We discuss the mathematical foundations of specifications, theories, and models with higher types. Higher type theories are presented by specifications either using the language of cartesian closure or a typed $\lambda$-calculus. We prove equivalence of both the specification methods, the main result being the equivalence of cartesian closure and a typed $\lambda$-calculus. Then we investigate "intensional" and extensional" models (the distinction is similar to that between $\lambda$-algebras and $\lambda$-models). We prove completeness of higher type theories with regard to intensional models as well as existence of free intensional models. For extensional models we prove that completeness and existence of an initial models implies that the theory itself already is the initial model. As a consequence intensional models seem to be better suited for the purposes of data type specification.


0. INTRODUCTION

Algebraic specification methods have been introduced by Guttag (1976) and Zilles (1975). Since then the mathematical foundations of algebraic specifications as well as modularization techniques have been investigated by several authors (Goguen, Thatcher, and Wagner, 1978; Ehrig et al., 1980, 1982; Burstall and Goguen, 1980; and many others). The result is a well developed theory of algebraic specifications which is widely accepted as a tool for the development of software systems.

Algebraic specifications are in general restricted to (conditional) equations. Several proposals have been made to extend the language such as to include a subsort mechanism (Goguen, 1978), to use first-order predicate calculus (Maibaum and Veloso, 1983) or to introduce functions of higher types (Goguen and Tardo 1978). Our paper will address the latter problem. We treat the mathematical foundations of specifications with higher types as well as we look for a suitable notion of language. Modularization techniques for such specifications are out of the scope of this paper.

Our interest in the subject is motivated by a project on compiler generation based on denotational semantics where it soon turned out that a modularization technique for the specification of denotational semantics (as in Gordon, 1979 for instance) is needed to make compiler generation comfortable and more efficient. At that time, only abstract data type theory provided modularization techniques. But algebraic specification does not include a handling of functions with higher types (at least not in a canonical way) which are a basic feature of denotational semantics. In Poigné (1983) we have made some suggestions how to combine abstract data type theory with higher type structure. Somewhat later we learned about the thesis of Parsaye–Ghomi (1981) which seems to be the first treatment of the mathematical foundations of specifications with higher type functions.

Functions of higher types have been studied in mathematics and computer science for a long time, one of the aims being to provide a foundation of mathematics respectively computer science which is not based on set theory. There are two major lines of investigation, one being concerned with \( \lambda \)-calculus, the other with cartesian closed categories. In the works of Lambek (1980) and Scott (1980) (and earlier papers), typed \( \lambda \)-calculus and cartesian closure are related, and an equivalence of the theories is stated. The second chapter of our paper may be understood as an elaboration of their ideas. It turned out that typed \( \lambda \)-calculus not only has to be extended by a product structure as suggested in (Scott, 1980) but that also a further binding structure has to be introduced to model composition of (formal) functions (=morphisms of a category). We refer to this calculus as \( A \)-calculus. We give a proof of the equivalence of the theories of cartesian closure and of the \( A \)-calculus as well as a proof of the equivalence of the categories of small cartesian closed categories and of \( A \)-theories.

To remain in the framework of algebraic specification techniques, cartesian closed categories are introduced as algebras with regard to two-level specifications. Two-level specifications, being discussed in the first chapter, extend algebraic specifications by specification schemes. A similar approach to the specification of structures with higher types is taken in Dybjer (1983), where the connection between cartesian closure, specification, and domain theory is discussed.

In the third chapter we define the notion of specifications with higher types, or higher type specifications for short, either being based on \( A \)-calculus or on cartesian closure. It is shown that the equivalence of the languages of cartesian closure and \( A \)-calculus can be extended to languages which include the axioms of the respective higher type specifications. Even if this results states that \( A \)-calculus and cartesian closure are equally powerful as a language for higher type specifications, the \( A \)-calculus seems to be better suited for pragmatic reasons, simply because the use of
variables often allows a shorter and more intuitive notation. The chapter contains some examples for higher type specifications.

Cartesian closed categories may be seen as theories in the sense of Lawvere (1963) but with higher types. Thus one can ask for a suitable notation of models. The third chapter develops elements of a model theory for theories with higher types. We suggest two notions of models which may be characterized as being intensional and extensional much in the same way as \( \lambda \)-algebras and \( \lambda \)-models (Barendregt, 1981). We prove that higher type theories are complete with regard to intensional models and that free algebras over a suitable sorted set can be constructed. This result is not too surprising, and the drawback is that intensional models lose the structural properties of the theories as they are not necessarily cartesian closed. For extensional models, which are concrete cartesian closed categories, the situation is far more unsettled. So far we do not know if theories are complete with regard to extensional models, but we conjecture that completeness does not hold in general. Our major result is that theories which are "extensionally" complete, and which have an initial model, are already concrete, and the initial model itself.

The results suggest that a theory of data types with higher type functions should be based on the notion of intensional models rather than on extensional ones.

As already pointed out, our work is based on (Lambek, 1980) and (Scott, 1980), and closely related to (Parsaye-Ghomi, 1981), where the same notion of theory and extensional model is used. Exact correspondences are given in the text. Dybjer (1983) uses cartesian closed categories as theories with higher types, and constructs free theories as free algebras in the standard way. He additionally introduces poset structures and investigates the relation to domain theory. About the same time Curien (1983) proves equivalences similar to ours. All the approaches have been developed independently being based on the pioneering work of Lambeck and Scott.

We assume familiarity with algebraic specification techniques as to be found in (Goguen et al., 1978). For category theory we refer to MacLane (1972) if no explicit references are given.

1. Two-Level Specifications

Algebraic specifications as considered in Goguen et al. (1978), for instance, consist of a set of sorts \( S \), a family \( \Sigma = (\Sigma_{w,s} \mid w \in S^*, s \in S) \) of operators and a family of equations \( E = (E_s \mid s \in S) \). Interpretations are given by \( \Sigma \)-algebras which consist of a data domain \( (A_s \mid s \in S) \) and operators \( \sigma_A : A_w \to A_s \) for all \( \sigma : w \to s \in \Sigma \) (where \( A_1 = 1, n = \{0, \ldots, n-1\} \),

\[ A_{w,s} := A_w \times A_s, \] and where \( \sigma : w \rightarrow s \in \Sigma \) is another notation for \( \sigma \in \Sigma_{w,s} \).

We use \( T_{\text{SPEC}}(X) \) to denote the canonical quotient term algebra w.r.t. the specification \( \text{SPEC} \) and a set of variables \( X = (X_s \mid s \in S) \). \( \Sigma_{\text{SPEC}} \) refers to the signature \( (S, \Sigma) \) of a specification \( \text{SPEC} \). We use mix fix notation.

