \text{otherwise, where} \\ \quad z = \text{newid (clashset } \langle f, x, \ell \rangle). \end{cases}$$

We have a general method for defining Λ-algebra homomorphisms whose domain is L. Do we expect Sub f to be a Λ-algebra homomorphism from L to L for each $f \varepsilon |L|^*$? The answer is no, because an affirmative answer would imply that for $x \varepsilon Id$, $\tau \varepsilon T$, $\ell \varepsilon L$,

$$\text{Sub f } [\![\lambda x{:}\tau.\ell]\!] = [\![\lambda x{:}\tau.(\text{Sub f } \ell)]\!]\ ,$$

an equation at odds with (Sub4). In fact, the complexity of condition (Sub4) makes it impossible to simply fix $f \varepsilon |L|^*$ and then define Sub f by induction on the structure of its argument because Sub f $[\![\lambda x{:}\tau.\ell]\!]$ is not expressed in terms of Sub f ℓ.

The class of functions satisfying the above four conditions is not too bizarre, as the next proposition shows.

Proposition 3.2: Suppose $S \varepsilon |L|^* \to |L| \to |L|$ satisfies (Sub1), (Sub2), (Sub3), and (Sub4). Then S emp_L is the identity function on $|L|$.

Proof: By induction on the structure of $\ell \varepsilon L$ we will show that, for all $f \varepsilon |L|^*$ such that $f^\infty = emp_L^\infty$, S f $\ell = \ell$.

From this the theorem follows.

(1) Let $\ell = g$ where $g \varepsilon G$. Then S f g = g by (Sub1).

(2) Let $\ell = [\![x]\!]$ where $x \varepsilon Id$. If $f^{\infty} = emp_L^{\infty}$, then S f $[\![x]\!] = f^{\infty}\ x = emp_L^{\infty}\ x = [\![x]\!]$ by (Sub2).

(3) Let $\ell = [\![mn]\!]$ where $m, n \varepsilon L$. If $f^{\infty} = emp_L^{\infty}$, then

S f $[\![mn]\!]$ = $[\![$(S f m) (S f n)$]\!]$, by (Sub3)

= $[\![mn]\!]$, by the induction hypothesis.

(4) Let $\ell = [\![\lambda x{:}\tau.m]\!]$ where $x \varepsilon Id$, $\tau \varepsilon T$, $m \varepsilon L$. Suppose $f^{\infty} = emp_L^{\infty}$. Then $x \notin$ clashset <f, x, m> because clashshet <f, x, m>

= $\cup\{$Free $(f^{\infty}\ y) \mid y \varepsilon$ Free m, $y \neq x\}$

= $\cup\{$Free $[\![y]\!] \mid y \varepsilon$ Free m, $y \neq x\}$

= $\{y \mid y \varepsilon$ Free m, $y \neq x\}$.

Therefore

S f $[\![\lambda x{:}\tau.m]\!]$ = $[\![\lambda x{:}\tau.$ (S [f | x:$[\![x]\!]$] m)$]\!]$

= $[\![\lambda x{:}\tau.m]\!]$,

because of the induction hypothesis and

[f | x:$[\![x]\!]$]$^{\infty}$ = $f^{\infty} = emp_L^{\infty}$.

□

It should come as no surprise that there is a Λ-algebra homomorphism connected with substitution. Its codomain, the Λ-algebra H, is suggested by observing that there is a one-to-one correspondence between $|L|^* \to |L| \to |L|$ and

$|L| \to |L|^* \to |L|$. Thus, we take the carrier of H to be

$$|H| = |L|^* \to |L|,$$

and we describe the interpretations of the operators by the equations

$$H_x\ f = f^\infty\ x \text{ where } x \varepsilon Id,\ f \varepsilon |L|^*,$$

$$H_{ap} \langle h, j \rangle\ f = [\![(hf)(jf)]\!] \text{ where } h, j \varepsilon H,\ f \varepsilon |L|^*,$$

$$H_{\lambda x:\tau}\ h\ f = \begin{cases} [\![\lambda x:\tau.(h\ [f \mid x:[\![x]\!]]\ )]\!], \\ \quad \text{if } x \notin \text{clashset } \langle f, x, h\ emp_L \rangle \\ [\![\lambda z:\tau.\ (h\ [f \mid x:[\![z]\!]]\ )]\!], \text{ otherwise,} \\ \quad \text{where } x \varepsilon Id,\ \tau \varepsilon T,\ h \varepsilon H,\ f \varepsilon |L|^*, \text{ and} \\ \quad z = \text{newid } (\text{clashset } \langle f, x, h\ emp_L \rangle) \end{cases}$$

Let

$$\text{Presub} \varepsilon L \xrightarrow[\Lambda\text{-Alg}]{} H$$

be the unique Λ-algebra homomorphism such that for all $g \varepsilon G$, Presub g is the constant function:

$$\text{Presub } g\ f = g \quad \text{for all} \quad f \varepsilon |L|^*.$$

Now define

$$\text{Sub} \varepsilon |L|^* \to |L| \to |L|$$

by

$$\text{Sub } f\ \ell = \text{Presub } \ell\ f$$

for all $f \varepsilon |L|^*$, $\ell \varepsilon L$.

The next theorem asserts that the substitution function determined by a phrase assignment f depends only on the canonical extension of f to a comprehensive phrase assignment.

Theorem 3.3: For all phrase assignments f and j, $f^{\infty} = j^{\infty}$ implies Sub f = Sub j.

Proof: The theorem will follow if we can prove by induction on the structure of $\ell \varepsilon L$ that for all phrase assignments f and j, $f^{\infty} = j^{\infty}$ implies Sub f ℓ = Sub j ℓ. The fact that both f and j are universally quantified in the induction hypothesis is crucial to the proof.

(1) Let ℓ = g where $g \varepsilon G$. Then, for all f, j,

Sub f g = Presub g f = g = Presub g j = Sub j g.

(2) Let $\ell = [\![x]\!]$ where $x \varepsilon Id$. Then $f^{\infty} = j^{\infty}$ implies

Sub f $[\![x]\!]$ = Presub $[\![x]\!]$ f = H_x f = f^{∞} x = j^{∞} x = Sub j $[\![x]\!]$.

(3) Let $\ell = [\![mn]\!]$ where $m, n \varepsilon L$. Then $f^{\infty} = j^{\infty}$ implies

Sub f $[\![mn]\!]$ = Presub $[\![mn]\!]$ f

= H_{ap} <Presub m, Presub n> f

= $[\![$ (Presub m f) (Presub n f)$]\!]$

= $[\![$ (Sub f m) (Sub f n)$]\!]$

= $[\![$ (Sub j m) (Sub j n)$]\!]$,

by the induction hypothesis,

= Sub j $[\![mn]\!]$

(4) Let $\ell = [\![\lambda x{:}\tau.m]\!]$ where $x \varepsilon Id$, $\tau \varepsilon T$, $m \varepsilon L$.

Suppose $f^{\infty} = j^{\infty}$. If $x \notin$ clashset <f, x, Presub m emp_L>, then

$$\begin{aligned}
\text{Sub } f\ [\![\lambda x{:}\tau.m]\!] &= \text{Presub } [\![\lambda x{:}\tau.m]\!]\ f \\
&= H_{\lambda x:\tau}(\text{Presub } m)\ f \\
&= [\![\lambda x{:}\tau.(\text{Presub } m\ [f \mid x{:}[\![x]\!]])\ ]\!] \\
&= [\![\lambda x{:}\tau.(\text{Sub } [f \mid x{:}[\![x]\!]]\ m)\ ]\!] \\
&= [\![\lambda x{:}\tau.(\text{Sub } [j \mid x{:}[\![x]\!]]\ m)\ ]\!], \\
&\quad \text{because of the induction hypothesis and} \\
&\quad [f \mid x{:}[\![x]\!]]^{\infty} = [j \mid x{:}[\![x]\!]]^{\infty}, \\
&= [\![\lambda x{:}\tau.(\text{Presub } m\ [j \mid x{:}[\![x]\!]])\ ]\!] \\
&= H_{\lambda x:\tau}(\text{Presub } m)\ j, \\
&\quad \text{because clashset } \langle j, x, \text{Presub } m\ \text{emp}_L\rangle \\
&\qquad = \text{clashset } \langle f, x, \text{Presub } m\ \text{emp}_L\rangle, \\
&= \text{Sub } j\ [\![\lambda x{:}\tau.m]\!]\ .
\end{aligned}$$

If $x \in \text{clashset } \langle f, x, \text{Presub } m\ \text{emp}_L\rangle$, then a similar argument gives $\text{Sub } f\ [\![\lambda x{:}\tau.m]\!] = \text{Sub } j\ [\![\lambda x{:}\tau.m]\!]$. □

Our next task is to prove that the substitution function induced by the empty phrase assignment is the identity function on $|L|$. We base the proof on the obvious observation that the identity function is a homomorphism.

Theorem 3.4: $\text{Sub emp}_L = 1_L$.

Proof: We claim Sub emp_L is a Λ-algebra homomorphism. This is implied by the following three computations.

(1) Let $x \in \text{Id}$. Then

$$\begin{aligned}
\text{Sub emp}_L\ [\![x]\!] &= \text{Presub } [\![x]\!]\ \text{emp}_L \\
&= H_x\ \text{emp}_L \\
&= \text{emp}_L^{\infty}\ x \\
&= [\![x]\!].
\end{aligned}$$

(2) Let $\ell, m \in L$. Then

$$\begin{aligned}\text{Sub emp}_L \llbracket \ell m \rrbracket &= \text{Presub } \llbracket \ell m \rrbracket \text{ emp}_L \\ &= H_{ap} \langle \text{Presub } \ell, \text{ Presub } m \rangle \text{ emp}_L \\ &= \llbracket (\text{Presub } \ell \text{ emp}_L) \ (\text{Presub } m \text{ emp}_L) \rrbracket \\ &= \llbracket (\text{Sub emp}_L \ \ell) \ (\text{Sub emp}_L \ m) \rrbracket .\end{aligned}$$

(3) Let $x \in \text{Id}$, $\tau \in T$, $\ell \in L$. Observe that

$$x \notin \text{clashset} \langle \text{emp}_L, x, \text{Presub } \ell \text{ emp}_L \rangle$$

because

$$\begin{aligned}&\text{clashset} \langle \text{emp}_L, x, \text{Presub } \ell \text{ emp}_L \rangle \\ &= \cup \{\text{Free } (\text{emp}_L^{\infty} y) \mid y \in \text{Free } (\text{Presub } \ell \text{ emp}_L), y \neq x\} \\ &= \cup \{\text{Free } \llbracket y \rrbracket \mid y \in \text{Free } (\text{Presub } \ell \text{ emp}_L), y \neq x\} \\ &= \{y \mid y \in \text{Free } (\text{Presub } \ell \text{ emp}_L), y \neq x\}.\end{aligned}$$

Thus

$$\begin{aligned}\text{Sub emp}_L \llbracket \lambda x{:}\tau.\ell \rrbracket &= \text{Presub } \llbracket \lambda x{:}\tau.\ell \rrbracket \text{ emp}_L \\ &= H_{\lambda x:\tau} \ (\text{Presub } \ell) \text{ emp}_L \\ &= \llbracket \lambda x{:}\tau. \ (\text{Presub } \ell \ [\text{emp}_L \mid x{:}\llbracket x \rrbracket]) \rrbracket \\ &= \llbracket \lambda x{:}\tau. \ (\text{Sub } [\text{emp}_L \mid x{:}\llbracket x \rrbracket] \ \ell) \rrbracket \\ &= \llbracket \lambda x{:}\tau. \ (\text{Sub emp}_L \ \ell) \rrbracket, \text{ by Theorem 3.3.}\end{aligned}$$

So Sub emp_L is a Λ-algebra homomorphism. By comparing the actions of Sub emp_L and 1_L on generators ($\text{Sub emp}_L \ g = \text{Presub } g \text{ emp}_L = g = 1_L \ g$ for $g \in G$), we see $\text{Sub emp}_L = 1_L$. □

Now we are ready to prove the most important theorem on substitution.

Theorem 3.5: Sub is the unique function in $|L|^* \to |L| \to |L|$ such that for all $f \in |L|^*$, conditions (Sub1), (Sub2), (Sub3) and (Sub4) are all satisfied.

Proof: First we will verify that the four conditions are satisfied by Sub. In this proof let $f \varepsilon |L|^*$, $g \varepsilon G$, $x \varepsilon Id$, $\ell \varepsilon L$, $m \varepsilon L$, $\tau \varepsilon T$, and z = newid (clashset <f, x, ℓ>).

By definition of Presub on generators,

(1) Sub f g = Presub g f = g.

Because Presub is a homomorphism,

(2) Sub f $[\![x]\!]$ = Presub $[\![x]\!]$ f = H_x f = f^{∞} x ,

(3) Sub f $[\![\ell m]\!]$ = Presub $[\![\ell m]\!]$ f

= H_{ap} <Presub ℓ, Presub m> f

= $[\![$ (Presub ℓ f) (Presub m f)$]\!]$

= $[\![$ (Sub f ℓ) (Sub f m)$]\!]$, and

(4) if x $\notin$ clashset <f, x, ℓ>, then

Sub f $[\![\lambda x{:}\tau.\ell]\!]$ = Presub $[\![\lambda x{:}\tau.\ell]\!]$ f

= $H_{\lambda x:\tau}$ (Presub ℓ) f

= $[\![\lambda x{:}\tau.$(Presub ℓ [f | x:$[\![$x$]\!]$])$]\!]$,

because clashset <f, x, Presub ℓ emp_L>

= clashset <f, x, ℓ > by Theorem 3.4

= $[\![\lambda x{:}\tau.$ (Sub [f | x:$[\![$x$]\!]$] ℓ)$]\!]$,

whereas, if x ε clashset <f, x, ℓ>, then

Sub f $[\![\lambda x{:}\tau.\ell]\!]$ = $H_{\lambda x:\tau}$ (Presub ℓ) f

= $[\![\lambda z{:}\tau.$ (Presub ℓ [f | x:$[\![$z$]\!]$])$]\!]$

(here again we apply Theorem 3.4)

= $[\![\lambda z{:}\tau.$ (Sub [f | x:$[\![$z$]\!]$] ℓ)$]\!]$.

So the four conditions are satisfied by Sub.

Let Sub' be any function in $|L|^* \to |L| \to |L|$ satisfying the four conditions. Define Presub' ε $|L| \to |L|^* \to |L|$

by Presub' ℓ f = Sub' f ℓ . Using (Sub2) we see

$$\text{Presub' } [\![x]\!]\ f = \text{Sub' } f^{\infty}\ x = H_x\ f\ .$$

Thus

$$\text{Presub' } [\![x]\!] = H_x .$$

Using (Sub3) we see

$$\begin{aligned}\text{Presub' } [\![\ell m]\!]\ f &= \text{Sub' } f\ [\![\ell m]\!] \\ &= [\![(\text{Sub' } f\ \ell)\ (\text{Sub' } f\ m)]\!] \\ &= [\![(\text{Presub' } \ell\ f)\ (\text{Presub' } m\ f)]\!] \\ &= H_{ap} \langle \text{Presub' } \ell, \text{Presub' } m\rangle\ f\ .\end{aligned}$$

Thus

$$\text{Presub' } [\![\ell m]\!] = H_{ap} \langle \text{Presub' } \ell, \text{Presub' } m \rangle$$

Next we carry out some computations using (Sub4). There are two cases. First suppose $x \notin \text{clashset} \langle f, x, \ell \rangle$; then

$$\begin{aligned}\text{Presub' } [\![\lambda x{:}\tau.\ \ell]\!]\ f &= \text{Sub' } f\ [\![\lambda x{:}\tau.\ \ell]\!] \\ &= [\![\lambda x{:}\tau.\ (\text{Sub' } [f \mid x{:}[\![x]\!]]\ \ell)]\!] \\ &= [\![\lambda x{:}\tau.\ (\text{Presub' } \ell\ [f \mid x{:}[\![x]\!]])]\!] \\ &= H_{\lambda x:\tau}(\text{Presub' } \ell)\ f.\end{aligned}$$

The last equality is valid because

$$\begin{aligned}\text{clashset} \langle f, x, \text{Presub' } \ell\ \text{emp}_L\rangle &= \text{clashset} \langle f, x, \text{Sub' } \text{emp}_L\ \ell\rangle \\ &= \text{clashset} \langle f, x, \ell\rangle, \\ &\quad \text{by Proposition 3.2.}\end{aligned}$$

Second, suppose $x \in \text{clashset} \langle f, x, \ell\rangle$. Let $z = \text{newid (clashset} \langle f, x, \ell\rangle)$. Then

$$\begin{aligned}\text{Presub' } [\![\lambda x{:}\tau.\ \ell]\!]\ f &= \text{Sub' } f\ [\![\lambda x{:}\tau.\ \ell]\!] \\ &= [\![\lambda z{:}\tau.\ (\text{Sub' } [f \mid x{:}[\![z]\!]]\ \ell)\ ]\!] \\ &= [\![\lambda z{:}\tau.\ (\text{Presub' } \ell\ [f \mid x{:}[\![z]\!]])]\!] \\ &= H_{\lambda x:\tau}(\text{Presub' } \ell)\ f\ .\end{aligned}$$

Thus

$$\text{Presub}' \llbracket \lambda x{:}\tau.\ell \rrbracket = H_{\lambda x:\tau}\ (\text{Presub}'\ \ell)\ .$$

Hence

$$\text{Presub}' \in L \xrightarrow[\Lambda\text{-Alg}]{} H\ .$$

On generators the homomorphisms Presub' and Presub agree since for $g \in G$, $f \in |L|^*$,

$$\text{Presub}'\ g\ f = \text{Sub}'\ f\ g = g = \text{Presub}\ g\ f\ .$$

Therefore Presub' = Presub and consequently Sub' = Sub. □

The next theorem tells how to compute the set of identifiers occurring free in a phrase created by applying a substitution function to another phrase.

Theorem 3.6: Let $\ell \in L$. For all $f \in |L|^*$,

$$\text{Free}\ (\text{Sub}\ f\ \ell) = \cup \{\text{Free}\ (f^{\infty}\ y) \mid y \in \text{Free}\ \ell\}\ .$$

Proof: We proceed by induction on $\ell \in L$.

(1) Let $\ell = g$ where $g \in G$. For all $f \in |L|^*$,

$$\begin{aligned}\text{Free}\ (\text{Sub}\ f\ g) &= \text{Free}\ g, \quad \text{by (Sub1)},\\ &= \emptyset\ , \text{ by Theorem 3.1},\\ &= \cup \{\text{Free}\ (f^{\infty}\ y) \mid y \in \text{Free}\ g\ \}.\end{aligned}$$

(2) Let $\ell = \llbracket x \rrbracket$ where $x \in \text{Id}$. For all $f \in |L|^*$,

$$\begin{aligned}\text{Free}\ (\text{Sub}\ f\ \llbracket x \rrbracket) &= \text{Free}\ (f^{\infty}\ x)\ , \text{ by (Sub2)},\\ &= \cup \{\text{Free}\ (f^{\infty}\ y) \mid y = x\}\\ &= \cup \{\text{Free}\ (f^{\infty}\ y) \mid y \in \text{Free}\ \llbracket x \rrbracket\}\ ,\\ &\qquad \text{by Theorem 3.1.}\end{aligned}$$

(3) Let $\ell = \llbracket mn \rrbracket$ where $m, n \in L$. For all $f \in |L|^*$,

$$\begin{aligned}&\text{Free}\ (\text{Sub}\ f\ \llbracket mn \rrbracket)\\ &\qquad = \text{Free}\ \llbracket (\text{Sub}\ f\ m)\ (\text{Sub}\ f\ n) \rrbracket\ , \text{ by (Sub3)}\end{aligned}$$

$$= \text{Free (Sub f } m) \cup \text{Free (Sub f } n),$$

by Theorem 3.1 ,

$$= \cup \{\text{Free } (f^{\infty}\ y) \mid y \in (\text{Free } m) \cup (\text{Free } n)\},$$

by the induction hypothesis,

$$= \cup \{\text{Free } (f^{\infty}\ y) \mid y \in \text{Free } [\![mn]\!]\},$$

by Theorem 3.1.

(4) Let $\ell = [\![\lambda x{:}\tau.m]\!]$ where $x \in Id$, $\tau \in T$, $m \in L$. Suppose $f \in |L|^{*}$ and z = newid (clashset <f, x, m>). If $x \notin$ clashset <f, x, m>, then

Free (Sub f $[\![\lambda x{:}\tau.m]\!]$)

$$= \text{Free } [\![\lambda x{:}\tau.\ (\text{Sub } [f \mid x{:}[\![x]\!]]\ m)]\!], \text{ by (Sub4)},$$

$$= \text{Free (Sub } [f \mid x{:}[\![x]\!]]m) - \{x\}, \text{ by Theorem 3.1},$$

$$= (\cup \{\text{Free } ([f \mid x{:}[\![x]\!]]^{\infty}\ y) \mid y \in \text{Free } m\}) - \{x\},$$

by the induction hypothesis ,

$$= ((\cup\{\text{Free}([f \mid x{:}[\![x]\!]]^{\infty}\ y) \mid y \in \text{Free } m,\ y \neq x\}) \cup \{x\}) - \{x\},$$

because Free $([f \mid x{:}[\![x]\!]]^{\infty}\ x) = \text{Free } [\![x]\!] = \{x\}$,

$$= ((\cup\{\text{Free}(f^{\infty}\ y) \mid y \in \text{Free } m,\ y \neq x\}) \cup \{x\}) - \{x\}$$

$$= (\text{clashset } <f, x, m> \cup \{x\}) - \{x\}$$

$$= \text{clashset } <f, x, m>$$

$$= \cup \{\text{Free } (f^{\infty}\ y) \mid y \in \text{Free } [\![\lambda x{:}\tau.m]\!]\} \ .$$

Similarly, if $x \in$ clashset <f, x, m>, then

Free (Sub f $[\![\lambda x{:}\tau.m]\!]$)

$$= \text{Free } [\![\lambda z{:}\tau.\ (\text{Sub } [f \mid x{:}[\![z]\!]]\ m)]\!]$$

$$= \text{Free (Sub } [f \mid x{:}[\![z]\!]]\ m) - \{z\}$$

$$= (\cup \{\text{Free } ([f \mid x{:}[\![z]\!]]^{\infty}\ y) \mid y \in \text{Free } m\}) - \{z\}$$

$$= ((\cup \{\text{Free } ([\, f \mid x{:}[\![z]\!]]^{\infty}\ y) \mid y \in \text{Free } m,\ y \neq x\})$$
$$\cup \{z\}) - \{z\}$$
$$= ((\cup \{\text{Free } (f^{\infty}\ y) \mid y \in \text{Free } m,\ y \neq x\})$$
$$\cup \{z\}) - \{z\}$$
$$= (\text{clashset } \langle f, x, m\rangle \cup \{z\}) - \{z\}$$
$$= \text{clashset } \langle f, x, m\rangle, \text{ because } z \notin \text{clashset } \langle f, x, m\rangle,$$
$$= \cup \{\text{Free } (f^{\infty}\ y) \mid y \in \text{Free } [\![\lambda x{:}\tau.m]\!]\}.$$

□

The next item on the agenda is the consideration of the nature of type checking. To this end, we suppose that in this and the next chapter we are given a fixed function

$$\text{type} \in G \to \text{Ob } |T|,$$

which assigns a type to each linguistic constant. For instance, if bool-exp and int-exp are primitive phrase types and true, 2, + ε G, then perhaps

$$\text{type true} = \underline{\text{bool-exp}},\ \text{type } 2 = \underline{\text{int-exp}},$$
$$\text{type } + = \underline{\text{int-exp}} \Rightarrow \underline{\text{int-exp}} \Rightarrow \underline{\text{int-exp}}.$$

Exactly what role does the function type play if we attempt to assign phrase types to all phrases of L?

The first difficulty is how do we assign phrase types to expressions containing free occurrences of identifiers. We expect that the type "connected" to x in $[\![\lambda x{:}\tau.\ell]\!]$ is τ, but what is the type of $[\![x]\!]$? At first glance, we might be tempted to assign phrase types only to those $\ell \varepsilon L$ such that Free $\ell = \emptyset$. This is entirely inadequate because it prevents us from expressing the type of an expression in terms of the types of its constituents (which possibly contain free

identifiers). A better idea is to assign phrase types to expressions in the context of an assignment of types to identifiers. Thus, the phrase type associated with ⟦x y⟧ will be int-exp in the context in which x is assigned bool-exp ⇒ int-exp and y is assigned bool-exp, whereas ⟦x y⟧ will be assigned ns = T_{ns} (= nonsense) if x is assigned bool-exp and y is assigned int-exp. Note our intended use of the nonsense type: the type checker assigns nonsense to phrases that in the context of a type assignment are meaningless, i.e. not executable, due to mismatched types.

Thus we are led to introduce the poset A, an element of which is a type assignment, i.e. a function

$$\alpha \in F \to Ob\ |T|,$$

where F is a finite set of identifiers. Hence

$$Ob\ A = (Ob\ |T|)^{*}.$$

The partial order on A is given by

$$\alpha \leq_A \beta \quad \text{iff} \quad \text{dom } \beta \subseteq \text{dom } \alpha, \text{ and, for all } x \in \text{dom } \beta,$$
$$\alpha\ x \leq_{|T|} \beta\ x.$$

Intuitively $\alpha \leq_A \beta$ means α has more information than β, and α can have more information in two ways: (1) α can have more components than β, and (2) each component of α can have more information than the corresponding component of β. Consistent with this intuition is the observation that the empty type assignment, denoted

$$emp_T \in \emptyset \to Ob\ |T|$$

which contains no information, is the maximal element of A.

A <u>comprehensive type assignment</u> is a function

$$a \in Id \to Ob\ |T| \ .$$

The collection of comprehensive type assignments can be made into a poset A^∞ by letting

$$a \leq_{A^\infty} b \quad \text{iff} \ , \quad \text{for all } x \in Id, \quad ax \leq_{|T|} bx \ .$$

We can write

$$Ob\ A^\infty \ = \ (Ob\ |T|)^\infty \ .$$

Comprehensive type assignments bear roughly the same relationship to type assignments that comprehensive phrase assignments bear to phrase assignments. If $\alpha \in A$, then there is a canonical $\alpha^\infty \in A^\infty$ given by

$$\alpha^\infty \ x \ = \begin{cases} \alpha\ x \ , & \text{if} \quad x \in \text{dom } \alpha, \\ \\ \underline{ns} \ , & \text{otherwise} \ . \end{cases}$$

Observe that $\alpha \leq_A \beta$ implies $\alpha^\infty \leq_{A^\infty} \beta^\infty$.

Our aim is to define and investigate a function

$$Type \in |L| \to Ob\ A \to Ob\ |T|$$

which assigns to a phrase ℓ in the context of a type assignment α an intuitively satisfying phrase type $Type\ \ell\ \alpha$. We expect this function to have four properties. Since a linguistic constant contains no identifiers we expect its type to be independent of the type assignment. This gives the first property.

(Type1) Type g α = type g for all $g \varepsilon G$, $\alpha \varepsilon A$.

Phrase types of identifiers should be determined by the type assignment, provided the type assignment has enough information in it to do so. This yields the second property.

(Type2) Type $[\![x]\!]\ \alpha = \alpha^{\infty}\ x$ for all $x \varepsilon Id$, $\alpha \varepsilon A$.

The third property says that for an application to make sense in a context, the operator in that context must possess an arrow type, and it must be possible to coerce the type of the operand in that context into the appropriate argument (or parameter) type; then the whole phrase has the appropriate result (or call) type.

(Type3)

$$\text{Type } [\![\ell m]\!]\ \alpha = \begin{cases} \theta, \text{ if Type } \ell\ \alpha = \tau \Rightarrow \theta \text{ and Type } m\ \alpha \leq \tau. \\ \underline{\text{ns}}, \text{ otherwise ,} \\ \text{for all } \ell, m \varepsilon L,\ \alpha \varepsilon A. \end{cases}$$

Finally, lambda expressions have procedural types in which the type of the result is computed with an altered type assignment.

(Type4) Type $[\![\lambda x{:}\tau.\ell]\!]\ \alpha = \tau \Rightarrow (\text{Type } \ell\ [\alpha \mid x{:}\tau])$,

for all $x \varepsilon Id$, $\tau \varepsilon T$, $\ell \varepsilon L$, $\alpha \varepsilon A$.

Of course, the four properties above suffice to define Type ℓ by induction on the structure of ℓ. However that path fails to uncover fully the role played by Λ-algebras in type checking. A more fruitful approach begins by defining the Λ-algebra E whose carrier is

$$|E| = A \to |T| \quad ,$$

the collection of functors (i.e. monotone functions) from A to $|T|$. The fact that the carrier of E is the collection of objects of the poset $A \Rightarrow |T|$ will be exploited in Theorem 3.12. The definitions of interpretations of the operations on E are contained in the next theorem.

Theorem 3.7: Let e and f be monotone functions from A to $|T|$. Let $x \varepsilon Id$ and $\tau \varepsilon |T|$. The equations

(1) $E_x\ \alpha = \alpha^\infty\ x$,

(2) $$E_{ap}\ \langle e, f \rangle\ \alpha = \begin{cases} \theta \text{ if } e\alpha = \tau \Rightarrow \theta \text{ and } f\alpha \leq \tau \\ \underline{ns} \text{ otherwise,} \end{cases}$$

(3) $E_{\lambda x:\tau}\ e\ \alpha = \tau \Rightarrow (e\ [\alpha \mid x{:}\tau])$,

where $\alpha \varepsilon A$, define monotone functions from A to $|T|$.

Proof: Suppose $\alpha \leq_A \beta$.

(1) From $\alpha^\infty \leq_{A^\infty} \beta^\infty$ we get

$$E_x\ \alpha = \alpha^\infty\ x \leq \beta^\infty\ x = E_x\ \beta.$$

(2) Suppose $e\beta = \tau \Rightarrow \theta$ and $f\beta \leq \tau$, so that $E_{ap}\ \langle e,f \rangle\ \beta = \theta$. Since $e\alpha \leq e\beta$, Theorem 2.2 shows that $e\alpha$ must be a procedural type, say $e\alpha = \tau' \Rightarrow \theta'$ where $\tau \leq \tau'$ and $\theta' \leq \theta$. Then $f\alpha \leq f\beta \leq \tau \leq \tau'$, which implies $E_{ap}\ \langle e,f \rangle\ \alpha = \theta' \leq \theta = E_{ap}\ \langle e,f \rangle\ \beta$, as desired.

Otherwise $E_{ap}\ \langle e,f \rangle\ \beta = \underline{ns}$, and the inequality $E_{ap}\ \langle e,f \rangle\ \alpha \leq E_{ap}\ \langle e,f \rangle\ \beta$ is trivially true.

(3) Clearly $[\alpha \mid x{:}\tau] \leq [\beta \mid x{:}\tau]$. Thus $E_{\lambda x:\tau}\, e\, \alpha = \tau \Rightarrow (e\, [\alpha \mid x{:}\tau]) \leq \tau \Rightarrow (e\, [\beta \mid x{:}\tau])$. □

Note that among the elements of $|E| = A \to |T|$ are the constant functions. Thus it makes sense to define the Λ-algebra homomorphism

$$\text{Type} \in L \xrightarrow[\Lambda\text{-Alg}]{} E$$

to be the unique homomorphism such that, for all $g \in G$, $\alpha \in A$,

$$\text{Type}\ g\ \alpha = \text{type}\ g\ .$$

Theorem 3.8: The Λ-algebra homomorphism Type is the unique function in $|L| \to (A \to |T|)$ such that (Type1), (Type2), (Type3), and (Type4) all hold.

Proof: Trivial. □

It is interesting to note that approaching Type as a Λ-algebra homomorphism tells us immediately that Type ℓ, for each ℓ, is monotone. This would certainly not be obvious if we attempted to define Type by structural induction instead of using the algebraic method we have chosen.

The next theorem and its two corollaries show that only the free identifiers in a phrase affect its type.

Theorem 3.9: Let $\ell \in L$. For all $\alpha, \beta \in A$,

$$\alpha^{\infty}\, x = \beta^{\infty}\, x \text{ , for all } x \in \text{Free } \ell \text{ ,}$$

implies

$$\text{Type } \ell\ \alpha = \text{Type } \ell\ \beta.$$

Proof: We proceed by induction on the structure of $\ell \in L$.

(1) Suppose $\ell = g$ where $g \in G$. Then, for all $\alpha, \beta \in A$,

Type g α = type g = Type g β.

(2) Suppose $\ell = [\![x]\!]$ where $x \in Id$. Recalling that Free $[\![x]\!] = \{x\}$, we let $\alpha, \beta \in A$ be such that

$$\alpha^{\infty}\ x = \beta^{\infty}\ x.$$

Then

Type $[\![x]\!]$ $\alpha = \alpha^{\infty}\ x = \beta^{\infty}\ x$ = Type $[\![x]\!]$ β.

(3) Suppose $\ell = [\![mn]\!]$ where $m, n \in L$. Assume that $\alpha, \beta \in A$ satisfy

$\alpha^{\infty}\ x = \beta^{\infty}\ x$ for all $x \in$ (Free m) $\cup$ (Free n)

By the induction hypothesis

Type m α = Type m β, and Type n α = Type n β.

From condition (Type3) it is then immediate that

Type $[\![mn]\!]$ α = Type $[\![mn]\!]$ β .

(4) Suppose $\ell = [\![\lambda x{:}\tau.m]\!]$ where $x \in Id$, $\tau \in T$, $m \in L$. Recall that Free $[\![\lambda x{:}\tau.m]\!]$ = (Free m) - $\{x\}$. Thus, take $\alpha, \beta \in A$ such that

$$\alpha^{\infty}\ y = \beta^{\infty}\ y, \quad \text{for all } y \in (\text{Free } m) - \{x\}.$$

Then

$$[\alpha \mid x{:}\tau]^{\infty}\ y = [\beta \mid x{:}\tau]^{\infty}\ y, \text{ for all } y \in \text{Free } m.$$

We conclude by the induction hypothesis that

$$\begin{aligned}\text{Type } [\![\lambda x{:}\tau.m]\!]\ \alpha &= \tau \Rightarrow (\text{Type } m\ [\alpha \mid x{:}\tau]) \\ &= \tau \Rightarrow (\text{Type } m\ [\beta \mid x{:}\tau]) \\ &= \text{Type } [\![\lambda x{:}\tau.m]\!]\ \beta.\end{aligned}$$

□

Corollary 3.10: Suppose $\ell \in L$, $\alpha \in A$, $\beta \in A$, and $\text{dom } \alpha = \text{dom } \beta = F$. If

$$\alpha x = \beta x, \quad \text{for all} \quad x \in (\text{Free } \ell) \cap F ,$$

then

$$\text{Type } \ell\ \alpha = \text{Type } \ell\ \beta .$$

Proof: Let $x \in \text{Free } \ell$. If $x \in F$, then

$$\alpha^{\infty} x = \alpha x = \beta x = \beta^{\infty} x ;$$

otherwise,

$$\alpha^{\infty} x = \underline{ns} = \beta^{\infty} x .$$

Therefore $\text{Type } \ell\ \alpha = \text{Type } \ell\ \beta$. □

Corollary 3.11: Let $\ell \in L$, $\alpha \in A$, $x \in \text{Id}$, $\tau \in T$. If $x \notin \text{Free } \ell$, then

$$\text{Type } \ell\ \alpha = \text{Type } \ell\ [\alpha \mid x{:}\tau].$$

Proof: Apply Theorem 3.9 with $\beta = [\alpha \mid x{:}\tau]$. □

The final goal of this chapter is to obtain a satisfying description of the type of an expression which is itself the result of a substitution and to investigate some of its consequences. To this end, we will need to define Type^*, a function which is derived from the homomorphism Type and

which applies to phrase assignments rather than phrases. To avoid confusion, let's review our use of the infixed operator $\rightarrow$. The notation $X \rightarrow Y$ denotes the collection of functions from X to Y, unless X and Y are both categories, in which case it denotes the collection of functors from X to Y. Therefore $A \rightarrow A$ denotes the set of monotone functions from A to A, and $|L|^* \rightarrow (A \rightarrow A)$ designates the set of all functions from the collection of phrase assignments to $A \rightarrow A$. Whereas Type accepts phrases and in an appropriate context produces a type, the function Type^* accepts phrase assignments and in an appropriate context produces a type assignment. Actually,

$$\text{Type}^* \in |L|^* \rightarrow (A \rightarrow A) .$$

If $f \in |L|^*$ and $\alpha \in A$, then

$$\text{Type}^* \; f \; \alpha \in \text{dom } f \rightarrow \text{Ob } |T| ,$$

and

$$\text{Type}^* \; f \; \alpha \; x = \text{Type } (fx) \; \alpha , \quad \text{for all } x \in \text{dom } f .$$

We must verify that $\text{Type}^* \; f$ is monotone. Suppose $\alpha \leq_A \beta$. Observe that

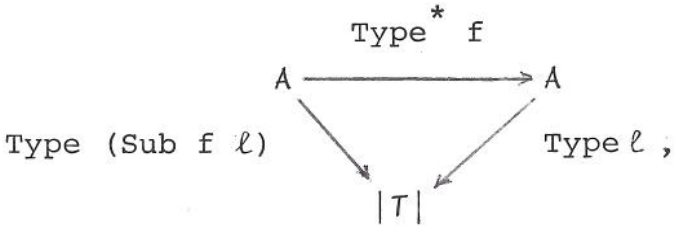
$$\text{dom } (\text{Type}^* \; f \; \alpha) = \text{dom } f = \text{dom } (\text{Type}^* \; f \; \beta).$$

For $x \in \text{dom } f$,

$$\begin{aligned} \text{Type}^* \; f \; \alpha \; x &= \text{Type } (fx) \; \alpha \\ &\leq \text{Type } (fx) \; \beta, \quad \text{by the monotonicity of Type } (fx), \\ &= \text{Type}^* \; f \; \beta \; x . \end{aligned}$$

Hence $\text{Type}^* \; f \; \alpha \leq_A \text{Type}^* \; f \; \beta$, as desired.

As an aid to contemplating the statement of the next theorem, consider the diagram



where $\ell \in L$, $f \in |L|^*$. In particular, suppose $\ell = [\![x]\!]$, where $x \in Id$. It is easy to see that if f maps x to $[\![y]\!]$, where $y \in Id$, then the diagram commutes; whereas if $f = emp_L$ then it does not.

Theorem 3.12: Let $\ell \in L$. For all $f \in |L|^*$,

$$\text{Type (Sub } f\, \ell\text{)} \leq_{A \Rightarrow |T|} (\text{Type } \ell) \circ (\text{Type}^* f) ,$$

and the inequality becomes equality whenever Free $\ell \subseteq$ dom f.

Proof: Use induction on the structure of ℓ.

(1) Let $\ell = g$ where $g \in G$. For all $f \in |L|^*$, $\alpha \in A$,

$$\begin{aligned} \text{Type (Sub } f\ g)\ \alpha &= \text{Type } g\ \alpha \\ &= \text{type } g \\ &= \text{Type } g\ (\text{Type}^*\ f\ \alpha), \end{aligned}$$

as desired.

(2) Let $\ell = [\![x]\!]$ where $x \in Id$. Suppose $f \in |L|^*$ and $\alpha \in A$. If $x \in$ dom f, then

$$\begin{aligned} \text{Type (Sub } f\ [\![x]\!])\ \alpha &= \text{Type } (fx)\ \alpha \text{ , since } f^\infty\ x = f\ x , \\ &= \text{Type}^*\ f\ \alpha\ x \\ &= (\text{Type}^*\ f\ \alpha)^\infty\ x \\ &= \text{Type } [\![x]\!]\ (\text{Type}^*\ f\ \alpha) . \end{aligned}$$

If $x \notin \text{dom } f$, then

$$\text{Type (Sub f } [\![x]\!]) \ \alpha = \text{Type } [\![x]\!] \ \alpha \text{ , since } f^{\infty} x = [\![x]\!],$$

$$\leq \underline{\text{ns}}$$

$$= (\text{Type}^{*} f \ \alpha)^{\infty} x,$$

since dom $(\text{Type}^{*} f \ \alpha) = \text{dom } f$,

$$= \text{Type } [\![x]\!] \ (\text{Type}^{*} f \ \alpha)$$

Hence Type (Sub f $[\![x]\!]$) $\leq_{A \Rightarrow |T|}$ (Type $[\![x]\!]$) $\circ$ (Type* f), and, upon recalling that Free $[\![x]\!] = \{x\}$, we have equality when Free $[\![x]\!] \subseteq$ dom f.

(3) Let $\ell = [\![mn]\!]$ where $m, n \in L$. Let $f \in |L|^{*}$ and $\alpha \in A$. We must carry out two computations. First,

$$\text{Type (Sub f } [\![mn]\!]) \ \alpha$$

$$= \text{Type } [\![(\text{Sub f } m) \ (\text{Sub f } n)]\!] \ \alpha$$

$$= \begin{cases} \theta & \text{if Type (Sub f } m) \ \alpha = \tau \Rightarrow \theta, \\ & \text{Type (Sub f } n) \ \alpha \leq \tau \ , \\ \underline{\text{ns}} & \text{otherwise.} \end{cases}$$

Second,

$$\text{Type } [\![mn]\!] \ (\text{Type}^{*} f \ \alpha)$$

$$= \begin{cases} \theta' & \text{if Type } m \ (\text{Type}^{*} f \ \alpha) = \tau' \Rightarrow \theta', \\ & \text{Type } n \ (\text{Type}^{*} f \ \alpha) \leq \tau', \\ \underline{\text{ns}} & \text{otherwise.} \end{cases}$$

If

$$\text{Type } m \ (\text{Type}^{*} f \ \alpha) = \tau' \Rightarrow \theta' \text{ and Type } n (\text{Type}^{*} f \ \alpha) \leq \tau',$$

then by the induction hypothesis and Theorem 2.2

(*) $\text{Type (Sub f } m) \alpha = \tau \Rightarrow \theta \leq \tau' \Rightarrow \theta'$ (so that $\tau' \leq \tau$ and $\theta \leq \theta'$),

$$(*) \qquad \text{Type (Sub f } n) \ \alpha \leq \text{Type } n \ (\text{Type}^* \ f \ \alpha)$$
$$\leq \tau'$$
$$(*) \qquad \leq \tau,$$

and consequently

$$\text{Type (Sub } [\![mn]\!] \ \alpha) = \theta$$
$$(*) \qquad \leq \theta'$$
$$= \text{Type } [\![mn]\!] \ (\text{Type}^* \ f \ \alpha);$$

otherwise

$$(*) \qquad \text{Type (Sub } [\![mn]\!] \ \alpha) \leq \underline{ns}$$
$$= \text{Type } [\![mn]\!] \ (\text{Type}^* \ f \ \alpha).$$

This gives the desired inequality. Suppose in addition that

$$(\text{Free } m) \cup (\text{Free } n) = \text{Free } [\![mn]\!] \subseteq \text{dom } f.$$

Then we may change all inequalities in lines marked (*) to equalities, yielding the desired equality.

(4) Let $\ell = \lambda x{:}\tau.m$ where $x \in \text{Id}$, $\tau \in T$, $m \in L$. Take any $f \in |L|^*$, $\alpha \in A$. Let X = clashset <f, x, m> and z = newid X. There are two relevant equations. First,

$$\text{Type (Sub f } [\![\lambda x{:}\tau.m]\!]) \ \alpha$$
$$= \begin{cases} \text{Type } [\![\lambda x{:}\tau.\ (\text{Sub } [f \mid x{:}\ [\![x]\!]]\ m)]\!] \ \alpha \text{ if } x \notin X, \\ \text{Type } [\![\lambda z{:}\tau.\ (\text{Sub } [f \mid x{:}\ [\![z]\!]]\ m)]\!] \ \alpha \text{ if } x \in X \end{cases}$$
$$= \begin{cases} \tau \Rightarrow (\text{Type (Sub } [f \mid x{:}[\![x]\!]]\ m)\ [\alpha \mid x{:}\tau]) \text{ if } x \notin X, \\ \tau \Rightarrow (\text{Type (Sub } [f \mid x{:}[\![z]\!]]\ m)\ [\alpha \mid z{:}\tau]) \text{ if } x \in X. \end{cases}$$

Second

$$\text{Type } [\![\lambda x{:}\tau.m]\!] \ (\text{Type}^* \ f \ \alpha)$$
$$= \tau \Rightarrow (\text{Type } m\ [\text{Type}^* \ f \ \alpha \mid x{:}\tau]).$$

On the one hand, suppose $x \notin X$. By the induction hypothesis

$$\text{Type } (\text{Sub } [f \mid x:[\![x]\!]]\ m)\ [\alpha \mid x:\tau]$$

$$(\dagger) \qquad \leq \text{Type } m\ (\text{Type}^*\ [f \mid x:[\![x]\!]]\ [\alpha \mid x:\tau]).$$

We will show that

$$\text{Type } m\ \beta = \text{Type } m\ \gamma$$

where

$\beta = \text{Type}^*\ [f \mid x:[\![x]\!]]\ [\alpha \mid x:\tau]$ and $\gamma = [\text{Type}^*\ f\ \alpha \mid x:\tau]$

To see this, note that the domains of β and γ are both equal to $(\text{dom } f) \cup \{x\}$. If the restrictions of β and γ to $(\text{Free } m) \cap ((\text{dom } f) \cup \{x\})$ are equal, then the desired equality follows by Corollary 3.10. So let

$y \in (\text{Free } m) \cap ((\text{dom } f) \cup \{x\})$. If $y = x$, then

$$\begin{aligned}\beta\ y &= \text{Type } ([f \mid x:[\![x]\!]]\ x)\ [\alpha \mid x:\tau] \\ &= \text{Type } [\![x]\!]\ [\alpha \mid x:\tau] \\ &= \tau \\ &= \gamma\ y\end{aligned}$$

If $y \neq x$, then

$$\begin{aligned}\beta\ y &= \text{Type } ([f \mid x:[\![x]\!])\ y)\ [\alpha \mid x:\tau] \\ &= \text{Type } (fy)\ [\alpha \mid x:\tau] \\ &= \text{Type } (fy)\ \alpha, \\ &\quad \text{by Corollary 3.11 which applies since } x \notin X \\ &\quad\quad \text{implies } x \notin \text{Free } (fy), \\ &= \text{Type}^*\ f\ \alpha\ y \\ &= \gamma\ y.\end{aligned}$$

We conclude that $\text{Type } m\ \beta = \text{Type } m\ \gamma$. Therefore

$$\begin{aligned}
&\text{Type } (\text{Sub } f\ [\![\lambda x{:}\tau.m]\!])\ \alpha \\
&\quad = \tau \Rightarrow (\text{Type } (\text{Sub } [f \mid x{:}[\![x]\!]]\ m)\ [\alpha \mid x{:}\tau]) \\
(\dagger)\quad &\quad \leq \tau \Rightarrow (\text{Type } m\ \beta) \\
&\quad = \tau \Rightarrow (\text{Type } m\ \gamma) \\
&\quad = \text{Type } [\![\lambda x{:}\tau.m]\!]\ (\text{Type}^*\ f\ \alpha).
\end{aligned}$$

On the other hand, suppose $x \in X$. By the induction hypothesis

$$\begin{aligned}
&\text{Type } (\text{Sub } [f \mid x{:}[\![z]\!]]m)\ [\alpha \mid z{:}\tau] \\
(\dagger)\quad &\quad \leq \text{Type } m\ (\text{Type}^*\ [f \mid x{:}[\![z]\!]]\ [\alpha \mid z{:}\tau]).
\end{aligned}$$

We claim

$$\text{Type } m\ \delta = \text{Type } m\ \gamma$$

where γ is as before and

$$\delta = \text{Type}^*\ [f \mid x{:}[\![z]\!]]\ [\alpha \mid z{:}\tau].$$

Clearly the domains of γ and δ are both equal to $(\text{dom } f) \cup \{x\}$, and by Corollary 3.10 it suffices to show the equality of the restrictions of γ and δ to $(\text{Free } m) \cap ((\text{dom } f) \cup \{x\})$. Let $y \in (\text{Free } m) \cap ((\text{dom } f) \cup \{x\})$.

If $y = x$, then

$$\begin{aligned}
\delta\ y &= \text{Type } ([f \mid x{:}[\![z]\!]]\ x)\ [\alpha \mid z{:}\tau] \\
&= \text{Type } [\![z]\!]\ [\alpha \mid z{:}\tau] \\
&= \tau \\
&= \gamma\ y\ .
\end{aligned}$$

If $y \neq x$, then

$$\begin{aligned}
\delta\ y &= \text{Type } ([f \mid x{:}[\![z]\!]]\ y)\ [\alpha \mid z{:}\tau] \\
&= \text{Type } (fy)\ [\alpha \mid z{:}\tau] \\
&= \text{Type } (fy)\ \alpha,
\end{aligned}$$

by Corollary 3.11 which applies because

$z \notin X$ implies $z \notin \text{Free } (fy)$,

$$= \text{Type}^{*}\ f\ \alpha\ y$$

$$= \gamma\ y\ .$$

Much as before, we get

$$\text{Type}\ (\text{Sub}\ f\ [\![\lambda x{:}\tau.m]\!])\ \alpha$$

$$= \tau \Rightarrow (\text{Type}\ (\text{Sub}\ [f \mid x{:}[\![z]\!]]\ m)\ [\alpha \mid z{:}\tau])$$

$$(\dagger) \qquad \leq \tau \Rightarrow (\text{Type}\ m\ \delta)$$

$$= \tau \Rightarrow (\text{Type}\ m\ \gamma)$$

$$= \text{Type}\ [\![\lambda x{:}\tau.m]\!]\ (\text{Type}^{*}\ f\ \alpha)\ .$$

Hence

$$(\dagger)\quad \text{Type}\ (\text{Sub}\ f\ [\![\lambda x{:}\tau.m]\!]) \leq_{A \Rightarrow |T|} (\text{Type}\ [\![\lambda x{:}\tau.m]\!]) \circ (\text{Type}^{*}\ f)$$

Furthermore, if

$$(\text{free}\ m) - \{x\} = \text{free}\ [\![\lambda x{:}\tau.m]\!] \subseteq \text{dom}\ f,$$

then every inequality marked (†) may be strengthened to an equality. □

In the next proposition the expression $\text{Sub}\ [\text{emp}_L \mid x{:}[\![y]\!]]\ \ell$ has the easy reading "substitute y for x in ℓ." It should not be surprising that a satisfying result relating the type of ℓ with the type of the phrase arising from substituting y for x in ℓ requires the hypothesis that y not occur free in ℓ.

Proposition 3.13: Let $x, y \in \text{Id}$, $\ell \in L$, $\tau \in T$, $\alpha \in A$. Suppose $y \notin \text{Free}\ \ell$ and $m = \text{Sub}\ [\text{emp}_L \mid x{:}[\![y]\!]]\ \ell$. Then

$$\text{Type}\ \ell\ [\alpha \mid x{:}\tau] = \text{Type}\ m\ [\alpha \mid y{:}\tau]$$

Proof: Define the phrase assignment $f \in \text{Free}\ \ell \to |L|$ by

$$f\ z\ =\ \begin{cases} [\![z]\!] & \text{if}\quad z \neq x \\ [\![y]\!] & \text{if}\quad z = x\ . \end{cases}$$

By Theorem 3.3, Sub f = Sub [emp_L | x:[[y]]].

Hence m = Sub f ℓ. By Theorem 3.12,

$$\text{Type}\ m = (\text{Type}\ \ell) \circ (\text{Type}^*\ f).$$

Observe that

$$\text{Type}^*\ f\ [\alpha \mid y{:}\tau]\ \varepsilon\ \text{Free}\ \ell \to |L|$$

is the same as the restriction of $[\alpha \mid x{:}\tau]^\infty$ to Free ℓ, because, if $z \neq x$, then

$$\begin{aligned} \text{Type}^*\ f\ [\alpha \mid y{:}\tau]\ z &= \text{Type}\ (fz)\ [\alpha \mid y{:}\tau] \\ &= \text{Type}\ [\![z]\!]\ [\alpha \mid y{:}\tau] \\ &= \alpha^\infty\ z \\ &= [\alpha \mid x{:}\tau]^\infty\ z, \end{aligned}$$

whereas, if $z = x$, then

$$\begin{aligned} \text{Type}^*\ f\ [\alpha \mid y{:}\tau]\ z &= \text{Type}\ (fx)\ [\alpha \mid y{:}\tau] \\ &= \text{Type}\ [\![y]\!]\ [\alpha \mid y{:}\tau] \\ &= \tau \\ &= [\alpha \mid x{:}\tau]^\infty\ z\ . \end{aligned}$$

We may now apply Theorem 3.2 to conclude

$$\begin{aligned} \text{Type}\ m\ [\alpha \mid y{:}\tau] &= \text{Type}\ \ell\ (\text{Type}^*\ f\ [\alpha \mid y{:}\tau]) \\ &= \text{Type}\ \ell\ [\alpha \mid x{:}\tau]. \end{aligned}$$

□

Proposition 3.14 explores the connection between β-reduction and the function Type. It is used in Chapter IV.