While some efforts have been spent to combine given specifications in a structured way, less attention has been paid—at least in the abstract data type community—to the structure of specifications themselves. In a specification there may be implicit dependencies between the data, indicated for instance by indices. An example of this kind is the definition of categories (Schubert, 1970), where categories are introduced as algebras with hom-sets as carriers and composition and units as operators. In fact, a definition scheme is given, sorts and operators being indexed by objects. In order to obtain an explicit representation of such dependencies, we suggest the notion of hierarchical specifications, where the data on each level of specification can be indexed by an algebra of the preceding level.

Hierarchical specifications are generalized algebraic theories as defined by Cartmell (1978) (which we noticed only after we have introduced two-level specifications in Poigné, 1983). As we use hierarchical specifications only to specify categories with varying properties we prefer to avoid the full generality of generalized algebraic theories which would require quite an elaborate formal definition. In fact, we can restrict our attention to two-level specifications which are defined by an extension of the standard style of algebraic specifications.

A typical example for this kind of specifications is that of a category:

\[
\text{spec CATEGORY is} \\
\text{objects sorts type} \\
\text{var A, B, C, D: type in} \\
\text{morphisms sorts (A, B)} \\
\text{ops _ o A,B,C → (A, B)(B, C) → (A, C)} \\
1_A: (A, A) \\
\text{var f: (A, B), g: (B, C), h: (C, D) in} \\
\text{eqns f o (g o h) = (f o g) o h} \\
1_A o f = f \\
f o 1_B = f
\]

(The key-words “objects” and “morphisms” are added to support the intuition. The indices of \( _{\circ} \) are omitted in equations for readability.)

The first (object) level specifies the object structure which in case of categories simply is a set (formally denoted by the sort \text{type}). On the second level the morphism structure is specified, the carriers being hom-sets (denoted by \( (A, B) \)), with composition and units comprising the operators. Second-level sorts and operators depend on the first level data. The depen-
MODELS WITH HIGHER TYPES

decy is expressed by using variables over the first level sorts. A CATEGORY-algebra consists of a set (of objects), and (hom-) sets for every pair of objects, and suitably typed operations of composition and units over the hom-sets.

Formally, two-level specifications are introduced as follows: Let SIG = (S, Σ) be a signature.

(SIG-) dependent sorts are of the form \( s(x_0 : s_0, ..., x_n-1 : s_{n-1}) \), where \( s_i \in S \) and \( x_i \in X \), \( i \in \mathbf{n} \) (\( X \) is a set of variable names); \( s \) is called sort name, the variables \( x_i : s_i \) indicate the dependency.

A (SIG-) dependent operator \( \sigma(x_0 : s_0, ..., x_m-1 : s_{m-1}) : ds_0 \cdot \cdot \cdot ds_{n-1} \rightarrow ds_n \) consists of an operator name \( \sigma \), and variables \( x_i : s_i \) \( (x_i \in X, s_i \in S, i \in \mathbf{m}) \) expressing the dependency. Arity and coarity are given by derived dependent sorts \( ds_j \), \( j \in \mathbf{n} \), which are of the form \( s(t_0, ..., t_{k-1}) \), where \( s(y_0 : s_0, ..., y_k-1 : s_{k-1}) \) is a dependent sort and \( t_i \in T_{SIG} \{x_0 : s_0, ..., x_{m-1} : s_{m-1}\} \), \( 1 \leq k \).

A two-level signature SIG consists of a signature SIG and sets DS and \( D \Sigma \) of dependent sorts and operators. Let \( DV \) denote the set of dependency variables occurring in \( DS \) and \( D \Sigma \).

Given a SIG-algebra \( A \) and a SIG-homomorphism \( h : T_{SIG}(DV) \rightarrow A \), we extend the homomorphism to dependent sorts and operators by

\[
\begin{align*}
h(s(x_0 : s_0, ..., x_n-1 : s_{n-1})) & := s(h(x_0 : s_0), ..., h(x_{n-1} : s_{n-1})) \\
h(\sigma(x_0 : s_0, ..., x_m-1 : s_{m-1}) : ds_0 \cdot \cdot \cdot ds_{n-1} \rightarrow ds_n) & := \sigma(h(x_0 : s_0), ..., h(x_{m-1} : s_{m-1})): h(ds_0) \cdot \cdot \cdot h(ds_{n-1}) \rightarrow h(ds_n).
\end{align*}
\]

We call SIG\((A) = (DS(A), D \Sigma(A))\) with

\[
\begin{align*}
DS(A) & := \{ I(ds) \mid ds \in DS, I : T_{SIG}(DV) \rightarrow A \in SIG^b \} \\
D \Sigma(A) & := \{ I(da) \mid da \in D \Sigma, I : T_{SIG}(DV) \rightarrow A \in SIG^b \}
\end{align*}
\]

the A-signature of SIG, where \( A \) is a SIG-algebra. We abbreviate \( DS := DS(T_{SIG}(DV)), D \Sigma := D \Sigma(T_{SIG}(DV)) \).

Moreover we observe that every SIG-homomorphism \( h : A \rightarrow B \) induces a specification morphism (Goguen et al., 1978) \( h : SIG(A) \rightarrow SIG(B) \) (just extend the above definition), and thus a forgetful functor \( V_h : SIG(B)^b \rightarrow SIG(A)^b \).

A SIG-algebra \( \mathcal{A} \) consists of a SIG-algebra \( A \) and a \( SIG(A) \)-algebra \( DA \). A homomorphism \( h : \mathcal{A} \rightarrow \mathcal{B} \) of SIG-algebras is given by a SIG-homomorphism \( h : A \rightarrow B \) and a SIG\((A)\)-homomorphism \( dh : DA \rightarrow V_h(DB) \). With composition \( h' \circ h := (h' \circ h, V_h(dh') \circ dh) \) this defines a category we denote by SIG\(^b\).
Let \( V \subseteq \{ x : ds \mid x \in X, ds \in D \} \) be a (finite) set of variables, and let \( DT_{\text{SIG}}(DV, V) \) denote the free term algebra over \( V \) with regard to the signature \( \text{SIG} \). Observe that \( T_{\text{SIG}}(DV, V) = (T_{\text{SIG}}(DV), DT_{\text{SIG}}(DV, V)) \) is a \( \text{SIG} \)-algebra.

A \( \text{SIG} \)-equation consists of a pair \((t, t')\) of terms of the same sort either of \( T_{\text{SIG}}(DV) \) or of \( DT_{\text{SIG}}(DV, V) \). In the latter case \((t, t')\) is called a dependent equation.

A \( \text{SIG} \)-algebra \( \mathcal{A} \) satisfies an equation \((t, t')\) if \( \mathcal{A}(t) = \mathcal{A}(t') \) for all \( \text{SIG} \)-homomorphisms \( \mathcal{A} : T_{\text{SIG}}(DV, V) \rightarrow \mathcal{A} \) (more precisely, \( h(t) = h(t') \) if \( t, t' \in T_{\text{SIG}}(DV) \), or \( dh(t) = dh(t') \) if \( t, t' \in DT_{\text{SIG}}(DV, V) \)).

At last, a two-level specification \( \text{SPEC} \) consists of a specification \( \text{SPEC} = (S, S, E) \), a \( S \)-sorted set \( DV \) of dependency variables, and a dependent specification \( \text{DSPEC} = (DS, DS, DE) \), where \( DS, DS, DE \) are sets of dependent sorts, operators, and equations, respectively. A \( \text{SPEC} \)-algebra is a \( \text{SIG} \)-algebra which satisfies all the equations in \( E \cup DE \) (where \( \text{SIG} \) is the underlying two-level signature). The full subcategory of \( \text{SPEC} \)-algebras is denoted by \( \text{SPEC}^b \).

The specification of categories above is a two-level specification (in a typical style of presentation). CATEGORY-algebras are exactly small categories (MacLane, 1972), and CATEGORY-homomorphisms are functors between small categories.

The structure of our definition allows to use standard algebraic methods in proofs. For instance, if we define the notion of a two-level sorted set to consist of a \( S \)-sorted set \( X \) and a \( DS(X) \)-sorted set \( Y \), there is an obvious forgetful functor from \( \text{SPEC} \)-algebras to the underlying \((S, DS)\)-sorted set. The functor has a left adjoint \( (T_{\text{SPEC}}(X), DT_{\text{SPEC}}(X, Y)) \), where \( DT_{\text{SPEC}}(X, Y) \) is obtained from \( DT_{\text{SIG}}(X, Y) \) by factorization in \( \text{SIG}^b \) in the standard way. Especially, \( T_{\text{SIG}}(X, Y) \) is the free \( \text{SIG} \)-term algebra over \((X, Y)\). Another special instance of this proceeding is the construction of free categories over a given graph in (MacLane, 1972).

In the sequel we use overloading of operators for convenience \( (\circ \circ A,B,C\ldots) \) instead of \( \circ A,B,C\ldots \), for instance), without developing a corresponding theory (compare Gogolla, 1983; Poigné, 1984, for instance). We use

\[
\text{SPEC}(A) \vdash f = g
\]

to state that \( f = g \) is provable from the \( \text{SPEC}(A) \)-axioms.

2. Cartesian Closure versus \( \hat{\cdot} \)-Calculus

As an axiomatic language an algebraic specification naturally generates an equational theory. The pioneering work of Lawvere (1963) has demonstrated that categories with finite coproducts are an abstraction of
equational theories, and that algebras can be interpreted as contravariant functors which preserve finite products with natural transformations as homomorphisms. Basically, Lawvere observed that equational theories may be expressed via derived operators and derived equations, and that theories with different presentations but same structure on derived operators should be identified. With isomorphism classes of derived operators as morphisms and with substitution as composition, a category with finite coproducts is defined, the coproduct structure being used to express the arity of operations. Such categories are called \textit{algebraic theories}. In context of this paper we prefer to consider the dual notion of categories with finite products, as theories then more naturally extend to theories with higher types.

The specification of theories, i.e., categories with finite products, extends that of a category by adding a tupling mechanism.

\textbf{Categories with Finite Products.}

\texttt{spec THEORY is}
\texttt{ CATEGORY with}
\texttt{ objects ops 1 $\rightarrow$ type}
\texttt{ $\qquad \times \rightarrow$ type type $\rightarrow$ type}
\texttt{ var $A, B, C$: type in}
\texttt{ eqns $1 \times A = A = A \times 1$
\hfill $A \times (B \times C) = (A \times B) \times C$
\texttt{ var $A, B, C$: type in}
\texttt{ morphisms ops $\langle \_ , \_ \rangle : (A, B) \rightarrow (A, C) \rightarrow (A, B \times C)$
\texttt{ var $f: (A, B), g: (A, C), h: (A, B \times C), k: (A, 1)$ in
\texttt{ eqns $\langle \_ , \_ \rangle = k$
\hfill $\langle f, g \rangle \circ p_{B,C} = f$
\hfill $\langle f, g \rangle \circ q_{B,C} = g$
\hfill $\langle h \circ p_{B,C}, h \circ q_{B,C} \rangle = h.$

(The notation states that the specification CATEGORY is extended by the additional data.)

A THEORY-algebra $T$ consists of an \textit{algebra of objects} with operations \textquotedblleft terminal object\textquotedblright{} and \textquotedblleft binary product\textquotedblright{} and an \textit{algebra of morphisms} with carriers $T(A, B)$ being hom-sets and with operations such as tupling. Clearly, THEORY-algebras are small categories with finite products and with the additional property that the equations $X \times (Y \times Z) = (X \times Y) \times Z$, $X \times 1 = X = 1 \times X$ holds for all objects $X, Y, Z$ (This condition is technical, the intention being to simplify notation below).
Remark. The operations guarantee the existence of morphisms to be generated according to the definition of finite products (compare MacLane, 1972), the equation marked with (*) ensure the properties of these morphisms, and those marked with (**) the unicity. There is a canonical choice of products.

A specification technique involving higher types demands a well-defined concept of sorts of higher type. As canonically as products are used in theories to express the arity of operations, an abstract notion of “function space” should correspond to sorts of higher type. The most likely property to abstract from the set-theoretic concept of function space is the natural isomorphism

\[
\text{Set}[X \times Y, Z] \cong \text{Set}[X, Y \to Z]
\]

where \(\text{Set}[X, Y]\) denotes the set of functions with domain \(X\) and codomain \(Y\), and \(Y \to Z\) denotes the function space over \(Y\) and \(Z\). This property is axiomatized by cartesian closed categories. Considered as an abstraction of theories with higher types, cartesian closed categories play the same part as categories with products do for equational theories. This view, being folklore in category theory, has been adopted by Parsaye–Ghomi in (Parsaye–Ghomi, 1981) and independently in (Poigné, 1983).

Cartesian closed categories can be specified algebraically by the two-level specification mechanism. On the level of objects we add an operator to form “function spaces”, and on the level of morphisms “abstraction operators” and “evaluation morphisms” are introduced.