Proposition 3.14: Let $x \varepsilon Id$, $\tau \varepsilon T$, and $\ell, m \varepsilon L$.

Suppose $n = Sub\ [emp_L \mid x:m]\ \ell$ and $n' = [\![(\lambda x:\tau.\ell)\ m]\!]$.

Then

(1) $Type\ n \leq_{A \Rightarrow |T|} Type\ n'$, and

(2) $Type\ n\ \alpha = Type\ n'\ \alpha$ whenever $\alpha \varepsilon A$ and $Type\ m\ \alpha = \tau$.

Proof: (1) Let $\alpha \varepsilon A$. Define the phrase assignment

$$f_\alpha \varepsilon (Free\ \ell) \cup (dom\ \alpha) \cup \{x\} \to |L|$$

by

$$f_\alpha\ y = \begin{cases} [\![y]\!] & \text{if } y \neq x \\ m & \text{if } y = x \end{cases}.$$

Using Theorem 3.3, we see $Sub\ f = Sub\ [emp_L \mid x:m]$.

Therefore $n = Sub\ f_\alpha\ \ell$. Since $Free\ \ell \subseteq dom\ f$, Theorem 3.12 gives

$$Type\ n = (Type\ \ell) \circ (Type^*\ f_\alpha).$$

Observe that for $y \varepsilon (dom\ \alpha) \cup \{x\}$

$$Type^*\ f_\alpha\ \alpha\ y = Type\ (f_\alpha\ y)\ \alpha = \begin{cases} \alpha\ y & \text{if } y \neq x \\ Type\ m\ \alpha & \text{if } y = x \end{cases}.$$

Hence

$$Type^*\ f_\alpha\ \alpha \leq_A [\alpha \mid x:Type\ m\ \alpha]$$

There are two cases. First suppose $Type\ m\ \alpha \leq \tau$.

Then

$$\begin{aligned} Type\ n\ \alpha &= Type\ \ell\ (Type^*\ f\ \alpha) \\ &\leq Type\ \ell\ [\alpha \mid x:Type\ m\ \alpha] \\ &\leq Type\ \ell\ [\alpha \mid x:\tau] \\ &= Type\ [\![(\lambda x:\tau.\ell)\ m]\!]\ \alpha \\ &= Type\ n'\ \alpha. \end{aligned}$$

Second, suppose that it is not true that Type $m\ \alpha \leq \tau$. Then

$$\text{Type } n\ \alpha \leq \underline{\text{ns}} = \text{Type } [\![(\lambda x{:}\tau.\ell)m]\!]\ \alpha = \text{Type } n'\ \ \alpha.$$

Hence

$$\text{Type } n \leq_{A \Rightarrow |T|} \text{Type } n'.$$

(2) Consider the proof above with the added assumption that Type $m\ \alpha = \tau$. It is easy to see that in this case

$$(\text{Type}^*\ f\ \alpha)^\infty = [\alpha \mid x{:}\tau]^\infty .$$

Thus

$$\begin{aligned} \text{Type } n\ \alpha &= \text{Type } \ell\ (\text{Type}^*\ f\ \alpha) \\ &= \text{Type } \ell\ [\alpha \mid x{:}\tau] \\ &= \text{Type } n'\ \alpha. \end{aligned}$$

□

CHAPTER IV

THE FUNDAMENTAL THEOREM OF SEMANTICS

In this chapter we explore the process of assigning meanings to programs with the aim of uncovering the algebraic roots of programming language semantics.

We continue to assume as given the various entities assumed to exist in the last chapter. Thus, we suppose we are given

(1) P, the poset of primitive phrase types, which determines T, the free type algebra generated by P,

(2) Id, the infinite set of identifiers, which with T determines Λ, the language signature,

(3) G, the generating set of linguistic constants, which with Λ determines L, the coercive typed λ-calculus,

(4) newid ε $|F| \to$ Id, which enters into the definition of the substitution function
Sub ε $|L|^* \to |L| \to |L|$,

and

(5) type ε G $\to$ Ob $|T|$, which determines
Type ε $|L| \to (A \to |T|)$, the type-checking homomorphism.

Furthermore, we assume we are given

(6) K, a type algebra derived in the canonical way from its carrier $|K|$, which is assumed to be a Cartesian closed category - hence,

$$K_{ns} = |K|_{term} \quad \text{and} \quad K_{\Rightarrow} = |K|_{\Rightarrow} \ ,$$

(7) mng ε $P \to |K|$, a functor, which gives rise by Theorem 2.2 to a type algebra homomorphism

$$\text{Mng } \varepsilon \ T \xrightarrow[Type\ Alg]{} K \ ,$$

and

(8) a <u>basic semantic function</u>

$$\text{semf } \varepsilon \ G \to \text{Ar } |K|$$

such that, for all g ε G,

$$\text{semf g } \varepsilon \ \Pi_{\phi} \ \text{emp}_{|K|} \xrightarrow[|K|]{} \text{Mng (type g)}.$$

Of course, we must give some intuitively satisfying justification for calling something with the properties of semf a semantic function. Suppose for a moment that $|K| = Set$. The idea is that, for g ε G, semf g gives a proper meaning for g, i.e. an element of some set. The set to which semf g belongs is given by the typing function. A first guess would be

$$\text{semf g } \varepsilon \text{ Mng (type g)}.$$

If all the Cartesian closed categories of use had objects with a set-like structure, then this might suffice as a basis

for generalization. However, other kinds of Cartesian closed categories, namely functor categories, turn out to play a crucial role in programming language semantics. Thus we must reformulate the condition semf g ε Mng (type g). Let T = {t} be a terminal object of *Set*. The elements of any set X are in one-to-one correspondence with the functions T $\to$ X. So we are led to revise our notion of what a proper meaning of g is - no longer is it an element of Mng (type g), but rather it is a method of choosing an element of Mng (type g). More precisely, we want

$$\text{semf g } \varepsilon \text{ T} \xrightarrow[Set]{} \text{Mng (type g)} .$$

We have been a bit vague about what terminal object of *Set* should be used as T. In a sense it is immaterial, but for reasons of subsequent elegance it is best to use $\text{T} = \Pi_{\phi}\ \text{emp}_{Set}$, the set consisting of the unique 0-tuple. Our motivation for (8) is now complete.

It is significant that we have come to the point of view that a semantic function assigns to a phrase a morphism of a Cartesian closed category. The central question is how does

$$\text{semf } \varepsilon \text{ G} \to \text{Ar } |K|$$

determine a semantic function

$$\text{Semf } \varepsilon \ |L| \to ?$$

applicable to all phrases of *L*. The appropriate codomain for Semf is still in question. It is not Ar $|K|$ for much

the same reason that the codomain of Type is not Ob $|T|$. Just as the type of an expression cannot be determined except in the context of a type assignment, the meaning of a phrase requires for its complete determination something that yields an environment, i.e. an assignment of meanings to free identifiers.

Environments are certain product objects in $|K|$; they are products of collections of objects indexed by finite sets of identifiers. Environments are produced from type assignments by application of the functor

$$\text{Env} \in A \to |K| \quad ,$$

which is defined on objects of A by

$$\text{Env } \alpha = \Pi_{\text{dom } \alpha} (\text{Mng} \circ \alpha) \quad .$$

Suppose $\alpha \leq_A \beta$. Then

$$\text{Env } (\alpha \leq_A \beta) \in \text{Env } \alpha \xrightarrow{|K|} \text{Env } \beta$$

is the unique morphism such that for all $x \in \text{dom } \beta$ the diagram

$$\begin{array}{ccc} \Pi_{\text{dom } \alpha} (\text{Mng} \circ \alpha) & \xrightarrow{\text{proj}_{\text{dom } \alpha}\, x} & \text{Mng } (\alpha\, x) \\ \Big\downarrow \text{Env } (\alpha \leq_A \beta) & & \Big\downarrow \text{Mng } (\alpha x \leq \beta x) \\ \Pi_{\text{dom } \beta} (\text{Mng} \circ \beta) & \xrightarrow{\text{proj}_{\text{dom } \beta}\, x} & \text{Mng } (\beta\, x) \end{array}$$

commutes. The verification that Env is indeed a functor is routine.

Note that

$$\text{Env emp}_T = \Pi_\phi\ (\text{Mng} \circ \text{emp}_T) = \Pi_\phi\ \text{emp}_{|K|}.$$

Thus, the basic semantic function is such that for $g \varepsilon G$

$$\text{semf } g \varepsilon \text{ Env emp}_T \xrightarrow[|K|]{} \text{Mng (type } g).$$

We call Env emp_T the <u>empty environment</u>. In a sense it is the environment most appropriate to a phrase with no free identifiers, such as a linguistic constant.

It is now clear that the meaning of an expression ℓ should be calculated in the context of a type assignment α, which yields an environment Env α, and it should ultimately reside in Mng (Type ℓ α). Hence we expect Semf to be found among those functions

$$S \varepsilon |L| \longrightarrow \text{Ob } A \longrightarrow \text{Ar } |K|$$

(thus, S assigns to a phrase ℓ an (Ob A)-indexed collection S ℓ of arrows of $|K|$) such that

(Semf0) for all $\ell \varepsilon L$, $\alpha \varepsilon A$

$$S\ \ell\ \alpha \varepsilon \text{ Env } \alpha \xrightarrow[|K|]{} \text{Mng (Type } \ell\ \alpha).$$

In particular we now know the codomain of Semf:

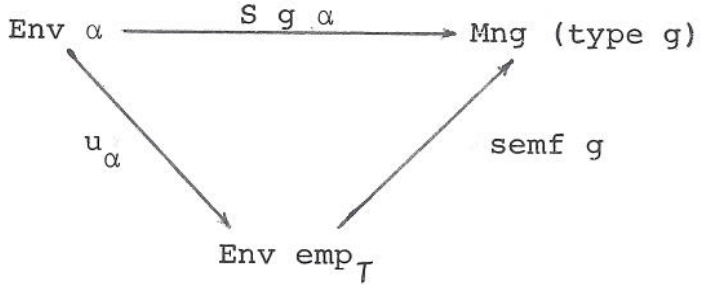
$$\text{Semf } \varepsilon\ |L| \to \text{Ob } A \to \text{Ar } |K|\ .$$

Some elements of Ob $A \to$ Ar $|K|$ are particularly nice; they are natural transformations, i.e. arrows of $A \Rightarrow |K|$. Indeed, we shall see that for each $\ell \varepsilon L$, Semf ℓ will be a natural transformation:

$$\text{Semf } \ell \varepsilon \text{ Env} \xrightarrow[A \Rightarrow |K|]{} \text{Mng} \circ (\text{Type } \ell).$$

Of course, the class of functions to which Semf belongs should have some other properties in addition to (Semf0) above. For instance a semantic function should give to a linguistic constant a meaning independent of the environment. This is stated more precisely as

(Semf1) for all $g \varepsilon G$, $\alpha \varepsilon A$



commutes, where u_α is the unique arrow to the empty environment.

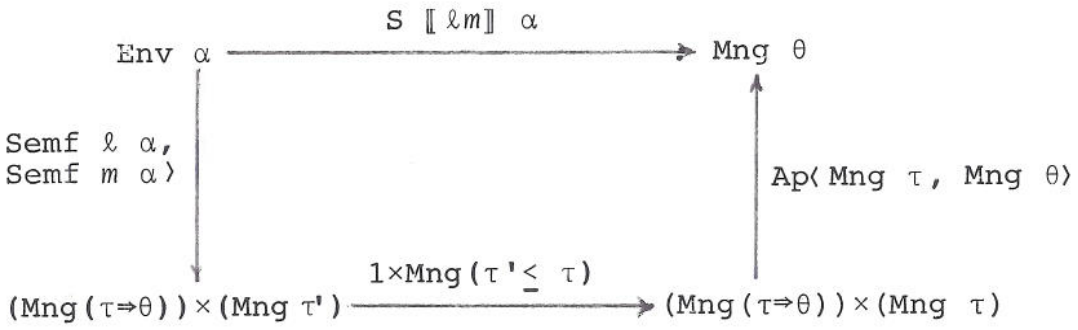
We expect identifiers to pick the right meaning out of an environment when this is possible. Thus,

(Semf2) for all $x \varepsilon \text{Id}$, $\alpha \varepsilon A$

$$S[\![x]\!]\,\alpha = \begin{cases} \text{proj}_{\text{dom }\alpha}\, x \varepsilon \text{ Env } \alpha \xrightarrow[|K|]{} \text{Mng}(\alpha x) & \text{if } x \varepsilon \text{ dom } \alpha, \\ u_\alpha \varepsilon \text{ Env } \alpha \xrightarrow[|K|]{} \text{Mng } \underline{\text{ns}} & \text{otherwise.} \end{cases}$$

In the case of an application we want to coerce the operand into a meaning of the right type and then apply the operator, provided there is no error due to mismatched types. This gives

(Semf3) for all ℓ, $m \in L$, $\alpha \in A$



commutes, whenever

Type $\ell\ \alpha = \tau \Rightarrow \theta$, and Type $m\ \alpha = \tau' \leq \tau$,

whereas

$$S\ [\![\ell\ m]\!]\ \alpha = u_\alpha \in \text{Env}\ \alpha \xrightarrow[|K|]{} \text{Mng}\ \underline{ns},$$

otherwise.

There is a fourth condition connected with the semantics of λ-expressions, but its statement requires some preliminary groundwork.

Let $x \in Id$. For each $\alpha \in A$, $\tau \in T$ we seek to define

$$P_x\langle \alpha,\tau\rangle \in (\text{Env}\ \alpha)\times(\text{Mng}\tau) \xrightarrow[|K|]{} \text{Env}\ [\alpha|x:\tau].$$

The idea is that $P_x\langle \alpha,\tau\rangle$ is the obvious isomorphism when $x \notin \text{dom}\ \alpha$, and $P_x\langle \alpha,\tau\rangle$ changes the meaning object associated with x from Mng (αx) to Mng τ when $x \in \text{dom}\ \alpha$. The morphism $P_x\langle \alpha,\tau\rangle$ has an important role to play in the treatment of the semantics of identifier binding. Denote by

$$\pi_\alpha \in (\text{Env}\ \alpha)\times(\text{Mng}\ \tau) \xrightarrow[|K|]{} \text{Env}\ \alpha$$

the projection onto the first component. For each $y \in \mathrm{dom}\,[\alpha \mid x{:}\tau]$ define

$$p_{x,\alpha,\tau}\ y \in (\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) \xrightarrow[|K|]{} \mathrm{Mng}\ ([\alpha \mid x{:}\tau]\, y)$$

as follows:

(1) if $y = x$, then

$$p_{x,\alpha,\tau}\ x \in (\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) \xrightarrow[|K|]{} \mathrm{Mng}\ \tau$$

is the projection onto the second component,

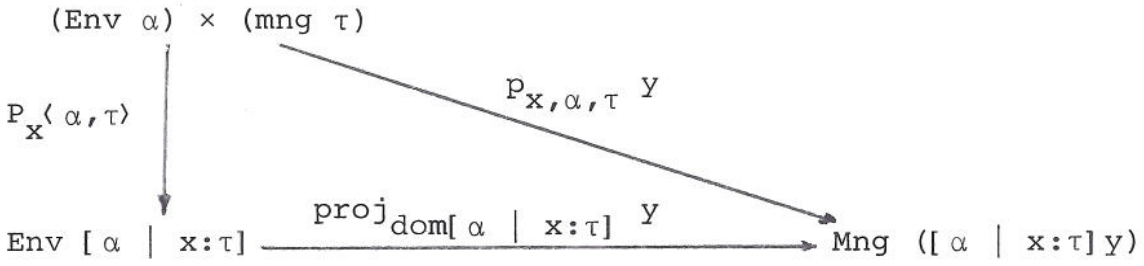
(2) if $y \neq x$, then

$$p_{x,\alpha,\tau}\ y \in (\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) \xrightarrow[|K|]{} \mathrm{Mng}\ (\alpha y)$$

is the composition

$$(\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) \xrightarrow{\pi_\alpha} \mathrm{Env}\ \alpha \xrightarrow{\mathrm{proj}_{\mathrm{dom}\ \alpha}\ y} \mathrm{Mng}\ (\alpha y)\ .$$

Now take $P_x\langle \alpha,\tau\rangle$ to be the unique morphism such that



commutes for all $y \in \mathrm{dom}\,[\alpha \mid x{:}\tau]$.

The following technical lemma is used in the proof of the proposition that follows it.

Lemma 4.1: Suppose $x \in \mathrm{Id}$, $\alpha \leq_A \beta$, and $\tau \leq_{|T|} \theta$. For all $y \in \mathrm{dom}\,[\beta \mid x{:}\tau]$, the diagram

$$\begin{array}{ccc} (\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) & \xrightarrow{p_{x,\alpha,\tau}\, y} & \mathrm{Mng}\ ([\alpha \mid x{:}\tau]\ y) \\ \Big\downarrow \mathrm{Env}(\alpha \leq \beta) \times \mathrm{Mng}(\tau \leq \theta) & & \Big\downarrow \mathrm{Mng}([\alpha \mid x{:}\tau]\ y \leq [\beta \mid x{:}\theta]\ y) \\ (\mathrm{Env}\ \beta) \times (\mathrm{Mng}\ \theta) & \xrightarrow{p_{x,\beta,\theta}\, y} & \mathrm{Mng}\ ([\beta \mid x{:}\theta]\ y) \end{array}$$

commutes.

Proof: Suppose y = x. Then the diagram becomes

$$\begin{array}{ccc} (\mathrm{Env}\ \alpha) \times (\mathrm{Mng}\ \tau) & \longrightarrow & \mathrm{Mng}\ \tau \\ \Big\downarrow \mathrm{Env}(\alpha \leq \beta) \times \mathrm{Mng}(\tau \leq \theta) & & \Big\downarrow \mathrm{Mng}\ (\tau \leq \theta)\ , \\ (\mathrm{Env}\ \beta) \times (\mathrm{Mng}\ \theta) & \longrightarrow & \mathrm{Mng}\ \theta \end{array}$$

where the horizontal arrows are projections onto the second component; this diagram commutes by the definition of the product morphism Env $(\alpha \leq \beta) \times$ Mng $(\tau \leq \theta)$.

Suppose $y \neq x$. By decomposing the horizontal arrows we get

$$\begin{array}{ccccc} (\mathrm{Env}\alpha) \times (\mathrm{Mng}\tau) & \xrightarrow{\pi_\alpha} & \mathrm{Env}\ \alpha & \xrightarrow{\mathrm{proj}_{\mathrm{dom}\,\alpha}\, y} & \mathrm{Mng}\ (\alpha y) \\ \Big\downarrow \mathrm{Env}(\alpha \leq \beta) \times \mathrm{Mng}(\tau \leq \theta) & & \Big\downarrow \mathrm{Env}(\alpha \leq \beta) & & \Big\downarrow \mathrm{Mng}(\alpha y \leq \beta y) \\ (\mathrm{Env}\beta) \times (\mathrm{Mng}\theta) & \xrightarrow{\pi_\beta} & \mathrm{Env}\ \beta & \xrightarrow{\mathrm{proj}_{\mathrm{dom}\,\beta}\, y} & \mathrm{Mng}\ (\beta y) \end{array} .$$

The left-hand square commutes by the definition of the product morphism Env $(\alpha \leq \beta) \times$ Mng $(\tau \leq \theta)$, and the right-hand square commutes by the definition of the functor Env. □

Proposition 4.2: Let $x \in \mathrm{Id}$. Then P_x is a natural transformation:

$$P_x \in (\text{Env } -) \times (\text{Mng } -) \xrightarrow[A\times|T|\Rightarrow|K|]{} \text{Env } [\,- \mid x{:}-\,]\,.$$

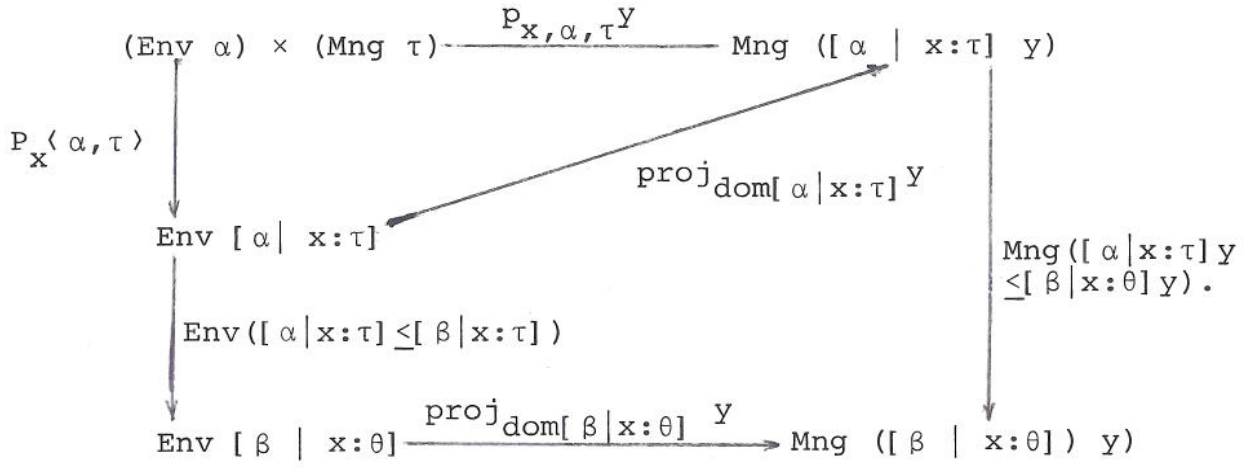
Proof: Suppose $\alpha \leq_A \beta$ and $\tau \leq_{|T|} \theta$. We must verify that

$$\begin{array}{ccc} (\text{Env } \alpha) \times (\text{Mng } \tau) & \xrightarrow{P_x\langle \alpha,\tau\rangle} & \text{Env } [\,\alpha \mid x{:}\tau\,] \\ \downarrow{\scriptstyle \text{Env}(\alpha\leq\beta)\times\text{Mng}(\tau\leq\theta)} & & \downarrow{\scriptstyle \text{Env}([\,\alpha\mid x:\tau\,]\leq[\,\beta\mid x:\theta\,])} \\ (\text{Env } \beta) \times (\text{Mng } \theta) & \xrightarrow{P_x\langle \beta,\theta\rangle} & \text{Env } [\,\beta \mid x{:}\theta\,] \end{array}$$

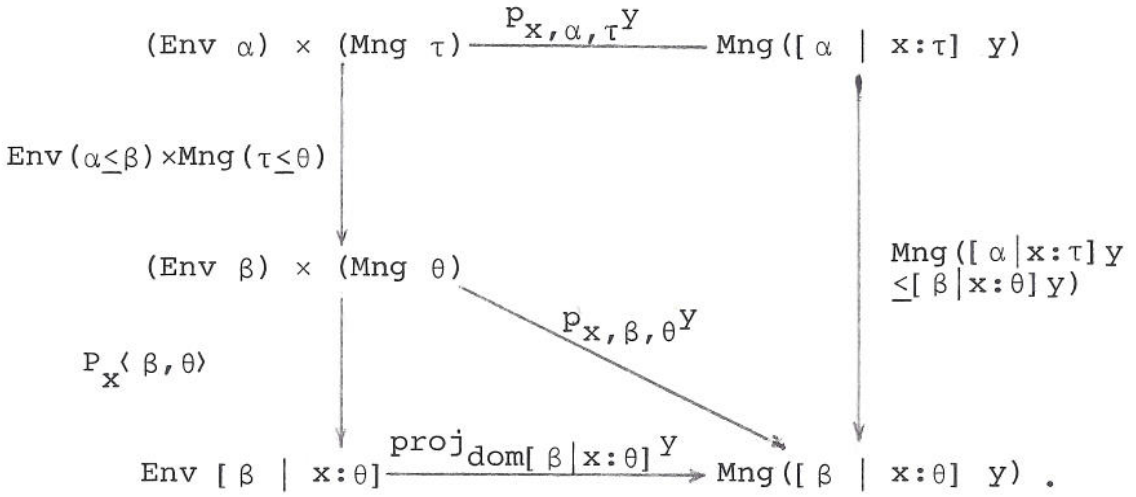
is a commutative diagram. We will use the fact that

$$\text{Env } [\,\beta \mid x{:}\theta\,] \;=\; \Pi_{\text{dom}[\,\beta \mid x:\theta\,]}\ (\text{Mng} \circ [\,\beta \mid x{:}\theta\,])$$

together with the universal mapping property for products. First, for each $y \in \text{dom } [\,\beta \mid x{:}\theta\,]$, consider the diagram



The inner diagram on the top commutes by the definition of $P_x\langle \alpha,\tau\rangle$ and commutativity of the inner diagram on the bottom follows from the definition of Env; thus, the outer compositions are equal. Second, for each $y \in \text{dom } [\,\beta \mid x{:}\theta\,]$, consider the diagram



Here, the top inner diagram commutes by the preceding lemma, and the lower inner diagram commutes by the definition of $P_x\langle \beta,\theta\rangle$. Consequently the outer diagram commutes. By the universal mapping property for products, the left-hand vertical compositions in the two diagrams must be equal. □

Now let's return to the statement of the last property that we deem desirable for the collection of those functions

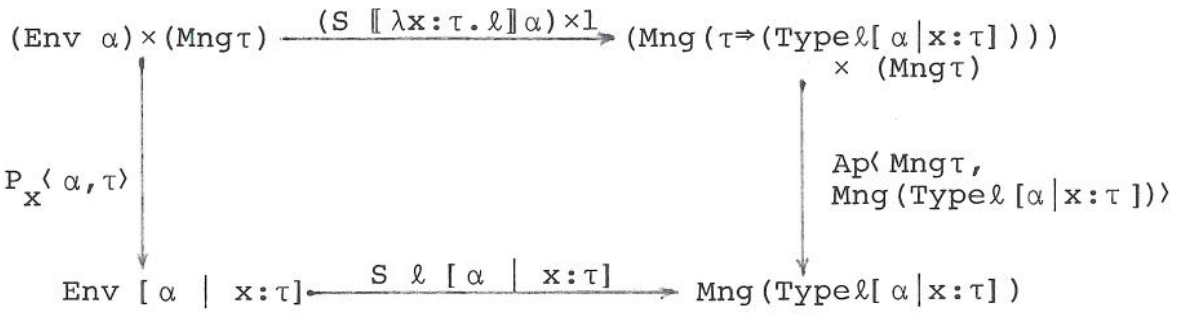
$$S \in |L| \to ObA \to Ar\ |K|$$

of which Semf is a member. We expect the meaning of a λ-expression in the context of a type assignment to be the abstraction of a morphism connected with the meaning of the body of the λ-expression in the context of the type assignment altered to take into account the binding of an identifier to a type. Therefore,

(Semf4) for all $x \in Id$, $\tau \in T$, $\ell \in L$, $\alpha \in A$,

$$S\ [\![\lambda x{:}\tau.\ell]\!]\ \alpha \in Env\ \alpha \xrightarrow{|K|} Mng\ (\tau \Rightarrow (Type\ \ell\ [\alpha \mid x{:}\tau]))$$

is the unique morphism such that



commutes.

Our investigation of Semf involves two Λ-algebras U and V, where U is a subalgebra of V. Take the carrier of V to be the set of ordered pairs ⟨ T, F⟩ such that

(1) $T \varepsilon A \to |T|$ (hence, T is monotone), and

(2) F is an (Ob A)- indexed collection of arrows of $|K|$ satisfying for all $\alpha \varepsilon A$

$$F\ \alpha \varepsilon \text{Env}\ \alpha \xrightarrow{\ |K|\ } \text{Mng}\ (T\ \alpha)$$

The carrier of the subalgebra U of V consists of those ⟨ T, F⟩ ε $|V|$ such that F is a natural transformation:

$$F \varepsilon \text{Env} \xrightarrow{\ A \Rightarrow |K|\ } \text{Mng} \circ T \quad .$$

(Thus, the carrier of U is the collection of objects of the comma category (Env ↓ A ⇒ Mng) where

$$A \Rightarrow \text{Mng} \varepsilon (A \Rightarrow |T|) \to (A \Rightarrow |K|)$$

is the functor induced by

$$\text{Mng} \varepsilon T \xrightarrow{\ TypeAlg\ } K \quad .$$

The significance of this observation is not completely clear.)

Recall that $A \to |T|$ is the carrier of the Λ-algebra E. We turn now to the interpretations in V of the operators

in Λ.

Let $x \varepsilon Id$. Let

$$V_x = \langle E_x , F_x \rangle ,$$

where

$$F_x\ \alpha\ \varepsilon\ Env\ \alpha \xrightarrow{|K|} Mng\ (E_x\ \alpha)\ ,\quad \alpha \varepsilon A\ ,$$

will be defined shortly. Upon recalling that

$$E_x\ \alpha = \alpha^{\infty}\ x\ = \begin{cases} \alpha x & \text{if} \quad x \varepsilon \text{dom } \alpha \\ \underline{ns} & \text{otherwise} , \end{cases}$$

we let

$$F_x\ \alpha\ = \begin{cases} proj_{dom\ \alpha}\ x \varepsilon Env\ \alpha \xrightarrow{|K|} Mng\ (\alpha x) & \text{if} \quad x \varepsilon \text{dom } \alpha, \\ u_\alpha \varepsilon Env\ \alpha \xrightarrow{|K|} Mng\ \underline{ns} & \text{otherwise} . \end{cases}$$

Let $\langle T, F\rangle$ and $\langle T', F'\rangle$ be in $|V|$. Let

$$V_{ap}\ \langle\langle T, F\rangle\ ,\ \langle T', F'\rangle\rangle\ =\ \langle T_{ap}\ ,\ F_{ap}\rangle$$

where

$$T_{ap}\ =\ E_{ap}\ \langle T, T'\rangle$$

and

$$F_{ap}\ \alpha\ \varepsilon\ Env \xrightarrow{|K|} Mng\ (T_{ap}\ \alpha)\ ,\quad \alpha \varepsilon A,$$

will be defined below. Note that, for $\alpha \varepsilon A$,

$$T_{ap}\ \alpha\ = \begin{cases} \theta & \text{if} \quad T\alpha = \tau \Rightarrow \theta \quad \text{and} \quad T'\alpha \leq \tau\ , \\ \underline{ns} & \text{otherwise} . \end{cases}$$

This leads to two cases in the definition of F_{ap}. If $\alpha \varepsilon A$, $T\alpha = \tau \Rightarrow \theta$, and $T'\alpha \leq \tau$, then let $F_{ap}\ \alpha$ be defined by the commutative diagram

$$\begin{array}{ccc} \text{Env } \alpha & \xrightarrow{F_{ap}{}^{\alpha}} & \text{Mng } \theta \\ \Big\downarrow \langle F\alpha,\ F'\alpha\rangle & & \Big\uparrow \text{Ap}\langle \text{Mng}\tau,\ \text{Mng}\theta\rangle \\ (\text{Mng } (T\alpha)) \times (\text{Mng } (T'\alpha)) & \xrightarrow{1 \times \text{Mng } (T'\alpha \leq \tau)} & (\text{Mng}(\tau \Rightarrow \theta)) \times (\text{Mng } \tau) \end{array}$$

For other $\alpha \varepsilon A$, $F_{ap}\alpha$ is the unique arrow to the nonsense object:

$$F_{ap}\alpha = u_{\alpha} \varepsilon \text{ Env } \alpha \xrightarrow{|K|} \text{Mng } \underline{ns}$$

Let $x \varepsilon \text{Id}$, $\tau \varepsilon T$, $\langle T, F\rangle \varepsilon |V|$. Let

$$V_{\lambda x:\tau} \langle T, F\rangle = \langle E_{\lambda x:\tau}\ T\ ,\ F_{\lambda x:\tau}\rangle$$

where the definition of

$$F_{\lambda x:\tau}\ \alpha \varepsilon \text{ Env } \alpha \xrightarrow{|K|} \text{Mng } (E_{\lambda x:\tau}\ T\ \alpha)\ ,\ \alpha \varepsilon A,$$

follows. Recall that

$$E_{\lambda x:\tau}\ T\ \alpha = \tau \Rightarrow (T\ [\alpha \mid x:\tau])\ .$$

For $\alpha \varepsilon A$, let $F_{\lambda x:\tau}\ \alpha$ be the unique morphism in $|K|$ such that

$$\begin{array}{ccc} (\text{Env}\alpha) \times (\text{Mng}\tau) & \xrightarrow{(F_{\lambda x:\tau}{}^{\alpha}) \times 1} & (\text{Mng}(\tau \Rightarrow (T[\alpha \mid x:\tau]))) \times (\text{Mng}\tau) \\ \Big\downarrow P_x\langle \alpha, \tau\rangle & & \Big\downarrow \text{Ap}\langle \text{Mng}\tau, \text{Mng}(T[\alpha \mid x:\tau])\rangle \\ \text{Env } [\alpha \mid x:\tau] & \xrightarrow{F[\alpha \mid x:\tau]} & \text{Mng } (T\ [\alpha \mid x:\tau]) \end{array}$$

commutes. This completes the definition of the Λ-algebra V.

Theorem 4.3: U is a subalgebra of the Λ-algebra V.

Proof: We must show that $|U|$ is closed under the interpretations in V of the operators in Λ.

Let $x \in Id$, and consider $V_x = \langle E_x, F_x \rangle$. We will show $V_x \in |U|$ by demonstrating that F_x is a natural transformation. Suppose $\alpha \leq_A \beta$, so that $\alpha^\infty x \leq \beta^\infty x$. Consider the diagram

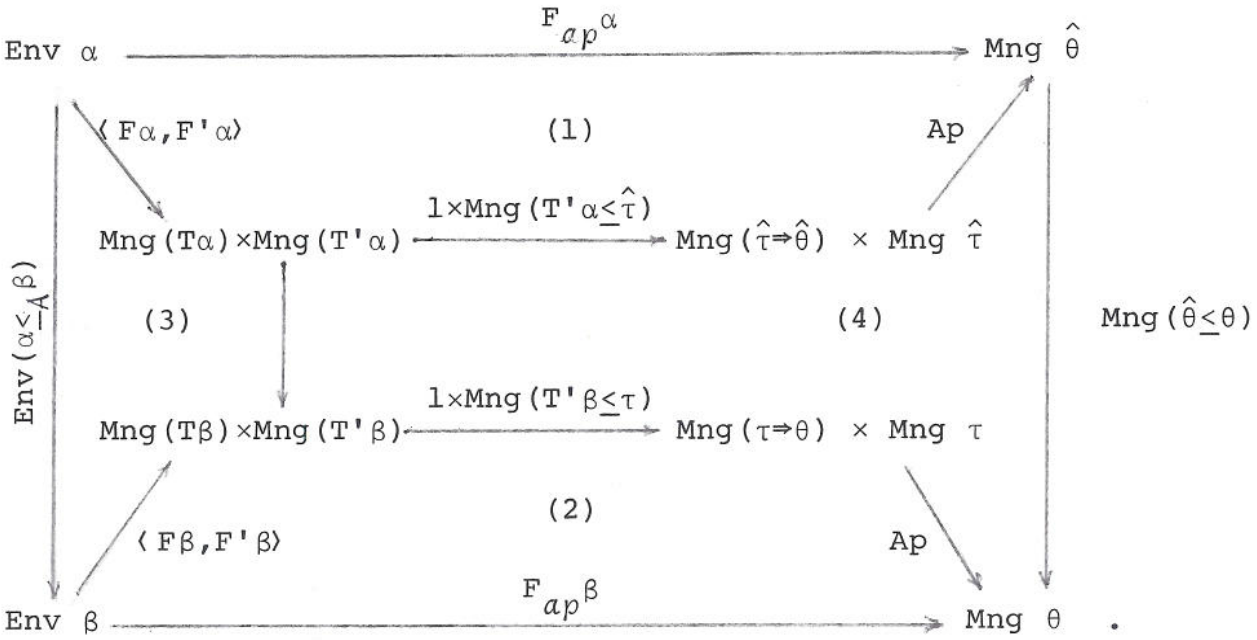
$$\begin{array}{ccc} \text{Env } \alpha & \xrightarrow{F_x\alpha} & \text{Mng } (\alpha^\infty x) \\ \downarrow \text{Env}(\alpha \leq \beta) & & \downarrow \text{Mng}(\alpha x \leq \beta x) \\ \text{Env } \beta & \xrightarrow{F_x\beta} & \text{Mng } (\beta^\infty x) \end{array} .$$

If $x \notin \text{dom } \beta$, then $\beta^\infty x = \underline{ns}$ and the fact that Mng $(\beta\ x)$ is terminal gives commutativity of the diagram. If $x \in \text{dom } \beta$, then also $x \in \text{dom } \alpha$, $\alpha x = \alpha^\infty x$, $\beta x = \beta^\infty x$, and commutativity follows from the definition of Env $(\alpha \leq \beta)$.

Let $\langle T, F \rangle$ and $\langle T', F' \rangle$ be in $|U|$, and consider

$$V_{ap}\ \langle \langle T, F \rangle\ , \langle T', F' \rangle \rangle = \langle T_{ap}\ , F_{ap} \rangle .$$

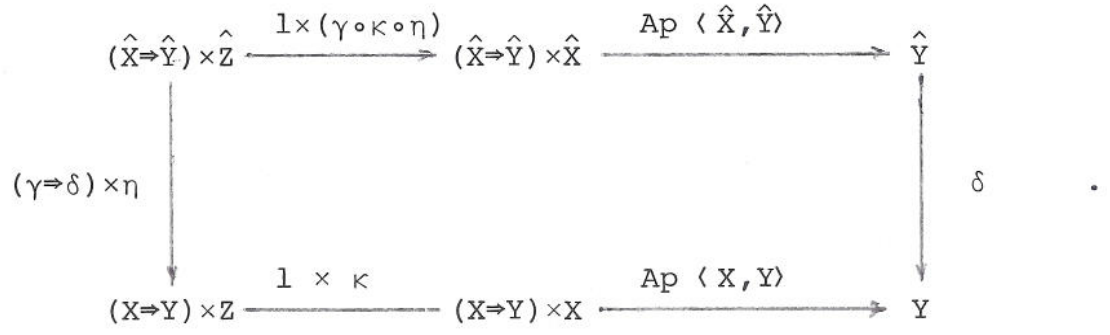
We wish to show this is in $|U|$, i.e. to show F_{ap} is natural. To do this let $\alpha \leq_A \beta$. Consider what happens when $T\beta = \tau \Rightarrow \theta$ and $T'\beta \leq \tau$. Then we can write $T\alpha = \hat{\tau} \Rightarrow \hat{\theta}$ where $T'\alpha \leq T'\beta \leq \tau \leq \hat{\tau}$ and $\hat{\theta} \leq \theta$. We wish to demonstrate the commutativity of the outer square in the diagram



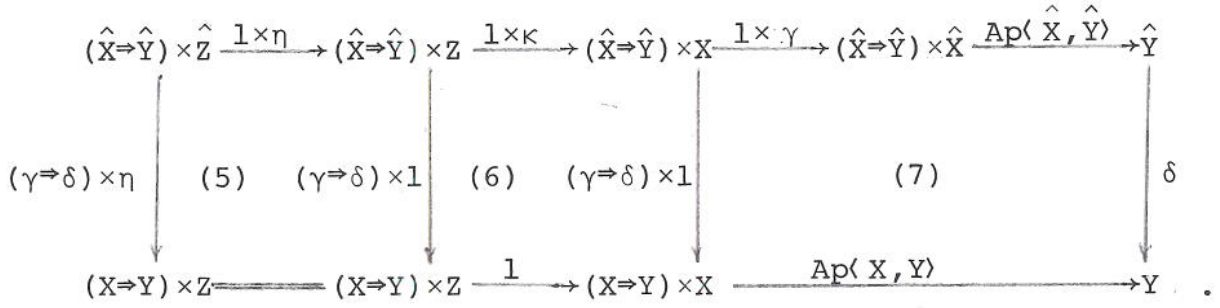
The interior diagrams labelled (1) and (2) commute by definition, and the square labelled (3) commutes by the naturality of F and F' . Let

$$X = \mathrm{Mng}\ \tau\ , \quad \hat{X} = \mathrm{Mng}\ \hat{\tau}\ , \quad \gamma = \mathrm{Mng}\ (\tau \leq \hat{\tau})\ ,$$
$$Y = \mathrm{Mng}\ \theta\ , \quad \hat{Y} = \mathrm{Mng}\ \hat{\theta}\ , \quad \delta = \mathrm{Mng}\ (\hat{\theta} \leq \theta)\ ,$$
$$Z = \mathrm{Mng}(T'\beta)\ , \quad \hat{Z} = \mathrm{Mng}(T'\alpha), \quad \eta = \mathrm{Mng}\ (T'\alpha \leq T'\beta)\ ,$$
$$\kappa = \mathrm{Mng}\ (T'\beta \leq \tau)\ .$$

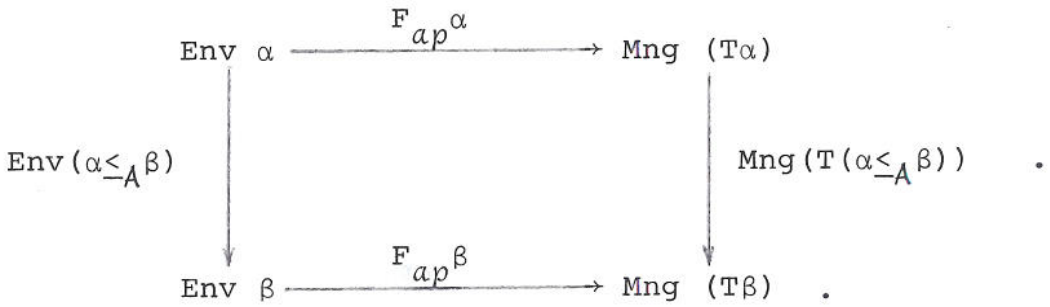
Then the interior diagram labelled (4) becomes



Equivalently, we have



Squares (5) and (6) obviously commute. Diagram (7) commutes by the definition of $\gamma\Rightarrow\delta$. Hence diagram (4) commutes, yielding commutativity of the original diagram. In other cases $T\beta$ = ns, which automatically gives commutativity of



This concludes the proof that F_{ap} is a natural transformation.

Let $x \in Id$, $\tau\varepsilon T$, $\langle T,F\rangle \in |U|$. Let

$$V_{\lambda x:\tau}\langle T,F\rangle = \langle E_{\lambda x:\tau}\ T,\ F_{\lambda x:\tau}\rangle \ ;$$

we aim to show $F_{\lambda x:\tau}$ is a natural transformation.

Suppose $\alpha \leq_A \beta$. Since Mng is a type algebra homomorphism the morphisms

$$\mathrm{Mng}\ ((\tau \Rightarrow (T\ [\alpha \mid x{:}\tau])) \leq (\tau \Rightarrow (T\ [\beta \mid x{:}\tau])))$$

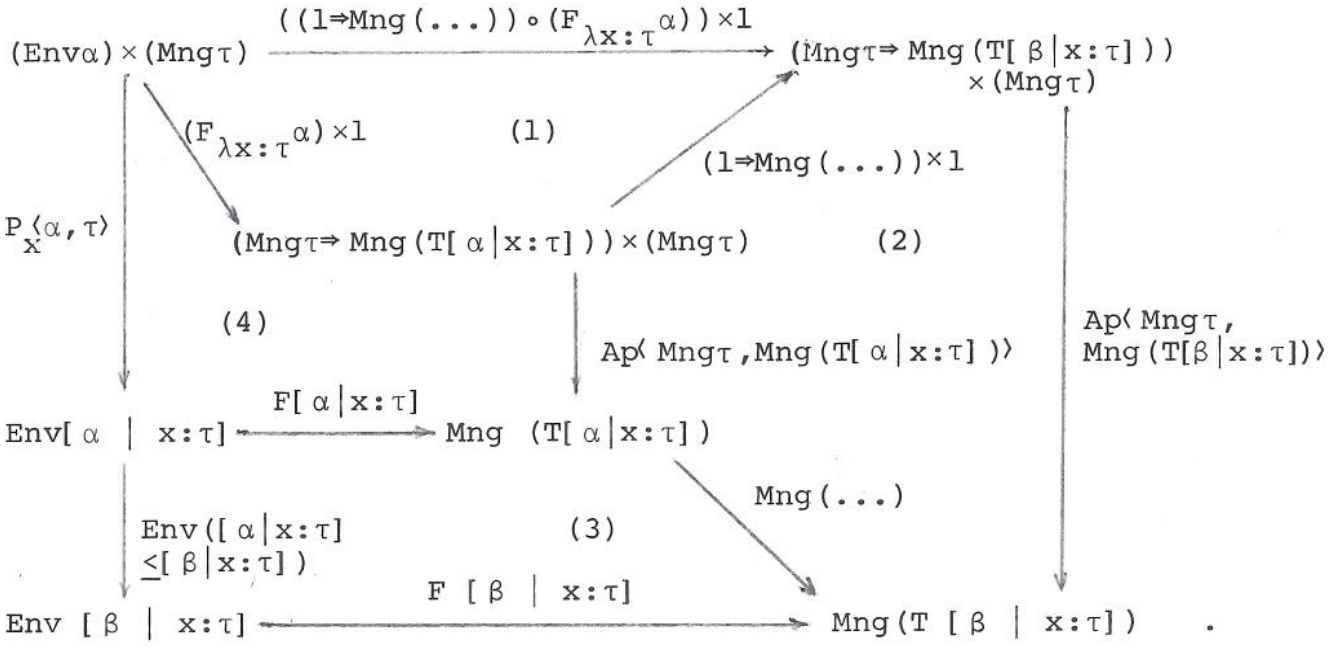
and

$$1_{\mathrm{Mng}\tau} \xRightarrow[|K|]{} \mathrm{Mng}\ ((T\ [\alpha \mid x{:}\tau]) \leq (T\ [\beta \mid x{:}\tau]))$$

are equal. Therefore our aim is to prove that

$$\begin{array}{ccc}
\mathrm{Env}\ \alpha & \xrightarrow{F_{\lambda x:\tau}{}^{\alpha}} & \mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ (T\ [\alpha \mid x{:}\tau]) \\
\Big\downarrow {\scriptstyle \mathrm{Env}(\alpha \leq \beta)} & & \Big\downarrow {\scriptstyle 1 \Rightarrow \mathrm{Mng}((T\ [\alpha \mid x:\tau]) \leq (T\ [\beta \mid x:\tau]))} \\
\mathrm{Env}\ \beta & \xrightarrow{F_{\lambda x:\tau}{}^{\beta}} & \mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ (T\ [\beta \mid x{:}\tau])
\end{array}$$

commutes. The method of proof is based on the universal mapping property (Cart3), with X = Env α, Y = Mng τ, Z = Mng $(T\ [\beta \mid x{:}\tau])$. Consider the diagram



The four small diagrams commute for the following reasons:

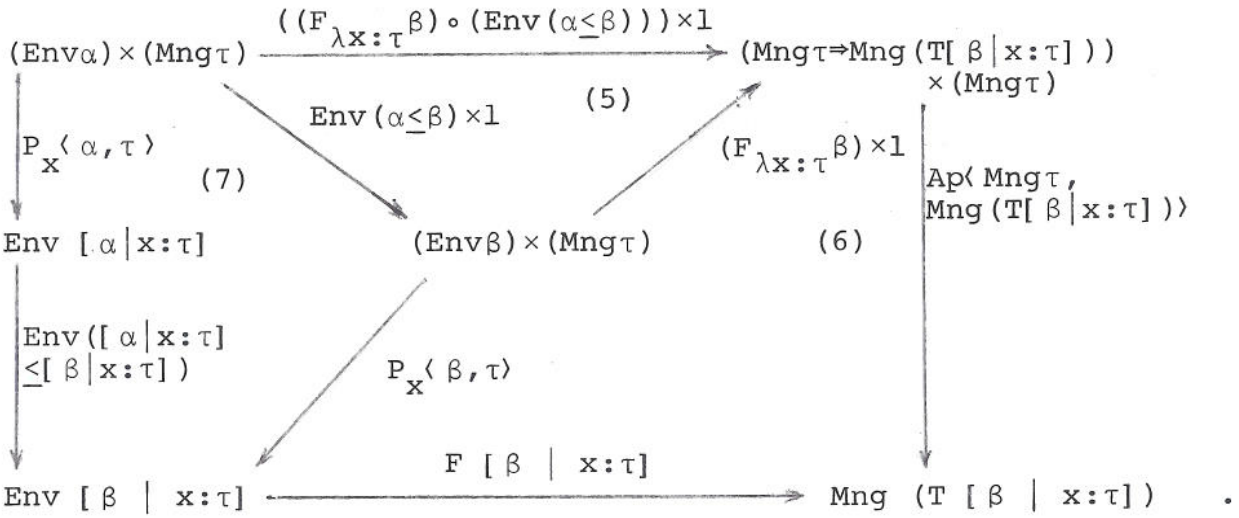
(1) $- \times 1$ is a functor,

(2) naturality of Ap in its second argument,

(3) F is a natural transformation,

(4) definition of $F_{\lambda x:\tau}\alpha$.

Thus the whole diagram commutes. Next, consider the diagram



The reasons for the commutativity of the small diagrams is as follows:

(5) $- \times 1$ is a functor,

(6) definition of $F_{\lambda x:\tau}\beta$,

(7) naturality of P_x.

So the entire diagram commutes. We finally apply the universal mapping property (Cart3) to get commutativity of the square in question. □

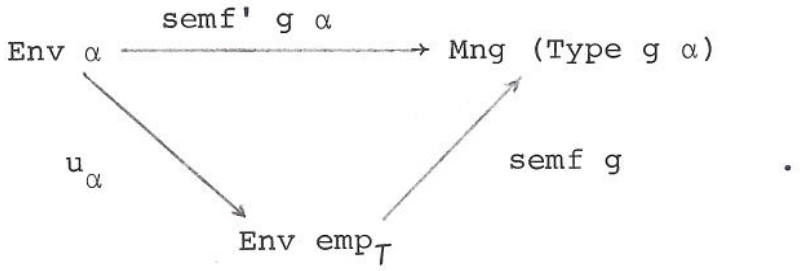
The condition (Semf1) suggests that we should use the function

$$\text{semf} \in G \to \text{Ar } |K|$$

to create

$$\text{semf}' \in G \to \text{Ob } A \to \text{Ar } |K| \ ,$$

where semf' is defined for $g \in G$, $\alpha \in A$ by the commutative diagram



Now define

$$\text{sf} \in G \longrightarrow |V|$$

by

$$\text{sf } g = \langle \text{Type } g, \ \text{semf}' \ g \rangle$$

for $g \in G$. Extend sf to get

$$\text{Sf} \in L \xrightarrow[\Lambda\text{-Alg}]{} V,$$

the unique Λ-algebra homomorphism such that

$$\text{Sf } g = \text{sf } g \quad \text{for all } g \in G \ .$$

Let

$$\text{Pr}_1 \in |V| \to |E| \ , \qquad \text{Pr}_2 \in |V| \to \text{Ob}A \to \text{Ar } |K|$$

be the functions which select the components of the elements of

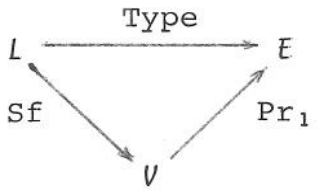
$|V|$. Thus $Pr_1 \langle T, F \rangle = T$ and $Pr_2 \langle T, F \rangle = F$.

Proposition 4.4: The function Pr_1 is a Λ-algebra homomorphism:

$$Pr_1 \in V \xrightarrow[\Lambda\text{-Alg}]{} E.$$

Proof: Clear. □

Corollary 4.5: The diagram



of Λ-algebra homomorphisms is commutative.