**Cartesian Closed Categories.**

\[
\text{spec CCC is}
\]

\[
\text{THEROY with}
\]

\[
\text{objects ops } \mathbin{\cdots} \to \mathbin{\cdots}: \text{type type } \to \text{type}
\]

\[
\text{var } A, B, C: \text{type}
\]

\[
\text{morphisms ops } \text{ev}_{B,C}: \to ([B \to C] \times B, C)
\]

\[
A: (A \times B, C) \to (A, B \to C)
\]

\[
\text{var } f: (A \times B, C), g: (A, B \to C)
\]

\[
\text{eqns } \langle p_{A,B} \circ A(f), q_{A,B} \circ \text{ev}_{B,C} \rangle = f \quad (*)
\]

\[
A(\langle p_{A,B} \circ g, q_{A,B} \circ \text{ev}_{B,C} \rangle) = g. \quad (**) 
\]

**Remarks.** (i) A comparison with the definition given in (MacLane, 1972) immediately shows that CCC-algebras are small cartesian closed categories, but with monoid structure on products. Again the equation (*) ensures the properties of the generated morphisms, while (**) guarantees their unicity. A canonical choice of exponentiation is implicit in the definition.
MODELS WITH HIGHER TYPES

(ii) The operator $A: (A \times B, C) \to (A, B \to C)$ constitutes one component of the isomorphism mentioned above while the derived operator $\langle p_{a,b} \circ \eta, q_{a,b} \circ ev_{b,c} \rangle$ defines the inverse morphism. In semantics of programming languages the operators are well known as “curry” and “uncurry” (Gordon, 1979).

As pointed out by Lambek (1980) and Scott (1980), there exists a close connection between cartesian closed categories and typed $\lambda$-calculus. In fact cartesian closure is like the theory of combinators (Barendregt, 1981) a variable-free approach towards a theory of functions. The type structure of our $\lambda$-calculus is more elaborate because of the addition of products.

The definition of $\lambda$-calculus proceeds in the style of two-level specifications fixing the type structure on the first level, and the term structure on the second level. Generalized algebraic theories (Cartmell, 1978), and thus especially two-level specifications, do not seem to cover the binding mechanism of $\lambda$-calculus: In generalized algebraic theories assumptions are of the form $\gamma_{0} \in A_{0}, ..., x_{n} \in A_{n}(x_{0}, ..., x_{n-1})$, where each type $A_{i}$ depends on the preceding types $A_{j}$ with $i < j \leq n + 1$, while in $\lambda$-calculus assumptions of the form $x \in A \leadsto M \in B$ (“If $x$ is a variable of type $A$ then it can be proved that $M$ is of type $B$.”) are needed. It may be an interesting task to extend the notion of generalized algebraic theories in such a way, and to see if our results hold in such a unified framework.

A TYPED $\lambda$-CALCULUS WITH FINITE PRODUCTS

$$\begin{align*}
spec \ A \ TYPE \ is \\
sorts \ type \\
ops \ l: A TYPE \\
\quad \times : type \ type \ -> \ type \\
\quad \rightarrow : type \ type \ -> \ type \\
var \ A, B, C: type \\
eqns \ l \times A = A = A \times 1 \\
\quad (A \times B) \times C = A \times (B \times C)
\end{align*}$$

$A \ TYPE$ is called the specification of types. Clearly, every $A \ TYPE$-algebra is an algebra of objects w.r.t. the specification CCC.

Let $A: type$ be a type variable. Then $V_{A} := \{ x : A \mid x \in X \}$ is the set of variables of type $A$, $X$ being denumerable. Moreover let

$$V* := \{ \langle x_{0} : A_{0}, ..., x_{n-1} : A_{n-1} \rangle \mid x_{i} : A_{i} \in V_{A_{i}}, x_{i} \neq x_{j} \text{ if } i \neq j \text{ for } i, j \in n \}.$$

Type-sorted sets $A_{A}$ of terms are defined by induction using a scheme indexed by $A \ TYPE$-terms over type variables $A, B, ...$. Replacement of the
type variables by *concrete types*, i.e., elements of a $\mathcal{A}$ *TYPE*-algebra $\mathcal{D}$, and evaluation of the resulting $\mathcal{A}$ *TYPE*-terms in $\mathcal{D}$, then yields the concrete $\mathcal{A}$-terms. The set $FV(M)$ of free variables of a term $M$ is defined simultaneously:

(i) $V_\mathcal{A} \subseteq A_\mathcal{A}$, $FV(x) = \{x\}$ for $x \in V_\mathcal{A}$;

(ii) $M \in A_\mathcal{A} \rightarrow B_\mathcal{A}$, $N \in A_\mathcal{A}$ $\Rightarrow (MN) \in B_\mathcal{A}$, $FV(MN) = FV(M) \cup FV(N)$,

\[
\langle x_0: A_0, \ldots, x_{n-1}: A_{n-1} \rangle \in V^*, \quad M \in A_B
\]

\[
\Rightarrow (\lambda x_0: A_0, \ldots, x_{n-1}: A_{n-1} \cdot M) \in A_{A_0 \times \cdots \times A_{n-1}} \rightarrow B.
\]

$FV(\langle x_0: A_0, \ldots, x_{n-1}: A_{n-1} \cdot M \rangle) = FV(M) \setminus \{x_0: A_0, \ldots, x_{n-1}: A_{n-1}\}$;

(iii) $\langle \rangle \in A_1$, $A(\langle \rangle) = \emptyset$;

$M_i \in A_\mathcal{A}$, $i \in \mathbb{n}$ $\Rightarrow \langle M_0, \ldots, M_{n-1} \rangle \in A_{A_0 \times \cdots \times A_{n-1}}$

\[
FV(\langle M_0, \ldots, M_{n-1} \rangle) = \bigcup_{i \in \mathbb{n}} FV(M_i).
\]

For notational convention we use script letters $\varpi, \psi, \ldots$, for tuples of variables $\langle x_0, \ldots, x_{n-1} \rangle$, $x_i \in V_\mathcal{A}$, and $\mathcal{M}, \mathcal{N}, \ldots$, for tuples of terms $\langle M_0, \ldots, M_{n-1} \rangle$, with $M_i \in A_\mathcal{A}$, $i \in \mathbb{n}$. $x_i$ (resp. $M_i$) denotes the $i$th component of a tuple, and $\mid x \mid$ (resp. $\mid M \mid$) the length of the respective tuple. Note that a tuple of variables is a tuple of terms. We often omit brackets for tuples of length 1.

As in Barendregt (1981) we consider terms modulo $\alpha$-conversion. Then substitution, which is simultaneous, is defined by

\[
x[x/\mathcal{N}] = \mathcal{N}
\]

if $x = x_i$,

\[
\equiv x \quad \text{else}
\]

\[
(MN)[x/\mathcal{N}] = (M[x/\mathcal{N}]) \cdot N[x/\mathcal{N}]
\]

\[
(\lambda y \cdot M)[x/\mathcal{N}] = (\lambda y \cdot M[x/\mathcal{N}])
\]

\[
\langle M_0, \ldots, M_{n-1} \rangle[x/\mathcal{N}] = \langle M_0[x/\mathcal{N}], \ldots, M_{n-1}[x/\mathcal{N}] \rangle
\]

with $\mid x \mid = \mid \mathcal{N} \mid$ and $\mathcal{N}$ of suitable type.

The set of closed terms of type $\mathcal{A}$ is denoted by $\mathcal{A}_\mathcal{A}^0$. The terms are to satisfy the following axioms (axiom scheme):

$\beta$ \quad $(\lambda x \cdot M)[x/\mathcal{N}] = M[x/\mathcal{N}]$

$\eta$ \quad $\lambda x \cdot (M[x]) = M$ \quad if $FV(M) \cap FV(x) = \emptyset$. 

MODELS WITH HIGHER TYPES

\[(\pi) \quad M = \langle \rangle \quad \text{if} \quad M \in A_1,\]
\[\langle p_0 M, \ldots, p_{n-1} M \rangle = M \quad \text{with} \quad p_i := \lambda x_i \cdot x_i\]
\[\langle M, \langle \rangle \rangle = M = \langle \langle \rangle, M \rangle\]
\[\langle L, \langle M, N \rangle \rangle = \langle \langle L, M \rangle, N \rangle\]
\[M = M\]
\[M = N \Rightarrow N = M\]
\[L = M, M = N \Rightarrow L = N\]
\[M = M \Rightarrow L M = L N, \quad ML' = NL', \quad \lambda x \cdot M = \lambda x \cdot N\]
\[\langle L, M \rangle = \langle L, N \rangle, \quad \langle M, L' \rangle = \langle N, L' \rangle.\]

We refer to the calculus as \textit{A-calculus}. \(A(D) \vdash M = N\) is used as notation to state that \(M = N\) is provable w.r.t. a given type algebra \(D\).

2.1. Proposition.

\[\begin{align*}
\lambda \langle x_0 : A_0, \ldots, x_i : 1, \ldots, x_{n-1} : A_{n-1} \rangle \cdot M \\
= \lambda \langle x_0 : A_0, \ldots, x_i : A_{i-1}, x_{i+1}, \ldots, x_{n-1} : A_{n-1} \rangle \cdot M \\
\lambda \langle x_0 : A_0, \ldots, x_i : B_0 \times \cdots \times B_{m-1}, \ldots, x_{n-1} : A_{n-1} \rangle \cdot M \\
= \lambda \langle x_0 : A_0, \ldots, y_0 : B_0, \ldots, y_{m-1}, y_{m+1}, \ldots, x_{n-1} : A_{n-1} \rangle \\
\cdot M[y_i / \langle y_0, \ldots, y_{m-1} \rangle].
\end{align*}\]

\textbf{Proof.} As \(\langle x_0, \ldots, y_0, y_{m-1}, \ldots, x_{n-1} \rangle = \langle x_0, \ldots, y_0, y_{m-1}, \ldots, x_{n-1} \rangle\) and by application of the \(\eta\)-rule.

\textbf{Remark.} The use of “parameter lists” in \(\lambda x \cdot M\) tacitly assumes that products have a monoid structure. This is reflected in our definition of type algebras. Without the monoid axioms on type algebras, the parameter lists are to be structured (i.e. \(\lambda \langle x : A, y : B, z : C \rangle \cdot M\)). In this case the development is technically more complicated but does not give essentially different insights. Another proceeding is to allow only unary parameters ("\(\lambda x : A \cdot M\)"), but then projection operators must be added. We believe that the latter possibility is less intuitive as it does not exploit the implicit projection facilities provided by variables.

In order to compare the theories of cartesian closed categories and of \textit{A}-calculus, we need translations from \textit{A}-terms to CCC-terms and vice versa. We follow the approach suggested in (Scott, 1980).
Translation of A-Terms to CCC-Terms

Any A-term \( M \) of type \( B \) is translated to a morphism (= constant CCC-term)

\[
\llbracket M \rrbracket: \prod_{x:A \in \text{FV}(M)} A \to B
\]

for a suitably chosen product. It turns out that the choice of the product is essential when comparing the theories of cartesian closed categories and A-calculus. We use lists of variables for an explicit coding of tuples to products:

Let \( \text{union} \) and \( \text{deletion} \) on variable lists \( V^* \) be functions

\[
\cup: V^* \times V^* \to V^*, \quad \setminus: V^* \times V^* \to V^*
\]

uniquely defined by

\[
x \cup y = x \quad \text{if} \quad y = x_i,
\]

\[
= \langle x, y \rangle \quad \text{else}
\]

\[
x \cup \langle \rangle = x = \langle \rangle \cup x \quad x \cup (y \cup z) = (x \cup y) \cup z
\]

\[
x \setminus \langle \rangle = x \quad y \setminus = \langle \rangle \quad \langle \rangle \setminus = \langle \rangle
\]

\[
(x \cup y) \setminus = (x \setminus x) \cup (y \setminus x) \quad x \setminus (y \cup z) = (x \setminus y) \cup (x \setminus z)
\]

\((x, y, z, y \in V^*, |y| = 1)\).

The list \( \Phi(M) \) of free variables of a A-term \( M \) is inductively defined by

\[
\Phi(x) = x \quad \text{if} \quad x \in V_A,
\]

\[
\Phi(MN) = \Phi(M) \cup \Phi(N)
\]

\[
\Phi(\lambda x \cdot M) = \Phi(M) \setminus x
\]

\[
\Phi(M) = \bigcup_{i \in \mathcal{I}} \Phi(M_i) \quad \text{with canonically induced iterated union.}
\]

For \( x \in V^* \) let

\[
\langle \rangle := 1
\]

\[
x := A_0 \times \cdots \times A_{n-1} \quad \text{if} \quad x = \langle x_0:A_0, \ldots, x_{n-1}:A_{n-1} \rangle.
\]

We observe that for \( x \in V^* \) and \( y = \langle y_0:A_0, \ldots, y_{n-1}:A_{n-1} \rangle \) with \( \{ y_i: A_i \mid i \in \mathbf{n} \} \subseteq \text{FV}(x) \) there is a canonical substitution morphism

\[
y(x \cdot y) := \langle p_0, \ldots, p_{n-1} \rangle: x \to y
\]
where $y_i = x_i$, and $p_i : x \rightarrow x_i$ is the projection to the $i$th component of $x$
For convenience we extend the notation to $A$-terms:
\[ M = \Phi(M). \]