Proof: For all $g \in G$

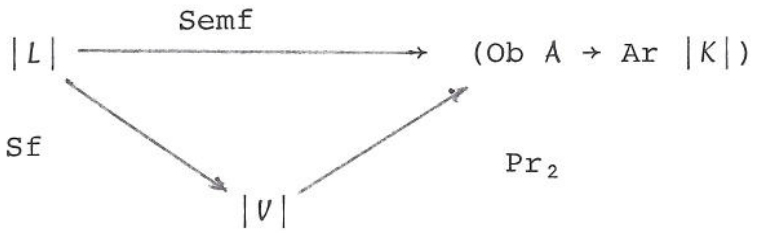
$$Type\ g = Pr_1\ (Sf\ g) \quad .$$

□

Define the semantic function

$$Semf \in |L| \to Ob\ A \to Ar\ |K|$$

by the commutative diagram



We are now ready to present Theorem 4.6, The Fundamental Theorem of Semantics for the coercive typed λ-calculus.

Theorem 4.6: (1) The function

$$\text{Semf} \in |L| \to \text{Ob } A \to \text{Ar } |K|$$

is the unique function satisfying conditions (Semf0), (Semf1), (Semf2), (Semf3) and (Semf4).

(2) Furthermore, for each $\ell \in |L|$, Semf ℓ is a natural transformation:

$$\text{Semf } \ell \in \text{Env} \xrightarrow[A \Rightarrow |K|]{} \text{Mng} \circ (\text{Type } \ell) .$$

Proof: (1) First we will verify that Semf satisfies the conditions. Let $\ell \in L$, $\alpha \in A$, $g \in G$, $m \in L$, $x \in \text{Id}$, $\tau \in T$.

From Corollary 4.5 and the definition of Semf,

$$\text{Sf } \ell = \langle \text{Type } \ell, \text{Semf } \ell \rangle .$$

Since Sf $\ell \in |V|$, we see

$$\text{Semf } \ell\ \alpha \in \text{Env } \alpha \xrightarrow[|K|]{} \text{Mng } (\text{Type } \ell\ \alpha).$$

This is (Semf0). Observe

$$\langle \text{Type } g, \text{Semf } g \rangle = \text{Sf } g = \text{sf } g = \langle \text{Type } g, \text{semf' } g \rangle .$$

Thus Semf g = semf' g. This is (Semf1). From the equation

$$\langle \text{Type } [\![x]\!], \text{Semf } [\![x]\!] \rangle = \text{Sf } [\![x]\!] = V_x = \langle E_x, F_x \rangle$$

we get Semf $[\![x]\!]\ \alpha = F_x\ \alpha$, which is just (Semf2).

The equation

$$\begin{aligned}\langle \text{Type } [\![\ell m]\!], \text{Semf } [\![\ell m]\!]\rangle &= \text{Sf } [\![\ell m]\!] \\ &= V_{ap}\ \langle (\text{Sf } \ell)\ (\text{Sf } m)\rangle \\ &= V_{ap}\ \langle\langle \text{Type } \ell, \text{Semf } \ell\rangle, \\ &\qquad \langle \text{Type } m, \text{Semf } m\rangle\rangle \\ &= \langle T_{ap}, F_{ap}\rangle\end{aligned}$$

implies

$$\text{Semf } [\![\ell\ m]\!]\ \alpha = F_{ap}\ \alpha\ ,$$

and this is (Semf3). Finally,

$$\begin{aligned}\langle \text{Type } [\![\lambda x{:}\tau.\ell]\!],\ \text{Semf } [\![\lambda x{:}\tau.\ell]\!]\rangle &= \text{Sf } [\![\lambda x{:}\tau.\ell]\!] \\ &= V_{\lambda x:\tau}\ (\text{Sf } \ell) \\ &= V_{\lambda x:\tau}\ \langle E_{\lambda x:\tau}, F_{\lambda x:\tau}\rangle\end{aligned}$$

gives

$$\text{Semf } [\![\lambda x{:}\tau.\ell]\!]\ \alpha = F_{\lambda x:\tau}\ \alpha\ ;$$

this is (Semf4). So Semf satisfies the five conditions.

Suppose $S \in |L| \to \text{Ob } A \to \text{Ar } |K|$ satisfies the five conditions. By condition (Semf0), for each $\ell \in L$, $\alpha \in A$,

$$S\ \ell\ \alpha \in \text{Env } \alpha \xrightarrow{|K|} \text{Mng } (\text{Type } \ell\ \alpha)\ ,$$

so that we can define

$$S' \in |L| \to |V|$$

by

$$S'\ \ell = \langle \text{Type } \ell, S\ \ell\rangle .$$

Using (Semf2), (Semf3), and (Semf4) it is straightforward to

show that S' is a homomorphism:

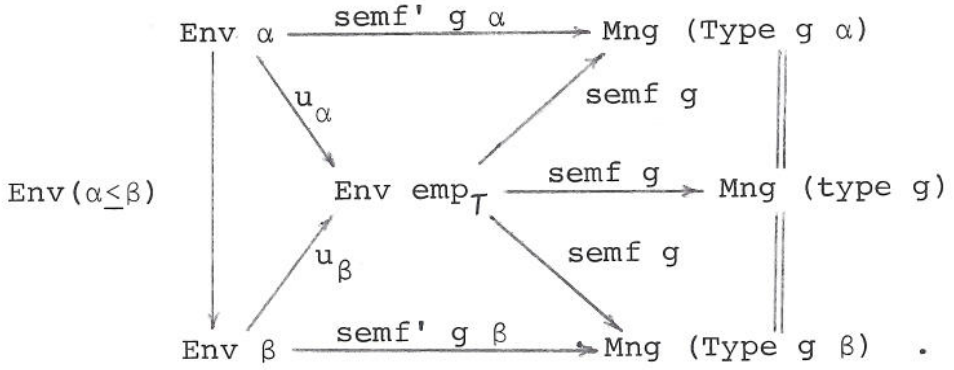
$$S' \in L \xrightarrow[\Lambda\text{-}Alg]{} V.$$

By condition (Semf1), the restriction of S' to G is sf. Hence S' = Sf, and S = $Pr_2 \circ S' = Pr_2 \circ Sf$ = Semf. This proves the uniqueness of Semf.

(2) We claim that for each g ∈ G, semf' g is a natural transformation:

$$\text{semf' g} \in \text{Env} \xrightarrow[A \Rightarrow |K|]{} \text{Mng} \circ (\text{Type g}) .$$

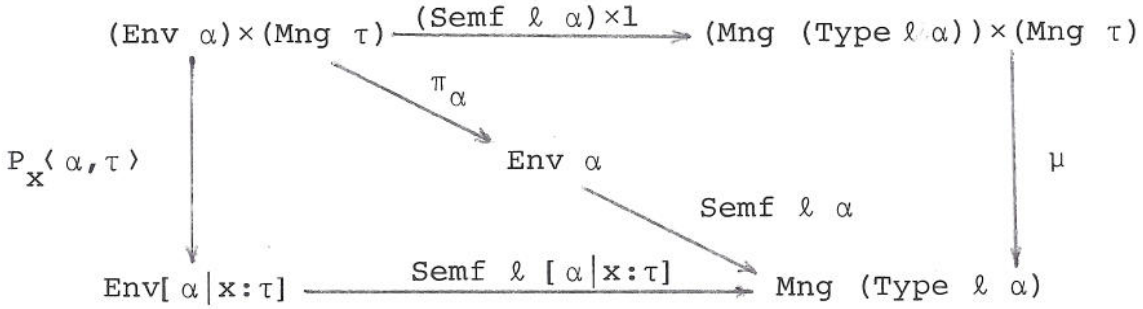
This is a consequence of the obvious commutativity of the interior triangles in



Thus the image of sf is contained in the subalgebra U of V. Consequently the image of Sf is contained in U. Therefore, for each ℓ the second component of Sf ℓ is a natural transformation, i.e. Semf ℓ is a natural transformation. □

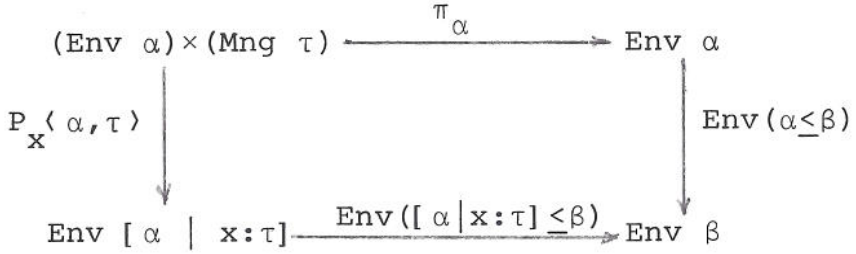
The next proposition explores the semantic effect of expanding the domain of a type assignment by adding an identifier which is not free in a phrase ℓ. Note the role played by the naturality of Semf ℓ in the proof.

Proposition 4.7: Let $\ell \in L$, $\alpha \in A$, $\tau \in T$, $x \in \mathrm{Id}$. Suppose $x \notin \mathrm{Free}\ \ell$. Then the diagram



commutes, where μ is the projection onto the first component. (Remark: The diagram makes sense by Corollary 3.11 which insures that $\mathrm{Mng}\ (\mathrm{Type}\ \ell\ [\alpha \mid x{:}\tau]) = \mathrm{Mng}\ (\mathrm{Type}\ \ell\ \alpha)$.)

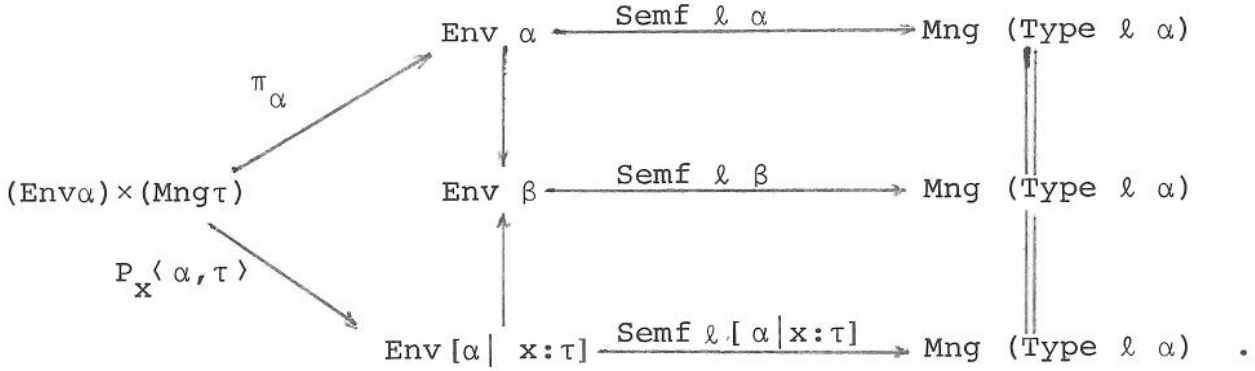
Proof: The upper half of the diagram obviously commutes, so we may turn our attention to the lower half. Denote the restriction of α to $(\mathrm{dom}\ \alpha) - \{x\}$ by β. Then $\alpha \leq \beta$ and $[\alpha \mid x{:}\tau] \leq \beta$. It is routine to check that



commutes. Since $\alpha = [\beta \mid x{:}\alpha x]$, Corollary 3.11 gives

$$\mathrm{Type}\ \ell\ \alpha = \mathrm{Type}\ \ell\ \beta \quad .$$

From the naturality of Semf ℓ we now get the commutativity of



Thus, the lower half of the diagram also commutes. □

In Chapter 3 we introduced Type^*, a typing function applicable to phrase assignments; Type^* was derived from Type, the typing function applicable to phrases. Similarly we will introduce Semf^*, a semantic function that is applicable to phrase assignments and is derived from Semf.

Let I be a finite set. It is well-known and easy to check that the distinguished product functor $\Pi_I^{|K|}$ for the Cartesian closed category $|K|$ gives rise in a natural way to a distinguished product functor $\Pi_I^{A \Rightarrow |K|}$ for the category $A \Rightarrow |K|$; simply construct products in $A \Rightarrow |K|$ at the level of $|K|$. To elaborate, let

$$F \in I \to (A \to |K|)$$

be an I-indexed collection of functors. Regard I as a discretely ordered poset, so that I may be viewed as a category. Then there is no difference between the functor category $I \Rightarrow |K|$ and the I-ary Cartesian product $|K|^I$. Let the functor

$$F' \in A \to (I \Rightarrow |K|)$$

be obtained from F by interchange of arguments:

$$F'\ \alpha\ x = F\ x\ \alpha\ .$$

The product of F,

$$\Pi_I^{A \Rightarrow |K|}\ F \in A \to |K|$$

is defined as follows:

(1) for $\alpha \in A$,

$$\Pi_I^{A \Rightarrow |K|}\ F\ \alpha = \Pi_I^{|K|}\ (F'\ \alpha)$$

and

(2) for $\alpha \leq_A \beta$, $\Pi_I^{A \Rightarrow |K|}\ F\ (\alpha \leq \beta)$ is the unique arrow in $|K|$ such that for all $x \in I$

$$\begin{array}{ccc} \Pi_I^{|K|}\ (F'\ \alpha) & \xrightarrow{\text{proj}} & F'\ \alpha\ x = F\ x\ \alpha \\ \Big\downarrow \Pi_I^{A \Rightarrow |K|}\ F(\alpha \leq \beta) & & \Big\downarrow F'(\alpha \leq \beta)\ x = F\ x\ (\alpha \leq \beta) \\ \Pi_I^{|K|}\ (F'\ \beta) & \xrightarrow{\text{proj}} & F'\ \beta\ x = F\ x\ \beta \end{array}$$

commutes.

Let f be a phrase assignment. We have a particular interest in the (dom f)-indexed collection of functors

$$V\ f \in \text{dom}\ f \to (A \to |K|)$$

given by

$$V\ f\ x = \text{Mng} \circ (\text{Type}\ (fx))\ .$$

Let

$$(V\ f)'\ \varepsilon\ A \to (\mathrm{dom}\ f \Rightarrow |K|)$$

be the functor obtained by interchanging the arguments of V f.

Proposition 4.8: Let $f \varepsilon |L|^*$. Then

$$\mathrm{Env} \circ (\mathrm{Type}^*\ f) = \Pi^{A \Rightarrow |K|}_{\mathrm{dom}\ f}\ (V\ f)\ .$$

Proof: Let α be a type assignment. If $x \varepsilon \mathrm{dom}\ f$, then

$$\begin{aligned}(V\ f)'\ \alpha\ x &= V\ f\ x\ \alpha \\ &= \mathrm{Mng}\ (\mathrm{Type}\ (fx)\ \alpha) \\ &= \mathrm{Mng}\ (\mathrm{Type}^*\ f\ \alpha\ x)\ .\end{aligned}$$

Hence $(V\ f)'\ \alpha = \mathrm{Mng} \circ (\mathrm{Type}^*\ f\ \alpha)$. By the definition of Env,

$$\begin{aligned}(\mathrm{Env} \circ (\mathrm{Type}^*\ f))\ \alpha &= \mathrm{Env}\ (\mathrm{Type}^*\ f\ \alpha) \\ &= \Pi^{|K|}_{\mathrm{dom}\ f}\ (\mathrm{Mng}\ \ (\mathrm{Type}^*\ f\ \alpha)) \\ &= \Pi^{|K|}_{\mathrm{dom}\ f}\ ((V\ f)'\ \alpha) \\ &= \Pi^{A \Rightarrow |K|}_{\mathrm{dom}\ f}\ (V\ f)\ \ \alpha.\end{aligned}$$

So the two functors are equal on objects.

Suppose $\alpha \leq_A \beta$. By the monotonicity of $\mathrm{Type}^*\ f$,

$$(\mathrm{Env} \circ (\mathrm{Type}^*\ f))\ (\alpha \leq \beta) = \mathrm{Env}\ ((\mathrm{Type}^*\ f\ \alpha) \leq_A (\mathrm{Type}^*\ f\ \beta)).$$

From the definition of Env we obtain for each $x \varepsilon \mathrm{dom}\ f$ a commutative diagram

$$\begin{array}{ccc}
\Pi^{|K|}_{domf}\ (Mng \circ (Type^{*}\ f\ \alpha)) = \Pi^{|K|}_{domf}\ ((Vf)'\ \alpha) & \xrightarrow{proj} & (Vf)'\ \alpha\ x \\
\Big\downarrow (Env \circ (Type^{*} f))(\alpha \leq \beta) & & \Big\downarrow (Vf)'(\alpha \leq \beta) x \\
\Pi^{|K|}_{dom}\ (Mng \circ (Type^{*}\ f\ \beta)) = \Pi^{|K|}_{domf}\ ((Vf)'\ \beta) & \xrightarrow{proj} & (Vf)'\ \beta\ x
\end{array}$$

Therefore

$$(Env \circ (Type^{*}\ f))(\alpha \leq \beta) = \Pi^{A \Rightarrow |K|}_{dom\ f}\ (V\ f)(\alpha \leq \beta),$$

which shows that the functors are equal on arrows. □

For each phrase assignment $f \in |L|^{*}$, let

$$Semf^{*}\ f \in Env \xrightarrow[A \Rightarrow |K|]{} Env \circ (Type^{*}\ f)$$

be the unique natural transformation such that for all $x \in dom\ f$,

$$\begin{array}{ccccc}
 & & Env & & \\
 & \swarrow^{Semf^{*}\ f} & & \searrow^{Semf\ (fx)} & \\
Env \circ (Type^{*} f) = \Pi^{A \Rightarrow |K|}_{domf}\ (V\ f) & & \xrightarrow{proj} & & V\ f\ x = Mng \circ (Type(fx))
\end{array}$$

commutes.

We next present the basic theorem on the semantics of substitution. Beyond its intrinsic interest, it is a most useful tool for the investigation of the semantics of α-, β-, and η-reduction.

In the statement of Theorem 4.9, Godement multiplication, denoted by an infixed *, makes its first appearance. Godement multiplication is nothing more than a way of combining appropriate functors with natural transformations to get new natural transformations. Thus, let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ be categories. Suppose

$$F,\ F' \in \mathcal{A} \to \mathcal{B}, \quad G \in \mathcal{B} \to \mathcal{C}, \quad H,\ H' \in \mathcal{C} \to \mathcal{D}$$

and

$$\eta \in F \xrightarrow[\mathcal{A}\Rightarrow\mathcal{B}]{} F' \quad , \qquad \delta \in H \xrightarrow[\mathcal{C}\Rightarrow\mathcal{D}]{} H' \quad .$$

We obtain natural transformations

$$G * \eta \in G \circ F \xrightarrow[\mathcal{A}\Rightarrow\mathcal{C}]{} G \circ F' \quad ,$$

$$\delta * G \in H \circ G \xrightarrow[\mathcal{B}\Rightarrow\mathcal{D}]{} H' \circ G$$

by letting

$$(G * \eta)\ A = G(\eta A) \quad \text{for} \quad A \in \mathrm{Ob}\ \mathcal{A},$$

$$(\delta * G)\ B = \delta(GB) \quad \text{for} \quad B \in \mathrm{Ob}\ \mathcal{B} \quad .$$

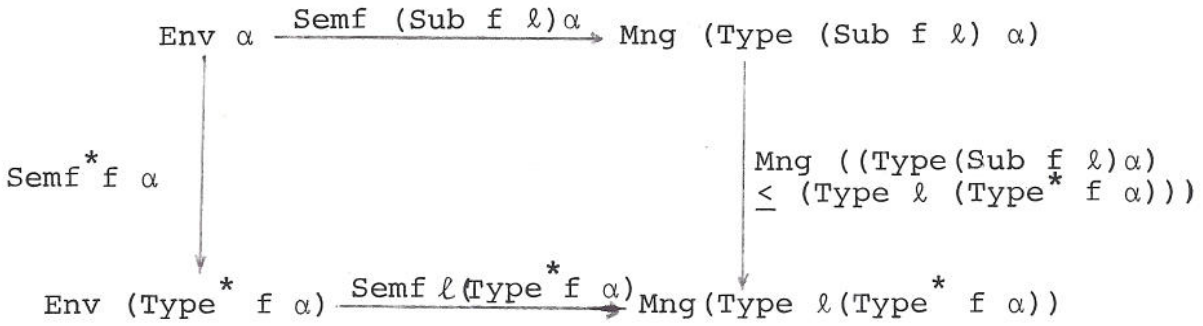
For more details about Godement multiplication, see Manes [5].

Theorem 4.9: Let $\ell \in L$. For all $f \in |L|^*$, the diagram

$$\begin{array}{ccc} \mathrm{Env} & \xrightarrow{\mathrm{Semf}\ (\mathrm{Sub}\ f\ \ell)} & \mathrm{Mng} \circ (\mathrm{Type}\ (\mathrm{Sub}\ f\ \ell)) \\ \Big\downarrow{\scriptstyle \mathrm{Semf}^* f} & & \Big\downarrow{\scriptstyle \mathrm{Mng}^*(\mathrm{Type}(\mathrm{Sub}\ f\ \ell) \leq (\mathrm{Type}\ \ell) \circ (\mathrm{Type}^* f))} \\ \mathrm{Env} \circ (\mathrm{Type}^* f) & \xrightarrow{(\mathrm{Semf}\ \ell)^*(\mathrm{Type}^* f)} & \mathrm{Mng} \circ (\mathrm{Type}\ \ell) \circ (\mathrm{Type}^* f) \end{array}$$

commutes in $A \Rightarrow |K|$.

Proof: We will show by induction on the structure of $\ell \in L$ that for all $f \in |L|^*$ and all $\alpha \in A$,

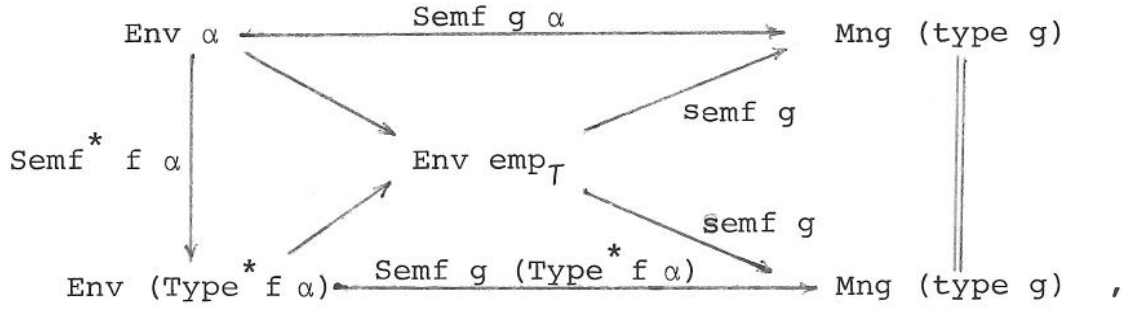


commutes in $|K|$.

(1) Let $\ell = g$ where $g \in G$. By (Subl), Sub f g = g. By (Typel),

$$\text{Type } g\ \alpha = \text{type } g = \text{Type } g\ (\text{Type}^*\ f\ \alpha).$$

Using (Semfl) and the fact that Env emp_T is terminal, we get commutativity of



as desired.

(2) Let $\ell = [\![x]\!]$ where $x \in Id$. On the one hand, suppose $x \notin$ dom f. Since dom (Type* f α) = dom f, we are able to use (Type2) to get

$$\text{Type } \ell \text{ (Type}^* \text{ f } \alpha) = \underline{\text{ns}}$$

Therefore the object in the lower right corner of the square whose commutativity is desired turns out to be terminal. Thus commutativity is automatic in this case. On the other hand, suppose $x \in$ dom f. By (Sub2), Sub f $[\![x]\!]$ = f x. By Theorem 3.12, the right-hand arrow is an identity morphism, and the diagram whose commutativity we want may be written as

$$\begin{array}{ccc} \text{Env } \alpha & \xrightarrow{\quad \text{Semf (fx) } \alpha \quad} & \text{Mng (Type (fx) } \alpha) \\ \Big\downarrow \text{Semf}^* \text{ f } \alpha & & \Big\| \\ \text{Env (Type}^* \text{f } \alpha) & \xrightarrow{\quad \text{Semf } [\![x]\!] \text{ (Type}^* \text{ f } \alpha) \quad} & \text{Mng (Type (fx) } \alpha) \end{array}$$

By (Semf2), the bottom arrow is the projection on the xth component. This diagram commutes by the definition of Semf* f.

(3) Let $\ell = [\![mn]\!]$ where $m, n \in L$. First consider the case in which

$$\text{Type } m \text{ (Type}^* \text{ f } \alpha) = \tau' \Rightarrow \theta' \text{ and Type } n \text{ (Type}^* \text{ f } \alpha) = \overline{\tau}' \leq \tau'$$

Then

$$\text{Type } [\![mn]\!] \text{ (Type}^* \text{ f } \alpha) = \theta' \ .$$

By Theorem 3.12 and Theorem 2.2,

$$\text{Type (Sub f } m) \ \alpha = \tau \Rightarrow \theta \leq \tau' \Rightarrow \theta' \ ,$$

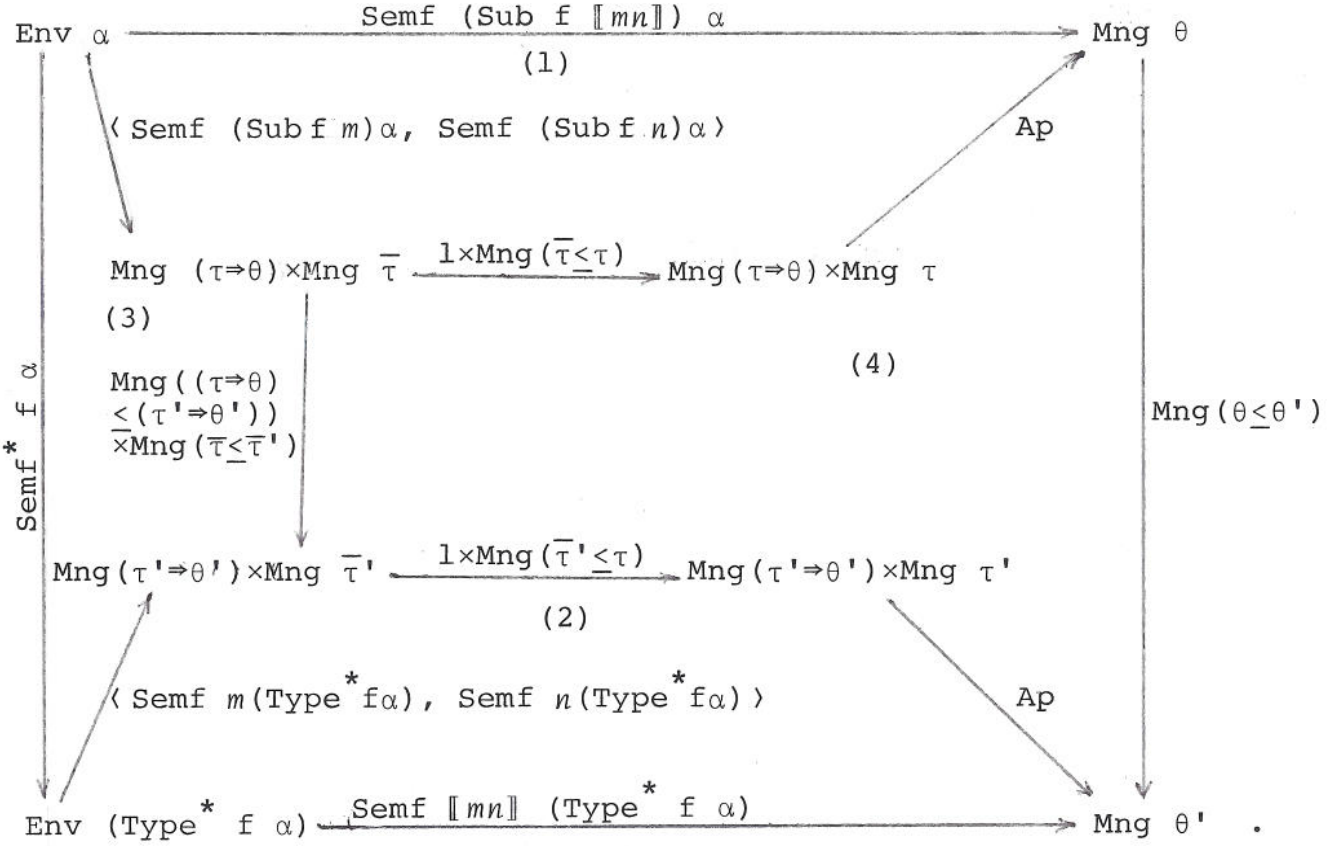
$$\text{Type (Sub f } n) \ \alpha = \overline{\tau} \leq \overline{\tau}'$$

Since Sub f $[\![mn]\!]$ = $[\![$(Sub f m) (Sub f n)$]\!]$,

$\overline{\tau} \leq \overline{\tau}' \leq \tau' \leq \tau$, and $\theta \leq \theta'$, we see

$$\text{Type (Sub f } [\![mn]\!])\alpha = \theta \leq \theta' \quad .$$

The diagram which we wish to show commutative may now be written as



Interior diagrams (1) and (2) commute by (Semf3), diagram (3) commutes by the induction hypothesis, and commutativity

of diagram (4) follows by the argument used to show F_{ap} is natural in the proof of Theorem 4.3. So the outer square commutes, as desired. The other case is that in which

$$\text{Type } [\![mn]\!] \ (\text{Type}^* \ f \ \alpha) = \underline{ns}$$

The relevant diagram commutes in this case because the object in its lower right corner is terminal.

(4) Let $\ell = [\![\lambda x{:}\tau.m]\!]$ where $x \in Id$, $\tau \in T$, $m \in L$. Let $X = \text{clashset} \langle f, x, m \rangle$ and $z = \text{newid } X$.

First we consider the case $x \notin X$. In this case

$$\text{Sub } f \ [\![\lambda x{:}\tau.m]\!] = [\![\lambda x{:}\tau. \ (\text{Sub } [f \mid x{:}[\![x]\!]] m)]\!] \ ,$$

and

$$\text{Type } (\text{Sub } f \ [\![\lambda x{:}\tau.m]\!])\alpha = \tau \Rightarrow \theta,$$

$$\text{where } \theta = \text{Type } (\text{Sub } [f \mid x{:}[\![x]\!]] m) \ [\alpha \mid x{:}\tau] .$$

Also,

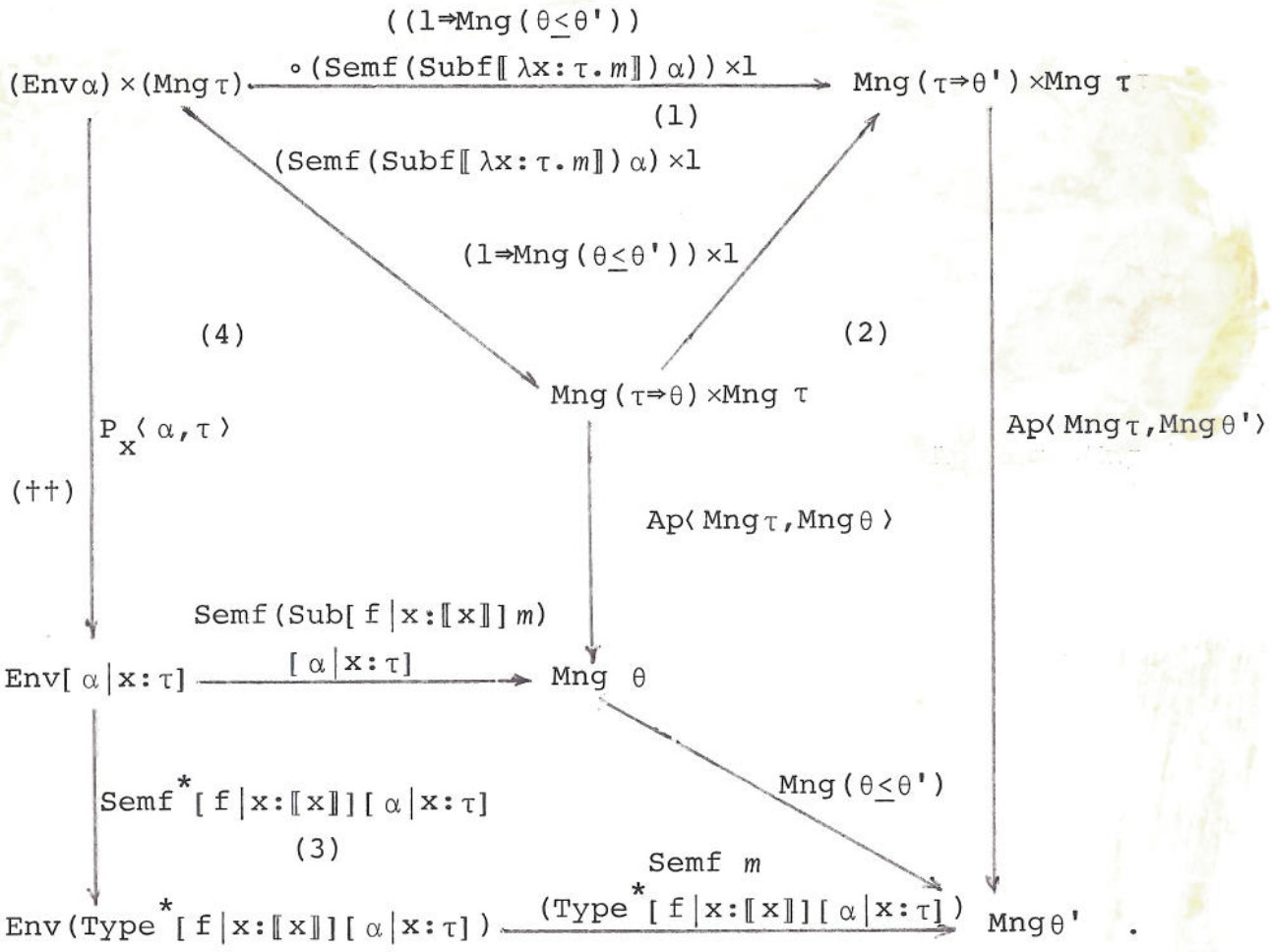
$$\text{Type } [\![\lambda x{:}\tau.m]\!] (\text{Type}^* \ f \ \alpha) = \tau \Rightarrow \theta' \ ,$$

$$\text{where } \theta' = \text{Type } m \ [\text{Type}^* \ f \ \alpha \mid x{:}\tau]$$

By Theorem 3.12, we know $(\tau \Rightarrow \theta) \leq (\tau \Rightarrow \theta')$; hence, $\theta \leq \theta'$. Our aim is to show that

$$\begin{array}{ccc} \text{Env } \alpha & \xrightarrow{\text{Semf } (\text{Sub } f \ [\![\lambda x{:}\tau.m]\!])\alpha} & \text{Mng } (\tau \Rightarrow \theta) \\ \Big\downarrow (\dagger) \ \text{Semf}^* \ f \ \alpha & & \Big\downarrow 1 \Rightarrow \text{Mng}(\theta \leq \theta') \\ \text{Env } (\text{Type}^* \ f \ \alpha) & \xrightarrow{\text{Semf}[\![\lambda x{:}\tau.m]\!] (\text{Type}^* f \alpha)} & \text{Mng } (\tau \Rightarrow \theta') \end{array}$$

commutes in $|K|$. Our method is to show both composites satisfy the same universal mapping property, namely, (Cart3). Consider the diagram



The interior diagrams commute for the following reasons:

(1) × is a functor,

(2) naturality of Ap in its second argument,

(3) induction hypothesis,

(4) (Semf4).

Thus, the outer square commutes.

Let

$$\beta_1 = [\,\mathrm{Type}^*\ f\ \alpha \mid x{:}\tau]$$

and

$$\beta_2 = \mathrm{Type}^*\ [\,f \mid x{:}[\![x]\!]]\ [\,\alpha \mid x{:}\tau]$$

We claim that β_1 and β_2 agree on Free m. If $x \in$ Free m, then

$$\beta_1\ x = \tau = \mathrm{Type}\ [\![x]\!]\ [\,\alpha \mid x{:}\tau] = \beta_2\ x\,.$$

If $y \in$ Free m and $y \neq x$, then

$$\begin{aligned}\beta_1\ y &= \mathrm{Type}\ (fy)\ \alpha \\ &= \mathrm{Type}\ (fy)\ [\,\alpha \mid x{:}\tau]\ , \\ &\qquad \text{by observing } x \notin X \ \text{ (hence,} \\ &\qquad x \notin \text{Free } (fy)) \text{ and applying Cor. 3.10,} \\ &= \beta_2\ y\,.\end{aligned}$$

Define the type assignment

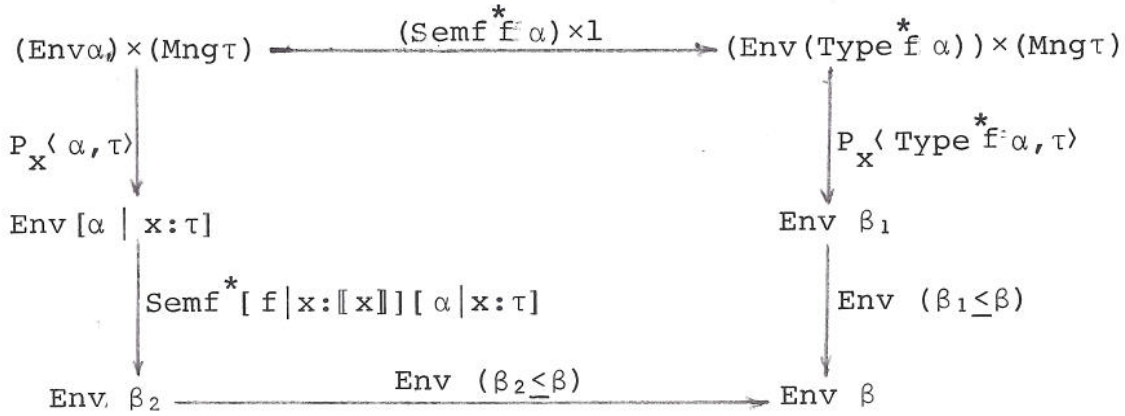
$$\beta \in ((\mathrm{dom}\ f) \cup \{x\}) \cap (\text{Free } m) \to \mathrm{Ob}\ |\mathcal{T}|$$

by

$$\beta\ y = \beta_1\ y = \beta_2\ y\ , \qquad y \in \mathrm{dom}\ \beta\ .$$

Clearly $\beta_1 \leq_A \beta$ and $\beta_2 \leq_A \beta$.

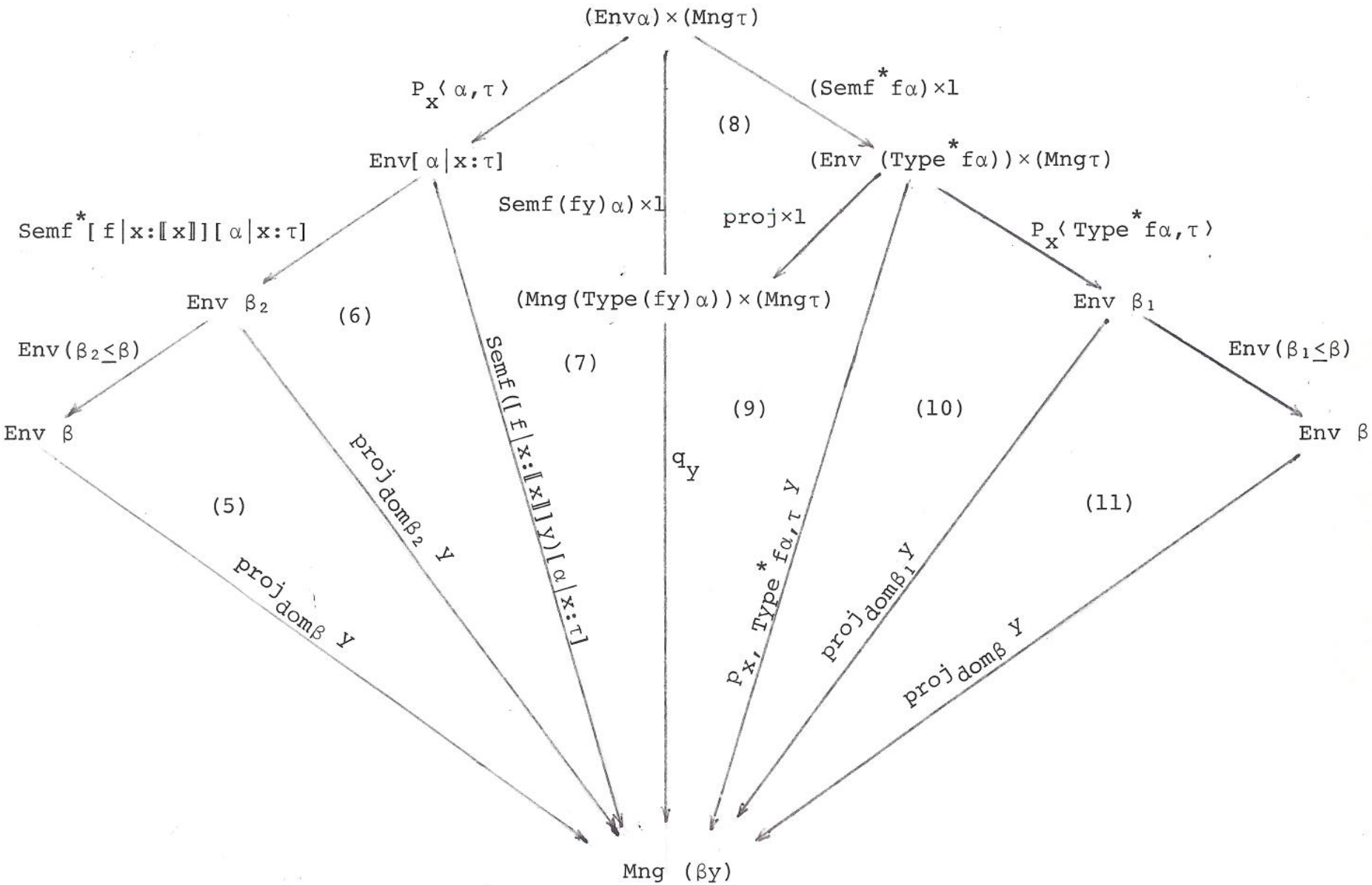
We claim that the diagram



commutes. The object in the lower right-hand corner is a product:

$$\text{Env } \beta = \Pi_{\text{dom } \beta}^{|K|} \ (\text{Mng} \circ \beta) \ .$$

Thus, the claim will be proved if we can verify for all $y \in \text{dom } \beta$ the commutativity of the outer square of the diagram


(Envα)×(Mngτ)
P_x⟨α,τ⟩
(Semf*fα)×1
(8)
Env[α|x:τ]
(Env (Type*fα))×(Mngτ)
Semf*[f|x:[[x]]][α|x:τ]
Semf(fy)α)×1
proj×1
P_x⟨Type*fα,τ⟩
Env β2
(6)
(Mng(Type(fy)α))×(Mngτ)
Env β1
Env(β2≤β)
Semf([f|x:[[x]]y)[α|x:τ]
(7)
Env(β1≤β)
Env β
(9)
(10)
Env β
q_y
(5)
(11)
proj dom β2 y
proj dom β y
p_x, Type*fα,τ y
proj dom β1 y
proj dom β y
Mng (βy)

In this last diagram the arrow q_y is the projection onto the first component if $y \neq x$ and the projection onto the second component if $y = x$. Consider each interior diagram in turn; they all commute for reasons we will now explain:

(5) definition of the functor Env,

(6) definition of Semf^*,

(7) when $y \neq x$, $x \notin \text{Free}\ (fy)$ follows from $x \notin X$, and therefore commutativity follows from Proposition 3.7; when $y = x$, commutativity is a consequence of the fact that $\text{Semf}\ [\![x]\!]\ [\alpha \mid x{:}\tau]$ is the projection onto the xth component of $\text{Env}[\alpha \mid x{:}\tau]$,

(8) definition of Semf^*,

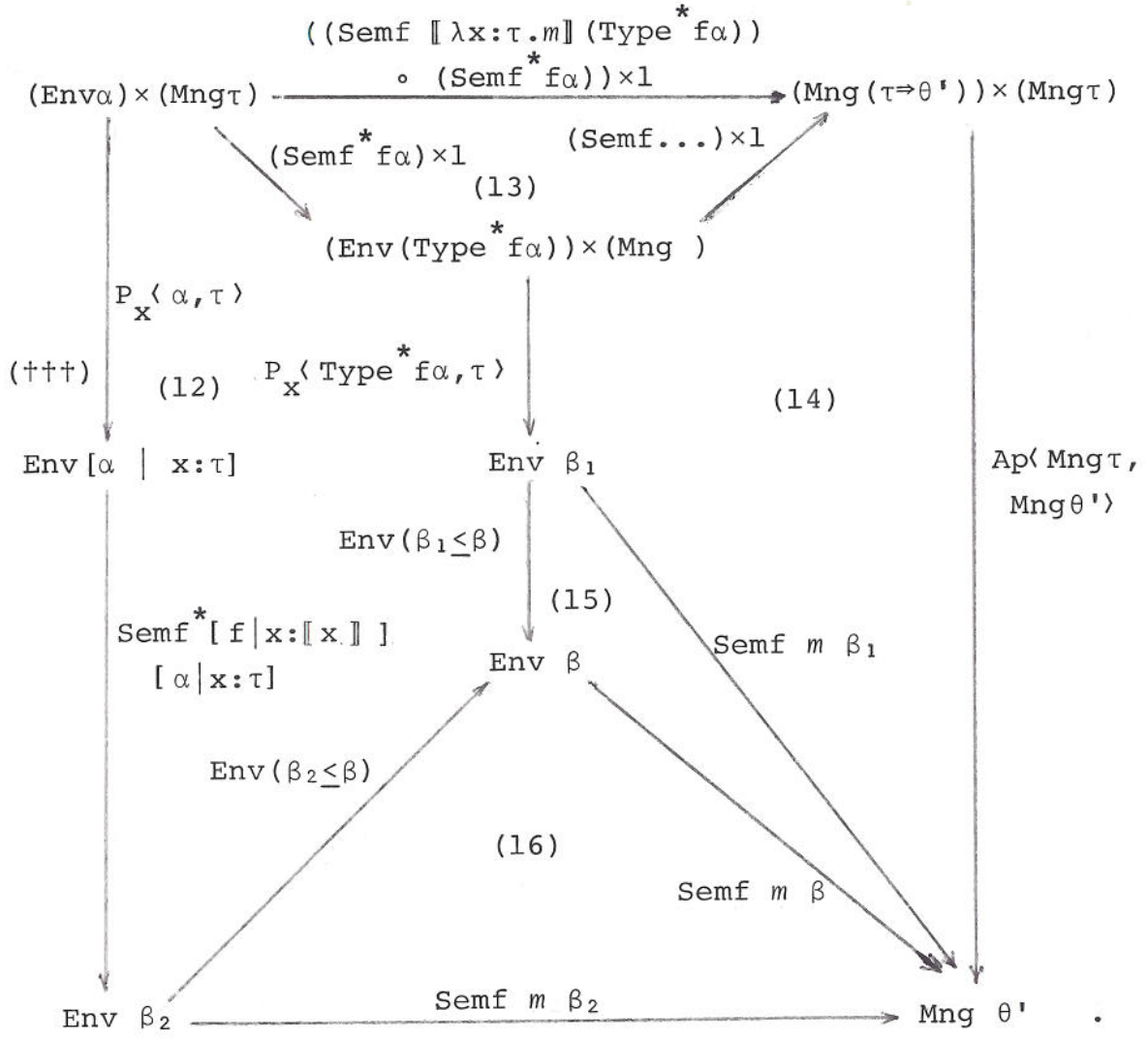
(9) definition of $p_{x,\ \text{Type}^* f\ \alpha,\tau\ Y}$,

(10) definition of $P_x \langle \text{Type}^*\ f\ \alpha,\ \tau \rangle$,

(11) definition of the functor Env.

Thus, the entire diagram commutes, proving the claim.

Next consider the diagram



Commutativity of interior diagram (12) has been demonstrated. Diagram (13) commutes because × is a functor. Diagram (14) commutes by (Semf4). The diagrams labelled (15) and (16) are commutative since Semf *m* is a natural transformation. Thus the outer square commutes.

Compare the outer squares of diagrams (††) and (†††). By applying the uniqueness assertion of (Cart3) we get commutativity of (†).

Finally, consider the case $x \in X$. In the case

$$\text{Sub } f \llbracket \lambda x{:}\tau.m \rrbracket = \llbracket \lambda z{:} \,.\, (\text{Sub } [f \mid x{:}\llbracket z \rrbracket]\ m) \rrbracket \ ,$$

and this time

$$\text{Type } (\text{Sub } f \llbracket \lambda x{:}\tau.m \rrbracket) = \tau \Rightarrow \theta \ ,$$

where $\theta = \text{Type } (\text{Sub } [f \mid x{:}\llbracket z \rrbracket]\ m)\ [\alpha \mid z{:}\tau]$.

Again,

$$\text{Type } \llbracket \lambda x{:}\tau.m \rrbracket\ (\text{Type}^*\ f\ \alpha) = \tau \Rightarrow \theta' \ ,$$

where $\theta' = \text{Type } m\ [\text{Type}^*\ f\ \alpha \mid x{:}\tau]$. By slightly altering the argument in the preceding case, we again obtain commutativity of (†). □

The next three theorems study in turn the semantics of α-reduction, β-reduction, and η-reduction.

Theorem 4.10: Let $x, y \in \text{Id}$, $\ell \in L$, $\tau \in T$. Suppose $y \notin \text{Free } \ell$ and $m = \text{Sub } [\text{emp}_L \mid x{:}\llbracket y \rrbracket]\ \ell$. Then

$$\text{Semf } \llbracket \lambda x{:}\tau.\ell \rrbracket = \text{Semf } \llbracket \lambda y{:}\tau.m \rrbracket \ .$$

Proof: Let $\alpha \in A$. By $\hat{\alpha} \in A$ we mean the restriction of α to dom α - {y}. Thus $\hat{\alpha} \in \text{dom } \alpha - \{y\} \to \text{Ob } |\tau|$. Extend the phrase assignment $[\text{emp}_L \mid x{:}\llbracket y \rrbracket]$ to

$$f \in (\text{dom } \hat{\alpha}) \quad \{x\} \to |L|$$

via

$$f\ z \qquad \begin{cases} \llbracket z \rrbracket & \text{if} \quad z \neq x \\ \llbracket y \rrbracket & \text{if} \quad z = x. \end{cases}$$

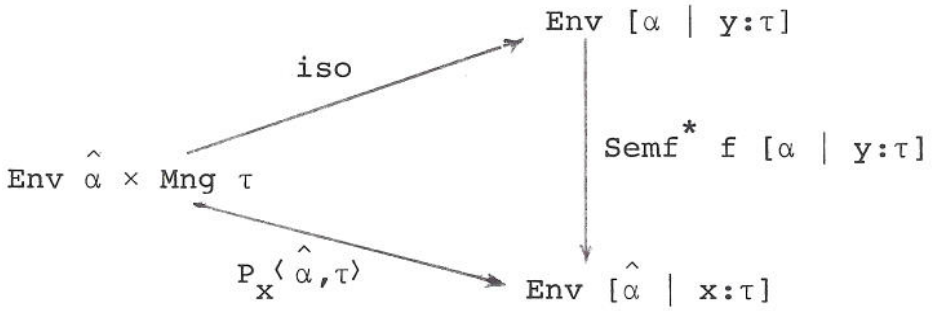
By Theorem 3.3, Sub f = Sub [emp_L | x:⟦y⟧], so that m = Sub f ℓ. It is easy to see that

$$\text{Type}^* \; f \; [\alpha \mid y{:}\tau] = [\hat{\alpha} \mid x{:}\tau].$$

Therefore

$$\text{Semf}^* \; f \; [\alpha \mid y{:}\tau] \in \text{Env} \; [\alpha \mid y{:}\tau] \xrightarrow[|K|]{} \text{Env} \; [\hat{\alpha} \mid x{:}\tau].$$

Using the universal mapping property that defines $P_x\langle \alpha,\tau\rangle$, it is an easy exercise to prove that



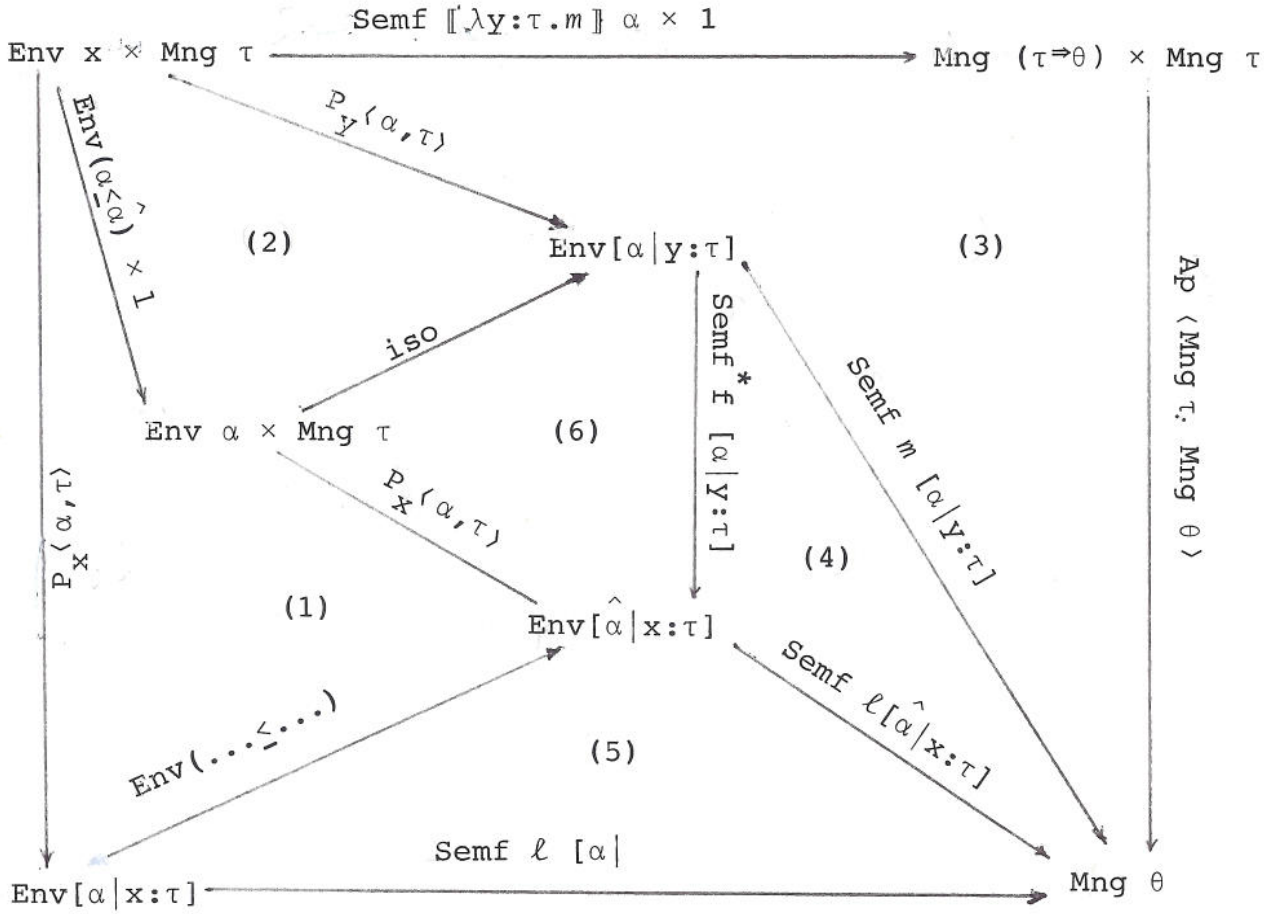
commutes, where iso is the obvious isomorphism. By Proposition 3.13, we may let

$$\theta = \text{Type} \; \ell \; [\alpha \mid x{:}\tau] = \text{Type} \; m \; [\alpha \mid y{:}\tau];$$

then also θ = Type ℓ [$\hat{\alpha}$ | x:τ] (by Corollary 3.11) and

$$\tau \Rightarrow \theta = \text{Type} \; [\![\lambda x{:}\tau.\ell]\!] \quad \alpha = \text{Type} \; [\![\lambda y{:}\tau.m]\!] \; \alpha.$$

Consider the diagram



The interior diagrams commute for the following reasons:

(1) naturality of P_x,

(2) definition of $P_y\langle \alpha\ \tau \rangle$

(3) (Semf 4),

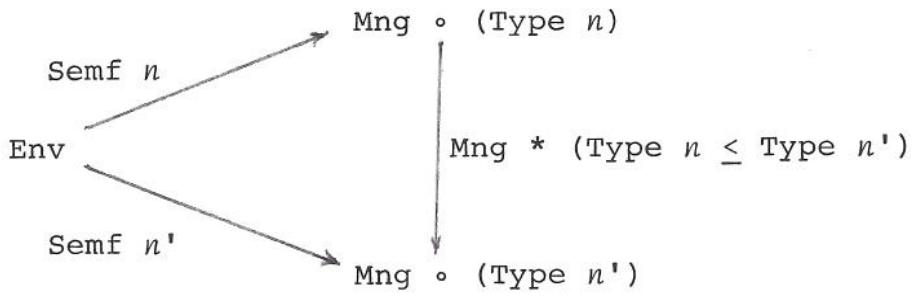
(4) Theorem 4.9,

(5) naturality of Semf ℓ,

(6) by earlier remarks.

Hence the outer square commutes. Apply (Semf 4) again to get Semf $[\![\lambda x{:}\tau.\ell]\!]\ \alpha$ = Semf $[\![\lambda y{:}\tau.m]\!]\ \alpha$. □

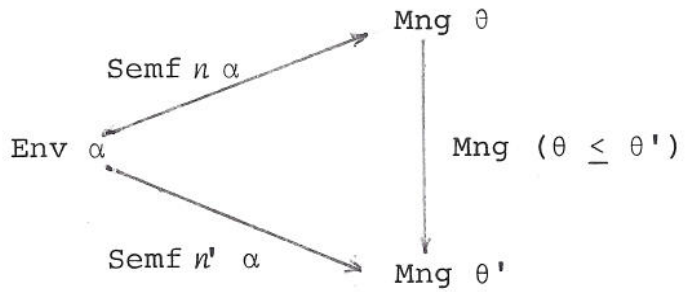
Theorem 4.11: Let $x \in Id$, $\tau \in T$, $\ell, m \in L$. Suppose $n = Sub\,[emp_L \mid x{:}m]\,\ell$ and $n' = [\![(\lambda x{:}\tau.\ell)m]\!]$. Then



commutes in $A \Rightarrow |K|$.

(Remark: Thus, if Type n = Type n', it follows that Semf n = Semf n'.)

Proof: Note that Type $n \leq$ Type n' by Proposition 3.14. Let $\alpha \in A$, θ = Type $n\ \alpha$, θ' = Type $n'\ \alpha$. We must verify that



commutes. If θ' = ns, then this diagram obviously commutes. So suppose that $n' \neq$ ns. Upon recalling that $n' = [\![(\lambda x{:}\tau.\ell)m]\!]$, we see this can only happen if

$$\text{Type } m\ \alpha = \overline{\tau}\ , \quad \text{where} \quad \overline{\tau} \leq \tau\ .$$

As in the proof of Proposition 3.14, the phrase assignment

$$f_\alpha \in (\text{Free } \ell) \cup (\text{dom } \alpha) \cup \{x\} \to |L|$$

is given by

$$f_\alpha\, y = \begin{cases} [\![y]\!], & \text{if } y \neq x \\ m, & \text{if } y = x \end{cases} .$$

Recall that

$$\text{Type } n = (\text{Type } \ell) \circ (\text{Type}^* f_\alpha),$$

and

$$\text{Type}^* f_\alpha\, \alpha \leq [\alpha \mid x{:}\overline{\tau}\,] .$$

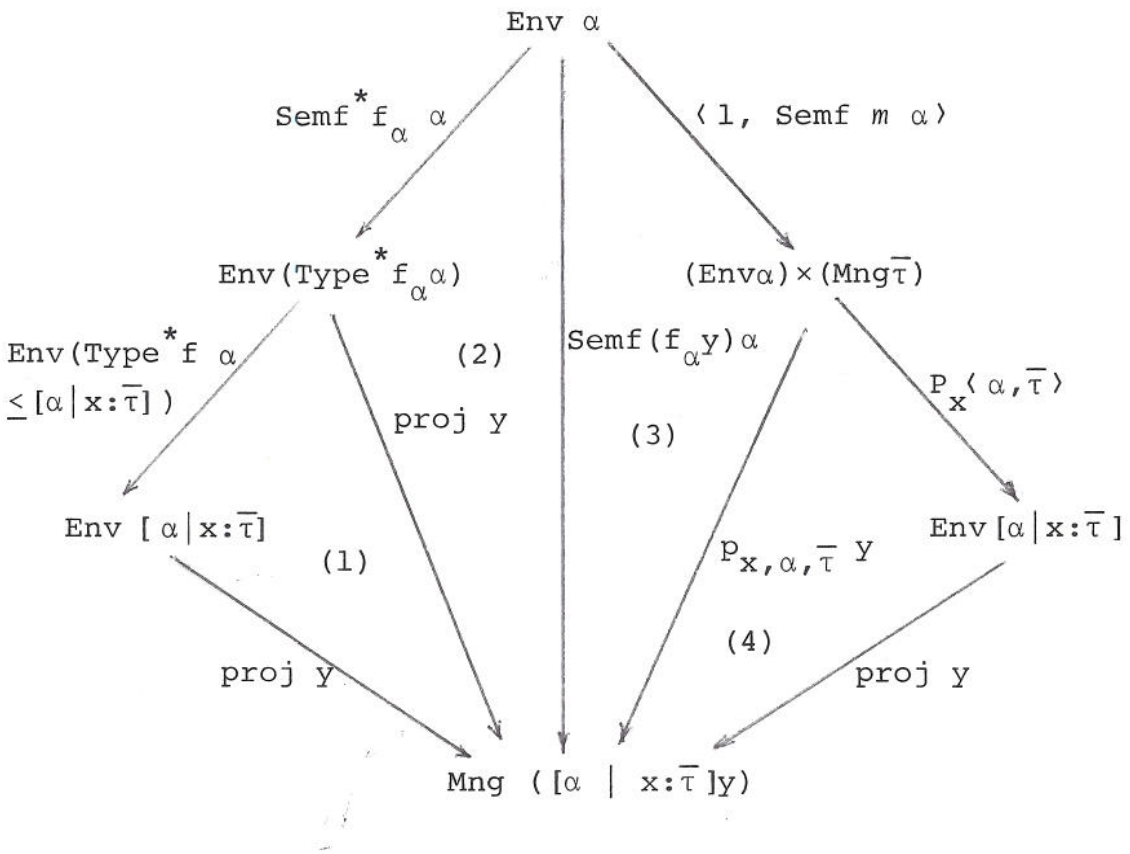
This implies that

$$\theta = \text{Type } n\, \alpha = \text{Type } \ell\, (\text{Type}^* f_\alpha\, \alpha) ,$$

so that

$$\text{Semf } \ell\, (\text{Type}^* f_\alpha\, \alpha) \in \text{Env } (\text{Type}^* f_\alpha\, \alpha) \xrightarrow{|K|} \text{Mng } \theta .$$

For each $y \in \text{dom } [\alpha \mid x{:}\overline{\tau}]$ consider the diagram



The interior diagrams commute for the following reasons:

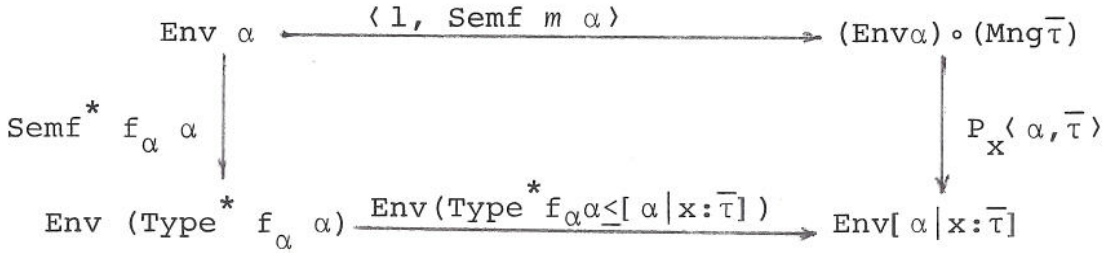
(1) definition of Env,

(2) definition of Semf*

(3) definition of $p_{x,\alpha,\bar{\tau}}$, together with observation that Semf $(f_\alpha y)\ \alpha$ = proj y when $y \neq x$, and Semf $(f_\alpha y)\ \alpha$ = Semf $m\ \alpha$ when $y = x$,

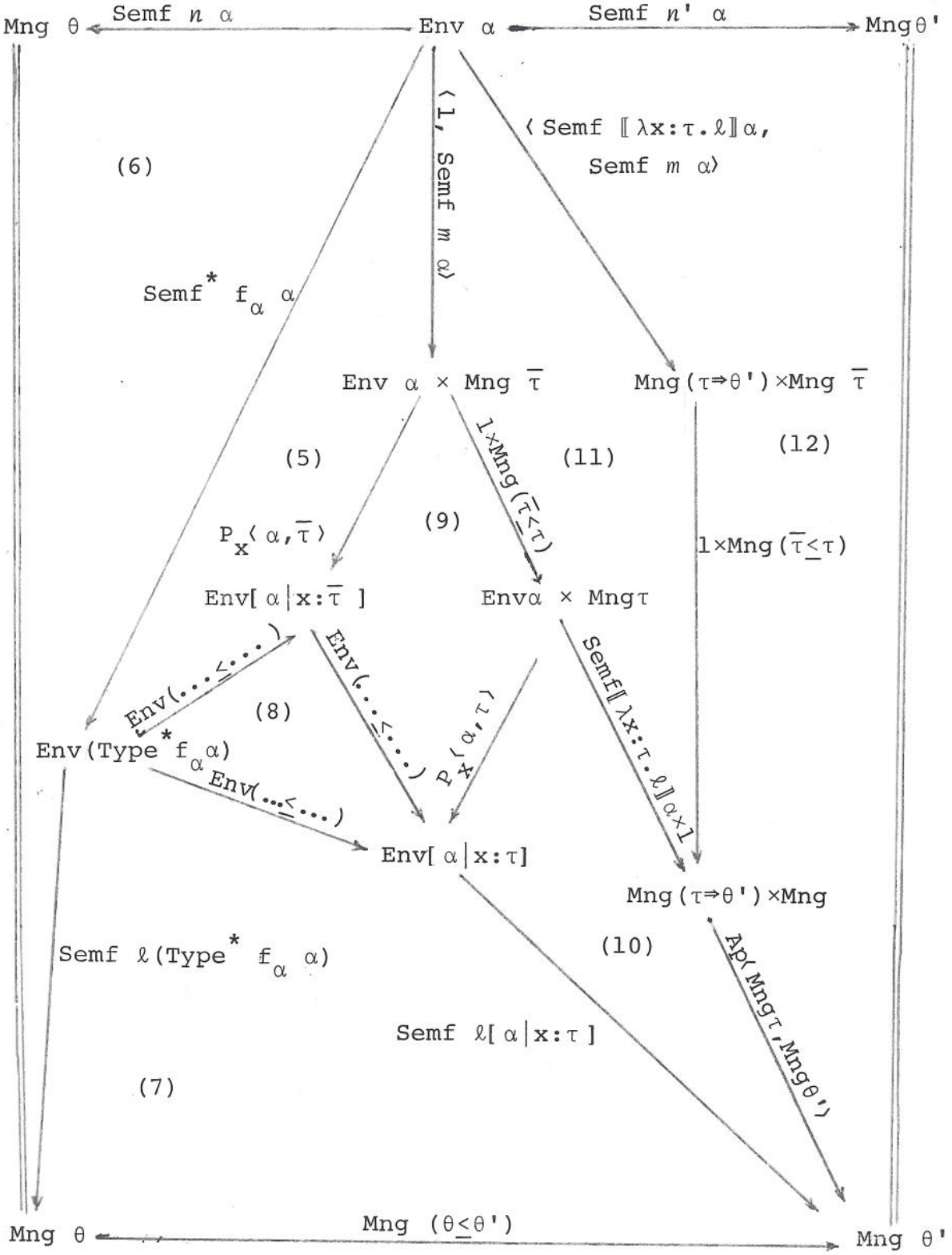
(4) definition of $P_x\langle \alpha, \bar{\tau} \rangle$.

By the universal mapping property for products, we see that



commutes.

Finally, consider the diagram



We claim this diagram commutes. This follows from the commutativity of each of the numbered subdiagrams, which commute for the following reasons:

(5) commutativity demonstrated earlier in this proof,

(6) Theorem 4.9 ,

(7) naturality of Semf ℓ,

(8) functoriality of Env,

(9) naturality of P_x,

(10) (Semf4),

(11) clearly commutes,

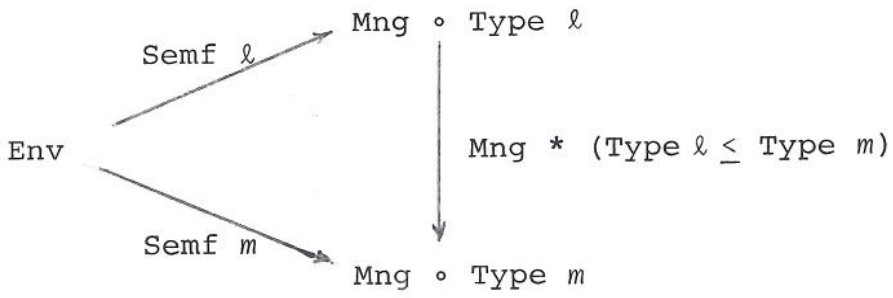
(12) (Semf3).

□

Theorem 4.12: Let $x \in \mathrm{Id}$, $\tau \in T$, and $\ell \in L$. Suppose $m = [\![\lambda x{:}\tau.[\![\ell[\![x]\!]]\!]]\!]$, $x \notin \mathrm{Free}\ \ell$, and

$$\mathrm{Type}\ \ell \leq_{A \Rightarrow |T|} \mathrm{Type}\ m \quad .$$

Then



commutes in $A \Rightarrow |K|$.

(Remark: As in Theorem 4.11, we conclude that Semf ℓ = Semf m if Type ℓ = Type m.)

Proof: Let $\alpha \in A$. We can write

$$\text{Type } m\ \alpha = \tau \Rightarrow \theta \ ,$$

where $\theta = \text{Type } [\![\ell \ [\![x]\!]]\!] \ [\alpha \mid x{:}\tau \ ]$. Since

$\text{Type } \ell\ \alpha \leq \text{Type } m\ \alpha$, we can write

$$\text{Type } \ell\ \alpha = \tau' \Rightarrow \theta' \ ,$$

where $\tau \leq \tau'$ and $\theta' \leq \theta$. Since $x \notin \text{Free } \ell$, Corollary 3.11 tells us

$$\text{Type } \ell\ [\alpha \mid x{:}\tau] = \tau' \Rightarrow \theta' \ .$$

Combining this with $\tau \leq \tau'$ and

$$\text{Type } [\![x]\!]\ [\alpha \mid x{:}\tau] = [\alpha \mid x{:}\tau]\ x = \tau,$$

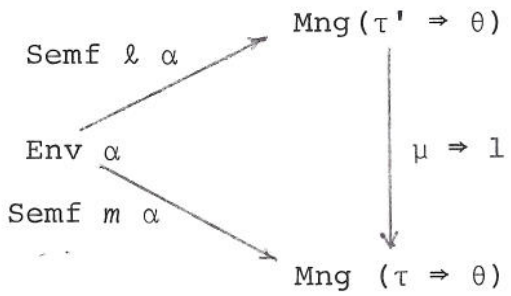
we see

$$\theta = \text{Type } [\![\ell\ [\![x]\!]]\!]\ [\alpha \mid x{:}\tau] = \theta' \ .$$

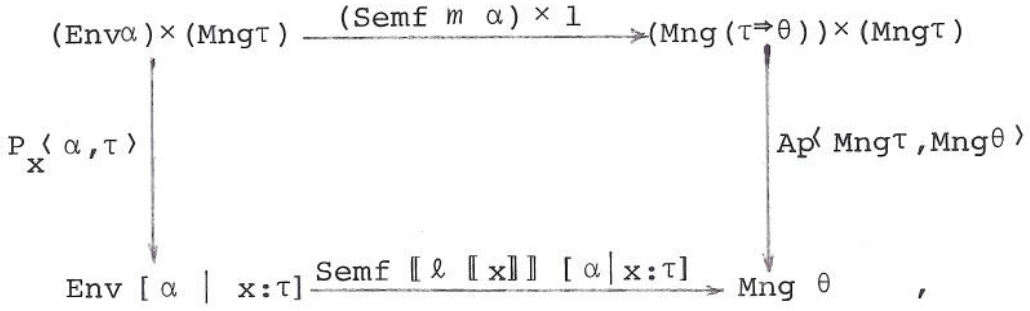
Let

$$\mu = \text{Mng } (\tau \leq \tau') \in \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \tau' \ .$$

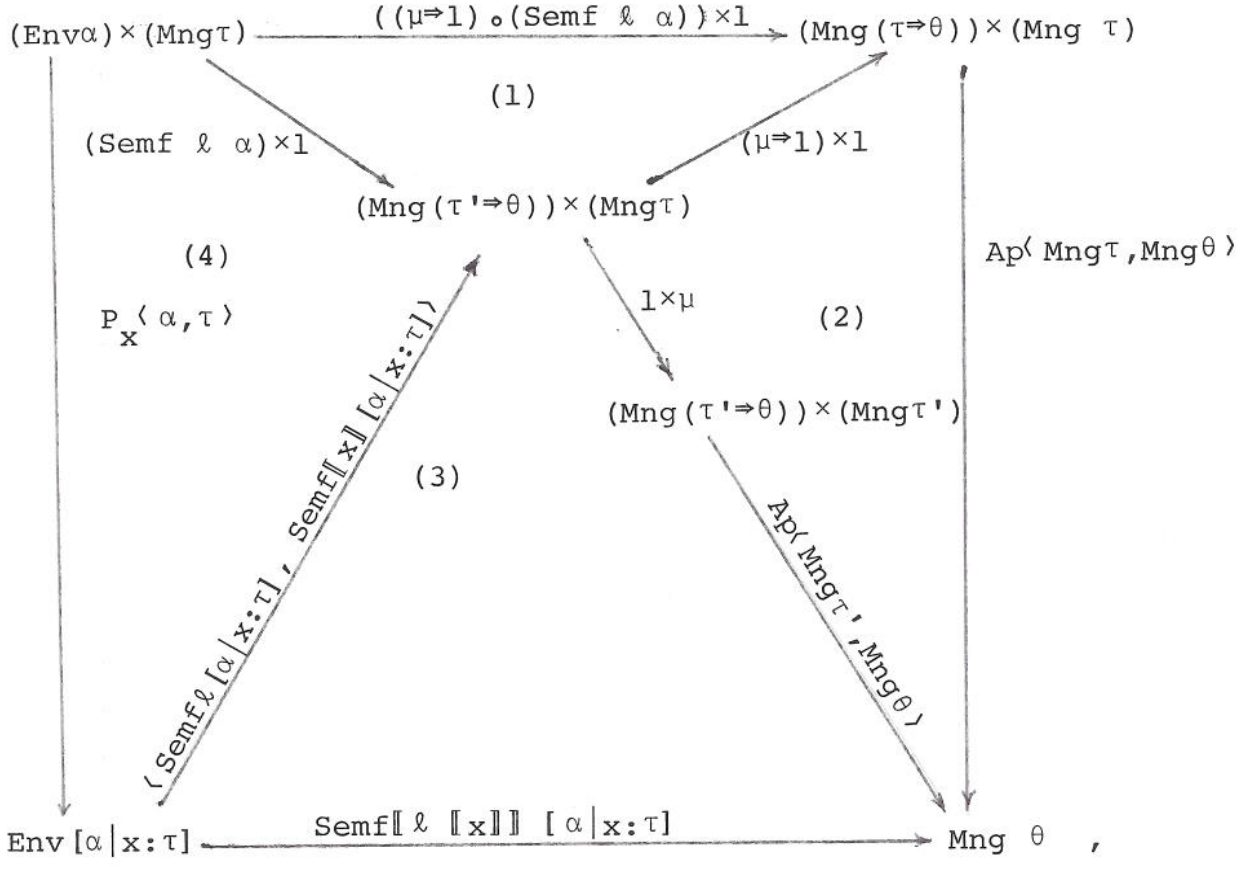
We must verify that



commutes in $|K|$. We know that Semf m α is the unique arrow in $|K|$ such that



commutes. Thus, if we can verify the commutativity of



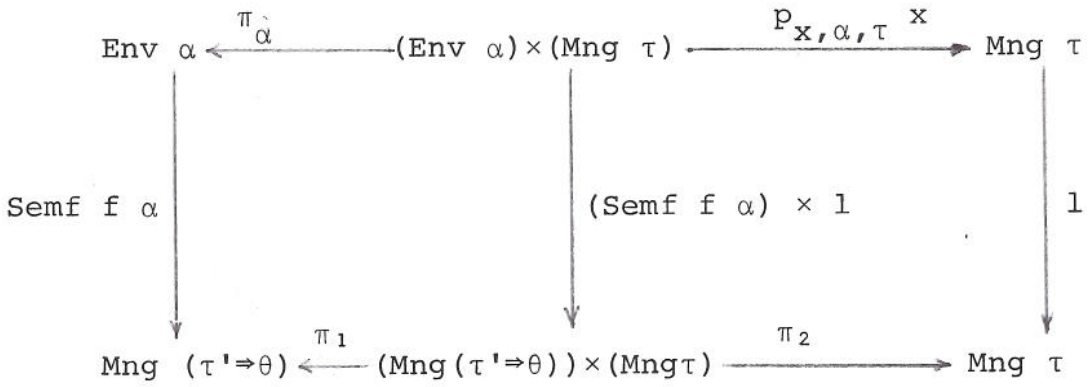
then the commutativity of the triangle in question follows. The interior diagrams (1), (2), and (3) commute for the following reasons:

(1) $- \times 1$ is a functor,

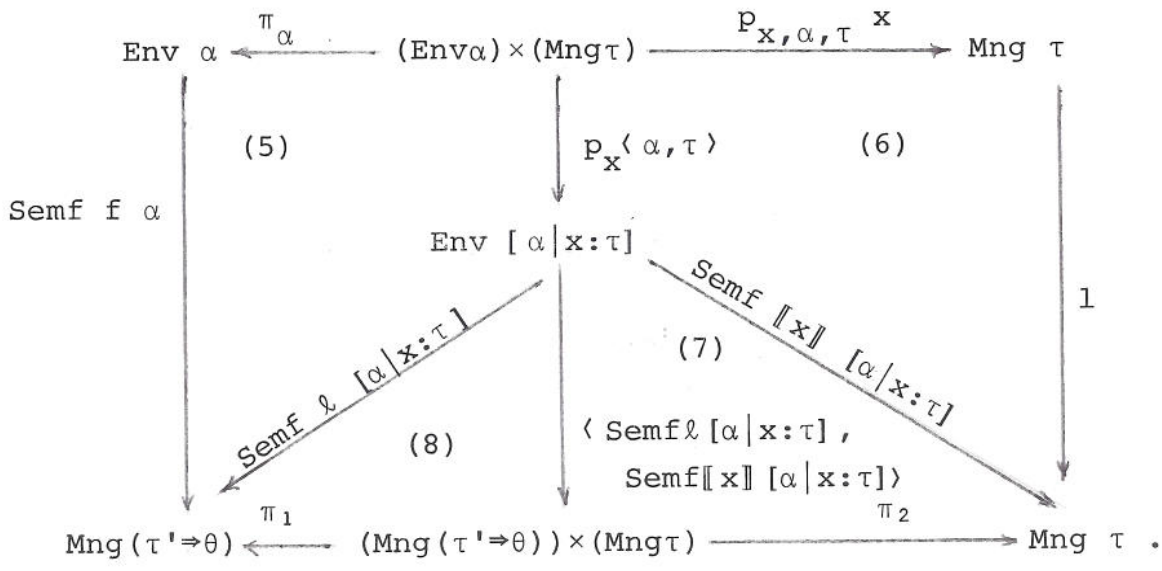
(2) the definition of $\mu \Rightarrow 1$ (see Chapter I),

(3) (Semf3).

The proof that (4) commutes involves a short argument. The demonstration is based on the fact that (Semf f α) $\times$ 1 is the unique morphism such that



commutes where π_1 and π_2 are the indicated projections. Consider the diagram



Interior diagram (5) commutes by Theorem 4.7 and interior diagram (6) commutes by the definition of $P_x\langle \alpha,\tau\rangle$ along with the observation that Semf $[\![x]\!]$ $[\alpha \mid x{:}\tau]$ is the projection on the xth component. Finally, (7) and (8) commute by the universal mapping property defining $\langle$ Semf ℓ $[\alpha \mid x{:}\tau]$, Semf $[\![x]\!]$ $[\alpha \mid x{:}\tau]\rangle$. Hence the entire diagram commutes, and commutativity of (4) is now immediate. □

CHAPTER V

CARTESIAN CLOSED CATEGORIES

In order to investigate the semantics of ALGOL-like languages it is necessary to specify a Cartesian closed category, among whose arrows are found the denotations of program fragments. In this chapter we will establish that certain semantically important categories are Cartesian closed. Because of the motivation given in Chapter I, we are particularly interested in categories whose objects are functors.

We begin by establishing some notational conventions. If f is a function and X is a set of arguments to which f may be applied, then we write

$$fX = \{fx \mid x \in X\}.$$

Similarly, if F is a set of functions and x is an argument in the domain of each function in F, then we write

$$Fx = \{fx \mid f \in F\}.$$

We may even write

$$FX = \{fx \mid f \in F, x \in X\}.$$

A poset is a pair consisting of an underlying set and a partial order. Routinely, a poset is viewed as a category whose collection of objects is the underlying set. Thus, whenever P is a poset, its underlying set may be denoted as Ob P.

We remarked in Chapter I that $\mathcal{Set}$ is a Cartesian closed category. The point of the next two theorems is that the

structure of *Set* as a Cartesian closed category can be extended to *Pdom*, and that the result of that extension can be restricted to *Dom*, so that both *Pdom* and *Dom* are Cartesian closed categories.

As a prelude to what follows note that both *Pdom* and *Dom* are categories with finite products specifically given; the products are constructed by partially ordering componentwise the products in *Set* of the underlying sets.

Theorem 5.1: The category *Pdom* is Cartesian closed.

Proof: The distinguished terminal object and the distinguished binary product functor for *Pdom* are constructed in the obvious way from their counterparts in *Set*. In particular, $Pdom_{term}$ is E Set_{term}, a singleton set regarded as a predomain.

We now turn to exponentiation. Suppose Q and R are predomains; we will define the predomain $Q \overset{\Longrightarrow}{Pdom} R = Q \Rightarrow R$. The underlying set of $Q \Rightarrow R$ is

$$Ob\ (Q \Rightarrow R) = Q \overset{\longrightarrow}{Pdom} R,$$

the set of continuous functions from Q to R. The partial order for $Q \Rightarrow R$ is defined by $f \leq_{Q \Rightarrow R} f'$ if $fq \leq_R f'q$ for all $q \in Ob\ Q$. We must show that $Q \Rightarrow R$ is directed-complete. To this end, let D be a directed subset of $Q \Rightarrow R$. Define the function

$$g_D \in Ob\ Q \longrightarrow Ob\ R$$

by

$$g_D\ q = \sqcup_R (Dq) \text{ for all } q \in Ob\ Q.$$

We claim g_D is continuous. Let X be a directed subset of Q. Continuity of g_D follows from the computation

$$g_D\ (\sqcup_Q X) = \sqcup_R\ (D\ (\sqcup_Q X))$$
$$= \sqcup_R\ (DX), \text{ since every function in D is continuous,}$$
$$= \sqcup_R\ \{fx \mid f \in D, x \in X\}$$
$$= \sqcup_R\ \{\sqcup_R\ (Dx) \mid x \in X\}$$
$$= \sqcup_R\ \{g_D x \mid x \in X\}$$
$$= \sqcup_R\ (g_D X).$$

Thus

$$g_D \in Q \xrightarrow[Pdom]{} R.$$

Next we claim that g_D is the least upper bound of D. Surely, if $f \in D$ then, for all $q \in \text{Ob } Q$,

$$fq \leq_R \sqcup_R\ (Dq) = g_D q\ ;$$

so $f \leq_{Q \Rightarrow R} g_D$ and g_D is an upper bound for D. If h is an arbitrary upper bound for D, then $f \leq_{Q \Rightarrow R} h$ for $f \in D$, which implies that for all $q \in \text{Ob } Q$

$$g_D q = \sqcup_R\ (Dq) \leq \sqcup_R\ hq.$$

So g_D is indeed the least upper bound of D. This proves $Q \Rightarrow R$ is a predomain.

Application in *Pdom* is the usual functional application. In more detail, suppose Q and R are predomains, and define

$$\text{Ap}\ \langle Q, R\rangle = \text{Ap} \in (Q \Rightarrow R) \times Q \xrightarrow[Pdom]{} R$$

by

$$\text{Ap}\ \langle f, q\rangle = fq.$$

It is necessary to verify that Ap is continuous. Suppose D is a directed subset of $(Q \Rightarrow R) \times Q$. Let

$$D_1 = \{f \mid \langle f, x\rangle \in D \text{ for some } x \in \text{Ob } Q\},$$
$$D_2 = \{x \mid \langle f, x\rangle \in D \text{ for some } f \in \text{Ob } (Q \Rightarrow R)\}.$$

Then D_1 is a directed subset of $Q \Rightarrow R$, D_2 is a directed subset of Q,

$$\sqcup_{(Q \Rightarrow R) \times Q} D = \langle \sqcup_{Q \Rightarrow R} D_1, \sqcup_Q D_2 \rangle ,$$

and

$$\sqcup_R (\mathrm{Ap}\ D) = \sqcup_R (D_1 D_2)$$

The computation

$$\begin{aligned} \mathrm{Ap}\ (\sqcup_{(Q \Rightarrow R) \times Q} D) &= \mathrm{Ap} \langle \sqcup_{Q \Rightarrow R} D_1, \sqcup_Q D_2 \rangle \\ &= \sqcup_{Q \Rightarrow R} D_1\ (\sqcup_Q D_2) \\ &= \sqcup_R (D_1 D_2), \\ &\quad \text{since each function in } D_1 \text{ is continuous} \\ &= \sqcup_R (\mathrm{Ap}\ D) \end{aligned}$$

shows that Ap is continuous.

Next, for predomains P, Q, and R consider the function

$$\mathrm{Ab}\ \langle P,Q,R \rangle = \mathrm{Ab} \in (P \times Q \xrightarrow[Pdom]{} R) \to (\mathrm{Ob}\ P \to (\mathrm{Ob}\ Q \to \mathrm{Ob}\ R))$$

defined by

$$\mathrm{Ab}\ f\ p\ q = f \langle p, q \rangle$$

where $f \in P \times Q \xrightarrow[Pdom]{} R$, $p \in \mathrm{Ob}\ P$, $q \in \mathrm{Ob}\ Q$.

We claim that

$$\mathrm{Ab} \in (P \times Q \xrightarrow[Pdom]{} R) \to (P \xrightarrow[Pdom]{} (Q \Rightarrow R))$$

First we will show Ab f p is continuous, where $f \in P \times Q \xrightarrow[Pdom]{} R$ and $p \in \mathrm{Ob}\ P$. Let D be a directed subset of Q. Then $\{p\} \times D$ is a directed subset of $P \times Q$ and

$$\sqcup_{P \times Q} (\{p\} \times D) = \langle p, \sqcup_Q D \rangle .$$

Compute:

$$\begin{aligned} \mathrm{Ab}\ f\ p\ (\sqcup_Q D) &= f \langle p, \sqcup_Q D \rangle \\ &= f\ (\sqcup_{P \times Q} (\{p\} \times D)) \\ &= \sqcup_R (f\ (\{p\} \times D)), \text{ by the continuity of } f \\ &= \sqcup_R (\mathrm{Ab}\ f\ p\ D). \end{aligned}$$

So Ab f p is continuous. Second we will show ab f is continuous. Let X be a directed subset of P. For all $q \in \mathrm{Ob}\ Q$,

$X \times \{q\}$ is a directed subset of $P \times Q$ and

$$\sqcup_{P\times Q} (X \times \{q\}) = \langle \sqcup_P X, q \rangle .$$

Now the equation

$$Ab\ f\ (\sqcup_P X) = \sqcup_{Q\Rightarrow R}\ (Ab\ f\ X)$$

follows from the observation that for all $q \varepsilon Ob\ Q$

$$\begin{aligned} Ab\ f\ (\sqcup_P X)\ q &= f\ \langle \sqcup_P X, q \rangle \\ &= f\ (\sqcup_{P\times Q}\ (X \times \{q\})) \\ &= \sqcup_R\ (f\ (X \times \{q\})), \text{ by the continuity of } f\ , \\ &= \sqcup_R\ (Ab\ f\ X\ q) \\ &= \sqcup_{Q\Rightarrow R}\ (Ab\ f\ X)\ q. \end{aligned}$$

This proves the claim.

To finish the proof the *Pdom* is a Cartesian closed category, for any $f \varepsilon P \times Q \xrightarrow[Pdom]{} R$ consider the diagram

$$\begin{array}{ccc} P \times Q & \xrightarrow{f' \times 1_Q} & (Q \Rightarrow R) \times Q \\ & {\scriptstyle f}\searrow \quad \swarrow{\scriptstyle Ap} & \\ & R & \end{array}$$

If we take $f' = Ab\ f$, then the diagram clearly commutes. Furthermore, if f' makes the diagram commute, then by applying the forgetful functor $U \varepsilon Pdom \to Set$ we get a commutative diagram in the Cartesian closed category *Set*, from which it follows that $f' = Ab\ f$. □

Suppose *K* is a Cartesian closed category, and *M* is a full subcategory of *K*. Suppose further that the distinguished product functors for *K* give objects of *M* when applied to objects of *M*, and that the terminal object *K* is an object of *M*. What else do we need to know about *M* to conclude that it, too, is a Cartesian closed category? The answer is simple: if, for all pairs M, M' of objects of *M*, $M \underset{K}{\Longrightarrow} M'$ is an object of

M, then M is a Cartesian closed category with $\Rightarrow$, Ap, and Ab for M defined by restricting to M the corresponding entities for K. We apply this observation in the proof of the following theorem.

Theorem 5.2: The category *Dom* is Cartesian closed.

Proof: Suppose P and Q are domains. It suffices to show that the predomain $P \xrightarrow[Pdom]{} Q$ is a domain by exhibiting a minimal element. Clearly the minimal element is given by

$$\perp_{P \Rightarrow Q} x = \perp_Q, \text{ for all } x \in \text{Ob } P. \qquad \square$$

Of course, *Sdom* and *Dom* share the same distinguished terminal object. Also, restriction of the distinguished product functors for *Dom* to *Sdom* yields distinguished product functors for *Sdom*. Perhaps surprisingly, we will show in the next theorem that *Sdom* is not Cartesian closed. Note that this is a rather strong claim because it asserts not only that the restrictions of $\xrightarrow[Dom]{}$, Ap_{Dom}, and Ab_{Dom} do not behave properly but also that no other way of defining $\xrightarrow[Sdom]{}$, Ap_{Sdom}, and Ab_{Sdom} works.

As a preliminary, consider an example due to John C. Reynolds. Let P be the domain with Hasse diagram

Let

$$f \in P \times P \xrightarrow[Dom]{} P$$

be the strict, continuous function given by

$$f\ x = \begin{cases} \top_P & \text{if } x \neq \langle \perp_P, \perp_P \rangle \\ \perp_P & \text{if } x = \langle \perp_P, \perp_P \rangle \end{cases}$$

Then

$$Ab_{\mathcal{D}om}\ f \in P \xrightarrow[\mathcal{D}om]{} (P \xRightarrow[\mathcal{D}om]{} P)$$

is not strict because the computation

$$Ab_{\mathcal{D}om}\ f\ \perp_P\ p = f\langle \perp_P, p \rangle = 1_P\ p \text{ , for } p \in Ob\ P,$$

shows

$$Ab_{\mathcal{D}om}\ f\ \perp_P = 1_P \neq \perp_{P \Rightarrow P} .$$

Theorem 5.3: The category $\mathcal{S}dom$ is not Cartesian closed.

Proof: Suppose to the contrary that $\mathcal{S}dom$ is Cartesian closed with internal hom functor $\xRightarrow[\mathcal{S}dom]{}$, application $Ap_{\mathcal{S}dom}$, and abstraction $Ab_{\mathcal{S}dom}$.

Suppose Q and R are arbitrary domains. To avoid subscripts on subscripts, let

$$X = Q \xRightarrow[\mathcal{S}dom]{} R \text{ and } Y = Q \xRightarrow[\mathcal{D}om]{} R.$$

Let

$$H = Ap_{\mathcal{S}dom}\ \langle Q, R \rangle \text{ and } K = Ab_{\mathcal{D}om}\ \langle X, Q, R \rangle,$$

so that

$$H \in X \times Q \xrightarrow[\mathcal{S}dom]{} R \text{ and } K\ H \in X \xrightarrow[\mathcal{D}om]{} Y.$$

We claim that K H is strict. To see this, let S be a one-point domain, and let

$$g \in S \times Q \xrightarrow[\mathcal{S}dom]{} R$$

be the constant function which maps everything to $\perp_R$. Then the diagram

$$\begin{array}{ccc} S \times Q & \xrightarrow{(Ab_{\mathcal{S}dom}\ \langle S, Q, R \rangle g) \times 1_Q} & X \times Q \\ & {\scriptstyle g} \searrow \quad \swarrow {\scriptstyle H} & \\ & R & \end{array}$$

commutes in *Sdom*. For $q \in \text{Ob } Q$, we compute as follows:

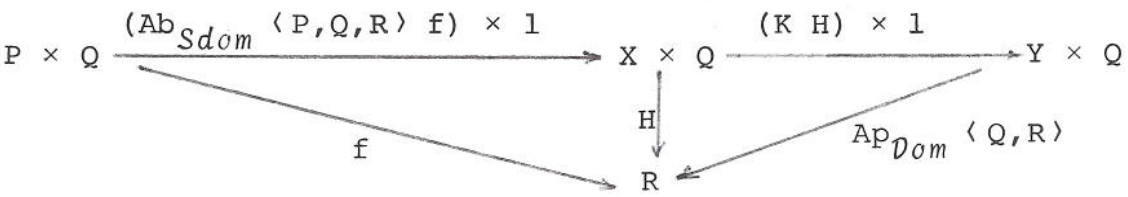
$$\begin{aligned} K\ H\ \perp_X q &= H \langle \perp_X, q \rangle, \text{ by definition of } K, \\ &= (H \circ ((Ab_{Sdom} \langle S, Q, R \rangle g) \times 1_Q)) \langle \perp_S, q \rangle \\ &= g \langle \perp_S, q \rangle \\ &= \perp_R \\ &= \perp_Y q \end{aligned}$$

Hence, $K\ H \perp_X = \perp_Y$, as desired.

Let P be any domain, and let

$$f \in P \times Q \xrightarrow[Sdom]{} R.$$

Consider the diagram



The left-hand triangle commutes in *Sdom* and the right-hand triangle commutes in *Dom*. Thus the entire diagram commutes in *Dom*. Consequently

$$Ab_{Dom} \langle P,Q,R \rangle f = (K\ H) \circ (Ab_{Sdom} \langle P,Q,R \rangle f),$$

and we have expressed $Ab_{Dom} \langle P,Q,R \rangle f$ as the composition of two strict functions, so it too must be strict. However, we have earlier seen an example of a strict continuous function f and domains P, Q, R (actually all are equal in the example) such that $Ab_{Dom} \langle P,Q,R \rangle f$ is not strict. Therefore *Sdom* is not Cartesian closed. □

In the rest of this chapter Σ denotes a category, and our focus is on the functor category $K = \Sigma \Rightarrow Pdom$ and its subcategories.

In Chapter IV we discussed how a product functor such as Π_I^{Pdom} induces a product functor Π_I^K. Furthermore each terminal object of *Pdom* determines a terminal object in *K*,

namely the constant functor which assigns the terminal object to each object of Σ.

The next theorem is nontrivial in the sense that it is not deducible from the well-known fact (see [3]) that $\Sigma \Rightarrow Top$ is a topos whenever Top is a topos,because $Pdom$ is not a topos. To see that $Pdom$ is not a topos simply observe that in $Pdom$ there are arrows which are both monomorphisms and epimorphisms but not isomorphisms (e.g. use a nonconstant arrow from a discrete, two-element predomain to a nondiscrete two element predomain). This sort of thing doesn't happen in a topos. Again, see [3].

Theorem 5.4: For any category Σ, the functor category $K = \Sigma \Rightarrow Pdom$ is Cartesian closed.

Proof: We have seen that distinguished products and terminal objects in K are inherited from $Pdom$. Therefore, we may turn our attention to the nature of exponentiation in K.

By

$$\hom_\Sigma \varepsilon \Sigma^{op} \times \Sigma \longrightarrow Set$$

we mean the usual hom functor. Thus for objects S and T of Σ we have $\hom_\Sigma \langle S, T \rangle = S \xrightarrow[\Sigma]{} T$. As before, let

$$E \varepsilon Set \longrightarrow Pdom$$

be the embedding functor which associates a discrete partial order with each set. Hence

$$E \circ \hom_\Sigma \varepsilon \Sigma^{op} \times \Sigma \longrightarrow Pdom .$$

Denote by

$$\hom^\Sigma \varepsilon \Sigma^{op} \to (\Sigma \Rightarrow Pdom) = \Sigma^{op} \longrightarrow K$$

the curried version of $E \circ \hom_\Sigma$. Thus, for all objects S

and S' of Σ, $\hom^{\Sigma}$ S S' is the discretely ordered predomain whose underlying set is $S \xrightarrow[\Sigma]{} S'$.

By taking the product of $\hom^{\Sigma}$ and the identity functor on K we obtain

$$\hom^{\Sigma} \times 1_K \in \Sigma^{op} \times K \longrightarrow K \times K.$$

We may compose this with the distinguished binary product functor for K, denoted

$$\Pi_2^K \in K \times K \longrightarrow K,$$

getting

$$\Pi_2^K \circ (\hom^{\Sigma} \times 1_K) \in \Sigma^{op} \times K \longrightarrow K.$$

The curried version of this last functor is denoted

$$H \in \Sigma^{op} \longrightarrow (K \Rightarrow K).$$

Hence if S $\in$ Ob Σ and F $\in$ Ob K, then

$$H\ S\ F = (\hom^{\Sigma} S) \times_K F;$$

furthermore, if T $\in$ Ob Σ, then

$$H\ S\ F\ T = (\hom^{\Sigma} S\ T) \times_{Pdom} (F\ T).$$

For each pair F, F' of functors in $\Sigma \longrightarrow Pdom$ we seek to define

$$F \xRightarrow[K]{} F' = F \Rightarrow F' \in \Sigma \longrightarrow Pdom.$$

Thus, for each object S of Σ we must describe a predomain (F $\Rightarrow$ F') S. The underlying set of (F $\Rightarrow$ F') S is

$$Ob\ ((F \Rightarrow F')\ S) = H\ S\ F \xrightarrow[K]{} F' = ((\hom^{\Sigma} S) \times F) \xrightarrow[K]{} F',$$

which is a set of natural transformations. The partial order on (F $\Rightarrow$ F') S is given by

$$\eta \leq_{(F \Rightarrow F')S} \eta' \text{ iff } \eta T \leq_{HSFT \Rightarrow F'T} \eta' T \text{ for all } T \in Ob\ \Sigma.$$

(Note that the underlying set of the predomain $H\ S\ F\ T \xRightarrow[Pdom]{} F'\ T$ is the set $H\ S\ F\ T \xrightarrow[Pdom]{} F'T$, of which both ηT and $\eta' T$ are members.) We must check that this partial order is directed-complete. Let D be a directed subset of (F $\Rightarrow$ F') S. The

definition of $\leq_{(F \Rightarrow F')S}$ implies that, for each $T \in \mathrm{Ob}\ \Sigma$, DT is a directed subset of $H\ S\ F\ T \Rightarrow F'T$. Hence, we can define an $(\mathrm{Ob}\ \Sigma)$-indexed collection ν_D of continuous functions by

$$\nu_D T = \bigsqcup_{HSFT \Rightarrow F'T} (DT).$$

Of course, we aim to show that $\nu_D = \bigsqcup_{(F \Rightarrow F')S} D$. First we must verify that $\nu_D \in \mathrm{Ob}\ ((F \Rightarrow F')S)$, i.e. ν_D is a natural transformation. Let $\tau \in T \xrightarrow[\Sigma]{} T'$. We must show that

$$\begin{array}{ccc} (\hom^{\Sigma} S\ T) \times (FT) & \xrightarrow{\nu_D T} & F'T \\ \downarrow{\scriptstyle (\hom^{\Sigma} S\ \tau) \times (F\tau)} & & \downarrow{\scriptstyle F'\tau} \\ (\hom^{\Sigma} S\ T') \times (FT') & \xrightarrow{\nu_D T'} & F'T' \end{array}$$

commutes in *Pdom*. For each $\eta \in D$, the commutativity of

$$\begin{array}{ccc} (\hom^{\Sigma} S\ T) \times (F\ T) & \xrightarrow{\eta T} & F'T \\ \downarrow{\scriptstyle (\hom^{\Sigma} S\ \tau) \times (F\tau)} & & \downarrow{\scriptstyle F'\tau} \\ (\hom^{\Sigma} S\ T') \times (FT') & \xrightarrow{\eta T'} & F'T' \end{array}$$

implies that, whenever $\sigma \in S \xrightarrow[\Sigma]{} T$ and $t \in \mathrm{Ob}\ (FT)$, we have

$$\begin{aligned} \eta T' \langle \tau \circ \sigma, F\tau t \rangle &= ((\eta T') \circ ((\hom^{\Sigma} S\ \tau) \times (F\tau))) \langle \sigma, t \rangle \\ &= F'\tau\ (\eta T \langle \sigma, t \rangle). \end{aligned}$$

Thus, we may write

$$D\ T' \langle \tau \circ \sigma, F\ \tau\ t \rangle = F'\ \tau\ (D\ T \langle \sigma, t \rangle).$$

Now take any $\sigma \in S \xrightarrow[\Sigma]{} T$ and $t \in \mathrm{Ob}\ (FT)$, and compute:

$$\begin{aligned} &((\nu_D T') \circ ((\hom^{\Sigma} S\ \tau) \times (F\tau))) \langle \sigma, t \rangle \\ &\quad = \nu_D T' \langle \sigma \circ \tau, F\tau t \rangle \\ &\quad = \bigsqcup_{HSFT' \Rightarrow F'T} (D\ T') \langle \sigma \circ \tau, F\tau t \rangle \\ &\quad = \bigsqcup_{F'T'} (F'\tau\ (D\ T \langle \sigma, t \rangle)) \\ &\quad = F'\tau\ (\bigsqcup_{F'T} (DT \langle \sigma, t \rangle)), \text{ by the continuity of } F'\tau, \end{aligned}$$

$$= F'\tau\ (\sqcup_{HSFT \Rightarrow F'T}\ (D\ T)\ \langle \sigma, t\rangle)$$

$$= F'\tau\ (\nu_D T \langle \sigma, t\rangle)$$

$$= ((F'\tau) \circ (\nu_D T))\ \langle \sigma, t\rangle ,$$

proving ν_D is indeed a natural transformation. Next we will demonstrate that ν_D is actually the least upper bound of D. Let $\eta \in D$. For all $T \in Ob\ \Sigma$, $\eta T \in D\ T$; consequently

$$\eta T \leq \sqcup_{HSFT \Rightarrow F'T}\ (D\ T) = \nu_D T.$$

Thus, $\eta \leq_{(F \Rightarrow F')S} \nu_D$, which shows ν_D is an upper bound of D. Suppose μ is an upper bound of D, so that for all $\eta \in D$ and all $T \in Ob\ \Sigma$, $\eta T \leq_{HSFT \Rightarrow F'T} \mu T$. Then, for all $T \in Ob\ \Sigma$,

$$\nu_D T = \sqcup_{HSFT \Rightarrow F'T}\ (DT) \leq_{HSFT \Rightarrow F'T} \mu T$$

Thus $\nu_D \leq_{(F \Rightarrow F')S} \mu$, which demonstrates that ν_D is the least upper bound of D. This completes the proof that $(F \Rightarrow F')\ S$ is a predomain.