Thus any $A$-term $M$ of type $B$ is translated to a morphism \([M] : M \rightarrow B\).
With this preliminaries the translation of $A$-terms to CCC-terms is given by
the following diagramatic definitions:
\[
\begin{array}{c}
\text{\([x : A]\) := 1_A} \\
\text{\([\langle \rangle\] := 1_1)} \\
(MN) \xrightarrow{[MN]} B \\
\gamma \\
M \times N \xrightarrow{[M] \times [N]} [A \rightarrow B] \times A
\end{array}
\]

where $M \in A_{A \rightarrow B}$, $N \in A_A$, and $\gamma = \gamma(\Phi(MN) \cdot \langle \Phi(M), \Phi(N) \rangle)$
\[
\begin{array}{c}
\text{\([\lambda x : M]\) \xrightarrow{[\lambda x : M] \times 1_e} [\lambda x : M] \times 1_e} \\
\gamma \\
(\lambda x : M) \xrightarrow{[\lambda x : M] \times \gamma} M
\end{array}
\]

where $M \in A_B$, and $\gamma = \gamma(\Phi(\lambda x : M) \cup x : \Phi(M))$
\[
\begin{array}{c}
\mathcal{M} \xrightarrow{[\lambda x : \mathcal{M}]} B_0 \times \cdots \times B_{n-1} \\
\gamma \\
\mathcal{M}_0 \times \cdots \times \mathcal{M}_{n-1}
\end{array}
\]

where $\mathcal{M} \in A_{B_0 \times \cdots \times B_{n-1}}$, and $\gamma = \gamma(\Phi(\mathcal{M}) \cdot \langle \Phi(\mathcal{M}_0), \ldots, \Phi(\mathcal{M}_{n-1}) \rangle)$.

Remark. Another, very elegant translation of substitutions, based on
the variable concept of deBruijn (Barendregt, 1981), is given by Curien
(1983). There the substitution morphisms are explicitly computed from
the index of variables. The reference has been pointed out by G. Huet.

2.3. Lemma. Let $x, y$ be tuples of variables with $x \in V^*$ and
$FV(y) \subseteq FV(x)$. Then
\[ [\lambda x \cdot y] = A(\gamma(x \cdot y)). \]
Proof. Immediately from the definition as \( \mathbb{P}_y = 1_y \).

2.4. PROPOSITION. Let \( D \) be a \( \lambda \) \( \text{TYPE} \)-algebra. Then

\[
A(D) \mapsto M = N \Rightarrow CCC(D) \mapsto \gamma_M \circ [M] = \gamma_N \circ [N]
\]

where \( \gamma_M = [\langle \Phi(M), \Phi(N) \rangle \cdot \Phi(M)] \) and \( \gamma_N = [\langle \Phi(M), \Phi(N) \rangle \cdot \Phi(N)] \).

Proof. The correctness of the \( \beta \)-axiom directly (by application of the definitions) follows from commutativity of the diagram

\[
\begin{array}{c}
(\Phi(M) \times N) \times N' \\
1 \times [M'] \\
\downarrow \quad \downarrow \\
M \times [N] \times [N'] \\
\downarrow \quad \downarrow \\
M[B]
\end{array}
\]

where \( M \in \Lambda_B, N' \in \Lambda_N \). Substitution morphisms are not labeled, but obvious from the context. We use that diagrams involving only substitution morphisms (composed from projections and tupling) commute because of coherence (MacLane, 1972). Commutativity of (*) is shown by induction on terms. The proofs are diagramatic, the outer diagrams are a specialization of (*).

(i) "\( M = x_i \)"

\[
\begin{array}{c}
N \xrightarrow{} N_0 \times \cdots \times N_{n-1} \xrightarrow{} N_i \\
\downarrow \quad \downarrow \quad \downarrow \\
x \xrightarrow{} x \xrightarrow{} x_i \\
\downarrow \quad \downarrow \quad \downarrow \\
\downarrow \quad \downarrow \quad \downarrow \\
\end{array}
\]

(1) commutes by definition of \([N]\), (2) because of product properties.

(ii) "\( M = y \in FV(x) \)"

\[
\begin{array}{c}
y \times N \xrightarrow{} y \xrightarrow{} y \\
\downarrow \quad \downarrow \quad \downarrow \\
y \times x \xrightarrow{} y \xrightarrow{} y \\
\downarrow \quad \downarrow \quad \downarrow \\
\end{array}
\]

(1) commutes because of product properties.
MODELS WITH HIGHER TYPES

(iii) \( M = PQ \):

\[
\Phi(PQ) \times \mathcal{N} \rightarrow (\Phi(P) \times \mathcal{N}) \times (\Phi(Q) \times \mathcal{N}) \rightarrow P[\alpha/\mathcal{N}] \times Q[\alpha/\mathcal{N}]
\]

\[
(\Phi(PQ) \times \mathcal{N}) \rightarrow (\Phi(P) \times \mathcal{N}) \rightarrow P[\alpha/\mathcal{N}]
\]

\[
(\Phi(PQ) \times \mathcal{N}) \rightarrow (\Phi(P) \times \mathcal{N}) \rightarrow P[\alpha/\mathcal{N}]
\]

(1) commutes because of product properties, (2) by inductive assumption.

(iv) \( M = \lambda y \cdot P \):

\[
\Phi(\lambda y \cdot P) \times \mathcal{N} \times y \rightarrow (\lambda y \cdot P[\alpha/\mathcal{N}]) \times y
\]

\[
(\Phi(\lambda y \cdot P) \times \mathcal{N} \times y) \rightarrow (\lambda y \cdot P[\alpha/\mathcal{N}]) \times y
\]

\[
(\Phi(\lambda y \cdot P) \times \mathcal{N} \times y) \rightarrow [\lambda y \cdot P[\alpha/\mathcal{N}] \times y
\]

commutes: (1) by inductive assumption, (2) by definition, and (3) by product properties. By abstraction we then conclude that

\[
\Phi(\lambda y \cdot P) \times \mathcal{N} \rightarrow \lambda y \cdot P[\alpha/\mathcal{N}]
\]

\[
(\Phi(\lambda y \cdot P) \times \mathcal{N}) \rightarrow \lambda y \cdot P[\alpha/\mathcal{N}]
\]

\[
(\Phi(\lambda y \cdot P) \times \mathcal{N}) \rightarrow \lambda y \cdot P[\alpha/\mathcal{N}]
\]

commutes.
(v) "\( \mathcal{M} = \mathcal{M}' \)" can be shown by a straightforward diagram chase using the inductive assumption for each component \( \mathcal{M}_i \):

For the \( \eta \)-axiom the statement directly follows from the definitions using unicity of abstraction. For the \( \pi \)-axioms, use that \( \llabel x, y \gg \cdot x = A_{\pi, \eta} \) and \( \llabel x, y \gg \cdot y = A_{\pi, \eta} \), Lemma 2.3, plus the unicity axioms.

For the translation of CCC-terms to \( \Lambda \)-terms one may be inclined to follow the lines of (Scott, 1980) and, for instance, define

\[
\# \text{1}_A \# = \lambda x : A \cdot x \\
\# T \cdot T' \# = \lambda x : A \cdot (\# T' \# (\# T \# x)).
\]

A closer look proves that the translation is inadequate if equivalence of the theories of cartesian closure and \( \Lambda \)-calculus is to be established. We have for instance

\[
[\# \text{1}_A \#] = [\lambda x : A \cdot x] = A(\gamma \cdot 1_A) \\
[\# x : A \#] = \# 1_A \# = \lambda x : A \cdot x.
\]

With this translation the theories only turn out to be equivalent up to abstraction in a sense to be made precise below.

Category theory distinguishes between "functions" as morphisms and "functions" as elements of a "function space," i.e., a morphism \( f : A \to B \) and a morphism \( A(\gamma \cdot f) : 1 \to [A \to B] \). This distinction is somewhat hidden in \( \Lambda \)-calculus: a \( \Lambda \)-term is a "function" in its free variables, while a "function name" is obtained by abstraction. For a "correct" translation of CCC-terms to \( \Lambda \)-terms, which preserves this distinction, a \( \Lambda \)-calculus is needed where substitution on free variables is an explicit concept. There are several possibilities for an explicit representation of the functional aspect, for instance to introduce different sets of variables for free and bound variables, or to use the technique of de Bruijn (Barendregt, 1981). We favour the following concept:

**Function Types in \( \Lambda \)-Calculus**

Let \( F \) be a denumerable set of names for function variables:

\[
f : (A, B) \in \Lambda_{(A, B)} \text{ for } f \in F \\
FV(f : (A, B)) = \emptyset \\
x \in V^*, M \in A_B, FV(M) \subseteq \{x\} \Rightarrow (x \cdot M) \in A_{(x,B)} \\
FV(x \cdot M) = \emptyset \\
M \in A_{(A,B)}, N \in A_A \Rightarrow (M'N) \in A_B \\
FV(M'N) = FV(N),
\]
The new terms, considered modulo $\alpha$-conversion, are to satisfy the axioms
\[(\beta') \quad (x \cdot M)' = M'[x/N']\]
\[(\eta') \quad x \cdot (M'x) = M \quad \text{if} \quad FV(M) \cap FV(x) = \emptyset\]
and the standard compatibility rules (binding and substitution as obvious). Terms $M \in \Lambda_{(A,B)}$ are called $\Lambda$-functions.

Remark. The function types correspond to hom-sets of categories. The new structure models algebraic substitution (compare Remark 2.10(i)).

We now rephrase the translation of CCC-terms to $\Lambda$-terms given in (Scott, 1980), but are careful about the use of free and bound variables.

**Translation of Cartesian Closure to $\Lambda$-Calculus.**

\[
\begin{align*}
\# f: (A, B) & = f: (A, B) \\
\# 1_A & = \langle x:A \rangle \cdot x \\
\# p_{A,B} & = \langle x:A, y:B \rangle \cdot x \\
\# q_{A,B} & = \langle x:A, y:B \rangle \cdot y \\
\# \langle >_A & = \langle x:A \rangle \cdot < > \\
\# ev_{A,B} & = \langle x:A \rightarrow B, y:A \rangle \cdot (xy) \\
\# T \cdot T' & = \langle x:A \rangle \cdot \# T' ' (\# T' ' x) \quad \text{if} \quad T: (A, B), T': (B, C), \\
\# \langle T, T' \rangle & = \langle x:A \rangle \cdot \langle \# T' ' x, \# T' ' x \rangle \quad \text{if} \quad T: (A, B), T': (A, C), \\
\# A(T) & = \langle x:A \rangle \cdot \lambda y:B \cdot \# T' ' \langle x, y \rangle \quad \text{if} \quad T: (A \times B, C). \\
\end{align*}
\]

2.5. **Proposition.** Let $D$ be a $\Lambda$ TYPE-algebra. Then

\[CCC(D) \rightrightarrows T = T = A(D) \rightrightarrows \# T = \# T' \]

**Proof.** By straightforward computations, for example,

\[
\begin{align*}
\# T \circ 1_B & = \langle x:A \rangle \cdot (\langle y:B \rangle \cdot y)'(\# T' ' x) \\
& = \langle x:A \rangle \cdot \# T' ' x = \# T' \quad \text{for} \quad T: (A, B), \\
\# \langle T \circ p_{B,C}, T \circ q_{B,C} \rangle & = \\
& = \langle x:A \rangle \cdot (\langle x:A \rangle \cdot (\langle y:B, z:C \rangle \cdot y)'(\# T' ' x))'x, \\
& (\langle x:A \rangle \cdot (\langle y:B, z:C \rangle \cdot z)'(\# T' ' x))'x \\
& = \langle x:A \rangle \cdot (\langle y:B, z:C \rangle \cdot y)'(\# T' ' x), (\langle y:B, z:C \rangle \cdot z)'(\# T' ' x) \\
& = \langle x:A \rangle \cdot \# T' ' x = \# T'
\end{align*}
\]
\[ # \langle T, T' \rangle \circ p_{B,C} # \]
\[ = \langle x : A \rangle \cdot (\langle y : B, z : C \rangle \cdot y)(\langle x : A \rangle \cdot \langle # T' \#' x, # T' \#' x \rangle)'x \]
\[ = \langle x : A \rangle \cdot # T' \#' x = # T'. \]

We have to extend the translation of \( A \)-terms to CCC-terms:

\[
\begin{align*}
[f : (A, B)] &= f : (A, B) \\
[x \cdot M] &= [x \cdot \Phi(M)] \circ [M] \\
[M'N] &= [N] \circ [M]
\end{align*}
\]

(note that terms of type \((A, B)\) are closed).

2.6. FACT. \( A([x \cdot M]) = [\lambda x \cdot M] \).

Proof. \([x \cdot M] = [\lambda x \cdot M] \times 1_\varepsilon \text{ev}_{x,B}\) as \( x \cdot M \) is closed. \(\)

2.7. LEMMA. Let \( x, y \) be tuples of variables with \( x \in V^* \) and with \( FV(y) \subseteq FV(x) \). Then

\( i \) \( [x \cdot y] = \gamma(x \cdot y) \)

\( ii \) \( # [x \cdot y] # = x \cdot y. \)

Proof. (i) Lemma 2.3 + Fact 2.6 or straightforward.

\( ii \) \( # [x \cdot y] # = # \gamma(x \cdot y) # = # \langle p_0, \ldots, p_{n-1} \rangle # \)
\[ = x \cdot \langle (x \cdot x_0)'x, \ldots, (x \cdot x_{n-1})'x \rangle \]
\[ = x \cdot \langle x_0, \ldots, x_{n-1} \rangle \]
\[ = x \cdot y. \]

2.8. PROPOSITION. Let \( D \) be a \( A \) TYPE-algebra. Then

\( i \) \( A(D) \vdash [M] # = \Phi(M) \cdot M \) for \( M \in A_A \),

\( A(D) \vdash [M] # = M \) for \( M \in A_{(A,B)} \),

\( ii \) \( CCC(D) \vdash [\# T \#] = T \) for a CCC-term of sort \((A, B)\).