Now we turn to the task of defining $F \Rightarrow F'$ on morphisms of Σ. Let $\sigma \in S \xrightarrow[\Sigma]{} S'$. We must give a continuous function.

$$(F \Rightarrow F')\ \sigma \quad \in (F \Rightarrow F')\ S \xrightarrow[Pdom]{} (F \Rightarrow F')\ S'.$$

So take $\eta \in ((hom^{\Sigma}\ S) \times F) \xrightarrow[K]{} F'$; we must define

$$(F \Rightarrow F')\ \sigma\ \eta \in ((hom^{\Sigma}\ S') \times F) \xrightarrow[K]{} F'.$$

Let $T \in Ob\ \Sigma$. The function

$$(F \Rightarrow F')\sigma\ \eta\ T \in ((hom^{\Sigma}\ S'\ T) \times (FT)) \xrightarrow[Pdom]{} (F'T)$$

is given by

$$(F \Rightarrow F')\sigma\ \eta\ T\ \langle \tau, t\rangle = \eta\ T\ \langle \tau \circ \sigma,\ t\rangle$$

where $\tau \in S' \xrightarrow{}_{\Sigma} T$ and $t \in Ob\ (F\ T)$. There are a number of points that need verification. First, we will show that the function $(F \Rightarrow F')\ \sigma\ \eta\ T$ is continuous. Because the partial order on the first component is discrete, each directed subset of $(hom^{\Sigma}\ S'\ T) \times (F\ T)$ has the form $\{\langle \tau, t\rangle \mid t \in D\}$ where

$\tau \in S' \xrightarrow[\Sigma]{} T$ and D is a directed subset of F T, and for each such pair τ, D we have the following computation:

$$(F \Rightarrow F')\ \sigma\ \eta\ T\ (\sqcup_{HS'FT} \{\langle \tau,t\rangle \mid t \in D\} = (F \Rightarrow F')\sigma\eta T\langle \tau, \sqcup_{FT} D\rangle$$

$$= \eta T\ \langle \tau\circ\sigma, \sqcup_{FT} D\rangle$$

$$= \eta T\ (\sqcup_{HSFT}\ \{\langle \tau\circ\sigma, t\rangle \mid t \in D\})$$

$$= \sqcup_{F'T}\ (\eta T\{\langle \tau\circ\sigma, t\rangle \mid t \in D\}), \text{ since } \eta T \text{ is continuous,}$$

$$= \sqcup_{F'T}\ \{(F \Rightarrow F')\ \sigma\ \eta\ T\ \langle \tau, t\rangle \mid t \in D\}\ .$$

Hence, $(F \Rightarrow F')\sigma\ \eta\ T$ is indeed continuous. Second, we must show that $(F \Rightarrow F')\sigma\ \eta$ is a natural transformation. Let $\tau \in T \xrightarrow[\Sigma]{} T'$. We must demonstrate that

$$\begin{array}{ccc} (\hom^{\Sigma} S'\ T) \times (FT) & \xrightarrow{(F\Rightarrow F')\sigma\eta T} & F'T \\ \Big\downarrow (\hom^{\Sigma} S'\tau)\times(F\tau) & & \Big\downarrow F'\tau \\ (\hom^{\Sigma} S'T') \times (FT') & \xrightarrow{(F\Rightarrow F)\sigma\ \eta\ T'} & F'T' \end{array}$$

commutes in *Pdom*. We will use the commutativity of

$$\begin{array}{ccc} (\hom^{\Sigma} S\ T)\times(FT) & \xrightarrow{\eta T} & F'T \\ \Big\downarrow (\hom^{\Sigma} S\ \tau)\times(F\tau) & & \Big\downarrow F'T \\ (\hom^{\Sigma} S\ T')\times(FT) & \xrightarrow{\eta T'} & F'T' \end{array}$$

which is a consequence of the naturality of η. Let $\rho \in S' \xrightarrow[\Sigma]{} T$ and $t \in Ob\ (FT)$. Then

$$(F \Rightarrow F')\sigma\ \eta\ T'\ (((\hom^{\Sigma} S'\ \tau)\ \times\ (F\tau))\ \langle \rho, t\rangle) = (F \Rightarrow F')\sigma\ \eta\ T'\langle \tau\circ\rho, F\tau t\rangle$$

$$= \eta T'\ \langle \tau\circ\rho\circ\sigma,\ F\tau t\ \rangle$$

$$= \eta T'\ (((\hom^{\Sigma} S\ \tau)\ \times\ (F\tau))\ \langle \rho\circ\sigma, t\rangle)$$

$$= F'\tau\ (\eta T\ \langle \rho\circ\sigma, t\rangle)$$

$$= F'\tau\ ((F \Rightarrow F')\sigma\ \eta\ T\ \langle \rho, t\rangle),$$

as desired. Finally we must check that $(F \Rightarrow F')\ \sigma$ is continuous.

Let D be a directed subset of $(F \Rightarrow F')\ S$. We will show that

$$(F \Rightarrow F')\ \sigma(\sqcup_{(F\Rightarrow F')S} D) = \sqcup_{(F\Rightarrow F')S'}((F \Rightarrow F')\ \sigma\ D).$$

Let $T \in \mathrm{Ob}\ \Sigma$, $\tau \in S' \xrightarrow[\Sigma]{} T$, $t \in \mathrm{Ob}\ (FT)$. Then

$$\begin{aligned}(F \Rightarrow F')\ \sigma\ (\sqcup_{(F\Rightarrow F')S} D)\,T\langle \tau,t\rangle &= (\sqcup_{(F\Rightarrow F')S} D)\ T\langle \tau\circ\sigma,\ t\rangle \\ &= (\sqcup_{HSFT\Rightarrow F'T}(DT))\ \langle \tau\circ\sigma,\ t\rangle \\ &= \sqcup_{F'T}(DT\ \langle \tau\circ\sigma,\ t\rangle) \\ &= \sqcup_{F'T}((F \Rightarrow F')\sigma\ D\ T\ \langle \tau,\ t\rangle) \\ &= \sqcup_{HS'FT\Rightarrow F'T}((F \Rightarrow F')\sigma\ DT)\ \langle \tau,t\rangle \\ &= \sqcup_{(F\Rightarrow F')S'}\ ((F \Rightarrow F')\ \sigma\ D)\ T\langle \tau,\ t\rangle\end{aligned}$$

from which the desired result follows immediately.

Now that we have defined $F \Rightarrow F'$ on both objects and arrows of Σ, we must verify that it is a functor by showing that $F \Rightarrow F'$ preserves composition and identity arrows. Let $\sigma \in S \xrightarrow[\Sigma]{} S'$ and $\sigma' \in S' \xrightarrow[\Sigma]{} S''$. The equalities

$$(F \Rightarrow F')\ (\sigma'\circ\sigma) = ((F \Rightarrow F')\sigma')\ \circ\ ((F \Rightarrow F')\ \sigma)$$

and

$$(F \Rightarrow F')\ \ 1^{\Sigma}_{S} \ = \ 1^{Pdom}_{(F\Rightarrow F')S}$$

follow from the computations

$$\begin{aligned}(F \Rightarrow F')\ (\sigma'\circ\sigma)\eta\ T\ \langle\tau,t\rangle &= \eta T\ \langle\tau\circ\sigma'\circ\sigma,\ t\rangle \\ &= (F \Rightarrow F')\ \sigma\ \eta\ T\langle\tau\circ\sigma',\ t\rangle \\ &= (F \Rightarrow F')\ \sigma'\ ((F \Rightarrow F')\sigma\ \eta)\ T\langle\tau,t\rangle \\ &= (((F \Rightarrow F')\ \sigma')\ \circ\ ((F \Rightarrow F')\ \sigma))\ \eta\ T\ \langle\tau,t\rangle\end{aligned}$$

and

$$\begin{aligned}(F \Rightarrow F')\ 1^{\Sigma}_{S}\ \eta\ T\ \langle\rho,t\rangle &= \eta T\ \langle\rho\circ 1^{\Sigma}_{S},\ t\rangle \\ &= \eta T\ \langle\rho,t\rangle \\ &= 1^{Pdom}_{(F\Rightarrow F')S}\ \eta\ T\ \langle\rho,t\rangle,\end{aligned}$$

where $\eta \in ((\hom^{\Sigma} S) \times F) \xrightarrow[K]{} F'$, $T \in Ob\ \Sigma$, $\tau \in S' \xrightarrow[\Sigma]{} T$, $t \in Ob\ (FT)$, and $\rho \in S \xrightarrow[\Sigma]{} T$. This completes the definition of $F \Rightarrow F'$ and the proof that it is a functor.

Given functors $F, F' \in \Sigma \longrightarrow Pdom$, we now turn to the definition of the natural transformation

$$Ap = Ap\ \langle F, F' \rangle \in ((F \Rightarrow F') \times_K F) \xrightarrow[K]{} F'.$$

Let $S \in Ob\ \Sigma$. Then

$$Ap\ S \in ((F \Rightarrow F')\ S) \times_{Pdom} (F\ S)) \xrightarrow[Pdom]{} F'S$$

is given by

$$Ap\ S\ \langle \eta, s \rangle = \eta S\ \langle 1^{\Sigma}_S, s \rangle$$

where $\eta \in ((\hom^{\Sigma} S) \times_K F) \xrightarrow[K]{} F'$ and $s \in Ob\ (FS)$.

Some verifications are in order. First, we must check that Ap S is continuous, i.e. if D denotes a directed subset of $((F \Rightarrow F')\ S) \times_{Pdom} (FS)$, then

$$Ap\ S(\sqcup_{((F \Rightarrow F')S) \times (FS)} D) = \sqcup_{F'S} (Ap\ S\ D).$$

A consequence of the continuity of the projection mappings from a product onto its factors is that the directed subset D gives rise to directed subsets

$$D_1 = \{\eta \mid \langle \eta, s \rangle \in D\},\ D_2 = \{s \mid \langle \eta, s \rangle \in D\} \text{ and } D' = D_1 \times D_2$$

Also,

$$\sqcup_{((F \Rightarrow F')S) \times (FS)} D = \langle \sqcup_{(F \Rightarrow F')S} D_1, \sqcup_{FS} D_2 \rangle = \sqcup_{((F \Rightarrow F')S) \times (FS)} D'.$$

A routine argument establishes that the least upper bounds of the two sets

$$Ap\ S\ D = \{\eta S\ \langle 1^{\Sigma}_S, s \rangle \mid \langle \eta, s \rangle \in D\}$$

$$Ap\ S\ D' = \{\eta S\ \langle 1^{\Sigma}_S, s \rangle \mid \langle \eta, s \rangle \in D'\}$$

are the same. Now compute:

$$\begin{aligned}
&\text{Ap } S \ (\sqcup_{((F\Rightarrow F')S)\times(FS)} D) \\
&\quad = \text{Ap } S \ \langle \sqcup_{(F\Rightarrow F')S} D_1, \sqcup_{FS} D_2 \rangle \\
&\quad = (\sqcup_{(F\Rightarrow F')S} D_1 S) \ \langle 1_S^{\Sigma}, \sqcup_{FS} D_2 \rangle \\
&\quad = (\sqcup_{HSFS\Rightarrow F'S} (D_1 S)) \ \langle 1_S^{\Sigma}, \sqcup_{FS} D_2 \rangle \\
&\quad = \sqcup_{F'S} \{\eta S \ \langle 1_S^{\Sigma}, \sqcup_{FS} \{s \mid s \in D_2\}\rangle \mid \eta \in D_1\} \\
&\quad = \sqcup_{F'S} \{\sqcup_{F'S} \{\eta S \ \langle 1_S^{\Sigma}, s\rangle \mid s \in D_2\} \mid \eta \in D_1\}, \text{ by continuity,} \\
&\quad = \sqcup_{F'S} \{\eta S \ \langle 1_S^{\Sigma}, s\rangle \mid \langle \eta, s\rangle \in D'\} \\
&\quad = \sqcup_{F'S} (\text{Ap } S \ D') \\
&\quad = \sqcup_{F'S} (\text{Ap } S \ D),
\end{aligned}$$

as desired. Second, we must verify that Ap is a natural transformation. Suppose $\sigma \in S \xrightarrow[\Sigma]{} S'$. We must establish the commutativity in *Pdom* of

$$\begin{array}{ccc}
((F\Rightarrow F')S)\times(FS) & \xrightarrow{\text{Ap } S} & F'S \\
\downarrow{\scriptstyle ((F\Rightarrow F')\sigma)\times(F\sigma)} & & \downarrow{\scriptstyle F'\sigma} \\
((F\Rightarrow F')S')\times(FS') & \xrightarrow{\text{Ap } S'} & F'S'
\end{array}$$

Let $\eta \in ((\hom^{\Sigma} S) \times F) \xrightarrow[K]{} F'$ and $s \in \text{Ob } (FS)$.

We know that

$$\begin{array}{ccc}
(\hom^{\Sigma} S\ S)\times(FS) & \xrightarrow{\eta S} & F'S \\
\downarrow{\scriptstyle (\hom^{\Sigma} S\ \sigma)\times(F\sigma)} & & \downarrow{\scriptstyle F'\sigma} \\
(\hom^{\Sigma} S\ S')\times(FS') & \xrightarrow{\eta S'} & F'S'
\end{array}$$

commutes in *Pdom*. Compute:

$$\begin{aligned}
&((\text{Ap } S') \circ (((F \Rightarrow F')\ \sigma) \times (F\sigma))\ \langle \eta, s\rangle \\
&\qquad = \text{Ap } S' \ ((((F \Rightarrow F')\ \sigma) \times (F\sigma))\ \langle \eta, s\rangle) \\
&\qquad = \text{Ap } S' \ \langle (F \Rightarrow F')\ \sigma\ \eta,\ F\ \sigma\ s\rangle
\end{aligned}$$

$$= (F \Rightarrow F')\ \sigma\ \eta\ S'\ \langle 1^{\Sigma}_{S'},\ ,\ F\sigma s\rangle$$

$$= \eta S'\ \langle 1^{\Sigma}_{S'} \circ \sigma,\ F\ \sigma\ s\rangle$$

$$= \eta S'\ \langle \sigma \circ 1^{\Sigma}_{S},\ F\ \sigma\ s\rangle$$

$$= \eta S'\ (((\hom^{\Sigma}\ S\ \sigma) \times (F\sigma))\ \langle 1^{\Sigma}_{S},\ s\rangle)$$

$$= F'\sigma\ (\eta S\ \langle 1^{\Sigma}_{S},\ s\rangle)$$

$$= F'\sigma\ (Ap\ S\ \langle \eta, s\rangle)$$

$$= ((F'\sigma) \circ (Ap\ S))\ \langle \eta, s\rangle.$$

Therefore, $Ap = Ap\ \langle F,\ F'\rangle$ is indeed a natural transformation.

Given functors F, F', F" in $\Sigma \longrightarrow Pdom$, our task is to define

$$Ab = Ab\ \langle F,\ F',\ F"\rangle\ \varepsilon\ ((F \times F') \xrightarrow[K]{} F") \xrightarrow[Set]{} (F \xrightarrow[K]{} (F' \Rightarrow F"))$$

Take a natural transformation $\eta\ \varepsilon\ (F \times F') \xrightarrow[K]{} F"$; we want a natural transformation

$$Ab\ \eta\ \varepsilon\ F \xrightarrow[K]{} (F' \Rightarrow F").$$

So let $S\ \varepsilon\ Ob\ \Sigma$. We must define

$$Ab\ \eta\ S\ \varepsilon\ FS \xrightarrow[Pdom]{} (F' \Rightarrow F")S.$$

Let $s\ \varepsilon\ Ob\ (FS)$; we need

$$Ab\ \eta\ S\ s\ \varepsilon\ (\hom^{\Sigma}\ S) \times F' \xrightarrow[K]{} F".$$

Let $T\ \varepsilon\ Ob\ \Sigma$. Define

$$Ab\ \eta\ S\ s\ T\ \varepsilon\ (\hom^{\Sigma}\ S\ T) \times (F'T) \xrightarrow[Pdom]{} F"T$$

by

$$Ab\ \eta\ S\ s\ T\ \langle \sigma, t\rangle = \eta T\ \langle F\sigma s, t\rangle$$

where $\sigma\ \varepsilon\ S \xrightarrow[\Sigma]{} T$ and $t\ \varepsilon\ Ob\ (F'T)$. Of course, each layer of this definition requires some sort of verification. First, we must check that $Ab\ \eta\ S\ s\ T$ is continuous. Since $\hom^{\Sigma}\ S\ T$ is discretely ordered, an arbitrary directed subset of $(\hom^{\Sigma}\ S\ T)$

$\times$ (F'T) may be described as $\{\langle \sigma,t\rangle \mid t \in D\}$ where $\sigma \in S \xrightarrow[\Sigma]{} T$ and D is a directed subset of F'T. The computation

$$\begin{aligned}
\text{Ab } \eta \text{ S s T } (\sqcup_{HSF'T} \{\langle \sigma,t\rangle \mid t \in D\}) &= \text{Ab } \eta \text{ S s T } \langle \sigma, \sqcup_{F'T} D\rangle \\
&= \eta T \langle F\,\sigma\, s, \sqcup_{F'T} D\rangle \\
&= \eta T\, (\sqcup_{(FT)\times(F'T)} \{\langle F\,\sigma\, s, t\rangle \mid t \in D\}) \\
&= \sqcup_{F''T} \{\eta T \langle F\sigma s, t\rangle \mid t \in D\}, \text{ by continuity of } \eta T, \\
&= \sqcup_{F''T} \{\text{Ab } \eta \text{ S s T } \langle \sigma, t\rangle \mid t \in D\}
\end{aligned}$$

demonstrates that Ab η S s T is continuous. Second, we must check that Ab η S s is a natural transformation. Let $\tau \in T \xrightarrow[\Sigma]{} T'$. We need to demonstrate the commutativity of

$$\begin{array}{ccc}
(\hom^{\Sigma} S\ T)\times(F'T) & \xrightarrow{\text{Ab } \eta \text{ S s T}} & F''T \\
\Big\downarrow{\scriptstyle (\hom^{\Sigma} S\ \tau)\times(F'\tau)} & & \Big\downarrow{\scriptstyle F''\tau} \\
(\hom^{\Sigma} S\ T')\times(F'T') & \xrightarrow{\text{Ab } \eta \text{ S s T'}} & F''T'
\end{array} .$$

Let $\sigma \in S \xrightarrow[\Sigma]{} T$ and $t \in$ Ob (F'T), compute:

$$\begin{aligned}
((\text{Ab } \eta \text{ S s T'}) \circ ((\hom^{\Sigma} S\ \tau) \times (F'\tau)))\ \langle \sigma,t\rangle &= \text{Ab } \eta \text{ S s T' } (((\hom^{\Sigma} S\ \tau) \times (F'\tau))\ \langle \sigma,t\rangle) \\
&= \text{Ab } \eta \text{ S s T' } \langle \tau\circ\sigma, F'\tau t\rangle \\
&= \eta T' \langle F\ (\tau\circ\sigma)\ s, F'\tau t\rangle \\
&= \eta T' \langle F\tau\ (F\sigma s), F'\tau t\rangle, \text{ because F is a functor,} \\
&= \eta T' \ (((F\tau) \times (F'\tau))\ \langle F\sigma s, t\rangle) \\
&= F''\tau\ (\eta T \langle F\sigma s, t\rangle), \text{ by the naturality of } \eta, \\
&= F''\tau\ (\text{Ab } \eta \text{ S s T } \langle \sigma, t\rangle) \\
&= ((F''\tau) \circ (\text{Ab } \eta \text{ S s T}))\ \langle \sigma,t\rangle.
\end{aligned}$$

Thus Ab η S s is natural. Third, we will check that Ab η S

is continuous. Let D be a directed subset of FS. If $T \in Ob\ \Sigma$, $\sigma \in S \xrightarrow[\Sigma]{} T$, and $t \in Ob\ (F'T)$, then

$$\begin{aligned}
&Ab\ \eta\ S\ (\sqcup_{FS} D)\ T\ \langle \sigma, t\rangle \\
&\quad = \eta T\ \langle F\sigma\ (\sqcup_{FS} D),\ t\rangle \\
&\quad = \eta T \langle \sqcup_{FT}\ (F\sigma D),\ t\rangle, \text{ because } F\sigma \text{ is continuous,} \\
&\quad = \eta T\ (\sqcup_{(FT)\times(F'T)} \{\langle F\sigma s, t\rangle \mid s \in D\}) \\
&\quad = \sqcup_{F''T} \{\eta T \langle F\sigma s,\ t\rangle \mid s \in D\}, \text{ because } \eta T \text{ is continuous,} \\
&\quad = \sqcup_{F''T} (Ab\ \eta\ S\ D\ T\ \langle \sigma,\ t\rangle) \\
&\quad = (\sqcup_{HSF'T \Rightarrow F''T} (Ab\ \eta\ S\ D\ T))\langle \sigma,\ t\rangle \\
&\quad = (\sqcup_{(F'\Rightarrow F'')S}\ (Ab\ \eta\ S\ D))\ T\ \langle \sigma,\ t\rangle .
\end{aligned}$$

Therefore,

$$Ab\ \eta\ S\ (\sqcup_{FS} D) = \sqcup_{(F'\Rightarrow F'')S}\ (Ab\ \eta\ S\ D),$$

which shows Ab η S is continuous. Finally we must prove that Ab η is a natural transformation. Suppose $\sigma \in S \xrightarrow[\Sigma]{} S'$. We are interested in establishing the commutativity in *Pdom* of

$$\begin{array}{ccc}
FS & \xrightarrow{\quad Ab\ \eta\ S\quad} & (F' \Rightarrow F'')S \\
{\scriptstyle F\sigma}\downarrow & & \downarrow{\scriptstyle (F'\Rightarrow F'')\sigma} \\
FS' & \xrightarrow{\quad Ab\ \eta\ S'\quad} & (F' \Rightarrow F'')S'
\end{array}$$

Let $s \in Ob\ (FS)$, $T \in Ob\ \Sigma$, $\tau \in S' \xrightarrow[\Sigma]{} T$, and $t \in Ob\ (F'T)$. Then

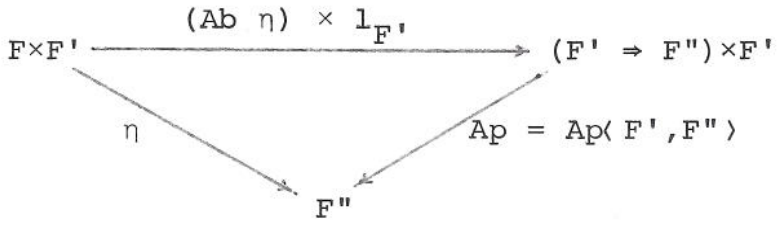
$$\begin{aligned}
&((Ab\ \eta\ S') \circ (F\sigma))\ s\ T\ \langle \tau,\ t\rangle \\
&\quad = Ab\ \eta\ S'\ (F\sigma s)\ T\ \langle \tau,\ t\rangle \\
&\quad = \eta T\ \langle F\tau\ (F\sigma s),\ t\rangle \\
&\quad = \eta T\ \langle F\ (\tau\circ\sigma)\ s,\ t\rangle, \text{ because } F \text{ is a functor,} \\
&\quad = Ab\ \eta\ S\ s\ T\ \langle \tau\circ\sigma,\ t\rangle
\end{aligned}$$

$$= (F' \Rightarrow F'')\ \sigma\ (Ab\ \eta\ S\ s)\ T\ \langle \tau, t \rangle$$

$$= (((F' \Rightarrow F'')\ \sigma) \circ (Ab\ \eta\ S))\ s\ T\ \langle \tau, t \rangle,$$

and from this computation commutativity of the diagram above is clear.

We now arrive at the heart of the proof that K is a Cartesian closed category. Given functors F. F', F" in $\Sigma \longrightarrow Pdom$, we claim that for each natural transformation $\eta \in F \times F' \xrightarrow[K]{} F''$ the natural transformation $Ab\ \eta \in F \xrightarrow[K]{} (F' \Rightarrow F'')$ is the unique arrow making the diagram



commute in K. Let $S \in Ob\ \Sigma$, $s \in Ob\ (FS)$, $s' \in Ob\ (F'S)$. Then

$$(Ap \circ ((Ab\ \eta) \times 1^{K}_{F'}))\ S\ \langle s, s' \rangle$$

$$= ((Ap\ S) \circ ((Ab\ \eta\ S) \times (1^{K}_{F'}, S)))\ \langle s, s' \rangle$$

$$= Ap\ S\ (((Ab\ \eta\ S) \times 1^{Pdom}_{F'S})\langle s, s' \rangle)$$

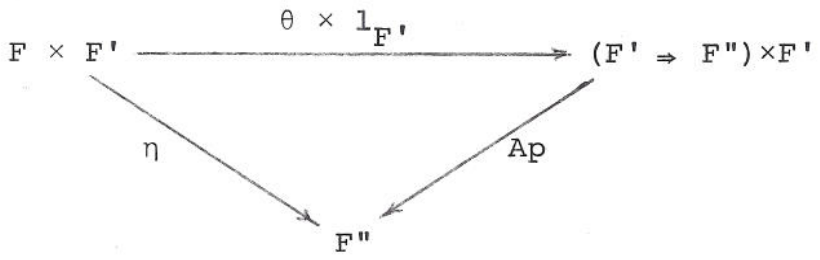
$$= Ap\ S\ \langle Ab\ \eta\ S\ s, s' \rangle$$

$$= Ab\ \eta\ S\ s\ S\ \langle 1^{\Sigma}_{S}, s' \rangle$$

$$= \eta S\ \langle F1^{\Sigma}_{S}\ s, s' \rangle$$

$$= \eta S\ \langle s, s' \rangle, \text{ because F is a functor.}$$

Thus, the diagram commutes. To deal with the uniqueness assertion, suppose $\theta \in F \xrightarrow[K]{} (F \Rightarrow F'')$ and the diagram



commutes. Let $S \varepsilon$ Ob Σ, $s \varepsilon$ Ob (FS), $T \varepsilon$ Ob Σ, $\sigma \varepsilon S \xrightarrow[\Sigma]{} T$, and $t \varepsilon$ Ob $(F'T)$. We compute as follows:

$\theta\ S\ s\ T\ \langle \sigma, t \rangle$

$= (F' \Rightarrow F")\ \sigma\ (\theta S s)\ T\ \langle 1^{\Sigma}_{T}, t \rangle$, by the definition of $F \Rightarrow F'$,

$= (((F' \Rightarrow F")\sigma) \circ (\theta S))\ \ s\ T\ \langle 1^{\Sigma}_{T}, t \rangle$

$= ((\theta T) \circ (F\sigma))\ s\ T\ \langle 1^{\Sigma}_{T}, t \rangle$, by the naturality of θ,

$= \theta T\ (F\sigma s)\ T\ \langle 1^{\Sigma}_{T}, t \rangle$

$= Ap\ T\ \langle \theta T\ (F\sigma s), t \rangle$, by the definition of Ap,

$= Ap\ T\ (((\theta T) \times 1^{Pdom}_{F'T})\ \langle F\sigma s, t \rangle)$

$= ((Ap\ T) \circ ((\theta T) \times (1^{K}_{F'}, T)))\ \langle F\sigma s, t \rangle$

$= (Ap \circ (\theta \times 1^{K}_{F'},))\ T\ \langle F\sigma s, t \rangle$

$= \eta T\ \langle F\sigma s, t \rangle$, by the above diagram,

$= Ab\ \eta\ S\ s\ T\ \langle \sigma, t \rangle$, by the definition of Ab η.

Therefore, $\theta = Ab\ \eta$, and this proves the claim. □

The category $\Sigma \Rightarrow Pdom$ is a little bit too large for the applications to the semantics of programming languages that we have in mind; its use would involve many annoying verifications that certain predomains are indeed domains. Thus, we might try using $\Sigma \Rightarrow Dom$ rather than $\Sigma \Rightarrow Pdom$. However, the obvious way of showing that $\Sigma \Rightarrow Dom$ is a Cartesian closed category, i.e. by embedding it in $\Sigma \Rightarrow Pdom$, fails! We suspect

that, for some Σ, $\Sigma \Rightarrow \mathcal{D}om$ cannot be made into a Cartesian closed category, although we are unable at this time to give a proof. Another idea is to use $\Sigma \Rightarrow \mathit{Sdom}$. This, too, is doomed, as one can see by taking Σ to be the category with one object and one arrow; then $\Sigma \Rightarrow \mathit{Sdom}$ is isomorphic to Sdom, which we have shown is not Cartesian closed. Fortunately, there is another alternative, which we now present.

Let M be the full subcategory of $K = \Sigma \Rightarrow \mathit{Pdom}$ whose collection of objects consists of those functors which factor through the inclusion functor $J \in \mathit{Sdom} \to \mathit{Pdom}$. Thus a functor $F \in \Sigma \longrightarrow \mathit{Pdom}$ is an object of M if (1) FS is a domain for every $S \in \mathrm{Ob}\ \Sigma$, and (2) $F\sigma$ is a <u>strict</u> continuous function for every $\sigma \in \mathrm{Ar}\ \Sigma$. The inclusion functor J induces an embedding

$$\Sigma \Rightarrow J \in \Sigma \Rightarrow \mathit{Sdom} \longrightarrow \Sigma \Rightarrow \mathit{Pdom}$$

given by

$$(\Sigma \Rightarrow J)\ F = J \circ F, \text{ for } F \in \mathrm{Ob}\ (\Sigma \Rightarrow \mathit{Sdom}),$$

and

$$(\Sigma \Rightarrow J)\ \eta = J * \eta, \text{ for } \eta \in \mathrm{Ar}\ (\Sigma \Rightarrow \mathit{Sdom}).$$

At first glance, it might appear that the image of $\Sigma \Rightarrow J$ is M. However, this is false, because the assertion $\eta \in F \underset{M}{\longrightarrow} G$ requires that ηS be a continuous function for each $S \in \mathrm{Ob}\ \Sigma$, whereas $\eta \in F \underset{(\Sigma \Rightarrow J)(\Sigma \Rightarrow \mathit{Sdom})}{\longrightarrow} G$ correspondingly requires that ηS be a strict, continuous function for each $S \in \mathrm{Ob}\ \Sigma$.

It is easy to see that the distinguished product functors for K give objects of M when applied to objects of M, and that the terminal object of K is an object of M. These observations are the groundwork for the following theorem.

Theorem 5.5: The category M is Cartesian closed, with $\Rightarrow$, Ap, and Ab defined by restricting to M the corresponding entities for K.

Proof: Let the functors F and G be objects of M. It suffices to show that $F \underset{K}{\Longrightarrow} G$ is an object of M.

Suppose $S \varepsilon$ Ob Σ. We wish to show that the predomain $(F \underset{K}{\Longrightarrow} G)S$ has a minimal element. Recall that the underlying set of this predomain is the set of natural transformations

$$(\hom^{\Sigma} S) \times F \xrightarrow[K]{} G.$$

We attempt to define a natural transformation μ in this set by

$$\mu T \langle \sigma, t \rangle = \perp_{GT}$$

where $T \varepsilon$ Ob Σ, $\sigma \varepsilon S \xrightarrow[\Sigma]{} T$, $t \varepsilon$ Ob (FT).

For each $T \varepsilon$ Ob Σ,

$$\mu T \varepsilon (\hom^{\Sigma} S\ T) \times (F\ T) \xrightarrow[Pdom]{} GT,$$

because μT is a constant function and, hence, a continuous function. Naturality of μ follows from consideration of the diagram

$$\begin{array}{ccc} (\hom^{\Sigma}\ S\ T) \times (FT) & \xrightarrow{\mu T} & GT \\ \downarrow & & \downarrow \\ (\hom^{\Sigma}\ S\ \tau) \times (F\tau) & & G\tau \\ \downarrow & & \downarrow \\ (\hom^{\Sigma}\ S\ T') \times (FT') & \xrightarrow{\mu T'} & GT' \end{array} ,$$

where $\tau \in T \xrightarrow[\Sigma]{} T'$; the diagram commutes because $G\tau$ is strict. (This is why the definition of M involves $Sdom$ rather than $\mathcal{D}om$.) Thus

$$\mu \in (\hom^{\Sigma}\ S) \times F \xrightarrow[K]{} G.$$

The fact that μ is minimal in $(F \underset{K}{\Longrightarrow} G)\ S$ follows from the minimality of μT in $H\ F\ S\ T \underset{Pdom}{\Longrightarrow} GT$. Therefore we may legitimately write

$$\mu = \perp_{(F \Rightarrow G)\, S}.$$

Suppose $\sigma \in S \xrightarrow[\Sigma]{} S'$. We aim to show that $(F \underset{K}{\Longrightarrow} G)\ \sigma$ is strict. (We know it is continuous.)

Let $\tau \in S' \xrightarrow[\Sigma]{} T$, $t \in Ob\ (F\ T)$. Then

$$\begin{aligned} (F \underset{K}{\Longrightarrow} G)\ \sigma\ \perp_{(F \Rightarrow G)\, S\ T}\ \langle \tau, t \rangle & \\ &= \perp_{(F \Rightarrow G)\, S\ T}\ \langle \tau \circ \sigma,\ t \rangle, \text{ by definition of } (F \underset{K}{\Longrightarrow} G)\ \sigma, \\ &= \perp_{GT} \\ &= \perp_{(F \Rightarrow G)\, S'\ T}\ \langle \tau, t \rangle. \end{aligned}$$

Hence

$$(F \underset{K}{\Longrightarrow} G)\ \sigma\ \perp_{(F \Rightarrow G)\, S} = \perp_{(F \Rightarrow G)\, S'}.$$

We conclude $F \underset{K}{\Longrightarrow} G$ is an object of M, as desired. □

CHAPTER VI

A CATEGORY OF STORE SHAPES

So far we have been vague about the structure of Σ, the category of store shapes. Obviously, the semantics of a language can be precisely described only if Σ is completely specified. There are actually a variety of choices for Σ. As one approach, we may posit the existence of a set of "locations" along with operations for creating and manipulating "stacks of locations," which are the objects of Σ. This approach has a very operational flavor. It has two drawbacks. First, the details turn out to be very complicated, and, second, we are uncomfortable with insisting at the outset that stores are "stacks of locations." Another candidate for Σ is the category of store shapes and expansions described in [6]. It avoids the drawbacks of the operational approach, but it does not seem possible to give continuation semantics by using that category for Σ.

The point of this section is to describe a category Σ that we feel is at the proper level of abstraction. There is behind it an accessible intuition. At the same time we can avoid unnecessary complexities that may almost be said to be implementation-dependent (i.e., what kind of data type values can be placed in what locations, and what do we do with values that need several locations.)

The basic idea is to let <u>any</u> set X be an object of Σ (i.e. a store shape); then the set of stores corresponding to the store shape X is just X itself. An expansion in $X \xrightarrow[\Sigma]{} Y$ is supposed to go from a "little" store shape X to a "big" store shape Y. Part of an expansion is a "forgetting" function $\phi \in Y \to X$, which allows a more complicated store to serve as a less complicated one. Another part of an expansion is a "replacement" function $\rho \in X \to (Y \to Y)$; it is used when one wishes to overwrite the part of a more complicated store that corresponds to a less complicated store. Thus an arrow of Σ is an ordered pair

$$\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y,$$

where

$$\phi \in Y \to X \quad \text{and} \quad \rho \in X \to (Y \to Y).$$

Furthermore, we require that an arrow $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$ satisfy the following three conditions:

(Σ1) $\phi\ (\rho\ x\ y) = x$,

(Σ2) $\rho\ (\phi\ y)\ y = y$, and

(Σ3) $\rho\ x\ (\rho\ x'\ y) = \rho\ x\ y$,

where $x, x' \in X$ and $y \in Y$. We may read (Σ1) loosely as replacing with x in y and then forgetting gives back x. Condition (Σ2) says replacing in y with part of itself gives back y. Finally, (Σ3) says only the last replacement counts. We could obtain a larger and simpler category by

not imposing these three conditions on morphisms, and all of the results of this chapter would hold, with the exception of the theorem on the structure of morphisms. However, the demonstrations of naturality in Chapters VII and VIII seem to require (Σ1), (Σ2), and (Σ3) or some similar conditions.

In order to define composition elegantly in Σ we need some notation. Suppose $\alpha \in X \to Y$ and $\beta \in W \to Z$. Then

$$\alpha \to \beta \in (Y \to W) \to (X \to Z)$$

is given by

$$(\alpha \to \beta)\ f = \beta \circ f \circ \alpha,$$

where $f \in Y \to W$. (So $\to$ is just the ordinary hom functor for *Set*.) Also, for each set X, define the diagonalization D_X for X to be

$$D_X \in (X \to (X \to X)) \to (X \to X),$$

given by

$$D_X\ f\ x = f\ x\ x$$

where $f \in X \to (X \to X)$ and $x \in X$.

Suppose

$$\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y \text{ and } \langle \phi', \rho' \rangle \in Y \xrightarrow[\Sigma]{} Z.$$

Define

$$\langle \phi', \rho' \rangle \circ \langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Z$$

by

$$\langle \phi', \rho' \rangle \circ \langle \phi, \rho \rangle = \langle \phi \circ \phi', D_Z \circ (\phi' \to \rho') \circ \rho \rangle$$

Obtaining a new forgetting function by composing given forgetting functions is not surprising. The second component on the right-hand side is the functional composition

$$X \xrightarrow{\rho} (Y \to Y) \xrightarrow{\phi' \to \rho'} (Z \to (Z \to Z)) \xrightarrow{D_Z} (Z \to Z).$$

For $x \in X$, $z \in Z$, we have

$$\begin{aligned}(D_Z \circ (\phi' \to \rho') \circ \rho)\ x\ z &= D_Z\ ((\phi' \to \rho')\ (\rho\ x))\ z \\ &= D_Z\ (\rho' \circ (\rho\ x) \circ \phi')\ z \\ &= (\rho' \circ (\rho\ x) \circ \phi')z\ z \\ &= \rho'\ (\rho\ x\ (\phi' z))\ z.\end{aligned}$$

Thus, we have an intuitiviely satisfying replacement function derived from two expansions. Given that $\langle \phi, \rho \rangle$ and $\langle \phi', \rho' \rangle$ satisfy the three conditions, it is necessary to check that their composite also satisfies them. Let $x, x' \in X$ and $z \in Z$. For (Σ1), we have

$$\begin{aligned}&(\phi \circ \phi')\ ((D_Z \circ (\phi' \to \rho') \circ \rho)\ x\ z\ ) \\ &\quad = (\phi \circ \phi')\ (\rho'\ (\rho\ x\ (\phi'\ z))\ z) \\ &\quad = \phi\ (\phi'\ (\rho'\ (\rho\ x\ (\phi'\ z))\ z) \\ &\quad = \phi\ (\rho\ x\ (\phi'\ z)), \text{ by } (\Sigma 1) \text{ for } \langle \phi', \rho' \rangle. \\ &\quad = x, \text{ by } (\Sigma 1) \text{ for } \langle \phi, \rho \rangle.\end{aligned}$$

For ($\Sigma 2$), we compute

$$(D_Z \circ (\phi' \to \rho') \circ \rho)\ ((\phi \circ \phi')\ z)z$$

$$= (D_Z \circ (\phi' \to \rho') \circ \rho)\ (\phi\ (\phi'\ z))\ z$$

$$= \rho'\ (\rho\ (\phi\ (\phi'\ z))\ (\phi'\ z))\ z$$

$$= \rho'\ (\phi'\ z)\ z, \text{ by } (\Sigma 2) \text{ for } \langle \phi, \rho \rangle$$

$$= z, \text{ by } (\Sigma 2) \text{ for } \langle \phi', \rho' \rangle.$$

Finally, for ($\Sigma 3$), we have

$$(D_Z \circ (\phi' \to \rho') \circ \rho)\ x\ ((D_Z \circ (\phi' \to \rho') \circ \rho)\ x'\ z)$$

$$= (D_Z \circ (\phi' \to \rho') \circ \rho)\ x\ (\rho'\ (\rho\ x'\ (\phi'\ z))\ z)$$

$$= \rho'\ (\rho x\ (\phi'\ (\rho'\ (\rho x'\ (\phi'\ z))\ z)))\ (\rho'\ (\rho x'\ (\phi'\ z))\ z)$$

$$= \rho'(\rho\ x\ (\rho\ x'\ (\phi\ z)))\ \ (\rho'\ (\rho\ x'\ (\phi'\ z))\ z),$$

$$\text{by } (\Sigma 1) \text{ for } \langle \phi', \rho' \rangle,$$

$$= \rho'\ (\rho\ x\ (\phi'\ z))\ (\rho'\ (\rho x'\ (\phi'\ z))\ z),$$

$$\text{by } (\Sigma 3) \text{ for } \langle \phi, \rho \rangle,$$

$$= \rho'\ (\rho\ x\ (\phi'\ z))\ z, \text{ by } (\Sigma 3) \text{ for } \langle \phi', \rho' \rangle,$$

$$= (D_Z \circ (\phi' \to \rho') \circ \rho)\ x\ z.$$

Note the curious fact that both ($\Sigma 1$) and ($\Sigma 3$) play roles in the last computation.

We must check that composition in Σ is associative. Suppose

$$\langle \phi_1, \rho_1 \rangle \in X \xrightarrow[\Sigma]{} Y,\ \langle \phi_2, \rho_2 \rangle \in Y \xrightarrow[\Sigma]{} Z,\ \langle \phi_3, \rho_3 \rangle \in Z \xrightarrow[\Sigma]{} W.$$

First we need to verify that

$$D_W \circ (\phi_3 \to \rho_3) \circ D_Z = D_W \circ (\phi_3 \to (D_W \circ (\phi_3 \to \rho_3))) \ ;$$

both sides are in $(Z \to (Z \to Z)) \to (W \to W)$. Let $h \in Z \to (Z \to Z)$, $w \in W$, $w' \in W$. Then

$$\begin{aligned} ((\phi_3 \to \rho_3) \circ D_Z)\ h\ w\ w' &= (\phi_3 \to \rho_3)\ (D_Z\ h)\ w\ w' \\ &= (\rho_3 \circ (D_Z\ h) \circ \phi_3)\ w\ w' \\ &= \rho_3\ (D_Z\ h\ (\phi_3\ w))\ w' \\ &= \rho_3\ (h\ (\phi_3 w)(\phi_3 w))\ w' \ , \end{aligned}$$

and

$$\begin{aligned} (\phi_3 \to (D_W \circ (\phi_3 \to \rho_3)))\ h\ w\ w' &= (D_W \circ (\phi_3 \to \rho_3) \circ h \circ \phi_3)\ w\ w' \\ &= D_W\ ((\phi_3 \to \rho_3)(h\ (\phi_3 w)))\ w' \\ &= D_W\ (\rho_3 \circ (h\ (\phi_3 w)) \circ \phi_3)\ w' \\ &= (\rho_3 \circ (h\ (\phi_3 w)) \circ \phi_3)\ w'\ w' \\ &= \rho_3\ (h\ (\phi_3 w)\ (\phi_3 w'))\ w' \ . \end{aligned}$$

Therefore

$$\begin{aligned} (D_W \circ (\phi_3 \to \rho_3) \circ D_Z)\ h\ w &= \rho_3\ (h\ (\phi_3 w)\ (\phi_3 w))\ w \\ &= (D_W \circ (\phi_3 \to (D_W \circ (\phi_3 \to \rho_3))))\ h\ w, \end{aligned}$$

from which the desired equality follows. Now let's verify associativity:

$$(\langle \phi_3, \rho_3 \rangle \circ \langle \phi_2, \rho_2 \rangle) \circ \langle \phi_1, \rho_1 \rangle$$

$$= \langle \phi_2 \circ \phi_3, D_W \circ (\phi_3 \to \rho_3) \circ \rho_2 \rangle \circ \langle \phi_1, \rho_1 \rangle$$

$$= \langle \phi_1 \circ \phi_2 \circ \phi_3, D_W \circ ((\phi_2 \circ \phi_3) \to (D_W \circ (\phi_3 \to \rho_3) \circ \rho_2)) \circ \rho_1 \rangle$$

$$= \langle \phi_1 \circ \phi_2 \circ \phi_3, D_W \circ (\phi_3 \to (D_W \circ (\phi_3 \to \rho_3))) \circ (\phi_2 \to \rho_2) \circ \rho_1 \rangle,$$

since $\to$ is a functor,

$$= \langle \phi_1 \circ \phi_2 \circ \phi_3, D_W \circ (\phi_3 \to \rho_3) \circ D_Z \circ (\phi_2 \to \rho_2) \circ \rho_1 \rangle,$$

by the equality demonstrated earlier,

$$= \langle \phi_3, \rho_3 \rangle \circ \langle \phi_1 \circ \phi_2, D_Z \circ (\phi_2 \to \rho_2) \circ \rho_1 \rangle$$

$$= \langle \phi_3, \rho_3 \rangle \circ (\langle \phi_2, \rho_2 \rangle \circ \langle \phi_1, \rho_1 \rangle),$$

as desired.

The identity morphisms for Σ are given by

$$\langle 1_X, I_X \rangle \in X \xrightarrow[\Sigma]{} X,$$

where $1_X \in X \to X$ is the identity function and $I_X \in X \to (X \to X)$ is given by $I_X\ x\ x' = x$, for all $x, x' \in X$. The computations needed to prove this are very easy.

The morphisms in Σ have a mysterious quality about them. The following theorem dispels some of the mystery. If one knows what "split epimorphisms" are in, say, the category of right modules over a ring, then this theorem can be interpreted as saying the arrows in Σ are "split

surjections" in *Set*.

Theorem 6.1: A morphism $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$ induces an equivalence relation $\sim$ on Y such that

$$(Y, \phi \in Y \to X, \pi \in Y \to Y/\sim)$$

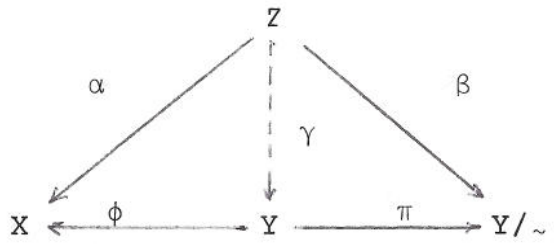
is a product in *Set* of X and $Y/\sim$, where $\pi \in Y \to Y/\sim$ maps each element of Y to its equivalence class. In particular, ϕ is surjective.

Proof: Define $\sim$ on Y by

$$y_1 \sim y_2 \quad \text{iff,} \quad \text{for all } x \in X, \ \rho \ x \ y_1 = \rho \ x \ y_2 \ .$$

Then $\sim$ is clearly an equivalence relation.

Consider in *Set* the diagram



We must show there exists a unique function γ making the diagram commute. Let $z \in Z$. Define γ by

$$\gamma \ z = \rho \ (\alpha \ z) \ y$$

where $y \in Y$ is any element such that $\pi \ y = \beta \ z$; that γ is well-defined is immediate from the definition of $\sim$. Compute:

$$\begin{aligned} (\phi \circ \gamma) z &= \phi \ (\gamma \ z) \\ &= \phi \ (\rho \ (\alpha \ z) \ y) \\ &= \alpha \ z, \text{ by } (\Sigma \ 1), \end{aligned}$$

and

$$(\pi \circ \gamma)\ z = \pi(\gamma\ z)$$

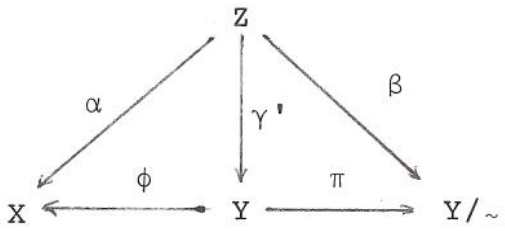
$$= \pi(\rho\ (\alpha\ z)\ y)$$

$$= \pi\ y$$

because, for all $x \in X$,

$$\rho\ x\ (\rho\ (\alpha\ z)\ y) = \rho\ x\ y, \text{ by } (\Sigma 3).$$

Hence γ makes the diagram commute. Suppose also that



commutes; we must show $\gamma = \gamma'$. Again, let $z \in Z$. Then $\gamma\ z \sim \gamma'\ z$ because

$$\pi(\gamma\ z) = \beta\ z = \pi\ (\gamma'\ z).$$

Therefore,

$$\gamma\ z = \rho(\phi\ (\gamma\ z))\ (\gamma\ z), \text{ by } (\Sigma 2),$$

$$= \rho(\alpha\ z)\ (\gamma\ z)$$

$$= \rho(\alpha\ z)\ (\gamma'\ z), \text{ since } \gamma\ z \sim \gamma'\ z,$$

$$= \rho(\phi\ (\gamma'\ z))\ (\gamma'\ z)$$

$$= \gamma'\ z, \text{ by } (\Sigma 2).$$

So $\gamma = \gamma'$, and we are done. □

CHAPTER VII

DESUGARED ALGOL: DIRECT SEMANTICS

In this chapter we will use the methods developed earlier to describe the semantics of a desugared version of that part of an ALGOL-like language which can be described without continuations.

Underlying an ALGOL-like language is a first-order data type language. Thus, there is a poset $\mathcal{D}$ of data types, a collection Op of operators, and two functions

$$\text{arity} \in \text{Op} \to (\text{Ob } \mathcal{D})^*, \quad \text{res} \in \text{Op} \to \text{Ob } \mathcal{D}.$$

Thus arity gives a finite sequence of data types for each operator, thereby specifying the data types to which an operator may meaningfully be applied, and res gives the data type that is the result of a meaningful application. This information amounts to a syntactic specification of the first-order data type language.

We also have in mind a specific semantics for these entities. Thus, we assume we are given a functor

$$\text{Val} \in \mathcal{D} \to \mathcal{S}et$$

and an appropriate interpretation for the operators in the form of m_{Op}, an Op-indexed collection of functions such that for each $g \in \text{Op}$,

$$m_{Op}\ g \in \text{Val } \delta_1 \times \cdots \times \text{Val } \delta_n \to \text{Val } \delta$$

where $\text{arity } g = \langle \delta_1, \cdots, \delta_n \rangle$ and $\text{res } g = \delta$.

For example, it may be that $\mathcal{D}$ is given by the Hasse diagram

and Op, arity, and res is given by the following table

g ε Op	arity g	res g
true	⟨⟩	bool
false	⟨⟩	bool
and	⟨bool, bool⟩	bool
0	⟨⟩	int
1	⟨⟩	int
π	⟨⟩	real
+	⟨real, real⟩	real
succ	int	int

It would then be reasonable for Val to take $\mathcal{D}$ to the diagram

and for the interpretations of the operators,

m_{Op} true ε $\{\langle\rangle\} \to \{true, false\}$

m_{Op} false ε $\{\langle\rangle\} \to \{true, false\}$

m_{Op} and ε $\{true, false\} \times \{true, false\} \to \{true, false\}$

m_{Op} 0 ε $\{\langle\rangle\} \to \mathbb{Z}$

m_{Op} 1 ε $\{\langle\rangle\} \to \mathbb{Z}$

m_{Op} π ε $\{\langle\rangle\} \to \mathbb{R}$

$$m_{Op}\ + \varepsilon\ \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

$$m_{Op}\ \text{succ}\ \varepsilon\ \mathbb{Z} \to \mathbb{Z},$$

to be the obvious ones.

An important idea is that for each data type $\delta \varepsilon \mathcal{D}$, Val δ is a set of storable values.

From $\mathcal{D}$ we get P, the poset of primitive phrase types. Let

$$\begin{aligned} \text{Ob } P = \{&\underline{\text{comm}}\} \\ &\cup \{\delta\text{-}\underline{\text{exp}} \mid \delta \varepsilon \mathcal{D}\} \\ &\cup \{\delta\text{-}\underline{\text{acc}} \mid \delta \varepsilon \mathcal{D}\} \\ &\cup \{\delta\text{-}\underline{\text{var}} \mid \delta \varepsilon \mathcal{D}\}. \end{aligned}$$

The partial order on P is given by

$\pi \leq_P \pi'$ iff $\pi = \pi'$,

or $\pi = \delta\text{-}\underline{\text{exp}}$, $\pi' = \delta'\text{-}\underline{\text{exp}}$, where $\delta \leq_{\mathcal{D}} \delta'$,

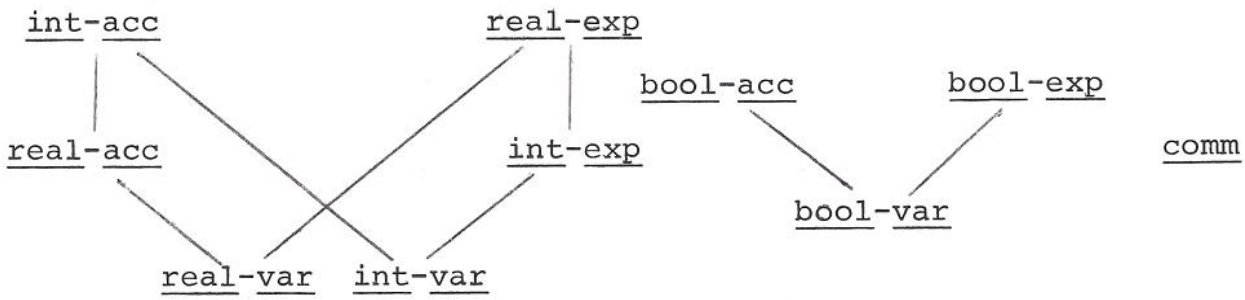
or $\pi = \delta'\text{-}\underline{\text{acc}}$, $\pi' = \delta\text{-}\underline{\text{acc}}$, where $\delta \leq_{\mathcal{D}} \delta'$,

or $\pi = \delta\text{-}\underline{\text{var}}$, $\pi' = \delta'\text{-}\underline{\text{exp}}$, where $\delta \leq_{\mathcal{D}} \delta'$,

or $\pi = \delta'\text{-}\underline{\text{var}}$, $\pi' = \delta\text{-}\underline{\text{acc}}$, where $\delta \leq_{\mathcal{D}} \delta'$.

Note that comm cannot be compared with the other primitive phrase types. For example, if $\mathcal{D}$ were given by

then P would be



For more discussion, see Reynolds [8]. (However, unlike Reynolds [8], we do not permit fully generic operations. As a consequence, the $\delta_1\delta_2$-variable construction is not needed.)

Now P generates the free type algebra T. Recall that Ob $|T|$ is the disjoint union

$$\{\underline{ns}\} \cup P \cup \{\tau \Rightarrow \theta \mid \tau, \theta \in T\}.$$

Once we are given Id, an infinite set of identifiers, we have A, the poset of phrase type assignments, which played an important role in the preceding development.

We view the language L (desugared ALGOL) as a free Λ-algebra generated by an appropriate set of G of linguistic constants. Recall that we must also give

$$\text{type} \in G \to \text{Ob } T,$$

so that G is really a typed set of linguistic constants. Take G to be the disjoint union of Op and Op' (to be defined shortly). For $g \in$ Op, let

$$\text{type } g = \delta_1\text{-}\underline{exp} \Rightarrow \delta_2\text{-}\underline{exp} \Rightarrow \ldots \Rightarrow \delta_n\text{-}\underline{exp} \Rightarrow \delta\text{-}\underline{exp},$$

where arity $g = \langle \delta_1, \ldots, \delta_n \rangle$ and res $g = \delta$. So, partly because product types are not in T, G contains curried versions of operators in Op. As we will see, this works out very neatly. The typed set Op' is given by the following

table, where $\delta \in \mathcal{D}$ is a data type and $\tau \in T$ is a phrase type.

$g \in Op'$	type g
skip	comm
;	comm ⇒ comm ⇒ comm
$:=_\delta$	δ-acc ⇒ δ-exp ⇒ comm
ifthenelse_τ	bool-exp ⇒ τ ⇒ τ ⇒ τ
rec_τ	$(\tau \Rightarrow \tau) \Rightarrow \tau$
newvar_δ	(δ-var ⇒ comm) ⇒ comm

It is the presence of Op' as a subset of G that entitles us to regard L as a desugared version of ALGOL. Intuitively, skip is the "do nothing" command, the operator ; concatenates commands, $:=_\delta$ is assignment for the data type δ, ifthenelse_τ is the conditional for the phrase type τ, rec_τ is used in the desugared denotation of recursively defined entities (e.g.

$$\underline{\text{while}}\ b\ \underline{\text{do}}\ c$$

is really

$$[\![\text{rec}_{\underline{\text{comm}}}\ [\![\lambda x{:}\underline{\text{comm}}.[\![\text{ifthenelse}_{\underline{\text{comm}}}\ b\ [\![c;x]\!]\ \text{skip}]\!]]\!]\!]),$$

and newvar_δ is used to desugar declarations of δ-variables (e.g.

$$\underline{\text{begin}}\ \delta\text{-}\underline{\text{var}}\ x\ ;\ c\ \underline{\text{end}}$$

is really

$$[\![\text{newvar}_\delta\ [\![\lambda x{:}\delta\text{-}\underline{\text{var}}.\ c]\!]\!]).$$

To give the semantics of this language we must do three things. First, we must give a Cartesian closed category and its associated canonical type algebra K. Second, we must give a functor $\text{mng} \in P \to |K|$, which by Theorem 2.2

can always be uniquely extended to a type algebra homomorphism Mng ε $T \xrightarrow{TypeAlg} K$. Finally, we must for each generator g ε G give its semantics, i.e.

$$\text{semf g} \varepsilon \text{ Env emp}_T \xrightarrow[|K|]{} \text{Mng (type g)}.$$

By the Fundamental Theorem of Semantics, this semantic function defined on generators uniquely determines a properly behaved semantic function defined on all program fragments.

Let Σ denote the category of store shapes described in Chapter VI. Let $|K|$ be the full subcategory of $\Sigma \Rightarrow Pdom$ whose objects are functors F ε $\Sigma \to Pdom$ such that

(1) F S is a domain for every S ε Ob Σ, and

(2) F ⟨ϕ,ρ⟩ is a strict, continuous function for every ⟨ϕ,ρ⟩ ε Ar Σ. In Chapter V we showed this category is Cartesian closed. Let K denote the corresponding type algebra. This takes care of the first, and simplest, part of giving the semantics of L.

We next embark on the description of the meaning functor

$$\text{mng} \varepsilon P \to |K|.$$

Therefore we must give functors

$$\left.\begin{array}{l} \text{mng } \underline{\text{comm}} \\ \text{mng } \delta\text{-}\underline{\text{exp}} \\ \text{mng } \delta\text{-}\underline{\text{acc}} \\ \text{mng } \delta\text{-}\underline{\text{var}} \end{array}\right\} \varepsilon \Sigma \to Pdom, \text{ where } \delta \varepsilon \mathcal{D},$$

and we must show these functors are actually objects of $|K|$ by verifying conditions (1) and (2) above. Before proceding, a definition is in order. If ρ ε X → (Y → Y), then

$\rho_\perp \in X_\perp \xrightarrow[Pdom]{} (Y \Rightarrow Y_\perp)$ is the strict extension of ρ, i.e.

for $x \in X_\perp$, $y \in Y$,

$$\rho_\perp\ x\ y = \begin{cases} \rho\ x\ y, & \text{if } x \neq \perp \\ \perp, & \text{if } x = \perp \end{cases}.$$

Intuitively, commands are possibly nonterminating transformations of the store. Thus, for $X \in Ob\ \Sigma$, let

$$mng\ \underline{comm}\ X = X \Rightarrow X_\perp.$$

For $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$, let

$$mng\ \underline{comm}\ \langle \phi, \rho \rangle \in (X \Rightarrow X_\perp) \xrightarrow[Pdom]{} (Y \Rightarrow Y_\perp)$$

be given by

$$mng\ \underline{comm}\ \langle \phi, \rho \rangle\ c\ y = \rho_\perp\ (c\ (\phi\ y))\ y,$$

where $c \in X \xrightarrow[Pdom]{} X_\perp$ and $y \in Y$. This last equation says that to view a command c, defined for a "little" store, as a command for a "big" store, do the following:

1) forget some of the "big" store via ϕ,
2) apply c,
3) overwrite the "big" store using the result of c, providing c terminates.

Note that we could almost write

$$mng\ \underline{comm}\ \langle \phi, \rho \rangle = D_Y \circ (\phi \to \rho_\perp),$$

but the domains and codomains don't quite match. We must check that $mng\ \underline{comm}\ \langle \phi, \rho \rangle$ is continuous. Suppose S is a directed subset of $X \Rightarrow X_\perp$. For all $y \in Y$,

$$\begin{aligned}
&\text{mng } \underline{\text{comm}} \ \langle \phi,\rho \rangle \ (\sqcup_{\text{mng } \underline{\text{comm}}\ X}\ S)\ y \\
&= \rho_{\perp}\ (\sqcup_{\text{mng } \underline{\text{comm}}\ X}\ S\ (\phi y))\ y \\
&= \rho_{\perp}\ (\sqcup_{X_{\perp}}\ (S\ (\phi y)))\ y \\
&= \sqcup_{\text{mng } \underline{\text{comm}}\ Y}\ (\rho_{\perp}\ (S\ (\phi y)))\ y, \\
&\qquad \text{since } \rho_{\perp} \text{ is continuous,} \\
&= \sqcup_{Y_{\perp}}\ (\rho_{\perp}\ (S\ (\phi y))\ y) \\
&= \sqcup_{Y_{\perp}}\ (\text{mng } \underline{\text{comm}}\ \langle \phi,\rho \rangle\ S\ y) \\
&= \sqcup_{\text{mng } \underline{\text{comm}}\ Y}\ (\text{mng } \underline{\text{comm}}\ \langle \phi,\rho \rangle\ S)\ y.
\end{aligned}$$

Therefore

$$\begin{aligned}
&\text{mng } \underline{\text{comm}} \ \langle \phi,\rho \rangle \ (\sqcup_{\text{mng } \underline{\text{comm}}\ X}\ S) \\
&= \sqcup_{\text{mng } \underline{\text{comm}}\ Y}\ (\text{mng } \underline{\text{comm}}\ \langle \phi,\rho \rangle\ S),
\end{aligned}$$

which implies mng $\underline{\text{comm}}$ $\langle \phi,\rho \rangle$ is continuous. Also, we must check that mng $\underline{\text{comm}}$ is truly a functor. It is trivial to verify that mng $\underline{\text{comm}}$ preserves identity morphisms. Suppose

$$\langle \phi,\rho \rangle \in X \xrightarrow[\Sigma]{} Y, \quad \langle \phi',\rho' \rangle \in Y \xrightarrow[\Sigma]{} Z.$$

We must establish

$$\begin{aligned}
\text{mng } \underline{\text{comm}}\ (\langle \phi', \rho' \rangle \circ \langle \phi,\rho \rangle) &= (\text{mng } \underline{\text{comm}}\ \langle \phi',\rho' \rangle) \\
&\quad \circ (\text{mng } \underline{\text{comm}}\ \langle \phi,\rho \rangle).
\end{aligned}$$

Let $c \in X \xrightarrow[Pdom]{} X_{\perp}$ and $z \in Z$. Then

$$\text{mng } \underline{\text{comm}}\ (\langle \phi', \rho' \rangle \circ \langle \phi,\rho \rangle)\ c\ z$$

$$= \text{mng } \underline{\text{comm}} \langle \phi\circ\phi', D_Z \circ (\phi'\to\rho')\circ\rho\rangle \; c \; z$$

$$= (D_Z \circ (\phi'\to\rho')\circ\rho)_\perp \; (c \; (\phi \; (\phi' z))) \; z$$

$$= \begin{cases} \perp \;, & \text{if } c \; (\phi \; (\phi' z)) = \perp \;, \\ D_Z \; ((\phi'\to\rho') \; (\rho \; (c \; (\phi \; (\phi' z))))) \; z, & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \perp \;, & \text{if } c \; (\phi \; (\phi' z)) = \perp \;, \\ ((\phi'\to\rho') \; (\rho \; (c \; (\phi \; (\phi' z))))) \; z \; z, & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \perp \;, & \text{if } c \; (\phi \; (\phi' z)) = \perp \;, \\ (\rho' \circ (\rho \; (c \; (\phi(\phi' z)))) \circ \phi') \; z \; z, & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \perp \;, & \text{if } c \; (\phi \; (\phi' z)) = \perp \;, \\ \rho' \; (\rho \; (c \; (\phi \; (\phi' z))) \; (\phi' z)) \; z, & \text{otherwise,} \end{cases}$$

$$= \rho'_\perp \; (\rho_\perp \; (c \; (\phi \; (\phi' z))) \; (\phi' z)) \; z$$

$$= \rho'_\perp \; (\text{mng } \underline{\text{comm}} \langle \phi,\rho\rangle \; c \; (\phi' z)) \; z$$

$$= \text{mng } \underline{\text{comm}} \langle \phi, \rho'\rangle \; (\text{mng } \underline{\text{comm}} \langle \phi,\rho\rangle \; c) \; z$$

$$= (\text{mng } \underline{\text{comm}} \langle \phi', \rho'\rangle) \circ (\text{mng } \underline{\text{comm}} \langle\phi,\rho\rangle) \; c \; z,$$

as desired. A moment's thought shows that

(1) for $X \in \text{Ob } \Sigma$, mng comm X is a domain, and

(2) for $\langle \phi,\rho\rangle \in \text{Ar } \Sigma$, mng comm $\langle \phi,\rho\rangle$ is strict.

Thus mng comm is an object of $|K|$.

Suppose $\delta \in \mathcal{D}$ is a data type. Our intuition about δ-expressions is that they try to produce elements of Val δ from the current store. Thus the functor mng δ-exp $\in \Sigma \to Pdom$ is defined on an object X of Σ by

$$\text{mng } \delta\text{-}\underline{\text{exp}} \; X = X \Rightarrow (\text{Val } \delta)_\perp$$

and on an arrow $\langle \phi,\rho\rangle : X \xrightarrow[\Sigma]{} Y$ by

$$\text{mng } \delta\text{-}\underline{\text{exp}} \; \langle \phi,\rho\rangle : (X \Rightarrow (\text{Val } \delta)_\perp) \xrightarrow[Pdom]{} (Y \Rightarrow (\text{Val}\delta)_\perp) \;,$$

and

$$\text{mng } \delta\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle \ e \ y = e \ (\phi y) \ ,$$

where $e \in X \xrightarrow[Pdom]{} (\text{Val } \delta)_\perp$ and $y \in Y$. Note that we use $(\text{Val } \delta)_\perp$, rather than Val δ, because the evaluation of an expression does not necessarily terminate. It is clear that if mng δ-exp is a functor, then it satisfies (1) and (2) and is therefore an object of $|K|$. The neatest way to see that mng δ-exp is indeed a functor is to consider the forgetful functor

$$U \in \Sigma \longrightarrow Pdom^{op},$$

given by

$$UX = X \text{ for } X \in \text{Ob}\Sigma, \ U\langle \phi, \rho \rangle = \phi \text{ for } \langle \phi, \rho \rangle \in \text{Ar } \Sigma,$$

and then to notice that

$$\text{mng } \delta\text{-}\underline{\text{exp}} = (U -) \Rightarrow (\text{Val } \delta)_\perp$$

where the heavy arrow is the internal hom functor for $Pdom$.

Next, we think about δ-acceptors. They take elements of Val δ (perhaps from the right-hand side of an assignment command) and produce transformations of the store. Therefore, for a store shape $X \in \Sigma$, let

$$\text{mng } \delta\text{-}\underline{\text{acc}} \ X = \text{Val } \delta \Rightarrow \text{mng } \underline{\text{comm}} \ X,$$

and, for $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$, let

$$\text{mng } \delta\text{-}\underline{\text{acc}} \langle \phi, \rho \rangle \in (\text{Val}\delta \Rightarrow \text{mng } \underline{\text{comm}} \ X) \xrightarrow[Pdom]{} (\text{Val } \delta \Rightarrow \text{mng } \underline{\text{comm}} \ Y)$$

be given by

$$\text{mng } \delta\text{-}\underline{\text{acc}} \langle \phi, \rho \rangle \ a \ v = \text{mng } \underline{\text{comm}} \langle \phi, \rho \rangle \ (a \ v).$$

If we rewrite this definition as

$$\text{mng } \delta\text{-}\underline{\text{acc}} = \text{Val } \delta \Rightarrow (\text{mng } \underline{\text{comm}} -),$$

where the heavy arrow is the internal hom functor for $\mathcal{P}dom$, it becomes clear that mng δ-$\underline{\text{acc}}$ is a functor. Once again, it is clear that mng δ-$\underline{\text{acc}}$ is an object of $|K|$.

Finally, δ-variables must have in them enough structure to be used as either δ-acceptors or δ-expressions. Therefore, let

$$\text{mng } \delta\text{-}\underline{\text{var}} = \text{mng } \delta\text{-}\underline{\text{acc}} \times \text{mng } \delta\text{-}\underline{\text{exp}},$$

where the $\times$ is the distinguished binary product in $|K|$. In particular, notice that for a store shape $X \in \text{Ob } \Sigma$

$$\text{mng } \delta\text{-}\underline{\text{var}}\ X = \text{mng } \delta\text{-}\underline{\text{acc}}\ X \times \text{mng } \delta\text{-}\underline{\text{exp}}\ X.$$

So far, we have defined mng on objects of $\mathcal{P}$. Next, for each $\pi \leq_{\mathcal{P}} \pi'$, we must give a natural transformation

$$\text{mng } (\pi \leq_{\mathcal{P}} \pi') \in \text{mng } \pi \xrightarrow[|K|]{} \text{mng } \pi'.$$

There are five cases, corresponding to the five kinds of implicit conversions which comprise the arrows of $\mathcal{P}$.

(1) If $\pi = \pi'$, then let mng $(\pi \leq_{\mathcal{P}} \pi)$ be the identity natural transformation on the functor mng π.

(2) The construction

$$\text{mng } \delta\text{-}\underline{\text{exp}} = (-) \Rightarrow (\text{Val } \delta)_{\perp}$$

is functorial in δ. Thus, if $\delta \leq_{\mathcal{D}} \delta'$, then we obtain a natural transformation.

$$\text{mng } (\delta\text{-}\underline{\text{exp}} \leq \delta'\text{-}\underline{\text{exp}}) \in \text{mng } \delta\text{-}\underline{\text{exp}} \xrightarrow[|K|]{} \text{mng } \delta'\text{-}\underline{\text{exp}}$$

by letting

$$\text{mng } (\delta\text{-}\underline{\text{exp}} \leq \delta'\underline{\text{exp}}) = (-) \Rightarrow (\text{Val } (\delta \leq_{\mathcal{D}} \delta'))_{\perp}$$

Therefore, if $X \varepsilon$ Ob Σ and $e \varepsilon X \xrightarrow[Pdom]{} (\text{Val } \delta)_\perp$, then

$$\text{mng } (\delta\text{-}\underline{\text{exp}} \leq \delta'\underline{\text{exp}})\ X \varepsilon X \Rightarrow (\text{Val } \delta)_\perp \xrightarrow[Pdom]{} X \Rightarrow (\text{Val } \delta')_\perp$$

and

$$\text{mng } (\delta\text{-}\underline{\text{exp}} \leq \delta'\text{-}\underline{\text{exp}})\ X\ e = (\text{Val } (\delta \leq \delta'))_\perp \circ e.$$

(3) Another construction functorial in δ (but contravariant!) is

$$\text{mng } \delta\text{-}\underline{\text{acc}} = \text{Val } \delta \Rightarrow (\text{mng } \underline{\text{comm}}\ -);$$

hence, if $\delta \leq_{\mathcal{D}} \delta'$, then we can get a natural transformation

$$\text{mng } (\delta'\text{-}\underline{\text{acc}} \leq_P \delta\text{-}\underline{\text{acc}}) \varepsilon \text{ mng } \delta'\text{-}\underline{\text{acc}} \xrightarrow[|K|]{} \text{mng } \delta\text{-}\underline{\text{acc}}$$

by letting

$$\text{mng } (\delta'\text{-}\underline{\text{acc}} \leq_P \delta\text{-}\underline{\text{acc}}) = (\text{Val } (\delta \leq_{\mathcal{D}} \delta') \Rightarrow (\text{mng } \underline{\text{comm}}\ -).$$

In this situation, if $X \varepsilon$ Ob Σ and $a \varepsilon \text{ Val } \delta' \xrightarrow[Pdom]{} \text{mng } \underline{\text{comm}}\ X$, then

$$\text{mng } (\delta'\text{-}\underline{\text{acc}} \leq_P \delta\text{-}\underline{\text{acc}})\ X \varepsilon \text{ mng } \delta'\text{-}\underline{\text{acc}}\ X \xrightarrow[Pdom]{} \text{mng } \delta\text{-}\underline{\text{acc}}\ X$$

and

$$\text{mng } (\delta'\text{-}\underline{\text{acc}} \leq_P \delta\text{-}\underline{\text{acc}})\ X\ a = a \circ \text{Val}(\delta \leq_{\mathcal{D}} \delta').$$

(4) If $\delta \leq_{\mathcal{D}} \delta'$, then define

$$\text{mng } (\delta\text{-}\underline{\text{var}} \leq_P \delta\text{-}\underline{\text{exp}}) \varepsilon \text{ mng } \delta\text{-}\underline{\text{var}} \xrightarrow[|K|]{} \text{mng } \delta\text{-}\underline{\text{exp}}$$

to be

$$\text{mng } (\delta\text{-}\underline{\text{var}} \leq_P \delta\text{-}\underline{\text{exp}}) = \text{mng } (\delta\text{-}\underline{\text{exp}} \leq_P \delta'\underline{\text{exp}}) \circ \text{proj}_2$$

where

$$\text{proj}_2 \varepsilon \text{ mng } \delta\text{-}\underline{\text{var}} = \text{mng } \delta\text{-}\underline{\text{acc}} \times \text{mng } \delta\text{-}\underline{\text{exp}} \xrightarrow[|K|]{} \text{mng } \delta\text{-}\underline{\text{exp}}$$

is the projection onto the second component.

(5) If $\delta \leq_{\mathcal{D}} \delta'$, then define

$$\text{mng } (\delta'\text{-}\underline{\text{var}} \leq_P \delta\text{-}\underline{\text{acc}}) \in \text{mng } \delta'\text{-}\underline{\text{var}} \xrightarrow[|K|]{} \text{mng } \delta\text{-}\underline{\text{acc}}$$

to be

$$\text{mng } (\delta'\text{-}\underline{\text{var}} \leq_P \delta\text{-}\underline{\text{acc}}) = \text{mng } (\delta'\text{-}\underline{\text{acc}} \leq_P \delta\text{-}\underline{\text{acc}}) \circ \text{proj}_1$$

where

$$\text{proj}_1 \in \text{mng } \delta'\text{-}\underline{\text{var}} = \text{mng } \delta'\text{-}\underline{\text{acc}} \times \text{mng } \delta'\text{-}\underline{\text{exp}} \xrightarrow[|K|]{} \text{mng } \delta'\text{-}\underline{\text{acc}}$$

is the projection onto the first component. An easy argument shows that we really have defined a functor

$$\text{mng} \in P \to |K|.$$

As we noted earlier, there is a unique type algebra homomorphism

$$\mathbf{Mng} \in T \to K$$

that extends mng.

The rest of this chapter is devoted to giving the semantics of the generating set G of linguistic constants.

Let $g \in \text{Op}$ with $\text{arity } g = \langle \delta_1, \ldots, \delta_n \rangle$, $\text{res } g = \delta$. We must define

$$\text{semf } g \in \text{Env emp}_T \xrightarrow[|K|]{} (\text{Mng } \delta_1\text{-}\underline{\text{exp}} \Rightarrow \ldots \Rightarrow \text{Mng } \delta_n\text{-}\underline{\text{exp}} \Rightarrow \text{Mng } \delta\text{-}\underline{\text{exp}}).$$

The codomain of this natural transformation is a rather formidable functor. However, since $|K|$ is a Cartesian closed category, there is a canonical bijection between

$$\text{Env emp}_T \xrightarrow[|K|]{} (\text{Mng } \delta_1\text{-}\underline{\text{exp}} \Rightarrow \ldots \Rightarrow \text{Mng } \delta_n\text{-}\underline{\text{exp}} \Rightarrow \text{Mng } \delta\text{-}\underline{\text{exp}})$$

and

$$(\text{Mng } \delta_1\text{-}\underline{\text{exp}} \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}}) \xrightarrow[|K|]{} \text{Mng } \delta\text{-}\underline{\text{exp}}$$

based on repeated abstraction and the fact that the product with a terminal object (Env $\text{emp}_{\mathcal{T}}$ in this case) is naturally isomorphic to the identity. So it suffices to give the corresponding

$$\hat{g} \in (\text{Mng } \delta_1\text{-}\underline{\text{exp}} \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}}) \xrightarrow[|K|]{} \text{Mng } \delta\text{-}\underline{\text{exp}}.$$

For $X \in \text{Ob } \Sigma$, let

$$\hat{g}\ X \in \text{Mng } \delta_1\text{-}\underline{\text{exp}}\ X \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}}\ X \xrightarrow[Pdom]{} \text{Mng } \delta\text{-}\underline{\text{exp}}\ X$$

be given by

$$\hat{g}\ X \langle e_1, \ldots, e_n \rangle\ x = \begin{cases} m_{Op}\ g \langle e_1 x, \ldots, e_n x \rangle & \text{if, for all } i,\ e_i x \neq \perp \\ \perp & \text{if, for some } i,\ e_i x = \perp, \end{cases}$$

where $x \in X$ and, for each i, $e_i \in X \xrightarrow[Pdom]{} (\text{Val } \delta_i)_\perp$.

It is easy to see that $\hat{g}\ X$ is continuous. We must verify that $\hat{g}$ is a natural transformation. To this end, suppose $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$, and consider the diagram

$$\begin{array}{ccc} \text{Mng } \delta_1\text{-}\underline{\text{exp}}\ X \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}}\ X & \xrightarrow{\hat{g}\ X} & \text{Mng } \delta\text{-}\underline{\text{exp}}\ X \\ \downarrow & & \downarrow \\ \text{Mng } \delta_1\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle & & \text{Mng } \delta\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle \\ \downarrow & & \downarrow \\ \text{Mng } \delta_1\text{-}\underline{\text{exp}}\ Y \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}}\ Y & \xrightarrow{\hat{g}\ Y} & \text{Mng } \delta\text{-}\underline{\text{exp}}\ Y. \end{array}$$

Let $e_i \in X \xrightarrow[Pdom]{} (\text{Val } \delta_i)_\perp$ for $i = 1, \ldots, n$, and let $y \in Y$. Then

$$(\hat{g}\ Y \circ (\text{Mng } \delta_1\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle \times \ldots \times \text{Mng } \delta_n\text{-}\underline{\text{exp}} \langle \phi, \rho \rangle))\ \langle e_1, \ldots, e_n \rangle\ y$$

$$= \hat{g}\ Y\ \langle \text{Mng } \delta_1\text{-}\underline{exp}\ \langle \phi,\rho\rangle\ e_1,\ \ldots,\ \text{Mng } \delta_n\text{-}\underline{exp}\ \langle \phi,\rho\rangle\ e_n\rangle\ y$$

$$= \begin{cases} m_{Op}\, g\, \langle e_1\ (\phi\ y),\ \ldots\ e_n\ (\phi\ y)\rangle & \text{if all } e_i\ (\phi\ y) \neq \perp \\ \perp & \text{if some } e_i\ (\phi\ y) = \perp \end{cases}$$

$$= \hat{g}\ X\ \langle e_1,\ \ldots,\ e_n\rangle\ (\phi\ y)$$

$$= \text{Mng } \delta\text{-}\underline{exp}\ \langle \phi,\rho\rangle\ (\hat{g}\ X\ \langle e_1,\ \ldots,\ e_n\rangle)\ y$$

$$= (\text{Mng } \delta\text{-}\underline{exp}\ \langle \phi,\rho\rangle \circ \hat{g}\ X)\ \langle e_1,\ \ldots,\ e_n\rangle\ y,$$

as desired.

Consider the "do nothing" command skip. We must give

$$\text{semf skip} \quad \varepsilon\ \text{Env emp}_T \xrightarrow[|K|]{} \text{Mng } \underline{comm}.$$

Let $X \varepsilon$ Ob Σ. Recall that Env emp_T X is the one-point predomain $\{\langle\rangle\}$ and Mng $\underline{comm}$ X = X $\Rightarrow$ $X_\perp$. Then

$$\text{semf skip X} \varepsilon\ \text{Env emp}_T\ X \xrightarrow[Pdom]{} \text{Mng } \underline{comm}\ X$$

is given by

$$\text{semf skip X } \langle\rangle x = x, \text{ for all } x \varepsilon X.$$

Clearly, then, skip does nothing to the current store x. Obviously, semf skip X is continuous. To see that semf skip is a natural transformation, let $\langle \phi,\rho\rangle \varepsilon\ X \xrightarrow[\Sigma]{} Y$. In *Pdom* consider the diagram

$$\begin{array}{ccc} \text{Env emp}_T\ X & \xrightarrow{\text{semf skip X}} & \text{Mng } \underline{comm}\ X \\ \| & & \downarrow \text{Mng } \underline{comm}\ \langle \phi,\rho\rangle \\ \text{Env emp}_T\ Y & \xrightarrow{\text{semf skip Y}} & \text{Mng } \underline{comm}\ Y. \end{array}$$

We compute, for $y \varepsilon Y$,

$$
\begin{aligned}
&(\text{Mng } \underline{\text{comm}} \langle \phi, \rho \rangle \circ \text{semf skip X}) \langle\rangle \text{ y} \\
&\quad = \text{Mng } \underline{\text{comm}} \langle \phi, \rho \rangle \text{ (semf skip X } \langle\rangle) \text{ y} \\
&\quad = \rho_{\perp} \text{ (semf skip X } \langle\rangle \text{ } (\phi \text{ y})) \text{ y} \\
&\quad = \rho_{\perp} \text{ } (\phi \text{ y}) \text{ y} \\
&\quad = \text{y}.
\end{aligned}
$$

The last equality follows from condition ($\Sigma 2$) in the definition of the category Σ of store shapes.

We wish to give

$$\text{semf} \; ; \varepsilon \; \text{Env emp}_{\mathcal{I}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}} \Rightarrow \text{Mng } \underline{\text{comm}} \Rightarrow \text{Mng } \underline{\text{comm}},$$

Using the Cartesian closed category structure of $|K|$, it suffices to define the corresponding natural transformation

$$\text{sc} \; \varepsilon \; \text{Mng } \underline{\text{comm}} \times \text{Mng } \underline{\text{comm}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}$$

For each store shape $X \varepsilon \text{ Ob } \Sigma$, we must give

$$\text{sc X} \varepsilon \; \text{Mng } \underline{\text{comm}} \; X \times \text{Mng } \underline{\text{comm}} \; X \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}} \; X$$

Recall that

$\text{Mng } \underline{\text{comm}} \; X = X \Rightarrow X_{\perp}$. For $f \varepsilon X \xrightarrow[Pdom]{} X_{\perp}$, let $f_{\perp} \varepsilon X_{\perp} \xrightarrow[Pdom]{} X_{\perp}$ be the strict extension of f. Now let

$$\text{sc X} \langle c, c' \rangle = c'_{\perp} \circ c,$$

where $c, c' \varepsilon X \xrightarrow[Pdom]{} X_{\perp}$. Note the order in which the commands are composed; it is determined by the operational rule that says (c;c') means execute c and then execute c'. (Of course, this is direct, not continuation, semantics.) Continuity of sc X follows from the continuity of composition. We must check that sc is indeed a natural transformation. Suppose $\langle \phi, \rho \rangle \varepsilon X \xrightarrow[\Sigma]{} Y$. We wish to show that

$$\begin{array}{ccc} \bar{M}ng\ \underline{comm}\ X \times Mng\ \underline{comm}\ X & \xrightarrow{sc\ X} & Mng\ \underline{comm}\ X \\ \downarrow{\scriptstyle Mng\ \underline{comm}\ \langle \phi,\rho\rangle \times Mng\ \underline{comm}\ \langle \phi\ \rho\rangle} & & \downarrow{\scriptstyle Mng\ \underline{comm}\ \langle \phi,\rho\rangle} \\ Mng\ \underline{comm}\ Y \times Mng\ \underline{comm}\ Y & \xrightarrow{sc\ Y} & Mng\ \underline{comm}\ Y \end{array}$$

commutes. Let $c, c' \in Mng\ \underline{comm}\ X$ and $y \in Y$. Then

$$(sc\ Y \circ (Mng\ \underline{comm}\ \langle \phi,\rho\rangle \times Mng\ \underline{comm}\ \langle \phi,\rho\rangle))\ \langle c, c'\rangle\ y$$

$$= sc\ Y\ \langle Mng\ \underline{comm}\ \langle \phi,\rho\rangle\ c, Mng\ \underline{comm}\ \langle \phi\ \rho\rangle\ c'\rangle\ y$$

$$= (Mng\ \underline{comm}\ \langle \phi,\rho\rangle\ c')_{\perp}\ (Mng\ \underline{comm}\ \langle \phi,\rho\rangle\ c\ y)$$

$$= (Mng\ \underline{comm}\ \langle \phi,\rho\rangle\ c')_{\perp}\ (\rho_{\perp}\ (c\ (\phi\ y))\ y)$$

$$= \begin{cases} \perp, \text{ if } c\ (\phi\ y) = \perp \\ (Mng\ \underline{comm}\ \langle \phi,\rho\rangle\ c')_{\perp}\ (\rho\ (c\ (\phi\ y))\ y), \text{ otherwise} \end{cases}$$

$$= \begin{cases} \perp, \text{ if } c\ (\phi\ y) = \perp \\ \rho_{\perp}\ (c'\ (\phi\ (\rho\ (c\ (\phi\ y))\ y)))\ (\rho\ (c\ (\phi\ y))\ y), \text{ otherwise} \end{cases}$$

$$= \begin{cases} \perp, \text{ if } c\ (\phi\ y) = \perp \\ \rho_{\perp}(c'\ (c\ (\phi\ y)))\ (\rho\ (c\ (\phi\ y))\ y), \text{ otherwise, by } (\Sigma 1), \end{cases}$$

$$= \begin{cases} \perp, \text{ if } c(\phi\ y) = \perp \text{ or } c'\ (c(\phi\ y)) = \perp, \\ \rho\ (c'\ (c\ (\phi\ y)))\ (\rho\ (c\ (\phi\ y))\ y), \text{ otherwise} \end{cases}$$

$$= \begin{cases} \perp \text{ if } c(\phi\ y) = \perp \text{ or } c'\ (c\ (\phi\ y)) = \perp, \\ \rho\ (c'\ (c\ (\phi\ y)))\ y, \text{ otherwise, by } (\Sigma 3), \end{cases}$$

$$= \rho_{\perp}\ (c'_{\perp}\ (c\ (\phi\ y)))\ y$$

$$= \rho_{\perp}\ (sc\ X\ \langle c, c'\rangle\ (\phi\ y))\ y$$

$$= Mng\ \underline{comm}\ \langle \phi\ \rho\rangle\ (sc\ X\ \langle c, c'\rangle)\ y$$

$$= (Mng\ \underline{comm}\ \langle \phi,\rho\rangle \circ sc\ X)\ \langle c, c'\rangle\ y,$$

which demonstrates commutativity. Note the usage of the fact that $\langle \phi,\rho\rangle$ satisfies $(\Sigma 1)$ and $(\Sigma 3)$.

Next we desire to define, for a data type δ,

$$\text{semf} :=_\delta \varepsilon \text{ Env emp}_T \xrightarrow[|K|]{} \text{Mng } \delta\text{-}\underline{\text{acc}} \Rightarrow \text{Mng } \delta\text{-}\underline{\text{exp}} \Rightarrow \text{Mng } \underline{\text{comm}}.$$

Because of the canonical bijection between

$$\text{Env emp}_T \xrightarrow[|K|]{} \text{Mng } \delta\text{-}\underline{\text{acc}} \Rightarrow \text{Mng } \delta\text{-}\underline{\text{exp}} \Rightarrow \text{Mng } \underline{\text{comm}}$$

and

$$\text{Mng } \delta\text{-}\underline{\text{acc}} \times \text{Mng } \delta\text{-}\underline{\text{exp}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}},$$

it is adequate to define the corresponding natural transformation

$$\text{assign}_\delta \varepsilon \text{ Mng } \delta\text{-}\underline{\text{acc}} \times \text{Mng } \delta\text{-}\underline{\text{exp}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}.$$

Suppose $X \varepsilon$ Ob Σ. We must give

$$\text{assign}_\delta \; X \varepsilon \text{ Mng } \delta\text{-}\underline{\text{acc}} \; X \times \text{Mng } \delta\text{-}\underline{\text{exp}} \; X \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}} \; X.$$

Recall

$$\text{Mng } \delta\text{-}\underline{\text{acc}} \; X = \text{Val } \delta \Rightarrow \text{Mng } \underline{\text{comm}} \; X,$$

$$\text{Mng } \delta\text{-}\underline{\text{exp}} \; X = X \Rightarrow (\text{Val } \delta)_\perp,$$

$$\text{Mng } \underline{\text{comm}} \; X = X \Rightarrow X_\perp.$$

For $a \varepsilon \text{ Val } \delta \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}} \; X$, $e \varepsilon X \xrightarrow[Pdom]{} (\text{Val } \delta)_\perp$,

and $x \varepsilon X$, let

$$\text{assign}_\delta \; X \langle a, e \rangle \; x = a_\perp \; (e \; x) \; x,$$

where $a_\perp$ is the strict extension of a. This definition reflects the idea that $(a :=_\delta e)$ means get a value from the store with e, then alter that same store with a.

The proof that assign_δ X is continuous is straightforward. We would like to check that assign_δ is a natural transformation. Let $\langle \phi, \rho \rangle \varepsilon X \xrightarrow[\Sigma]{} Y$. The diagram whose commutativity we must verify is

$$\begin{array}{ccc} \text{Mng } \delta\text{-}\underline{\text{acc}}\ X \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ X & \xrightarrow{\text{assign}_\delta\ X} & \text{Mng } \underline{\text{comm}}\ X \\ \downarrow & & \downarrow \\ \text{Mng } \delta\text{-}\underline{\text{acc}}\ \langle \phi,\rho\rangle \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ \langle \phi,\rho\rangle & & \text{Mng } \underline{\text{comm}}\ \langle \phi,\rho\rangle \\ \downarrow & \xrightarrow{\text{assign}_\delta\ Y} & \downarrow \\ \text{Mng } \delta\text{-}\underline{\text{acc}}\ Y \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ Y & \longrightarrow & \text{Mng } \underline{\text{comm}}\ Y \end{array}$$

Let $a \in \text{Val } \delta \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}}\ X$, $e \in X \xrightarrow[Pdom]{} (\text{Val } \delta)_\perp$,

and $y \in Y$. Then

$$\begin{aligned} &(\text{assign}_\delta\ Y \circ (\text{Mng } \delta\text{-}\underline{\text{acc}}\ \langle \phi,\rho\rangle \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ \langle \phi,\rho\rangle))\ \langle a,e\rangle\ y \\ &= \text{assign}_\delta\ Y\ \langle \text{Mng } \delta\text{-}\underline{\text{acc}}\ \langle \phi,\rho\rangle\ a,\ \text{Mng } \delta\text{-}\underline{\text{exp}}\ \langle \phi,\rho\rangle\ e\rangle\ y \\ &= (\text{Mng } \delta\text{-}\underline{\text{acc}}\ \langle \phi,\rho\rangle\ a)_\perp\ (\text{Mng } \delta\text{-}\underline{\text{exp}}\ \langle \phi,\rho\rangle\ e\ y)\ y \\ &= (\text{Mng } \delta\text{-}\underline{\text{acc}}\ \langle \phi\ \rho\rangle\ a)_\perp\ (e\ (\phi\ y))\ y \\ &= \begin{cases} \text{Mng } \underline{\text{comm}}\ \langle \phi,\rho\rangle\ (a\ (e\ (\phi\ y)))\ y & \text{if } e\ (\phi\ y) \neq \perp \\ \perp & \text{if } e\ (\phi\ y) = \perp \end{cases} \\ &= \begin{cases} \rho_\perp\ (a\ (e\ (\phi\ y))\ (\phi\ y))\ y & \text{if } e\ (\phi\ y) \neq \perp \\ \perp & \text{if } e\ (\phi\ y) = \perp \end{cases} \\ &= \rho_\perp\ (a_\perp\ (e\ (\phi\ y))\ (\phi\ y))\ y \\ &= \rho_\perp\ (\text{assign}_\delta\ X\ \langle a,e\rangle\ (\phi\ y))\ y \\ &= \text{Mng } \underline{\text{comm}}\ \langle \phi\ \rho\rangle\ (\text{assign}_\delta\ X\ \langle a,e\rangle)\ y \\ &= (\text{Mng } \underline{\text{comm}}\ \langle \phi,\rho\rangle \circ \text{assign}_\delta\ X)\ \langle a,e\rangle\ y, \end{aligned}$$

so that the diagram commutes.

The next generator on the list is the conditional for type τ; thus we seek

$$\text{semf ifthenelse}_\tau \in \text{Env emp}_T \xrightarrow[|K|]{} \text{Mng } \underline{\text{bool-exp}} \Rightarrow \text{Mng } \tau \Rightarrow \text{Mng } \tau \Rightarrow \text{Mng } \tau,$$

and it suffices to give the corresponding natural transformation in

$$\text{Mng } \underline{\text{bool-exp}} \times \text{Mng } \tau \times \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \tau.$$

To do this will require some preliminary work.

Let the functor $B \in \mathcal{T} \to |K|$ be the product in the functor category $\mathcal{T} \Rightarrow |K|$ of the constant functor $\tau \longmapsto$ Mng bool-exp, Mng, and Mng.

Thus for $\tau \in \mathcal{T}$

$$B\,\tau = \text{Mng bool-exp} \times \text{Mng}\ \tau \times \text{Mng}\ \tau$$

Shortly we will define a functor $N \in \mathcal{T} \to (\Sigma \Rightarrow Pdom)$.

Our plan is to define natural transformations

$$\Psi\,\tau \in B\,\tau \xrightarrow[\Sigma \Rightarrow Pdom]{} N\,\tau,$$

$$\Delta\,\tau \in N\,\tau \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng}\ \tau$$

and to then obtain the desired element of $B\,\tau \xrightarrow[|K|]{}$ Mng τ by composition:

$$\Delta\,\tau \circ \Psi\,\tau \in B\,\tau \xrightarrow[|K|]{} \text{Mng}\ \tau.$$

Our immediate aim is the definition of

$$N \in \mathcal{T} \to (\Sigma \Rightarrow Pdom)$$

as a composition of three functors. Note that the forgetful functor $U \in \Sigma \to Pdom^{op}$, which forgets the second component of morphisms, induces a functor

$$\langle U, - \rangle \in |K| \to (\Sigma \Rightarrow (Pdom^{op} \times Pdom))$$

such that for $F \in \text{Ob}\ |K| \subseteq \Sigma \to Pdom$ and $X \in \text{Ob}\ \Sigma$

$$\langle U, - \rangle\ F\ X = \langle U, F \rangle\ X = \langle UX, FX \rangle = \langle X, FX \rangle .$$

Next, the internal hom functor

$$Pdom_{\Rightarrow} \in Pdom^{op} \times Pdom \to Pdom$$

induces

$$(Pdom_{\Rightarrow})^{\Sigma} \in (\Sigma \Rightarrow (Pdom^{op} \times Pdom)) \to (\Sigma \Rightarrow Pdom),$$

which is such that

$$(Pdom_{\Rightarrow})^{\Sigma}\ G\ X = Pdom_{\Rightarrow}\ (G\ X),$$

where $G \varepsilon\ \Sigma \to (Pdom^{op} \times Pdom)$ and $X \varepsilon$ Ob Σ. Now let N be the composition

$$\mathcal{T} \xrightarrow{Mng} |K| \xrightarrow{\langle U,- \rangle} (\Sigma \Rightarrow (Pdom^{op} \times Pdom)) \xrightarrow{(Pdom_{\Rightarrow})^{\Sigma}} (\Sigma \Rightarrow Pdom).$$

Thus, if $\tau \varepsilon\ \mathcal{T}$ and $X \varepsilon$ Ob Σ, then

$$\begin{aligned} N\ \tau\ X &= (Pdom_{\Rightarrow})^{\Sigma}\ (\langle U, - \rangle\ (Mng\ \tau))\ X \\ &= (Pdom_{\Rightarrow})^{\Sigma}\ \langle U, Mng\ \tau \rangle\ X \\ &= Pdom_{\Rightarrow}\ \langle UX, Mng\ \tau\ X \rangle \\ &= X \xRightarrow[Pdom]{} Mng\ \tau\ X; \end{aligned}$$

therefore N τ X is the collection of functions from the set of stores of shape X to the collection of conventional meanings of type τ for stores of shape X.

Next we will define for each store shape $X \varepsilon$ Ob Σ the "choice function"

$$\Psi\ \tau\ X \varepsilon\ B\ \tau\ X \xrightarrow[Pdom]{} N\ \tau\ X$$

$$= (X \Rightarrow (Val\ \underline{bool}_{\perp})) \times Mng\ \tau\ X \times Mng\ \tau\ X \xrightarrow[Pdom]{} X \Rightarrow Mng\ \tau\ X$$

by

$$\Psi\ \tau\ X\ \langle b, t_1, t_2 \rangle\ x = \begin{cases} t_1 & \text{if } b\ x = true, \\ t_2 & \text{if } b\ x = false, \\ \perp_{Mng\ \tau\ X} & \text{if } b\ x = \perp, \end{cases}$$

where $b \varepsilon\ X \xrightarrow[Pdom]{} (Val\ \underline{bool})_{\perp}$ and $t_1, t_2 \varepsilon$ Mng τ X, and $x \varepsilon$ X.

Here we have used the fact that Mng τ X is actually a domain, and therefore it has a minimal element $\perp_{Mng\ \tau\ X}$.

It takes only a moment's thought to see that $\Psi\ \tau\ X$ is indeed continuous. We claim that $\Psi\ \tau$ is a natural transformation:

$$\Psi\ \tau \in B\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} N\ \tau \quad .$$

Suppose $\langle \phi,\rho \rangle \in X \xrightarrow[\Sigma]{} Y$. We must check the commutativity of the diagram

$$\begin{array}{ccc} (X \Rightarrow (\mathrm{Val}\ \underline{\mathrm{bool}})_\perp) \times \mathrm{Mng}\ \tau\ X \times \mathrm{Mng}\ \tau\ X & \xrightarrow{\Psi\ \tau\ X} & X \Rightarrow \mathrm{Mng}\ \tau\ X \\ \downarrow & & \downarrow \\ (\phi \Rightarrow 1) \times \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle \times \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle & & \phi \Rightarrow \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle \\ \downarrow & & \downarrow \\ (Y \Rightarrow (\mathrm{Val}\ \underline{\mathrm{bool}})_\perp) \times \mathrm{Mng}\ \tau\ Y \times \mathrm{Mng}\ \tau\ Y & \xrightarrow{\Psi\ \tau\ Y} & Y \Rightarrow \mathrm{Mng}\ \tau\ Y \end{array} .$$

Take $b \in X \xrightarrow[Pdom]{} (\mathrm{Val}\ \underline{\mathrm{bool}})_\perp$, $t_1 \in \mathrm{Mng}\ \tau\ X$, $t_2 \in \mathrm{Mng}\ \tau\ X$, $y \in Y$. Then

$$(\Psi\ \tau\ Y \circ ((\phi \Rightarrow 1) \times \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle \times \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle))\ \langle b, t_1, t_2 \rangle\ y$$

$$= \Psi\ \tau\ Y\ \langle b \circ \phi,\ \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_1,\ \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_2 \rangle\ y$$

$$= \begin{cases} \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_1 & \text{if } b\ (\phi\ y) = \mathit{true}, \\ \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_2 & \text{if } b\ (\phi\ y) = \mathit{false}, \\ \perp_{\mathrm{Mng}\ \tau\ Y} & \text{if } b\ (\phi\ y) = \perp\ , \end{cases}$$

$$= \begin{cases} \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_1 & \text{if } b\ (\phi\ y) = \mathit{true}, \\ \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ t_2 & \text{if } b\ (\phi\ y) = \mathit{false}, \\ \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ \perp_{\mathrm{Mng}\tau X} & \text{if } b\ (\phi\ y) = \perp, \end{cases}$$

because $\mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle$ is strict,

$$= \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ (\Psi\ \tau\ X\ \langle b, t_1, t_2 \rangle\ (\phi\ y))$$

$$= (\phi \Rightarrow \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle)\ (\Psi\ \tau\ X\ \langle b, t_1, t_2 \rangle)\ y$$

$$= ((\phi \Rightarrow \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle) \circ \Psi\ \tau\ X) \langle b, t_1, t_2 \rangle\ y.$$

This proves the claim of naturality.

Finally we will define for each $\tau \in T$ a "generalized diagonalization"

$$\Delta\ \tau\ \varepsilon\ N\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} Mng\ \tau.$$

The method of definition is by induction on the structure of $\tau\ \varepsilon\ T$ (There should be a better way.) Hence we will give

(1) $\Delta\ \pi$ for each primitive phrase type π,

(2) Δ ns

(3) $\Delta\ (\tau \Rightarrow \theta)$, assuming $\Delta\ \tau$ and $\Delta\ \theta$ are known.

Case 1a: π = comm. For $X\ \varepsilon$ Ob Σ, let

$$\Delta\ \underline{comm}\ X\ \varepsilon\ (X \Rightarrow Mng\ \underline{comm}\ X) \xrightarrow[Pdom]{} Mng\ \underline{comm}\ X$$

be given by

$$\Delta\ \underline{comm}\ X\ h\ x = h\ x\ x,$$

where $h\ \varepsilon\ X \xrightarrow[Pdom]{} Mng\ \underline{comm}\ X$ and $x\ \varepsilon\ X$.

It is very easy to see that Δ comm X is continuous. To check naturality of Δ comm, let $\langle \phi,\rho \rangle\ \varepsilon\ X \xrightarrow[\Sigma]{} Y$.

Consider the diagram

$$\begin{array}{ccc} X \Rightarrow Mng\ \underline{comm}\ X & \xrightarrow{\Delta\ \underline{comm}\ X} & Mng\ \underline{comm}\ X \\ \Big\downarrow \phi \Rightarrow Mng\ \underline{comm}\ \langle \phi,\rho \rangle & & \Big\downarrow Mng\ \underline{comm}\ \langle \phi,\rho \rangle \\ Y \Rightarrow Mng\ \underline{comm}\ Y & \xrightarrow{\Delta\ \underline{comm}\ Y} & Mng\ \underline{comm}\ Y. \end{array}$$

Let $h\ \varepsilon\ X \xrightarrow[Pdom]{} Mng\ \underline{comm}\ X$, $y\ \varepsilon\ Y$. Then

$$\begin{aligned} &(\Delta\ \underline{comm}\ Y \circ (\phi \Rightarrow Mng\ \underline{comm}\ \langle \phi,\rho \rangle))\ h\ y \\ &\quad = \Delta\ \underline{comm}\ Y\ (Mng\ \underline{comm}\ \langle \phi\ \rho \rangle \circ h \circ \phi)\ y \\ &\quad = (Mng\ \underline{comm}\ \langle \phi,\rho \rangle \circ h \circ \phi)\ y\ y \\ &\quad = Mng\ \underline{comm}\ \langle \phi,\rho \rangle\ (h\ (\phi\ y))\ y \end{aligned}$$

$= \rho_{\perp}\ (h\ (\phi\ y)\ (\phi\ y))\ y$

$= \rho_{\perp}\ (\Delta\ \underline{comm}\ X\ h\ (\phi\ y))\ y$

$= Mng\ \underline{comm}\ \langle \phi, \rho \rangle\ (\Delta\ \underline{comm}\ X\ h)\ y$

$= (Mng\ \underline{comm}\ \langle \phi, \rho \rangle \circ \Delta\ \underline{comm}\ X)\ h\ y,$

which shows Δ comm is natural.

Case 1b: $\pi = \delta\text{-}\underline{exp}$, where δ is a data type. For a store shape $X \in Ob\ \Sigma$, let

$$\Delta\ \delta\text{-}\underline{exp}\ X \in (X \Rightarrow Mng\ \delta\text{-}\underline{exp}\ X) \xrightarrow[Pdom]{} Mng\ \delta\text{-}\underline{exp}\ X$$

be given by

$$\Delta\ \delta\text{-}\underline{exp}\ X\ h\ x = h\ x\ x$$

where $h \in X \xrightarrow[Pdom]{} Mng\ \delta\text{-}\underline{exp}\ X$ and $x \in X$. Proofs of continuity and naturality are essentially the same as for Case 1a.

Case 1c: $\pi = \delta\text{-}\underline{acc}$ where δ is a data type. Let $X \in Ob\ \Sigma$ be a store shape and let

$$\Delta\ \delta\text{-}\underline{acc}\ X \in (X \Rightarrow Mng\ \delta\text{-}\underline{acc}\ X) \xrightarrow[Pdom]{} Mng\ \delta\text{-}\underline{acc}\ X$$

be given by

$$\Delta\ \delta\text{-}\underline{acc}\ X\ h\ v\ x = h\ x\ v\ x$$

where $h \in X \xrightarrow[Pdom]{} Mng\ \delta\text{-}\underline{acc}\ X = X \xrightarrow[Pdom]{} (Val\ \delta \Rightarrow Mng\ \underline{comm}\ X)$, $v \in Val\ \delta$, and $x \in X$. It is routine to check continuity of $\Delta\ \delta\text{-}\underline{acc}\ X$. To prove that $\Delta\ \delta\text{-}\underline{acc}$ is a natural transformation, let $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$, and consider the diagram

$$\begin{array}{ccc} X \Rightarrow \text{Mng } \delta\text{-}\underline{acc}\ X & \xrightarrow{\Delta\ \delta\text{-}\underline{acc}\ X} & \text{Mng } \delta\text{-}\underline{acc}\ X \\ \downarrow \phi \Rightarrow \text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle & & \downarrow \text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle \\ Y \Rightarrow \text{Mng } \delta\text{-}\underline{acc}\ Y & \xrightarrow{\Delta\ \delta\text{-}\underline{acc}\ Y} & \text{Mng } \delta\text{-}\underline{acc}\ Y. \end{array}$$

Let $h \varepsilon X \xrightarrow[Pdom]{} \text{Mng } \delta\text{-}\underline{acc}\ X$, $v \varepsilon \text{Val } \delta$, and $y \varepsilon Y$.

Then

$$\begin{aligned} &(\Delta\ \delta\text{-}\underline{acc}\ Y \circ (\phi \Rightarrow \text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle))\ h\ v\ y \\ &= \Delta\ \delta\text{-}\underline{acc}\ Y\ (\text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle \circ h \circ \phi)\ v\ y \\ &= (\text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle \circ h \circ \phi)\ y\ v\ y \\ &= \text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle\ (h\ (\phi\ y))\ v\ y \\ &= \text{Mng } \underline{comm}\ \langle \phi,\rho \rangle\ (h\ (\phi\ y)\ v)\ y \\ &= \rho_{\perp}\ (h\ (\phi\ y)\ v\ (\phi\ y))\ y \\ &= \rho_{\perp}\ (\Delta\ \delta\text{-}\underline{acc}\ X\ h\ v\ (\phi\ y))\ y \\ &= \text{Mng } \underline{comm}\ \langle \phi,\rho \rangle\ (\Delta\ \delta\text{-}\underline{acc}\ X\ h\ v)\ y \\ &= \text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle\ (\Delta\ \delta\text{-}\underline{acc}\ X\ h)\ v\ y \\ &= (\text{Mng } \delta\text{-}\underline{acc}\ \langle \phi,\rho \rangle \circ \Delta\ \delta\text{-}\underline{acc}\ X)\ h\ v\ y\ , \end{aligned}$$

which proves that the diagram commutes.

Case 1d: $\pi = \delta\text{-}\underline{var}$ where δ is a data type. Since Mng δ-$\underline{var}$ = Mng δ-$\underline{acc}$ × Mng δ-$\underline{exp}$, it would be nice to give

$$\Delta\ \delta\text{-}\underline{var} \varepsilon N\ \delta\text{-}\underline{var} \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng } \delta\text{-}\underline{var}$$

in terms of

$$\Delta\ \delta\text{-}\underline{acc} \varepsilon N\ \delta\text{-}\underline{acc} \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng } \delta\text{-}\underline{acc}$$

and

$$\Delta\ \delta\text{-}\underline{exp} \varepsilon N\ \delta\text{-}\underline{exp} \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng } \delta\text{-}\underline{exp}\ .$$

There is a natural isomorphism

$$\eta \in N\ \delta\text{-}\underline{var} \xrightarrow[\Sigma \Rightarrow Pdom]{} N\ \delta\text{-}\underline{acc} \times N\ \delta\text{-}\underline{exp}$$

where for $X \in Ob\ \Sigma$

$$\eta\ X \in X \Rightarrow (Mng\ \delta\text{-}\underline{acc}\ X \times Mng\ \delta\text{-}\underline{exp}\ X)$$

$$\xrightarrow[Pdom]{} (X \Rightarrow Mng\ \delta\text{-}\underline{acc}\ X) \times (X \Rightarrow Mng\ \delta\text{-}\underline{exp}\ X)$$

is the obvious bijection. Thus, we may take $\Delta\ \delta\text{-}\underline{var}$ to be the composite

$$\Delta\ \delta\text{-}\underline{var} = (\Delta\ \delta\text{-}\underline{acc} \times \Delta\ \delta\text{-}\underline{exp}) \circ \eta,$$

where the composition takes place in $\Sigma \Rightarrow Pdom$.

Case 2: We want to define

$$\Delta\ \underline{ns} \in N\ \underline{ns} \xrightarrow[\Sigma \Rightarrow Pdom]{} Mng\ \underline{ns}$$

There is only one way because Mng $\underline{ns}$ is a terminal object in $\Sigma \Rightarrow Pdom$.

Case 3: Assume that

$$\Delta\ \tau \in N\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} Mng\ \tau \text{ and } \Delta\ \theta \in N\ \theta \xrightarrow[\Sigma \Rightarrow Pdom]{} Mng\ \theta$$

are known. We will define

$$\Delta\ (\tau \Rightarrow \theta) \in N\ (\tau \Rightarrow \theta) \xrightarrow[\Sigma \Rightarrow Pdom]{} Mng\ (\tau \Rightarrow \theta).$$

(Actually, $\Delta\ \tau$ will not enter into the definition of $\Delta\ (\tau \Rightarrow \theta)$. This is a curious fact, and it should be connected with some interesting intuition.) We start with

$$Ap = Ap_{|K|}\ \langle Mng\ \tau,\ Mng\ \theta \rangle \in Mng\ (\tau \Rightarrow \theta) \times Mng\ \tau \xrightarrow[|K|]{} Mng\ \theta.$$

Hence, for $X \in Ob\ \Sigma$,

$$Ap\ X \in Mng\ (\tau \Rightarrow \theta)\ X \times Mng\ \tau\ X \xrightarrow[Pdom]{} Mng\ \theta\ X.$$

From Ap we get a natural transformation

$$\Phi\langle\tau,\theta\rangle \in N(\tau \Rightarrow \theta) \times Mng\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} N\,\theta$$

where for $X \in Ob\ \Sigma$

$$\Phi\langle\tau,\theta\rangle X \in (X \Rightarrow Mng(\tau \Rightarrow \theta)X) \times Mng\ \tau\ X \xrightarrow[Pdom]{} (X \Rightarrow Mng\ \theta\ X)$$

is given by

$$\Phi\langle\tau,\theta\rangle\ X\langle h,t\rangle\ x = Ap\ X\langle h\ x,\ t\rangle,$$

where $h \in X \xrightarrow[Pdom]{} Mng(\tau \Rightarrow \theta)X$, $t \in Mng\ \tau\ X$, and $x \in X$.

(The idea is this: a function like Ap X, say

$$a \in (U \times T) \to V,$$

induces a function

$$f \in ((X \to U) \times T) \to (X \to V)$$

via the equation

$$f\langle h,t\rangle\ x = a\langle h\ x,\ t\rangle.)$$

To verify that $\Phi\langle\tau,\theta\rangle$ is indeed a natural transformation, let $\langle\phi,\rho\rangle \in X \xrightarrow[\Sigma]{} Y$, and consider the diagram

$$\begin{array}{ccc}
(X \Rightarrow Mng(\tau \Rightarrow \theta)X) \times Mng\ \tau\ X & \xrightarrow{\Phi\langle\tau,\theta\rangle X} & X \Rightarrow Mng\ \theta\ X \\
\downarrow & & \downarrow \\
(\phi \Rightarrow Mng(\tau \Rightarrow \theta)\langle\phi,\rho\rangle) \times Mng\ \tau\langle\phi,\rho\rangle & & \phi \Rightarrow Mng\ \theta\ \langle\phi,\rho\rangle \\
\downarrow & & \downarrow \\
(Y \Rightarrow Mng(\tau \Rightarrow \theta)Y) \times Mng\ \tau\ Y & \xrightarrow{\Phi\langle\tau,\theta\rangle Y} & Y \Rightarrow Mng\ \theta\ Y.
\end{array}$$

Let $h \in X \xrightarrow[Pdom]{} Mng(\tau \Rightarrow \theta)X$, $t \in Mng\ \tau\ X$, $y \in Y$. Then compute:

$$\begin{aligned}
&(\Phi\langle\tau,\theta\rangle Y \circ ((\phi \Rightarrow Mng(\tau \Rightarrow \theta)\langle\phi,\rho\rangle) \times Mng\ \tau\ \langle\phi,\rho\rangle))\ \langle h,\ t\rangle\ y \\
&\quad = \Phi\langle\tau,\theta\rangle Y\ \langle Mng(\tau \Rightarrow \theta)\ \langle\phi,\rho\rangle \circ h \circ \phi,\ Mng\ \tau\ \langle\phi,\rho\rangle\ t\rangle\ y \\
&\quad = Ap\ Y\ \langle Mng(\tau \Rightarrow \theta)\ \langle\phi,\rho\rangle\ (h\ (\phi\ y)),\ Mng\ \tau\ \langle\phi,\rho\rangle t\rangle \\
&\quad = (Ap\ Y \circ (Mng(\tau \Rightarrow \theta)\ \langle\phi,\rho\rangle \times Mng\ \tau\ \langle\phi,\rho\rangle))\ \langle h\ (\phi\ y),\ t\rangle
\end{aligned}$$

$$= (\mathrm{Mng}\ \theta\ \langle \phi,\rho\rangle \circ \mathrm{Ap}\ X)\ \langle h\ (\phi\ y),\ t\rangle,$$

by the naturality of Ap,

$$= \mathrm{Mng}\ \theta\ \langle \phi,\rho\rangle\ (\mathrm{Ap}\ X\ \langle h\ (\phi\ y),\ t\rangle)$$

$$= \mathrm{Mng}\ \theta\ \langle \phi,\rho\rangle\ (\Phi\ \langle \tau,\theta\rangle\ X\ \langle h,\ t\rangle\ (\phi\ y))$$

$$= (\mathrm{Mng}\ \theta\ \langle \phi,\rho\rangle \circ \Phi\ \langle \tau,\theta\rangle\ X\ \langle h,\ t\rangle \circ \phi)\ y$$

$$= ((\phi \Rightarrow \mathrm{Mng}\ \theta\ \langle \phi,\rho\rangle)\ \Phi\ \langle \phi,\theta\rangle X\langle h,\ t\rangle)\ y$$

$$= ((\phi \Rightarrow \mathrm{Mng}\ \theta\langle \phi,\rho\rangle) \circ \Phi\ \langle \tau,\theta\rangle\ X)\ \langle h,\ t\rangle\ y.$$

This shows that $\Phi\ \langle \tau,\theta\rangle$ is natural. Now consider the composition of natural transformations

$$\mathrm{N}\ (\tau \Rightarrow \theta) \times \mathrm{Mng}\ \tau \xrightarrow{\Phi\ \langle \tau,\theta\rangle} \mathrm{N}\ \theta \xrightarrow{\Delta\ \theta} \mathrm{Mng}\ \theta.$$

(This is where $\Delta\ \theta$ is used.) By using the abstraction bijection

$$\mathrm{Ab}\ \varepsilon\ (\mathrm{N}\ (\pi \Rightarrow \theta) \times \mathrm{Mng}\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} \mathrm{Mng}\ \theta)$$

$$\longrightarrow (\mathrm{N}\ (\tau \Rightarrow \theta) \xrightarrow[\Sigma \Rightarrow Pdom]{} \mathrm{Mng}\ (\tau \Rightarrow \theta)),$$

we define

$$\Delta\ (\tau \Rightarrow \theta)\ \varepsilon\ \mathrm{N}\ (\tau \Rightarrow \theta) \xrightarrow[\Sigma \Rightarrow Pdom]{} \mathrm{Mng}\ (\tau \Rightarrow \theta)$$

by

$$\Delta\ (\tau \Rightarrow \theta) = \mathrm{Ab}\ (\Delta\ \theta \circ \Phi\ \langle \tau,\theta\rangle).$$

This completes the definition of $\Delta\ \tau$ for every phrase type τ.

Do not lose sight of the fact that the purpose of all this is to define

$$\mathrm{semf}\ \ \mathrm{ifthenelse}_{\tau}$$

as the natural transformation which is in canonical correspondence with

$$\Delta\ \tau \circ \Psi\ \tau\ \varepsilon\ \mathrm{B}\ \tau \xrightarrow[|K|]{} \mathrm{Mng}\ \tau.$$

Evidence that we have found the correct definition of ifthenelse_τ is provided by the following theorem.

Theorem 7.1: Let $\tau, \theta \in T$ and $b, r, s, m \in L$. Let $\alpha \in A$ be such that

$$\text{Type } r\ \alpha = \text{Type } s\ \alpha = \tau \Rightarrow \theta,$$

$$\text{Type } b\ \alpha = \underline{\text{bool-exp}},$$

$$\text{Type } m\ \alpha = \tau.$$

Let

$$p = [\![[\![\text{ifthenelse}_{\tau \Rightarrow \theta}\ b\ r\ s]\!]\ m\ ]\!], \text{ and}$$

$$q = [\![\text{ifthenelse}_\theta\ b\ [\![r\ m]\!]\ [\![s\ m]\!]\ ]\!].$$

Then

$$\text{Type } p\ \alpha = \text{Type } q\ \alpha = \theta, \text{ and}$$

$$\text{Semf } p\ \alpha = \text{Semf } q\ \alpha\ .$$

Proof: Since this theorem is peripheral to our development, we omit the straightforward, but lengthy, proof. □

We now take up the definition of

$$\text{semf rec}_\tau \in \text{Env emp}_T \xrightarrow[|K|]{} (\text{Mng } \tau \Rightarrow \text{Mng } \tau) \Rightarrow \text{Mng } \tau.$$

It suffices to define the corresponding natural transformation

$$\text{fix}_\tau \in \text{Mng } \tau \Rightarrow \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \tau.$$

Let $X \in \text{Ob } \Sigma$ be a store shape. Define

$$\text{fix}_\tau\ X \in (\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ X \xrightarrow[Pdom]{} \text{Mng } \tau\ X$$

by

$$\text{fix}_\tau\ X\ \eta = Y_{\text{Mng } \tau\ X}\ (A_X\ (\eta\ X)\ 1^\Sigma_X)$$

where $\eta \in \hom^\Sigma X \times \text{Mng } \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng } \tau$ (recall that

this is the underlying set of $(\mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ \tau)\ X)$,

$Y_{\mathrm{Mng}\ \tau\ X} \varepsilon\ (\mathrm{Mng}\ \tau\ X \Rightarrow \mathrm{Mng}\ \tau\ X) \xrightarrow[Pdom]{} \mathrm{Mng}\ \tau\ X$

is the least fixed point function $(Y_{\mathrm{Mng}\ \tau\ X}\ h = \bigsqcup_{n=0}^{\infty} h^n\ \perp_{\mathrm{Mng}\ \tau\ X})$,

and A_X is the abstraction function for *Pdom* such that

$$A_X \varepsilon\ ((X \underset{\Sigma}{\to} X \times \mathrm{Mng}\ \tau\ X) \xrightarrow[Pdom]{} \mathrm{Mng}\ \tau\ X)$$

$$\longrightarrow (X \underset{\Sigma}{\to} X \xrightarrow[Pdom]{} (\mathrm{Mng}\ \tau\ X \Rightarrow \mathrm{Mng}\ \tau\ X)).$$

The domains and codomains for abstraction functions for *Pdom* can be partially ordered in an obvious way to make them predomains; it is then easy to show that abstraction functions are continuous. From this observation a routine argument shows that fix_τ is a natural transformation. We require a technical lemma.

Lemma 7.2: Let $\langle \phi,\rho \rangle \varepsilon\ X \xrightarrow[\Sigma]{} Y$ and $\eta \varepsilon\ \mathrm{hom}^{\Sigma}\ X \times \mathrm{Mng}\ \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} \mathrm{Mng}\ \tau$. For each $n = 0, 1, 2, \ldots,$

$$(A_Y\ ((\mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ \tau)\ \langle \phi,\rho \rangle\ \eta\ Y)\ 1_Y^{\Sigma})^n\ \perp_{\mathrm{Mng}\ \tau\ Y}$$

$$= \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^n\ \perp_{\mathrm{Mng}\ \tau\ X}).$$

Proof: We will use induction on n. For n = 0, the assertion becomes

$$\perp_{\mathrm{Mng}\ \tau\ Y} = \mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle\ \perp_{\mathrm{Mng}\ \tau\ X'}$$

which is true because $\mathrm{Mng}\ \tau\ \langle \phi,\rho \rangle$ is strict.

Suppose the lemma is true for n = k. Since η is a natural transformation, we have a commutative diagram

$$\begin{array}{ccc} (X \xrightarrow[\Sigma]{} X) \times \mathrm{Mng}\ \tau\ X & \xrightarrow{\eta\ X} & \mathrm{Mng}\ \tau\ X \\ \downarrow{\scriptstyle \hom^{\Sigma} X\langle\phi,\rho\rangle \times \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle} & & \downarrow{\scriptstyle \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle} \\ (X \xrightarrow[\Sigma]{} Y) \times \mathrm{Mng}\ \tau\ Y & \xrightarrow{\eta\ Y} & \mathrm{Mng}\ \tau\ Y\ . \end{array}$$

Let $B = \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle\ ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^k\ \perp_{\mathrm{Mng}\ \tau\ X})$.

Then

$(A_Y\ ((\mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ \tau)\ \langle\phi,\rho\rangle\ \eta\ Y)\ 1_Y^{\Sigma})^{k+1}\ \perp_{\mathrm{Mng}\ \tau\ Y}$

$= A_Y\ ((\mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ \tau)\ \langle\phi,\rho\rangle\ \eta\ Y)\ 1_Y^{\Sigma}\ B,$

by the induction hypothesis,

$= (\mathrm{Mng}\ \tau \Rightarrow \mathrm{Mng}\ \tau)\ \langle\phi,\rho\rangle\ \eta\ Y\ \langle 1_Y,\ B\rangle,$

by the definition of abstraction in *Pdom*,

$= \eta\ Y\ \langle\langle\phi,\rho\rangle,\ B\ \rangle,$

by the definition of exponentiation in $|K|$,

$= (\eta\ Y \circ (\hom^{\Sigma} X\langle\phi,\rho\rangle \times \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle))\langle 1_X^{\Sigma}, (A_X(\eta\ X)1_X^{\Sigma})^k \perp_{\mathrm{Mng}\ \tau\ X}\rangle$

$= (\mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle \circ \eta\ X)\ \langle 1_X^{\Sigma},\ (A_X\ (\eta\ X)\ 1_X^{\Sigma})^k\ \perp_{\mathrm{Mng}\ \tau\ X},$

by the commutativity of the above diagram,

$= \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle\ (\eta\ X\ \langle 1_X^{\Sigma},\ (A_X\ (\eta\ X)\ 1_X^{\Sigma})^k\ \perp_{\mathrm{Mng}\ \tau\ X}\rangle)$

$= \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle\ (A_X\ (\eta\ X)\ 1_X^{\Sigma}\ ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^k\ \perp_{\mathrm{Mng}\ \tau\ X})\ )$

$= \mathrm{Mng}\ \tau\ \langle\phi,\rho\rangle\ ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^{k+1}\ \perp_{\mathrm{Mng}\ \tau\ X}).$

□

To return to the claim that fix_τ is a natural transformation, suppose $\langle\phi,\rho\rangle \in X \xrightarrow[\Sigma]{} Z$, and consider the diagram in *Pdom*

$$\begin{array}{ccc} (\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ X & \xrightarrow{\text{fix}_\tau\ X} & \text{Mng } \tau\ X \\ \downarrow{\scriptstyle (\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ \langle \phi,\rho \rangle} & & \downarrow{\scriptstyle \text{Mng } \tau\ \langle \phi,\rho \rangle} \\ (\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ Z & \xrightarrow{\text{fix}_\tau\ Z} & \text{Mng } \tau\ Z \end{array} .$$

Let $\eta \in \hom^{\Sigma}\ X \times \text{Mng } \tau \xrightarrow[\Sigma \Rightarrow Pdom]{} \text{Mng } \tau$. The computation

$$\begin{aligned} &(\text{fix}_\tau\ Z \circ (\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ \langle \phi,\rho \rangle)\ \eta \\ &= \text{fix}_\tau\ Z\ ((\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ \langle \phi,\rho \rangle\ \eta) \\ &= Y_{\text{Mng } \tau\ Z}\ (A_Z\ ((\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ \langle \phi,\rho \rangle\ \eta\ Z)\ 1_Z^{\Sigma}) \\ &= \bigsqcup_{n=0}^{\infty} ((A_Z\ ((\text{Mng } \tau \Rightarrow \text{Mng } \tau)\ \langle \phi,\rho \rangle\ \eta\ Z)\ 1_Z^{\Sigma})^n\ \perp_{\text{Mng } \tau\ Z}) \\ &= \bigsqcup_{n=0}^{\infty} (\text{Mng } \tau\ \langle \phi,\rho \rangle\ ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^n\ \perp_{\text{Mng } \tau\ X})), \end{aligned}$$

by Lemma 7.2,

$$\begin{aligned} &= \text{Mng } \tau\ \langle \phi,\rho \rangle \bigsqcup_{n=0}^{\infty} ((A_X\ (\eta\ X)\ 1_X^{\Sigma})^n\ \perp_{\text{Mng } \tau\ X}), \end{aligned}$$

by the continuity of Mng $\tau\ \langle \phi,\rho \rangle$,

$$\begin{aligned} &= \text{Mng } \tau\ \langle \phi,\rho \rangle\ (Y_{\text{Mng } \tau\ X}\ (A_X\ (\eta\ X)\ 1_X^{\Sigma})) \\ &= \text{Mng } \tau\ \langle \phi,\rho \rangle\ (\text{fix}_\tau\ X\ \eta) \\ &= (\text{Mng } \tau\ \langle \phi,\rho \rangle \circ \text{fix}_\tau\ X)\ \eta \end{aligned}$$

shows fix_τ is natural.

We must do some preliminary work with the category Σ of store shapes before giving the semantics of newvar_δ for a data type δ. We will give some definitions and three lemmas.

The functor

$$\text{Newshape}_\delta \in \Sigma \longrightarrow \Sigma$$

is defined for $X \varepsilon \text{ Ob } \Sigma$ by

$$\text{Newshape}_\delta \; X = X \times \text{Val } \delta$$

and for $\langle \phi, \rho \rangle \varepsilon X \underset{\Sigma}{\longrightarrow} Y$ by

$$\text{Newshape}_\delta \langle \phi, \rho \rangle = \langle \phi_\delta, \rho_\delta \rangle \varepsilon X \times \text{Val } \delta \underset{\Sigma}{\longrightarrow} Y \times \text{Val } \delta,$$

where

$$\phi_\delta \varepsilon Y \times \text{Val } \delta \longrightarrow X \times \text{Val } \delta$$

is given by

$$\phi_\delta \langle y, v \rangle = \langle \phi \; y, \; v \rangle \; ,$$

and

$$\rho_\delta \varepsilon X \times \text{Val } \delta \to (Y \times \text{Val } \delta \to Y \times \text{Val } \delta)$$

is given by

$$\rho_\delta \langle x, \; v \rangle \; \langle y, \; v' \rangle = \langle \rho \; x \; y, \; v \rangle \; .$$

It is not too hard to see that $\langle \phi_\delta, \rho_\delta \rangle$ satisfies (Σ1) (Σ2), and (Σ3) and that Newshape_δ really is a functor.

We will use Newshape_δ to create a local store shape from a global store shape, as must be done upon entering a block containing a declaration of a δ-variable.

We posit the existence of a specified element

$$\text{init } \delta \varepsilon \bar{\text{V}}\text{al } \delta,$$

which is used as an initial value for a newly declared variable. For each store shape X, this leads to the definition of

$$\text{enter}_\delta \; X \varepsilon X \longrightarrow X \times \text{Val } \delta,$$

given for $x \varepsilon X$ by

$$\text{enter}_\delta \; X \; x = \langle x, \; \text{init } \delta \rangle \; .$$

We also have, for each store shape X,

$$\text{exit}_\delta\ X\ \langle x, v\rangle = x.$$

Obviously, these functions will be used upon entering and exiting from blocks.

Lemma 7.3: Let $\langle \phi, \rho\rangle \in X \xrightarrow[\Sigma]{} Y$. Then

$$(\text{enter}_\delta\ Y \to \text{exit}_\delta\ Y) \circ \rho_\delta = \rho \circ \text{exit}_\delta\ X.$$

Thus, for all $x \in X$, $v \in V$, $y \in Y$,

$$\text{exit}_\delta\ Y\ (\rho_\delta\ \langle x, v\rangle\ (\text{enter}_\delta\ Y\ y)) = \rho(\text{exit}_\delta\ X\ \langle \hat{x}, v\rangle)\ y.$$

Proof: The proof is an easy computation left to the reader. □

The (Ob Σ)-indexed family exit_δ of functions is intimately related to a natural transformation

$$\text{Exit}_\delta \in 1_\Sigma \xrightarrow[\Sigma \Rightarrow \Sigma]{} \text{Newshape}_\delta,$$

given by

$$\text{Exit}_\delta\ X = \langle \text{exit}_\delta\ X,\ \nu_\delta\ X\rangle \in X \xrightarrow[\Sigma]{} X \times \text{Val}\ \delta$$

where

$$\nu_\delta\ X \in X \to (X \times \text{Val}\ \delta \to X \times \text{Val}\ \delta)$$

is defined for $x, x' \in X$, $v \in \text{Val}\ \delta$ by

$$\nu_\delta\ X\ x\ \langle x', v\rangle = \langle x, v\rangle .$$

Verifications that $\text{Exit}_\delta\ X$ is a morphism of Σ and that Exit_δ is natural are left to the reader. The utility of Exit_δ is that it contains the information about how to return to the global store shape from the local store shape. There is a way to relate enter_δ to a natural transformation, but it involves a slight change in the definition of Σ, and this

change would entail more complications that it is apparently worth.

Lemma 7.4: Let $\langle \phi, \rho \rangle \varepsilon\ X \xrightarrow[\Sigma]{} Y$. Then

$$\langle \phi_\delta, \rho_\delta \rangle \circ \mathrm{Exit}_\delta\ X = \mathrm{Exit}_\delta\ Y \circ \langle \phi, \rho \rangle$$

Proof: The proof is straightforward. □

It is not surprising that the denotation of newvar_δ involves a "canonical local δ-variable for each store shape X". Recall that Mng δ-var $(X \times \mathrm{Val}\ \delta) = \mathrm{Mng}\ \delta\text{-acc}\ (X \times \mathrm{Val}\ \delta) \times \mathrm{Mng}\ \delta\text{-exp}(X \times \mathrm{Val}\ \delta)$. Now for $X \varepsilon \mathrm{Ob}\ \Sigma$ let

$$\mathrm{localvar}_\delta\ X \varepsilon\ \mathrm{Mng}\ \delta\text{-var}\ (X \times \mathrm{Val}\ \delta)$$

be

$$\mathrm{localvar}_\delta\ X = \langle a_{\delta,X}, e_{\delta,X} \rangle,$$

where

$$a_{\delta,X} \varepsilon\ \mathrm{Val}\ \delta \xrightarrow[Pdom]{} \mathrm{Mng}\ \underline{\mathrm{comm}}\ (X \times \mathrm{Val}\ \delta)$$

is given for $v, v' \varepsilon V$ and $x \varepsilon X$ by

$$a_{\delta,X}\ v\ \langle x, v' \rangle = \langle x, v \rangle,$$

and

$$e_{\delta,X} \varepsilon\ X \times \mathrm{Val}\ \delta \xrightarrow[Pdom]{} (\mathrm{Val}\ \delta)_\perp$$

is given for $x \varepsilon X$, $v \varepsilon \mathrm{Val}\ \delta$ by

$$e_{\delta,X}\ \langle x, v \rangle = v.$$

Note the use of condition (Σ2) in the proof of the next lemma.

Lemma 7.5: Let $\langle \phi, \rho \rangle \varepsilon\ X \xrightarrow[\Sigma]{} Y$. Then

$$\mathrm{Mng}\ \delta\text{-var}\ \langle \phi_\delta, \rho_\delta \rangle\ (\mathrm{localvar}_\delta\ X) = \mathrm{localvar}_\delta\ Y.$$

Proof: We claim

$$\mathrm{Mng}\ \delta\text{-acc}\ \langle \phi_\delta, \rho_\delta \rangle\ a_{\delta,X} = a_{\delta,Y}.$$

To see this, let $v \in \text{Val}\ \delta$ and $\langle y, v' \rangle \in Y \in \text{Val}\ \delta$. Then

$$\begin{aligned}
&\text{Mng}\ \delta\text{-}\underline{\text{acc}}\ \langle \phi_\delta, \rho_\delta \rangle\ a_{\delta,X}\ v\ \langle y, v' \rangle \\
&= \text{Mng}\ \underline{\text{comm}}\ \langle \phi_\delta, \rho_\delta \rangle\ (a_{\delta,X}\ v)\ \langle y, v' \rangle \\
&= \rho_{\delta,\perp}\ (a_{\delta,X}\ v\ (\phi_\delta\ \langle y, v' \rangle))\ \langle y, v' \rangle \\
&= \rho_{\delta,\perp}\ (a_{\delta,X}\ v\ \langle \phi\ y, v' \rangle)\ \langle y, v' \rangle \\
&= \rho_{\delta,\perp}\ \langle \phi\ y, v \rangle\ \langle y, v' \rangle \\
&= \langle \rho\ (\phi\ y)\ y, v \rangle \\
&= \langle y, v \rangle, \text{ by } (\Sigma 2), \\
&= a_{\delta,Y}\ v\ \langle y, v' \rangle,
\end{aligned}$$

which proves the claim.

Furthermore, we claim

$$\text{Mng}\ \delta\text{-}\underline{\text{exp}}\ \langle \phi_\delta, \rho_\delta \rangle\ e_{\delta,X} = e_{\delta,Y}.$$

Let $\langle y, v \rangle \in Y \times \text{Val}\ \delta$. We compute:

$$\begin{aligned}
&\text{Mng}\ \delta\text{-}\underline{\text{exp}}\ \langle \phi_\delta, \rho_\delta \rangle\ e_{\delta,X}\ \langle y, v \rangle \\
&= e_{\delta,X}\ (\phi_\delta\ \langle y, v \rangle) \\
&= e_{\delta,X}\ \langle \phi\ y, v \rangle \\
&= v \\
&= e_{\delta,X}\ \langle y, v \rangle.
\end{aligned}$$

By combining these two facts we see

$$\begin{aligned}
&\text{Mng}\ \delta\text{-}\underline{\text{var}}\ \langle \phi_\delta, \rho_\delta \rangle\ (\text{localvar}_\delta\ X) \\
&= (\text{Mng}\ \delta\text{-}\underline{\text{acc}}\ \langle \phi_\delta, \rho_\delta \rangle \times \text{Mng}\ \delta\text{-}\underline{\text{exp}}\ \langle \phi_\delta, \rho_\delta \rangle)\ \langle a_{\delta,X}, e_{\delta,X} \rangle \\
&= \langle a_{\delta,Y}, e_{\delta,Y} \rangle \\
&= \text{localvar}_\delta\ Y,
\end{aligned}$$

as desired. □

We want to define the natural transformation

$$\text{semf newvar}_\delta \varepsilon \text{ Env emp}_T \xrightarrow[|K|]{} (\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) \Rightarrow \text{Mng } \underline{\text{comm}};$$

so it suffices to define the corresponding natural transformation

$$\text{nv}_\delta \varepsilon \text{ Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}.$$

Thus, we must give, for $X \varepsilon$ Ob Σ,

$$\text{nv}_\delta X \varepsilon (\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) X \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}} X.$$

For $\eta \varepsilon \hom^\Sigma X \times \text{Mng } \delta\text{-}\underline{\text{var}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}$ (recall that this is the underlying set of $(\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) X$), let

$$\text{nv}_\delta X \eta$$

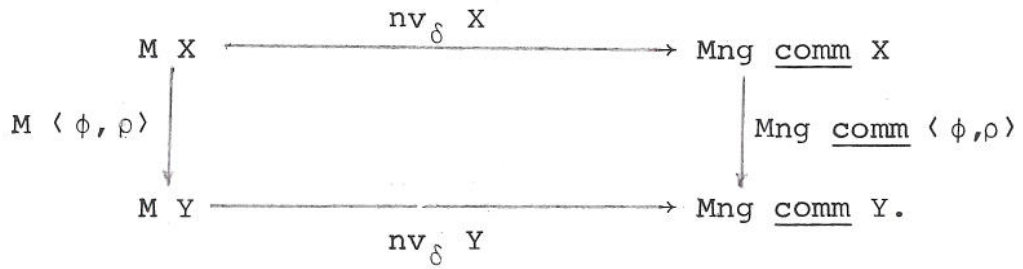
$$= (\text{exit}_\delta X)_\perp \circ \eta (\text{Newshape}_\delta X) \langle \text{Exit}_\delta X, \text{localvar}_\delta X \rangle \circ \text{enter}_\delta X,$$

where

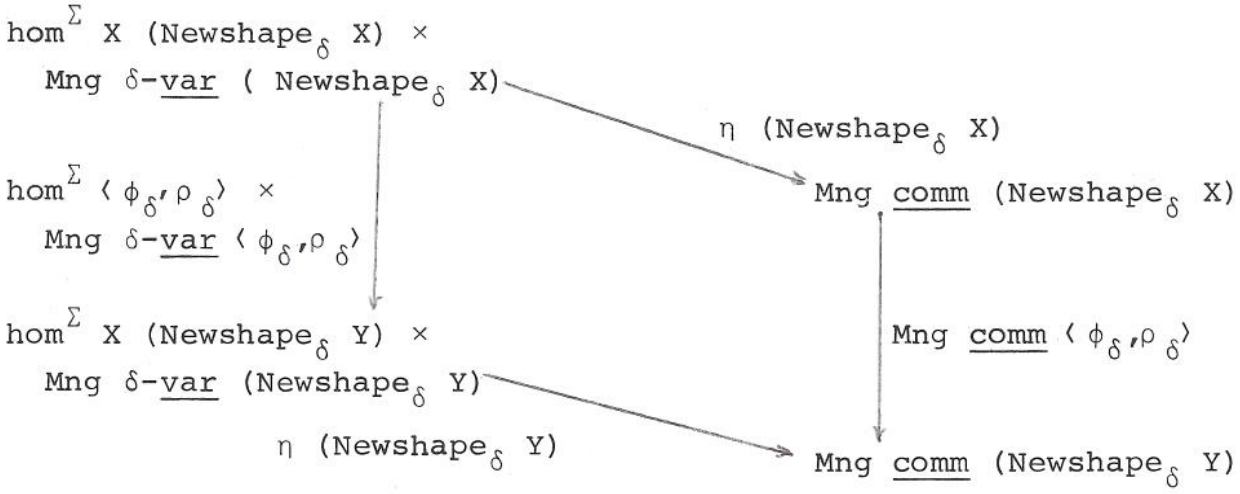
$$(\text{exit}_\delta X)_\perp \varepsilon (X \times \text{Val } \delta)_\perp \xrightarrow[Pdom]{} X_\perp$$

is the strict extension of $\text{exit}_\delta X$. Roughly speaking, we are saying that $\text{nv}_\delta X \eta$ means enter a new block, execute the command derived from η in the context of the new store shape, and finally exit from the new block in a manner that returns the store to its original shape. It is easy to see that $\text{nv}_\delta X$ is continuous.

We must verify that nv_δ is a natural transformation. Let M denote the functor Mng δ-var $\Rightarrow$ Mng comm. Suppose $\langle \phi, \rho \rangle \varepsilon X \xrightarrow[\Sigma]{} Y$. Consider the commutativity of



Let $\eta \in \hom^{\Sigma} X \times \text{Mng } \delta\text{-}\underline{\text{var}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}$ (the underlying set of M X), and let $y \in Y$. It is important to note that



commutes because η is a natural transformation. Then

$$
\begin{aligned}
&(nv_\delta\ Y \circ (1 \times M \langle \phi,\rho\rangle))\ \eta\ y \\
&\quad = nv_\delta\ Y\ (M \langle \phi,\rho\rangle\ \eta)\ y \\
&\quad = ((exit_\delta\ Y)_\perp \\
&\qquad \circ M \langle \phi,\rho\rangle\ \eta\ (Newshape_\delta\ Y) \langle Exit_\delta\ Y,\ localvar_\delta\ Y\rangle \\
&\qquad \circ enter_\delta\ Y)\ y \\
&\quad = ((exit_\delta\ Y)_\perp \\
&\qquad \circ \eta\ (Newshape_\delta\ Y) \langle Exit_\delta\ Y \circ \langle \phi,\rho\rangle,\ localvar_\delta\ Y\rangle \\
&\qquad \circ enter_\delta\ Y)\ y,
\end{aligned}
$$

by the definition of $\Rightarrow$ for $|K|$,

$= ((\text{exit}_\delta\ Y)_\perp$

$\circ\ \eta\ (\text{Newshape}_\delta\ Y)\ \langle\langle \phi_\delta, \rho_\delta\rangle \circ \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ Y\rangle$

$\circ\ \text{enter}_\delta\ Y)\ y,$

by Lemma 7.4,

$= ((\text{exit}_\delta\ Y)_\perp$

$\circ\ \eta\ (\text{Newshape}_\delta\ Y)$

$((\text{hom}^\Sigma\ X\ \langle \phi_\delta, \rho_\delta\rangle \times \text{Mng}\ \delta\text{-}\underline{\text{var}}\ \langle \phi_\delta, \rho_\delta\rangle)\ \langle \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ X\rangle)$

$\circ\ \text{enter}_\delta\ Y)\ y,$

by Lemma 7.5

$= ((\text{exit}\ \ Y)_\perp$

$\circ\ \text{Mng}\ \underline{\text{comm}}\ \langle \phi_\delta, \rho_\delta\rangle\ (\eta\ (\text{Newshape}_\delta\ X)\ \langle \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ X\rangle$

$\circ\ \text{enter}_\delta\ Y)\ y$

$= ((\text{exit}_\delta\ Y)_\perp$

$(\rho_{\delta,\perp}\ (\eta\ (\text{Newshape}_\delta\ X)\ \langle \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ X\rangle\ (\phi_\delta(\text{enter}_\delta Y\ y)))$

$(\text{enter}_\delta\ Y\ y))$

$= \rho_\perp\ ((\text{exit}_\delta\ X)_\perp$

$(\eta\ (\text{Newshape}_\delta\ X)\ \langle \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ X\rangle$
$(\phi_\delta(\text{enter}_\delta\ Y\ y))))\ y,$

by Lemma 7.3,

$= \rho_\perp\ ((\text{exit}_\delta\ X)_\perp$

$(\eta\ (\text{Newshape}_\delta\ X)\langle \text{Exit}_\delta\ X,\ \text{localvar}_\delta\ X\rangle\ (\text{enter}_\delta X\ (\phi\ y)))\ y$

$= \quad (\text{Mng } \underline{\text{comm}} \langle \phi, \delta \rangle \circ nv_{\delta} X) \eta y,$

and this proves nv_{δ} is a natural transformation.

CHAPTER VIII

DESUGARED ALGOL: CONTINUATION SEMANTICS

In this chapter we expand the language given in the preceding chapter to include goto's and labels. The main novelties consist of the introduction of the primitive phrase type compl (for "completion"), whose set of meanings in the context of a store shape are what are normally called continuations, and of the removal of comm from the collection of primitive phrase types. Our point of view is now that comm is just an abbreviation for the procedural phrase type compl ⇒ compl.

As in Chapter VII we assume we are given

(1) $\mathcal{D}$, a poset of data types,

(2) Op, a collection of data-type operators,

(3) arity ε Op → (Ob $\mathcal{D}$)* and res ε Op → Ob $\mathcal{D}$,

(4) Val ε $\mathcal{D}$ → $\mathcal{S}et$, a functor,

(5) m_{Op}, an appropriate interpretation for the operators in Op, and

(6) Id, an infinite set of identifiers.

We first diverge from Chapter VII by changing P, the poset of primitive phrase types. At first, the change appears superficial: comm is just replaced by compl. Thus

$$\begin{aligned} \text{Ob } P = \{&\underline{\text{compl}}\} \\ &\cup \{\delta\text{-}\underline{\text{exp}} \mid \delta \in \mathcal{D}\} \\ &\cup \{\delta\text{-}\underline{\text{acc}} \mid \delta \in \mathcal{D}\} \\ &\cup \{\delta\text{-}\underline{\text{var}} \mid \delta \in \mathcal{D}\} , \end{aligned}$$

and the partial order on P is described in exactly the same way as was the partial order on the poset of primitive phrase types in Chapter VII.

As before, P generates T, a free type algebra, and Ob T is the disjoint union

$$\{\underline{\text{ns}}\} \cup P \cup \{\tau \Rightarrow \theta \mid \tau, \theta \in T\} .$$

Throughout this chapter we consider comm to be an abbreviation for compl $\Rightarrow$ compl. So all the phrase types from Chapter VII, and more, are available to us. In a sense, we have added the "square root" of comm to the type algebra; we leave it to some algebraist to formalize that intuition.

Again, the language L is a free Λ-algebra generated by a typed set G of linguistic constants with typing function

$$\text{type} \in G \to \text{Ob } T.$$

As before, G is the disjoint union of Op and Op'. Op and the restriction of type to Op are the same as in Chapter VII. However, we add two new linguistic constants, halt and goto, to Op', so that Op' is given by the following table, where $\delta \in \mathcal{D}$ and $\tau \in T$.

$g \in Op'$	type g
skip	comm
;	comm ⇒ comm ⇒ comm
$:=_{\delta}$	δ-acc ⇒ δ-exp ⇒ comm
ifthenelse_{τ}	bool-exp ⇒ τ ⇒ τ ⇒ τ
rec_{τ}	$(\tau \Rightarrow \tau) \Rightarrow \tau$
newvar_{δ}	(δ-var ⇒ comm) ⇒ comm
halt	compl
goto	compl ⇒ comm

Intuitively, halt is the completion to which well-formed programs, which are commands, are applied when they are executed. The constant goto takes a completion ("label value") and turns it into a command; the resulting command when applied to another completion ignores it and passes control to the completion to which the goto was applied.

To see how to desugar blocks with label definitions, some discussion is necessary. Regard

$$\text{let } x{:}\tau \text{ be } m \text{ in } n$$

as syntactic sugar for

$$[\![(\lambda x{:}\tau.n)\ m]\!] ,$$

and regard

$$\text{letrec } x{:}\tau \text{ be } m \text{ in } n$$

as syntactic sugar for

$$[\![(\lambda x{:}\tau.n)\ (\text{rec}_{\tau}\ (\lambda x{:}\tau.m))]\!] .$$

Multiple letrec's can now be desugared (see Reynolds [6]). For instance, regard

$$\begin{array}{l}\text{letrec } x_1{:}\tau_1 \text{ be } m_1 \ \& \\ \qquad\quad x_2{:}\tau_2 \text{ be } m_2 \\ \quad \text{in } n\end{array}$$

as

$$\begin{array}{l}\text{letrec } x_1{:}\tau_1 \text{ be } (\text{letrec } x_2{:}\tau_2 \text{ be } m_2 \text{ in } m_1) \\ \quad \text{in } (\text{letrec } x_2{:}\tau_2 \text{ be } m_2 \text{ in } n).\end{array}$$

The block with label definitions

$$(c_0 \ ; \ \ell_1{:}c_1 \ ; \ \ell_2{:}c_2 \ ; \ . \ . \ . \ ; \ \ell_n{:}c_n) \ ,$$

which has phrase type $\underline{\text{comm}} = \underline{\text{compl}} \Rightarrow \underline{\text{compl}}$ may be regarded as

$$\begin{array}{l}\lambda e{:}\underline{\text{compl}}. \ \text{letrec } \ell_1{:}\underline{\text{compl}} \text{ be } [\![c_1 \ \ell_2]\!] \ \& \\ \qquad\qquad\qquad\quad \ell_2{:}\underline{\text{compl}} \text{ be } [\![c_2 \ \ell_3]\!] \ \& \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad\quad \ell_n{:}\underline{\text{compl}} \text{ be } [\![c_n \ \ e]\!] \\ \qquad\qquad\quad \text{in } [\![c_0 \ \ell_1]\!] \ .\end{array}$$

(Intuitively, e denotes the "normal" exit from the block. The labels $\ell_1, \ldots, \ell_n$ are "used" if commands of the form goto ℓ_i are executed during execution of the block.) Hence, a block such as

$$(c \ ; \ \ell{:}\text{skip})$$

that contains a single label, which is used via the goto ℓ command to accomplish an "abnormal" exit from the block, may

be desugared by means of the following sequence of steps:

(c ; ℓ:skip) ,

λe:compl. (letrec ℓ:compl be ⟦skip e⟧ in ⟦c ℓ⟧),

λe:compl. (letrec ℓ:compl be e in ⟦c ℓ⟧),

λe:compl. (let ℓ:compl be e in ⟦c ℓ⟧),

λe:compl. ((λℓ:compl. ⟦c ℓ⟧ e),

λℓ:compl. ⟦c ℓ⟧ .

The method by which one proceeds from one step to the next above is rather informal. (This indicates that more work needs to be done on the nature of desugaring.) As long as we are being informal, when c contains no occurrences of ℓ, we can add the step

c

to the above list.

The semantics of this language *L* involves specifying

(1) a type algebra *K* derived from a Cartesian closed category,

(2) a functor mng ε $P \to |K|$, which serves to give meaning objects for each primitive phrase type, and which extends to a uniquely defined type algebra homomorphism Mng ε $T \xrightarrow[TypeAlg]{} K$,

(3) for each g ε G, an arrow of $|K|$

semf g ε Env $emp_T \xrightarrow{|K|}$ Mng (type g).

In this chapter we will use the same type algebra *K* as was used in Chapter VII. Thus, *K* is the type algebra canonically associated with the Cartesian closed category

$|K|$, the full subcategory of $\Sigma \Rightarrow Pdom$ whose collection of objects consists of those functors which give domains when applied to store shapes and which give strict, continuous functions when applied to expansions.

Turning now to the definition of $mng \varepsilon P \rightarrow |K|$, we assume that we are given O, a domain of "outputs". The minimal element of O is to be viewed as the undefined output. Perhaps O might be the flat poset consisting of $\perp$ and all strings of data type values. Then again, perhaps not. We do not need to be more explicit about the story of O.

We must give functors

mng compl, mng δ-exp, mng δ-acc, mng δ-var ε $\Sigma \rightarrow Pdom$,

where $\delta \varepsilon \mathcal{D}$, and show they are objects of $|K|$

Let

$$mng\ \underline{compl} = (U\ -) \Rightarrow O\ ,$$

where $U \varepsilon \Sigma \longrightarrow Pdom^{op}$ is the forgetful functor which forgets the second component of an expansion. Thus, for a store shape $X \varepsilon Ob\ \Sigma$,

$$mng\ \underline{compl}\ X = X \Rightarrow O\ ,$$

the set of continuations (functions from stores to outputs) appropriate to the store shape X. Since O is a domain, $X \Rightarrow O$ is a domain, and mng compl applied to store shapes yields domains.

Suppose $\langle \phi, \rho \rangle \varepsilon X \xrightarrow[\Sigma]{} Y$. Then

$$mng\ \underline{compl}\ \langle \phi, \rho \rangle = \phi \Rightarrow 1_O \varepsilon X \Rightarrow O \xrightarrow[Pdom]{} Y \Rightarrow O\ .$$

Let $y \varepsilon Y$. The computation

$$\begin{aligned}(\phi \Rightarrow 1_0)\ \perp_{X\Rightarrow 0}\ y &= (1_0 \circ \perp_{X\Rightarrow 0} \circ \phi)\ y \\ &= \perp_{X\Rightarrow 0}\ (\phi\ y) \\ &= \perp_0 \\ &= \perp_{Y\Rightarrow 0}\ y\end{aligned}$$

shows that mng compl $\langle \phi, \rho \rangle$ is strict. Hence mng compl is an object of $|K|$.

Suppose $\delta \varepsilon \mathcal{D}$; then mng δ-exp $\varepsilon \Sigma \to Pdom$ is defined just as in Chapter VII, i.e.

$$\text{mng } \delta\text{-exp} = (U\ -) \Rightarrow (\text{Val } \delta)_\perp \ ,$$

so that for $X \varepsilon$ Ob Σ,

$$\text{mng } \delta\text{-exp } X = X \Rightarrow (\text{Val } \delta)_\perp$$

(This leads to the idea that a completion is an "output expression," but we will say no more about this intuition.)

Next we will define mng δ-acc for $\delta \varepsilon \mathcal{D}$. Since we can't talk about Mng comm until the definition of mng is complete, we introduce

$$\text{Cmd} = \text{mng compl} \Rightarrow \text{mng compl} \ ,$$

where the heavy arrow is the internal hom functor in $|K|$. Mimicking Chapter VII, let

$$\text{mng } \delta\text{-acc } \varepsilon \Sigma \to Pdom$$

be

$$\text{mng } \delta\text{-acc} = \text{Val } \delta \Rightarrow (\text{Cmd } -) \ ,$$

where this heavy arrow is exponentiation for *Pdom*. Hence, for X ε Ob Σ,

$$\text{mng } \delta\text{-}\underline{\text{acc}}\ X = \text{Val } \delta \Rightarrow \text{Cmd } X\ .$$

Just as in Chapter VII, mng δ-acc is an object of $|K|$.

For $\delta \varepsilon \mathcal{D}$, the definition of mng δ-var follows the pattern laid down in the last chapter. Thus

$$\text{mng } \delta\text{-}\underline{\text{var}} = \text{mng } \delta\text{-}\underline{\text{acc}} \times \text{mng } \delta\text{-}\underline{\text{exp}}\ ,$$

where $\times$ is the binary product in $|K|$. Thus, for a store shape X,

$$\text{mng } \delta\text{-}\underline{\text{var}}\ X = \text{mng } \delta\text{-}\underline{\text{acc}}\ X \times \text{mng } \delta\text{-}\underline{\text{exp}}\ X.$$

The definition of mng on primitive phrase types (objects of P) is now complete. To define mng on implicit conversions between primitive phrase types (arrows of P), we may use the definition for the corresponding entity in Chapter VII, taking care to make the trivial change of using Cmd rather than mng comm. This fully defines mng ε $P \to |K|$ and also defines Mng ε $T \xrightarrow[TypeAlg]{} K$, the type algebra homomorphism whose restriction to P is mng.

The final item on the agenda is giving the semantics of the generating set G of linguistic constants.

First of all, semf g where g ε G is a data type operator (i.e. g ε Op) can be defined the same way as in Chapter VII. The discussion there carries over verbatim.

Next, consider skip. We must give

$$\text{semf skip } \varepsilon \text{ Env emp}_T \xrightarrow{|K|} \text{Mng } \underline{\text{comm}}$$

where Mng $\underline{\text{comm}}$ = Mng $\underline{\text{compl}}$ ⇒ Mng $\underline{\text{compl}}$. By using the Cartesian closed category structure of $|K|$, it suffices to give the corresponding natural transformation

$$\text{skp } \varepsilon \text{ Mng } \underline{\text{compl}} \xrightarrow{|K|} \text{Mng } \underline{\text{compl}}.$$

The obvious definition is

$$\text{skp} = 1_{\text{Mng } \underline{\text{compl}}}.$$

On the list of generators we next find ;. We wish to give

$$\text{semf ; } \varepsilon \text{ Env emp}_T \xrightarrow{|K|} \text{Mng } \underline{\text{comm}} \Rightarrow \text{Mng } \underline{\text{comm}} \Rightarrow \text{Mng } \underline{\text{comm}},$$

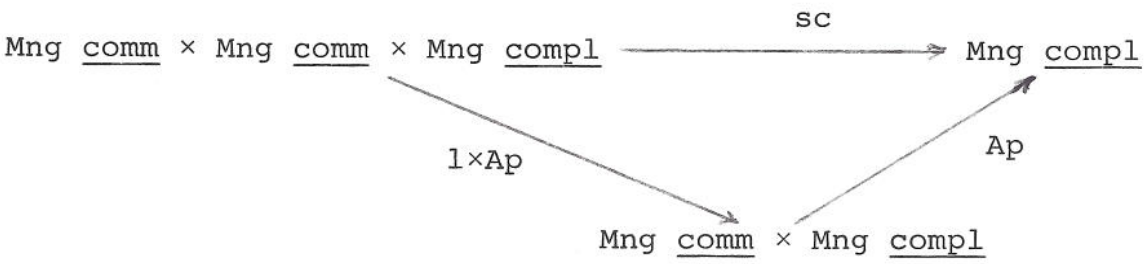
which we do by giving the corresponding natural transformation

$$\text{sc } \varepsilon \text{ Mng } \underline{\text{comm}} \times \text{Mng } \underline{\text{comm}} \times \text{Mng } \underline{\text{compl}} \xrightarrow{|K|} \text{Mng } \underline{\text{compl}}.$$

(Recall that Mng $\underline{\text{comm}}$ = Mng $\underline{\text{compl}}$ ⇒ Mng $\underline{\text{compl}}$.) Let

$$\text{Ap } \varepsilon \text{ Mng } \underline{\text{comm}} \times \text{Mng } \underline{\text{compl}} \xrightarrow{|K|} \text{Mng } \underline{\text{compl}}$$

denote the application arrow. Take sc to be such that



commutes. There is a deep difference here between direct and continuation semantics. Here the second command is applied first. Curiously, regarding comm as a procedural phrase type has made it possible to give the semantics of ; in a way that involves many fewer verification arguments than in Chapter VII. We are making strong and efficient use of the connection between ; and application in a Cartesian closed category.

Suppose $\delta \varepsilon \mathcal{D}$. We will define

$$\text{semf} :=_{\delta} \varepsilon \text{ Env emp}_T \xrightarrow{|K|} \text{Mng } \delta\text{-}\underline{\text{acc}} \Rightarrow \text{Mng } \delta\text{-}\underline{\text{exp}} \Rightarrow \text{Mng } \underline{\text{comm}}$$

by defining the corresponding natural transformation

$$\text{assign}_{\delta} \varepsilon \text{ Mng } \delta\text{-}\underline{\text{acc}} \times \text{Mng } \delta\text{-}\underline{\text{exp}} \times \text{Mng } \underline{\text{compl}} \xrightarrow{|K|} \text{Mng } \underline{\text{compl}}.$$

Let $X \varepsilon \text{Ob } \Sigma$ be a store shape. We seek

$$\text{assign}_{\delta}\ X \varepsilon (\text{Val } \delta \Rightarrow \text{Mng } \underline{\text{comm}}\ X) \times (X \Rightarrow (\text{Val } \delta)_{\perp}) \times (X \Rightarrow 0) \xrightarrow{Pdom} (X \Rightarrow 0)$$

Suppose $a \varepsilon \text{Val } \delta \xrightarrow{Pdom} \text{Mng } \underline{\text{comm}}\ X$, $e \varepsilon X \xrightarrow{Pdom} (\text{Val } \delta)_{\perp}$ and $k \varepsilon X \xrightarrow{Pdom} 0$. Let $a_{\perp}$ denote the strict extension of a. For each $x \varepsilon X$, note that

$$a_{\perp}\ (e\ x) \varepsilon \hom^{\Sigma} X \times \text{Mng } \underline{\text{compl}} \xrightarrow{|K|} \text{Mng } \underline{\text{compl}}\ .$$

Therefore, we may define

$$\text{assign}_{\delta}\ X \langle a, e, k \rangle \varepsilon X \xrightarrow{Pdom} 0$$

by

$$\text{assign}_{\delta}\ X \langle a, e, k \rangle\ x = a_{\perp}\ (e\ x)\ X \langle 1^{\Sigma}_{X}, k \rangle\ x, \text{ where } x \varepsilon X.$$

Note the two occurrences of x on the right-hand side of the

equation, indicating that the store used to produce a value is the store to be altered by the assignment command. The proof that $\text{assign}_\delta\ X$ is continuous is straightforward. We must prove that assign_δ is a natural transformation, so suppose $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$, and consider the diagram

$$\begin{array}{ccc}
(\text{Val}\,\delta \Rightarrow \text{Mng}\ \underline{\text{comm}}\ X)\times(X\Rightarrow(\text{Val}\,\delta)_\perp)\times(X\Rightarrow 0) & \xrightarrow{\text{assign}_\delta\ X} & (X \Rightarrow 0) \\
\downarrow{\scriptstyle (l\Rightarrow \text{Mng}\ \underline{\text{comm}}\langle\phi,\rho\rangle)\times(\phi\Rightarrow l)\times(\phi\Rightarrow l)} & & \downarrow{\scriptstyle \phi\Rightarrow l} \\
(\text{Val}\,\delta \Rightarrow \text{Mng}\ \underline{\text{comm}}\ Y)\times(Y\Rightarrow(\text{Val}\,\delta)_\perp)\times(Y\Rightarrow 0) & \xrightarrow{\text{assign}_\delta\ Y} & (Y \Rightarrow 0).
\end{array}$$

For $y \in Y$, compute:

$$\begin{aligned}
&(\text{assign}_\delta\ Y \circ ((l \Rightarrow \text{Mng}\ \underline{\text{comm}}\langle\phi,\rho\rangle)\times(\phi\Rightarrow l)\times(\phi\Rightarrow l)))\langle a,e,k\rangle\ y \\
&\quad = \text{assign}_\delta\ Y\ \langle \text{Mng comm}\ \langle \phi,\rho\rangle \circ a,\ e\circ\phi,\ k\circ\phi\rangle\ \ y \\
&\quad = (\text{Mng}\ \underline{\text{comm}}\ \langle \phi,\rho\rangle \circ a)_\perp\ ((e\circ\phi)\ y)\ Y\ \langle 1_Y^\Sigma,\ k\circ\phi\rangle\ y \\
&\quad = (\text{Mng}\ \underline{\text{comm}}\ \langle \phi,\rho\rangle \circ a_\perp)\ ((e\circ\phi)\ y)\ \ Y\ \langle 1_Y^\Sigma,\ k\circ\phi\rangle\ y, \\
&\qquad\qquad \text{because Mng}\ \underline{\text{comm}}\ \langle \phi,\rho\rangle \text{ is strict,} \\
&\quad = \text{Mng}\ \underline{\text{comm}}\ \langle \phi,\rho\rangle\ (a_\perp\ (e\ (\phi y)))\ Y\ \langle 1_Y^\Sigma,\ k\circ\phi\rangle\ y \\
&\quad = a_\perp\ (e\ (\phi y))\ Y\ \langle\langle \phi,\rho\rangle,\ k\circ\phi\rangle\ y, \\
&\qquad\qquad \text{since Mng}\ \underline{\text{comm}} = \text{Mng}\ \underline{\text{compl}} \Rightarrow \text{Mng}\ \underline{\text{compl}}, \\
&\quad = a_\perp\ (e\ (\phi y))\ X\ \langle 1_X^\Sigma,\ k\rangle\ (\phi y), \\
&\qquad\qquad \text{by the naturality of } a_\perp\ (e\ (\phi\ y)) \\
&\quad = \text{assign}_\delta\ X\ \langle a,\ e,\ k\rangle\ (\phi\ y) \\
&\quad = (\phi \Rightarrow 1)\ (\text{assign}_\delta\ X\ \langle a,\ e,\ k\rangle)\ y \\
&\quad = ((\phi \Rightarrow 1) \circ \text{assign}_\delta\ X)\ \langle a,\ e,\ k\rangle\ y,
\end{aligned}$$

and this proves assign_δ is natural.

In this chapter we take a different approach to the semantics of ifthenelse_τ, where $\tau \in T$. As before, we seek to define for each $\tau \in T$

$$\text{semf ifthenelse}_\tau \in \text{Env emp}_T \xrightarrow[|K|]{} \text{Mng } \underline{\text{bool-exp}} \Rightarrow \text{Mng } \tau \Rightarrow \text{Mng } \tau \Rightarrow \text{Mng } \tau,$$

and we do this by defining the corresponding

$$\text{cond}_\tau \in \text{Mng } \underline{\text{bool-exp}} \times \text{Mng } \tau \times \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \tau.$$

However, at this point the treatment diverges from that in the last chapter.

We first take up how to define $\text{cond}_{\tau\Rightarrow\theta}$ when cond_θ is known. Let

$$\pi_{\text{forget1}},\ \pi_{\text{forget2}} \in \text{Mng}(\tau\Rightarrow\theta) \times \text{Mng}(\tau\Rightarrow\theta) \times \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng}(\tau\Rightarrow\theta) \times \text{Mng } \tau$$

denote the projections obtained by forgetting the appropriate $\text{Mng } (\tau\Rightarrow\theta)$ factor in the domain.

Let

$$\text{Ap} \in \text{Mng } (\tau\Rightarrow\theta) \times \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \theta$$

be the application arrow in $|K|$. Then

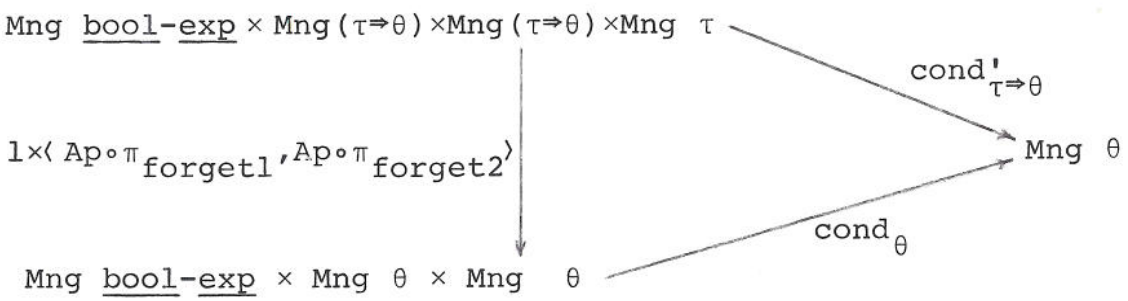
$$\text{Ap}\circ\pi_{\text{forget1}},\ \text{Ap}\circ\pi_{\text{forget2}} \in \text{Mng}(\tau\Rightarrow\theta)\times\text{Mng}(\tau\Rightarrow\theta)\times\text{Mng}\,\tau \xrightarrow[|K|]{} \text{Mng } \theta\ .$$

From the universal mapping property for products we obtain

$$\langle \text{Ap} \circ \pi_{\text{forget1}} , \text{Ap} \circ \pi_{\text{forget2}} \rangle \in$$

$$\text{Mng}(\tau\Rightarrow\theta) \times \text{Mng}(\tau\Rightarrow\theta) \times \text{Mng } \tau \xrightarrow[|K|]{} \text{Mng } \theta \times \text{Mng } \theta.$$

Now define $\text{cond}'_{\tau\Rightarrow\theta}$ so that the diagram



commutes in $|K|$. Thus assuming that cond_θ is known, we can define

$$\mathrm{cond}_{\tau\Rightarrow\theta} \varepsilon \text{ Mng } \underline{\text{bool-exp}} \times \text{Mng}(\tau\Rightarrow\theta)\times\text{Mng}(\tau\Rightarrow\theta) \xrightarrow[|K|]{} \text{Mng } \tau \Rightarrow \text{Mng } \theta$$

to be the abstraction of

$$\mathrm{cond}'_{\tau\Rightarrow\theta} \varepsilon \text{ Mng } \underline{\text{bool-exp}} \times \text{Mng}(\tau\Rightarrow\theta)\times\text{Mng}(\tau\Rightarrow\theta)\times\text{Mng}\tau \xrightarrow[|K|]{} \text{Mng } \theta.$$

So to define cond_τ for all phrase types τ, it suffices to define cond_π for primitive phrase types π. We now proceed to do this.

(1) π = $\underline{\text{compl}}$. Let $X \varepsilon \mathrm{Ob}\ \Sigma$ be a store shape. Define

$$\mathrm{cond}_{\underline{\text{compl}}}\ X \varepsilon$$

$$\text{Mng } \underline{\text{bool-exp}}\ X \times \text{Mng } \underline{\text{compl}}\ X \times \text{Mng } \underline{\text{compl}}\ X \xrightarrow[Pdom]{} \text{Mng } \underline{\text{compl}}\ X$$

by

$$\mathrm{cond}_{\underline{\text{compl}}}\ X\langle b, k_1, k_2\rangle\ x = \begin{cases} k_1\ x & \text{if} \quad b\ x = \mathit{true} \\ k_2\ x & \text{if} \quad b\ x = \mathit{false} \\ \perp & \text{if} \quad b\ x = \perp, \end{cases}$$

where $b \varepsilon X \xrightarrow[Pdom]{} (\text{Val } \underline{\text{bool}})_\perp$, $k_1, k_2 \varepsilon X \xrightarrow[Pdom]{} 0$, $x \varepsilon X$.

The proof that $\mathrm{cond}_{\underline{\text{compl}}}\ X$ is continuous is routine.

We must check that $\text{cond}_{\underline{\text{compl}}}$ is a natural transformation, so let $\langle \phi, \rho \rangle \varepsilon \; X \xrightarrow[\Sigma]{} Y$ be an expansion and consider the diagram

$$\begin{array}{ccc} (X \Rightarrow (\text{Val } \underline{\text{bool}})_\perp) \times (X \Rightarrow \mathcal{O}) \times (X \Rightarrow \mathcal{O}) & \xrightarrow{\text{cond}_{\underline{\text{compl}}}\ X} & (X \Rightarrow \mathcal{O}) \\ \downarrow (\phi \Rightarrow 1) \times (\phi \Rightarrow 1) \times (\phi \Rightarrow 1) & & \downarrow \phi \Rightarrow 1 \\ (Y \Rightarrow (\text{Val } \underline{\text{bool}})_\perp) \times (Y \Rightarrow \mathcal{O}) \times (Y \Rightarrow \mathcal{O}) & \xrightarrow{\text{cond}_{\underline{\text{compl}}}\ Y} & (Y \Rightarrow \mathcal{O}) \end{array} .$$

For $y \varepsilon Y$, compute:

$$\begin{aligned} &(\text{cond}_{\underline{\text{compl}}}\ Y \circ ((\phi \Rightarrow 1) \times (\phi \Rightarrow 1) \times (\phi \Rightarrow 1)))\ \langle b, k_1, k_2 \rangle\ y \\ &= \text{cond}_{\underline{\text{compl}}}\ Y\ \langle b \circ \phi, k_1 \circ \phi, k_2 \circ \phi \rangle\ y \\ &= \begin{cases} k_1\ (\phi\ y) & \text{if} \quad b(\phi\ y) = \mathit{true} \\ k_2\ (\phi\ y) & \text{if} \quad b(\phi\ y) = \mathit{false} \\ \perp & \text{if} \quad b(\phi\ y) = \perp \end{cases} \\ &= \text{cond}_{\underline{\text{compl}}}\ X\ \langle b, k_1, k_2 \rangle\ (\phi\ y) \\ &= (\text{cond}_{\underline{\text{compl}}}\ X\ \langle b, k_1, k_2 \rangle \circ \phi)\ y \\ &= ((\phi \Rightarrow 1) \circ \text{cond}_{\underline{\text{compl}}}\ X)\ \langle b, k_1, k_2 \rangle\ y , \end{aligned}$$

so the diagram commutes, and $\text{cond}_{\underline{\text{compl}}}$ is indeed natural.

(2) $\pi = \delta\text{-}\underline{\text{exp}}$ where $\delta \varepsilon \mathcal{D}$. For $X \varepsilon \text{Ob } \Sigma$, define

$$\text{cond}_{\delta\text{-}\underline{\text{exp}}}\ X \varepsilon$$
$$\text{Mng } \underline{\text{bool}}\text{-}\underline{\text{exp}}\ X \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ X \times \text{Mng } \delta\text{-}\underline{\text{exp}}\ X \xrightarrow{Pdom} \text{Mng } \delta\text{-}\underline{\text{exp}}\ X$$

by

$$\text{cond}_{\delta\text{-}\underline{\text{exp}}}\ X\ \langle b,\ e_1,\ e_2\rangle\ x = \begin{cases} e_1\ x & \text{if } b\ x = \mathit{true} \\ e_2\ x & \text{if } b\ x = \mathit{false} \\ \perp & \text{if } b\ x = \perp \end{cases}$$

where $b \in X \xrightarrow[Pdom]{} (\text{Val } \underline{\text{bool}})_\perp$, $e_1, e_2 \in X \xrightarrow[Pdom]{} (\text{Val}\,\delta)_\perp$, $x \in X$. The necessary verifications for $\text{cond}_{\delta\text{-}\underline{\text{exp}}}$ are completely parallel to those for $\text{cond}_{\underline{\text{compl}}}$.

(3) $\pi = \delta\text{-}\underline{\text{acc}}$ where $\delta \in \mathcal{D}$. Note that the remarks made earlier make it clear that $\text{cond}_{\underline{\text{comm}}} = \text{cond}_{\underline{\text{compl}} \Rightarrow \underline{\text{compl}}}$ is defined. Suppose $X \in \text{Ob}\ \Sigma$ is a store shape. We can define

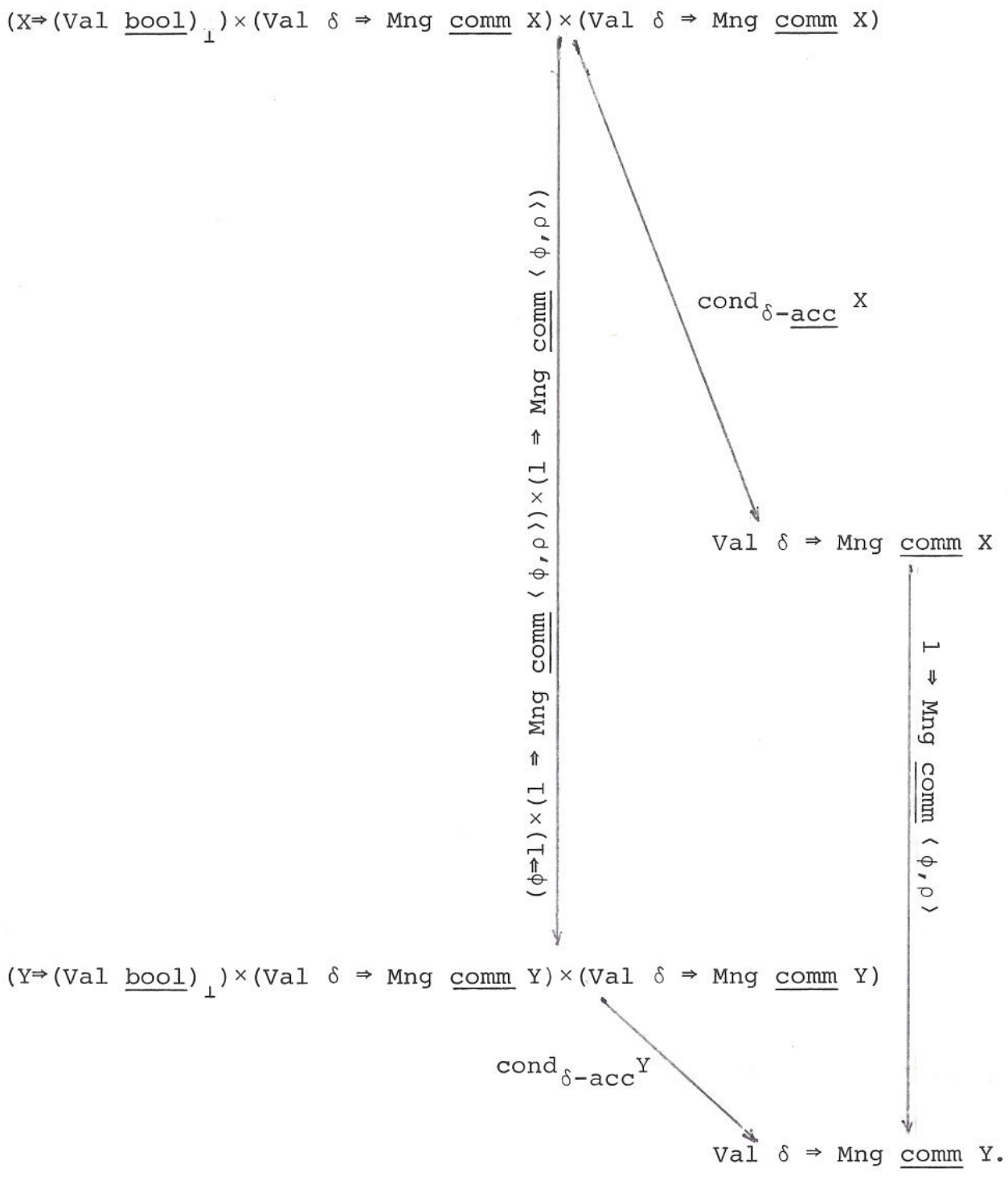
$$\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X\ \varepsilon$$

$$\text{Mng } \underline{\text{bool}}\text{-}\underline{\text{exp}}\ X \times \text{Mng } \delta\text{-}\underline{\text{acc}}\ X \times \text{Mng } \delta\text{-}\underline{\text{acc}}\ X \xrightarrow[Pdom]{} \text{Mng } \delta\text{-}\underline{\text{acc}}\ X$$

by

$$\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X\ \langle b,\ a_1,\ a_2\rangle\ v = \text{cond}_{\underline{\text{comm}}}\ X\ \langle b,\ a_1\ v,\ a_2\ v\rangle$$

where $b \in X \xrightarrow[Pdom]{} (\text{Val } \underline{\text{bool}})_\perp$, $a_1, a_2 \in \text{Val}\ \delta \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}}\ X$ and $v \in \text{Val}\ \delta$. It is easy to see that $\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X$ is continuous. To see that $\text{cond}_{\delta\text{-}\underline{\text{acc}}}$ is a natural transformation, let $\langle \phi, \rho\rangle \in X \xrightarrow[\Sigma]{} Y$ and consider the commutativity of the diagram



To establish this, we compute

$(\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ Y \circ ((\phi \Rightarrow 1) \times (1 \Rightarrow \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle)) \times$

$(1 \Rightarrow \text{Mng}\ \underline{\text{comm}}\ \ \phi, \rho\ )))\ \langle b, a_1, a_2 \rangle\ v$

$= \text{cond}_{\delta\text{-}\underline{\text{acc}}}\ Y \langle b \circ \phi,\ \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle \circ a_1,\ \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle \circ a_2 \rangle\ v$

$= \text{cond}_{\underline{\text{comm}}}\ Y\ \langle b \circ \phi,\ \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle\ (a_1\ v),\ \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle (a_2\ v) \rangle$

$= \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle\ (\text{cond}_{\underline{\text{comm}}}\ X\ \langle b,\ a_1\ v,\ a_2\ v \rangle\ ),$

by the naturality of $\text{cond}_{\underline{\text{comm}}}$,

$= \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle\ (\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X\ \langle b,\ a_1,\ a_2 \rangle\ v)$

$= (\text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle \circ \text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X\ \langle b,\ a_1,\ a_2 \rangle)\ v$

$= (1 \Rightarrow \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle)\ (\text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X\ \langle b,\ a_1,\ a_2 \rangle)\ v$

$= ((1 \Rightarrow \text{Mng}\ \underline{\text{comm}}\ \langle \phi, \rho \rangle) \circ \text{cond}_{\delta\text{-}\underline{\text{acc}}}\ X)\ \langle b,\ a_1,\ a_2 \rangle\ v.$

Hence $\text{cond}_{\delta\text{-}\underline{\text{acc}}}$ is a natural transformation.

(4) $\pi = \delta\text{-}\underline{\text{var}}$ where $\delta \in \mathcal{D}$. Recall that

$$\text{Mng}\ \delta\text{-}\underline{\text{var}} = \text{Mng}\ \delta\text{-}\underline{\text{acc}} \times \text{Mng}\ \delta\text{-}\underline{\text{exp}}.$$

The projection onto the first (respectively second) component of Mng δ-var induces a projection

$$\pi_{\delta\text{-}\underline{\text{acc}}} \in \text{Mng}\ \underline{\text{bool}}\text{-}\underline{\text{exp}} \times \text{Mng}\ \delta\text{-}\underline{\text{var}} \times \text{Mng}\ \delta\text{-}\underline{\text{var}}$$

$$\xrightarrow[|K|]{} \text{Mng}\ \underline{\text{bool}}\text{-}\underline{\text{exp}} \times \text{Mng}\ \delta\text{-}\underline{\text{acc}} \times \text{Mng}\ \delta\text{-}\underline{\text{acc}}$$

(respectively

$$\pi_{\delta\text{-}\underline{exp}} \varepsilon \text{ Mng } \underline{bool}\text{-}\underline{exp} \times \text{Mng } \delta\text{-}\underline{var} \times \text{Mng } \delta\text{-}\underline{var}$$

$$\xrightarrow{|K|} \text{Mng } \underline{bool}\text{-}\underline{exp} \times \text{Mng } \delta\text{-}\underline{exp} \times \text{Mng } \delta\text{-}\underline{exp}) \ .$$

We may compose the projections $\pi_{\delta\text{-}\underline{acc}}$ and $\pi_{\delta\text{-}\underline{exp}}$ with $\text{cond}_{\delta\text{-}\underline{acc}}$ and $\text{cond}_{\delta\text{-}\underline{exp}}$. Let the natural transformation

$$\text{cond}_{\delta\text{-}\underline{var}} \varepsilon \text{ Mng } \underline{bool}\text{-}\underline{exp} \times \text{Mng } \delta\text{-}\underline{var} \times \text{Mng } \delta\text{-}\underline{var}$$

$$\xrightarrow{|K|} \text{Mng } \delta\text{-}\underline{acc} \times \text{Mng } \delta\text{-}\underline{exp}$$

be

$$\text{cond}_{\delta\text{-}\underline{var}} = \langle \text{cond}_{\delta\text{-}\underline{acc}} \circ \pi_{\delta\text{-}\underline{acc}} \ , \ \text{cond}_{\delta\text{-}\underline{exp}} \circ \pi_{\delta\text{-}\underline{exp}} \rangle,$$

the arrow satisfying the obvious universal mapping property. This finishes the definition of semf ifthenelse_τ, where $\tau \varepsilon T$.

The semantics of rec_τ is the same for both continuation semantics and direct semantics. The definition of semf rec_τ, where $\tau \varepsilon T$, given in Chapter VII can be used here verbatim.

In Chapter VII we defined some objects related to the category of Σ of store shapes. Some of these entities, which were used in giving direct semantics for newvar_δ, are again useful for giving continuation semantics for newvar_δ. In fact, it is only the definition of the semantics of newvar_δ that seems to require properties (Σ1), (Σ2), and (Σ3); the semantics of the other linguistic constants can be given with $\mathcal{S}et^{op}$ as the category of store shapes (thus omitting the second component ρ of the arrows $\langle \phi, \rho \rangle$ of Σ). In this chapter, the following objects are

defined exactly as they were in Chapter VII:

(1) the functor $\text{Newshape}_\delta \in \Sigma \to \Sigma$, which is such that, for $X \in \text{Ob}\ \Sigma$,

$$\text{Newshape}_\delta\ X = X \times \text{Val}\ \delta,$$

(2) the functions

$$\text{enter}_\delta\ X \in X \to X \times \text{Val}\ \delta,$$

$$\text{exit}_\delta\ X \in X \times \text{Val}\ \delta \to X$$

used upon entering and exiting from blocks (recall that $\text{enter}_\delta\ X\ x = \langle x, \text{init}\ \delta \rangle$ and $\text{exit}_\delta\ X \langle x, v \rangle = x$),

and

(3) the natural transformation

$$\text{Exit}_\delta \in 1_\Sigma \xrightarrow[\Sigma \Rightarrow \Sigma]{} \text{Newshape}_\delta\ ,$$

which is such that for $X \in \text{Ob}\ \Sigma$,

$$\text{Exit}_\delta\ X = \langle \text{exit}_\delta\ X, \nu_\delta\ X \rangle \in X \xrightarrow[\Sigma]{} X \times \text{Val}\ \delta.$$

Furthermore, Lemma 7.3 and Lemma 7.4 are still valid.

However, for continuation semantics, the notion of "canonical local δ-variable for a store shape X" needs revision. For $X \in \text{Ob}\ \Sigma$ let

$$\text{localvar}_\delta\ X \in \text{Mng}\ \delta\text{-}\underline{\text{var}}\ (X \times \text{Val}\ \delta)$$

be

$$\text{localvar}_\delta\ X = \langle a_{\delta,X}\ ,\ e_{\delta,X} \rangle$$

where

$$a_{\delta,X} \varepsilon \text{ Val } \delta \xrightarrow[Pdom]{} \text{Mng } \underline{\text{comm}} \ (X \times \text{Val } \delta), \quad \text{and}$$

$$e_{\delta,X} \varepsilon \ X \times \text{Val } \delta \xrightarrow[Pdom]{} (\text{Val } \delta)_{\perp}$$

are defined below. Let $e_{\delta,X}$ be given by

$$e_{\delta,X} \langle x, v \rangle = v$$

where $x \varepsilon X$, $v \varepsilon$ Val δ (just as in Chapter VII).

For $v \varepsilon$ Val δ, we must give a natural transformation

$$a_{\delta,X} \ v \varepsilon \ \hom^{\Sigma} (X \times \text{Val } \delta) \times \text{Mng } \underline{\text{compl}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{compl}}.$$

Hence, for a store shape $Y \varepsilon$ Ob Σ, let

$$a_{\delta,X} \ v \ Y \varepsilon \ (X \times \text{Val } \delta \xrightarrow[\Sigma]{} Y) \times (Y \Rightarrow 0) \xrightarrow[Pdom]{} (Y \Rightarrow 0)$$

be given by

$$a_{\delta,X} \ v \ Y \langle\langle \phi, \rho \rangle, k \rangle \ y = k \ (\rho \langle \text{exit}_{\delta} \ X \ (\phi \ y), v \rangle \ y).$$

The idea behind the definition is that we apply the continuation k to an altered state derived from y by assigning v to the canonical local δ-variable. Verification that $a_{\delta,X} \ v \ Y$ is continuous is routine. We must verify that $a_{\delta,X} \ v$ is a natural transformation. Suppose that $\langle \phi' , \rho' \rangle \varepsilon \ Y \xrightarrow[\Sigma]{} Z$. Consider the diagram

$$\begin{array}{ccc}
(X \times \text{Val } \delta \xrightarrow[\Sigma]{} Y) \times (Y \Rightarrow 0) & \xrightarrow{a_{\delta,X} \ v \ Y} & (Y \Rightarrow 0) \\
\Big\downarrow \hom^{\Sigma} (X \times \text{Val}\delta)\langle \phi', \rho' \rangle \times (\phi' \Rightarrow 1) & & \Big\downarrow \phi' \Rightarrow 1 \\
(X \times \text{Val } \delta \xrightarrow[\Sigma]{} Z) \times (Z \Rightarrow 0) & \xrightarrow{a_{\delta,X} \ v \ Z} & (Z \Rightarrow 0) \quad .
\end{array}$$

For $\langle \phi,\rho \rangle \in X \times Val\ \delta \xrightarrow[\Sigma]{} Y$, $k \in Y \xrightarrow[Pdom]{} 0$, $z \in Z$, we have

$(a_{\delta,X}\ v\ Z \circ (hom^{\Sigma}\ (X \times Val\delta)\langle \phi',\rho' \rangle\ (\phi' \Rightarrow 1)))\ \langle\langle \phi,\rho \rangle, k \rangle\ z$

$= a_{\delta,X}\ v\ Z\ \langle\langle \phi',\rho' \rangle \circ \langle \phi,\rho \rangle, k \circ \phi' \rangle\ z$

$= a_{\delta,X}\ v\ Z\ \langle\langle \phi \circ \phi', D_Z \circ (\phi' \to \rho') \circ \rho \rangle, k \circ \phi' \rangle\ z$

$= (k \circ \phi')((D_Z \circ (\phi' \to \rho') \circ \rho)\langle exit_\delta\ X\ ((\phi \circ \phi')\ z), v \rangle\ z)$

$= (k \circ \phi')((D_Z((\phi' \to \rho')(\rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle))))\ z)$

$= (k \circ \phi')((D_Z(\rho' \circ \rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle \circ \phi'))\ z)$

$= (k \circ \phi')((\rho' \circ\ \rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle \circ \phi')\ z\ z)$

$= (k \circ \phi')(\rho'\ (\rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle\ (\phi'\ z))\ z)$

$= k\ (\phi'\ (\rho'\ (\rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle\ (\phi'\ z))\ z))$

$= k\ (\rho\ \langle exit_\delta\ X\ (\phi\ (\phi'\ z)), v \rangle\ (\phi'\ z))$, by $(\Sigma 1)$,

$= a_{\delta,X}\ v\ Y\ \langle\langle \phi,\ \rho \rangle, k \rangle\ (\phi'\ z)$

$= (a_{\delta,X}\ v\ Y\langle\langle \phi,\ \rho \rangle, k \rangle \circ \phi')\ z$

$= (\phi' \Rightarrow 1)(a_{\delta,X}\ v\ Y\ \langle\langle \phi,\rho \rangle, k \rangle)\ z$

$= ((\phi' \Rightarrow 1) \circ a_{\delta,X}\ v\ Y)\ \langle\langle \phi,\rho \rangle, k \rangle\ z,$

which shows that the diagram commutes, and $a_{\delta,X}\ v$ is a natural transformation.

Of course, it is necessary to prove a modified version of Lemma 7.5. Recall that $\langle \phi_\delta, \rho_\delta \rangle$ is another notation for $\text{Newshape}_\delta \langle \phi, \rho \rangle$ when $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$. Note the use of property (Σ2) in the proof.

Lemma 8.1: Let $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$. Then

$$\text{Mng } \delta\text{-var } \langle \phi_\delta, \rho_\delta \rangle \ (\text{localvar}_\delta \ X) = \text{localvar}_\delta \ Y.$$

Proof: Since

$$\text{Mng } \delta\text{-var} = \text{Mng } \delta\text{-acc} \times \text{Mng } \delta\text{-exp},$$

$$\text{localvar}_\delta \ X = \langle a_{\delta,X} \ , \ e_{\delta,X} \rangle \ , \text{ and}$$

$$\text{localvar}_\delta \ Y = \langle a_{\delta,Y} \ , \ e_{\delta,Y} \rangle \ ,$$

it suffices to show

$$\text{Mng } \delta\text{-acc } \langle \phi_\delta, \rho_\delta \rangle \ \ a_{\delta,X} = a_{\delta,Y} \ , \text{ and}$$

$$\text{Mng } \delta\text{-exp } \langle \phi_\delta, \rho_\delta \rangle \ \ e_{\delta,X} = e_{\delta,Y} \ .$$

Let $v \in \text{Val } \delta$, $Z \in \text{Ob } \Sigma$, $\langle \phi', \rho' \rangle \in Y \times \text{Val } \delta \xrightarrow[\Sigma]{} Z$, $k \in Z \xrightarrow[Pdom]{} 0$ and $z \in Z$. Observe that if we let $\phi' \ z = \langle y, w \rangle$ then we can carry out the following computation:

$$\begin{aligned}
&\rho_\delta \langle \text{exit}_\delta \ X \ (\phi_\delta \ (\phi' \ z)), \ v \rangle \ (\phi' \ z) \\
&\quad = \rho_\delta \langle \text{exit}_\delta \ X \ (\phi_\delta \ \langle y, \ w \rangle), \ v \rangle \ \langle y, \ w \rangle \\
&\quad = \rho_\delta \langle \text{exit}_\delta \ X \ \langle \phi \ y, \ w \rangle, \ v \rangle \ \langle y, \ w \rangle \\
&\quad = \rho_\delta \langle \phi \ \ y, \ v \rangle \ \langle y, \ w \rangle \\
&\quad = \langle \rho \ (\phi \ y) \ y, \ v \rangle
\end{aligned}$$

$$= \langle y, v\rangle, \quad \text{by } (\Sigma 2),$$

$$= \langle \mathrm{exit}_\delta\ Y \langle y, w\rangle\ , v\rangle$$

$$= \langle \mathrm{exit}_\delta\ Y\ (\phi'\ z), v\rangle\ .$$

Thus,

$$\mathrm{Mng}\ \delta\text{-}\underline{\mathrm{acc}} \langle \phi_\delta, \rho_\delta\rangle\ a_{\delta,X}\ v\ Z \langle\langle \phi', \rho'\rangle, k\rangle\ z$$

$$= \mathrm{Mng}\ \underline{\mathrm{comm}} \langle \phi_\delta, \rho_\delta\rangle\ (a_{\delta,X}\ v)\ Z \langle\langle \phi', \rho'\rangle, k\rangle\ z$$

$$= a_{\delta,X}\ v\ Z \langle\langle \phi', \rho'\rangle \circ \langle \phi_\sigma, \rho_\delta\rangle, k\rangle\ z,$$

since $\mathrm{Mng}\ \underline{\mathrm{comm}} = \mathrm{Mng}\ \underline{\mathrm{compl}} \Rightarrow \mathrm{Mng}\ \underline{\mathrm{compl}}$,

$$= a_{\delta,X}\ v\ Z \langle\langle \phi_\delta \circ \phi', D_Z \circ (\phi' \to \rho') \circ \rho_\delta\rangle\ , k\rangle\ z$$

$$= k\ ((D_Z \circ (\phi' \to \rho') \circ \rho_\delta) \langle \mathrm{exit}_\delta\ X\ ((\phi_\delta \circ \phi')z), v\rangle\ z)$$

$$= k\ (D_Z\ ((\phi' \to \rho')(\rho_\delta \langle \mathrm{exit}_\delta\ X\ (\phi_\delta\ (\phi'\ z)), v\rangle))\ z)$$

$$= k\ (D_Z\ (\rho' \circ \rho_\delta \langle \mathrm{exit}_\delta\ X\ (\phi_\delta(\phi'\ z)), v\rangle \circ \phi')\ z)$$

$$= k\ ((\rho' \circ \rho_\delta \langle \mathrm{exit}_\delta\ X\ (\phi_\delta(\phi'\ z)), v\rangle \circ \phi')\ z\ z\ )$$

$$= k\ (\rho'\ (\rho_\delta \langle \mathrm{exit}_\delta\ X\ (\phi_\delta(\phi'\ z)), v\rangle\ (\phi'\ z))\ z)$$

$$= k\ (\rho' \langle \mathrm{exit}_\delta\ Y\ (\phi'\ z), v\rangle\ z),$$

by the preceding computation,

$$= a_{\delta,Y}\ v\ Z \langle\langle \phi', \rho'\rangle, k\rangle\ z\ .$$

This demonstrates

$$\mathrm{Mng}\,\delta\text{-}\underline{\mathrm{acc}} \langle \phi_\delta, \rho_\delta\rangle\ \ a_{\delta,X} = a_{\delta,Y}\ .$$

The proof of the other equation is very simple and can be found in the proof of Lemma 7.5. □

We are now ready to give the semantics of $newvar_\delta$, where $\delta \in \mathcal{D}$. The aim is to give

$$semf\ newvar_\delta \in Env\ emp_T \xrightarrow[|K|]{} (Mng\ \delta\text{-}\underline{var} \Rightarrow Mng\ \underline{comm}) \Rightarrow Mng\ \underline{comm},$$

and we do this by giving the corresponding

$$nv_\delta \in (Mng\ \delta\text{-}\underline{var} \Rightarrow Mng\ \underline{comm}) \times Mng\ \underline{compl} \xrightarrow[|K|]{} Mng\ \underline{compl}.$$

Hence, for $X \in Ob\ \Sigma$, let

$$nv_\delta\ X \in (Mng\ \delta\text{-}\underline{var} \Rightarrow Mng\ \underline{comm})\ X \times (X \Rightarrow 0) \xrightarrow[Pdom]{} (X \Rightarrow 0)$$

be given by

$$nv_\delta\ X \langle \eta, k \rangle = \eta(X \times Val\delta)\langle Exit_\delta\ X,\ localvar_\delta\ X \rangle (X \times Val\delta)\langle 1^{\Sigma}_{X \times Val\delta},\ k \circ exit_\delta\ X \rangle \circ enter_\delta\ X,$$

where $\eta \in hom^{\Sigma}\ X \times Mng\ \delta\text{-}\underline{var} \xrightarrow[|K|]{} Mng\ \underline{comm}$ and $k \in X \xrightarrow[Pdom]{} 0$. Think of η as telling how to construct a command out of a δ-variable. Then $nv_\delta\ X \langle \eta, k \rangle$ means enter a new block, execute a command constructed with η from the canonical local δ-variable, then continue with $k \circ exit_\delta\ X$ (i.e. exit from the block and continue with k). It is straightforward to show that $nv_\delta\ X$ is continuous. We turn our attention to showing that nv_δ is a natural transformation. Suppose $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$.

The following technical lemma is useful.

Lemma 8.2: Let $\langle \phi, \rho \rangle \in X \xrightarrow[\Sigma]{} Y$ and $k \in X \xrightarrow[Pdom]{} 0$. Then

$$\text{Mng } \underline{\text{compl}} \langle \phi_\delta, \rho_\delta \rangle (k \circ \text{exit}_\delta\ X) = k \circ \phi \circ \text{exit}_\delta\ Y.$$

Proof: Both sides of the equation are elements of $Y \times \text{Val}\ \delta \xrightarrow[Pdom]{} 0$. Suppose $y \in Y$ and $v \in \text{Val}\ \delta$. The desired equality follows from the computation

$$\begin{aligned}
&\text{Mng } \underline{\text{compl}} \langle \phi_\delta, \rho_\delta \rangle\ (k \circ \text{exit}_\delta\ X)\ \langle y, v \rangle \\
&\quad = (k \circ \text{exit}_\delta\ X \circ \phi_\delta)\ \langle y, v \rangle \\
&\quad = k\ (\text{exit}_\delta\ X\ (\phi_\delta\ \langle y, v \rangle)) \\
&\quad = k\ (\text{exit}_\delta\ X\ \langle \phi\ y, v \rangle) \\
&\quad = k\ (\phi\ y) \\
&\quad = k\ (\phi\ (\text{exit}_\delta\ Y\ \langle y, v \rangle)) \\
&\quad = (k \circ \phi \circ \text{exit}_\delta\ Y)\ \langle y, v \rangle\ .
\end{aligned}$$

□

We are interested in the commutativity of

$$\begin{array}{ccc}
(\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}})\ X \times (X \Rightarrow 0) & \xrightarrow{\text{nv}_\delta\ X} & X \Rightarrow 0 \\
\Big\downarrow (\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) \langle \phi, \rho \rangle \times (\phi \Rightarrow 1) & & \Big\downarrow \phi \Rightarrow 1 \\
(\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}})\ Y \times (Y \Rightarrow 0) & \xrightarrow{\text{nv}_\delta\ Y} & Y \Rightarrow 0\ .
\end{array}$$

Let $\eta \in \hom^{\Sigma} X \times \text{Mng } \delta\text{-}\underline{\text{var}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{comm}}$, $k \in X \xrightarrow[Pdom]{} 0$ and $y \in Y$. Then

$$\begin{aligned}
&(\text{nv}_\delta\ Y \circ ((\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) \langle \phi, \rho \rangle\ (\phi \Rightarrow 1))) \langle \eta, k \rangle\ y \\
&\quad = \text{nv}_\delta\ Y \langle (\text{Mng } \delta\text{-}\underline{\text{var}} \Rightarrow \text{Mng } \underline{\text{comm}}) \langle \phi, \rho \rangle\ \eta, k \circ \phi \rangle\ y
\end{aligned}$$

$= \mathrm{Mng}(\delta\text{-}\underline{\mathrm{var}} \Rightarrow \mathrm{Mng}\ \underline{\mathrm{comm}})\langle \phi,\rho\rangle\ \eta\ (Y\times \mathrm{Val}\ \delta)\langle \mathrm{Exit}_\delta\ Y,\ \mathrm{localvar}_\delta\ Y\rangle$

$(Y \times \mathrm{Val}\ \delta)\langle 1^{\Sigma}_{Y\times \mathrm{Val}\delta},\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y)$

$= \eta\ (Y \times \mathrm{Val}\ \delta)\langle \mathrm{Exit}_\delta\ Y \circ \langle \phi,\rho\rangle,\ \mathrm{localvar}_\delta\ Y\rangle$

$(Y \times \mathrm{Val}\ \delta)\langle 1^{\Sigma}_{Y\times \mathrm{Val}\delta},\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y),$

by the definition of $\Rightarrow$ in $|K|$,

$= \eta\ (Y \times \mathrm{Val}\ \delta)\langle\langle \phi_\delta,\rho\rangle \circ\ \mathrm{Exit}_\delta\ X,\ \mathrm{localvar}_\delta\ Y\rangle$

$(Y \times \mathrm{Val}\ \delta)\langle 1^{\Sigma}_{Y\times \mathrm{Val}\delta},\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y),$

by Lemma 7.4 ,

$= \eta\ (Y \times \mathrm{Val}\ \delta)$

$((\mathrm{hom}^{\Sigma}\ X\ \langle \phi_\delta,\rho_\delta\rangle \times \mathrm{Mng}\ \delta\text{-}\underline{\mathrm{var}}\langle \phi_\delta,\rho_\delta\rangle)\langle \mathrm{Exit}_\delta\ X,\ \mathrm{localvar}_\delta\ X\rangle)$

$(Y \times \mathrm{Val}\ \delta)\langle 1^{\Sigma}_{Y\times \mathrm{Val}\delta},\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y),$

by Lemma 8.1,

$= \mathrm{Mng}\ \underline{\mathrm{comm}}\ \langle \phi_\delta,\rho_\delta\rangle(\eta(X \times \mathrm{Val}\ \delta)\langle \mathrm{Exit}_\delta\ X,\ \mathrm{localvar}_\delta\ X\rangle)$

$(Y \times \mathrm{Val}\delta)\langle 1^{\Sigma}_{Y\times \mathrm{Val}\delta},\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y),$

by the naturality of η,

$= \eta\ (X \times \mathrm{Val}\ \delta)\langle \mathrm{Exit}_\delta\ X,\ \mathrm{localvar}_\delta\ X\rangle(Y \times \mathrm{Val}\ \delta)$

$\langle\langle \phi_\delta,\rho_\delta\rangle,\ k \circ \phi \circ \mathrm{exit}_\delta\ Y\rangle(\mathrm{enter}_\delta\ Y\ y),$

because Mng $\underline{\mathrm{comm}}$ = Mng $\underline{\mathrm{compl}} \Rightarrow$ Mng $\underline{\mathrm{compl}}$,

$= \eta\ (X \times \mathrm{Val}\delta)\langle \mathrm{Exit}_\delta\ X,\ \mathrm{localvar}_\delta\ X\rangle(Y \times \mathrm{Val}\ \delta)$

$((\mathrm{hom}^{\Sigma}(X\times \mathrm{Val}\delta)\langle \phi_\delta,\rho_\delta\rangle \times \mathrm{Mng}\ \underline{\mathrm{compl}}\langle \phi_\delta,\rho_\delta\rangle)\langle 1^{\Sigma}_{X\times \mathrm{Val}\delta}, k \circ \mathrm{exit}_\delta X\rangle)$

$(\mathrm{enter}_\delta\ Y\ y),$ by Lemma 8.2,

$= \text{Mng } \underline{\text{compl}} \langle \phi_\delta, \rho_\delta \rangle$

$(\eta(X \times \text{Val}\delta)\langle \text{Exit}_\delta X, \text{localvar}_\delta X\rangle (X \times \text{Val}\delta)\langle 1^{\Sigma}_{X \times \text{Val}\delta}, k \circ \text{exit}_\delta X\rangle)$

$(\text{enter}_\delta\ Y\ y)$,

by the naturality of $\eta(X \times \text{Val}\delta)\langle \text{Exit}_\delta X, \text{localvar}_\delta X\rangle$,

$= (\eta(X \times \text{Val}\delta)\langle \text{Exit}_\delta X, \text{localvar}_\delta X\rangle (X \times \text{Val}\delta)\langle 1^{\Sigma}_{X \times \text{Val}\delta}, k \circ \text{exit}_\delta X\rangle \circ \phi_\delta)$

$(\text{enter}_\delta\ Y\ y)$

$= \eta(X \times \text{Val}\ \delta)\langle \text{Exit}_\delta\ X, \text{localvar}_\delta\ X\rangle (X \times \text{Val}\delta)\langle 1^{\Sigma}_{X \times \text{Val}\delta}, k \circ \text{exit}_\delta\ X\rangle$

$(\phi_\delta(\text{enter}_\delta\ Y\ y))$

$= \eta(X \times \text{Val}\delta)\langle \text{Exit}_\delta X, \text{localvar}_\delta\ X\rangle (X \times \text{Val}\delta)\langle 1^{\Sigma}_{X \times \text{Val}\delta}, k \circ \text{exit}_\delta\ X\rangle$

$(\text{enter}_\delta\ X\ (\phi\ y))$,

because $(\phi_\delta\ (\text{enter}_\delta\ Y\ y))$

$= \phi_\delta\langle y, \text{init}\ \delta\rangle = \langle \phi\ y, \text{init}\ \delta\rangle = \text{enter}_\delta\ X(\phi\ y)$,

$= \text{nv}_\delta\ X\ \langle \eta, k\rangle\ (\phi\ y)$

$= (\text{nv}_\delta\ X\ \langle \eta, k\rangle \circ \phi)\ y$

$= (\phi \Rightarrow 1)\ (\text{nv}_\delta\ X\ \langle \eta, k\rangle)\ y$

$= ((\phi \Rightarrow 1) \circ \text{nv}_\delta\ X)\ \langle \eta, k\rangle\ y,$

and this shows the diagram commutes. Hence nv_δ is a natural transformation.

The semantics of halt are provided by

$$\text{semf halt} \in \text{Env emp}_T \xrightarrow{|K|} \text{Mng } \underline{\text{compl}} .$$

We assume there is some element of 0 denoting normal program termination; call it endexecute $\in 0$. Then for a store shape X, let

$$\text{semf halt } X \in \text{Env emp}_T \; X \xrightarrow[Pdom]{} (X \Rightarrow 0)$$

be given by

$$\text{semf halt } X \langle\rangle \; x = \text{endexecute, for all } x \in X.$$

It is trivial to see that semf halt is a natural transformation.

(We remark that some other kinds of linguistic constants, such as output statements, must be given if we are to have programs that do more than either print a program termination message or run on indefinitely without giving output. See Reynolds [7]. We cannot delve more deeply into these matters without exploring the story of 0.)

The last linguistic constant whose semantics we must give is goto. We want

$$\text{semf goto} \in \text{Env emp}_T \xrightarrow[|K|]{} \text{Mng } \underline{\text{compl}} \Rightarrow \text{Mng } \underline{\text{comm}};$$

therefore we will give the corresponding natural transformation

$$\text{jump} \in \text{Mng } \underline{\text{compl}} \times \text{Mng } \underline{\text{compl}} \xrightarrow[|K|]{} \text{Mng } \underline{\text{compl}} \; .$$

Just let jump be the projection onto the first component. Thus goto interrupts the flow of control by sending control to its argument, rather than following the "normal" flow (represented by the second component of the domain of jump). After newvar_δ, the semantics of goto are mercifully brief.

BIBLIOGRAPHY

1. Arbib, M.A., and Manes, E.G., Arrows, Structures, and Functors - The Categorical Imperative, Academic Press, New York (1975).

2. Cohn, P.M., Universal Algebra, D. Reidel Pub. Co., Dordrecht (1981).

3. Goldblatt, R., Topoi, The Categorial Analysis of Logic, North-Holland, Amsterdam (1979).

4. MacLane, S., Categories for the Working Mathematician, Springer-Verlag, Berlin (1971).

5. Manes, E.G., Algebraic Theories, Springer-Verlag (1976).

6. Reynolds, J.C., "The Essence of Algol," Proc. International Symposium on Algorithmic Languages, North-Holland (1981).

7. Reynolds, J.C., "Semantics of the Domain of Flow Diagrams," Journal ACM 24, pp. 484-503 (1977).

8. Reynolds, J.C., "Using Category Theory to Design Implicit Conversions and Generic Operators," Lecture Notes in Computer Science 94, Springer-Verlag, Berlin, pp. 211-258.

9. Scott, D.S., "The Lattice of Flow Diagrams," Symposium on the Semantics of Algorithmic Languages, Springer-Verlag, Berlin, pp. 311-366 (1971).

BIOGRAPHICAL DATA

Name: Frank Joseph Oles

Date and Place of Birth: December 15, 1946
Kronberg, Germany

Elementary School: Maplewood School
Sylvania, Ohio

Stranahan School
Sylvania, Ohio

Grove Patterson School
Toledo, Ohio

Monac School
Toledo, Ohio
Graduated 1957

Junior High School: Washington Junior High School
Toledo, Ohio

Jefferson Junior High School
Toledo, Ohio
Graduated 1960

High School: Whitmer Senior High School
Toledo, Ohio
Graduated 1964

College: Case Western Reserve University
Cleveland, Ohio
B.S. 1968

Graduate School: Cornell University
Ithaca, New York
M.S. 1970

Syracuse University
Syracuse, New York