Proof. Again by straightforward computation. We give a few typical examples:

\( i \) \( # [x] # = # 1_A # = \langle x : A \rangle \cdot x \)
\[ # [x \cdot M] # = # [x \cdot \Phi(M)] \circ [M] # \]
\[ = x \cdot (\Phi(M) \cdot M)'((x \cdot \Phi(M))'x) \]
by inductive assumption and Lemma 2.7(ii)
\[ = x \cdot M \]
MODELS WITH HIGHER TYPES

# \[ \text{\# } [\lambda x \cdot M] \# = \#
\lambda (\lambda y \cdot x) \cdot \Phi(M) \] \[ \circ [M] \#' \]
\[ = y \cdot \lambda x \cdot (\langle y, x \rangle \circ \# [M] \#') \]
\[ = \text{by inductive assumption and Lemma 2.7(ii)} \]
\[ = y \cdot \lambda x \cdot (\Phi(M) \cdot M)' \circ \Phi(M) \]
\[ = y \cdot \lambda x \cdot M \quad \text{where } y = \Phi(\lambda x \cdot M); \]

(ii) \[ \# \langle T, T' \rangle \# = \#
\left[ \langle x : A \rangle \cdot \langle \# T', x, \# T' \#' \rangle \right]\]
\[ = \left[ \langle x : A \rangle \cdot \langle x, x \rangle \right] \circ (T \times T') \]
\[ = \left[ \langle x : A \rangle \cdot \langle x, x \rangle \right] \circ (T \times T') \]
\[ = \langle T, T' \rangle \]

\[ \# A(T) \# = \#
\left[ \langle x : A \rangle \cdot \lambda y : B \cdot \# T', y, \# T' \#' \rangle \right]\]
\[ = \left[ \langle x : A \rangle \cdot \lambda y : B \cdot (\langle x, y \rangle \circ \# T', y, \# T' \#') \right]\]
\[ = \left[ \langle x, y \rangle \cdot \langle x, y \rangle \right] \circ (T \times T') \]
\[ = A(T) \]
\[ \quad \text{by inductive assumption and Lemma 2.7(i)} \]

All the results obtained so far amount to the

2.9. THEOREM. The theories \( A(D) \) and \( CCC(D) \) are equivalent for all \( A \) TYPE-algebras \( D \), i.e.,

\[ A(D) \leftrightarrow \# [M] \# = M \]
\[ CCC(D) \leftrightarrow \# [\# T] \# = T \]
\[ A(D) \leftrightarrow M = N \Rightarrow CCC(D) \leftrightarrow [M] = [N] \]
\[ CCC(D) \leftrightarrow T = T' \Rightarrow A(D) \leftrightarrow \# T = \# T' \# \]

for \( M, N \in A(a,b) \), \( T, T' \) being CCC-term of sort \( (A, B) \).

Proof. By Propositions 2.4, 2.5, and 2.8 we have only to check well-definedness of \([M'N]\) and \([\lambda x \cdot M]\). We compute

\[ [(\lambda x \cdot M) \cdot N] = [[\lambda x \cdot M]] \circ [[\lambda x \cdot M]] \]
\[ = [[\lambda x \cdot M]] \circ (\alpha x \cdot M) \times \gamma x \cdot B \quad \text{(2.6)} \]
\[ = (\alpha x \cdot M) \times [[\lambda x \cdot M]] \circ \gamma x \cdot B \]
\[ = [[(\lambda x \cdot M) \cdot N]] \]
\[ = [[M | x = N]] \]
and

\[ [\text{ax} \cdot (M_\text{ax})] = [\text{ax} \cdot \text{ax}] \circ [M_\text{ax}] \quad (FV(M) = \emptyset) \]

\[ = [\text{ax}] \circ [M] \]

\[ = [M]. \]

2.10. Remarks. (i) Let a calculus of substitution be defined by restricting the \(\alpha\)-calculus to the type algebra

\[
\text{spec} \pi \quad \text{TYPE is}
\]

\[
\text{sorts} \quad \text{type}
\]

\[
\text{ops} \quad : \rightarrow \text{type}
\]

\[
\rightarrow \rightarrow \text{type} \rightarrow \text{type}
\]

\[
\text{var} \quad A, B, C: \text{type}
\]

\[
\text{eqns} \quad 1 \times A = A = A \times 1
\]

\[
A \times (B \times C) = (A \times B) \times C
\]

and to the term forming rules

\[
V_A \subseteq A_A
\]

\[
M \in A_{(A,B)}, N \in A_A \Rightarrow M \cdot N \in A_B
\]

\[
\text{ax} \in V^*, M \in A_B \quad FV(M) \subseteq \{\alpha\} \Rightarrow \text{ax} \cdot M \in A_{(\alpha,B)}
\]

\[
\langle \rangle \in A_1
\]

\[
M_i \in A_{A_i}, i \in n \Rightarrow \langle M_0, \ldots, M_{n-1} \rangle \in A_{A_0 \times \cdots \times A_{n-1}}
\]

and to the corresponding axioms. Refer to the calculus as \(\Gamma\)-calculus. Then we immediately conclude from the proof of Theorem 2.9 that

**Proposition.** For every \(\pi\) \(\text{TYPE}\)-algebra \(D\) the theories \(\text{THEORY}(D)\) and \(\Gamma(D)\) are equivalent.

The proposition states the standard fact that composition of algebraic theories (here in product form) corresponds to algebraic substitution (here expressed in a \(\lambda\)-calculus style).

(ii) If we restrict the \(\alpha\)-calculus by omitting the term forming rules for \(\text{ax} \cdot M\) and \(M \cdot N\) plus corresponding axioms, and if we replace \(\text{ax} \cdot M\) by \(\lambda \text{ax} \cdot M\) and \(M \cdot N\) by \((MN)\) in the translation \(\# \rightarrow \#\) of CCC-terms to \(\alpha\)-terms we obtain
PROPOSITION. For every $\Lambda$ TYPE-algebra $D$ the theories $\text{CCC}(D)$ and $\Lambda'(D)$ are equivalent up to abstraction, where

$\Lambda'(D) \vdash \# [M] \# = \lambda \Phi(M) \cdot M$

$\text{CCC}(D) \vdash [\# T \#] = \hat{T}$

$\Lambda'(D) \vdash M = N \Rightarrow \text{CCC}(D) \vdash [M] = [N]$

$\text{CCC}(D) \vdash T = T' \Rightarrow \Lambda'(D) \vdash \# T \# = \# T' \#$

with $M, N \in \Lambda(A, B)$ and $T, T'$ being CCC-terms of sort $A(A, B)$, and where

$\hat{T} : \prod_{f : \Gamma(A, B) \in FV(T)} [A \to B] \to [C \to D]$ is defined by $(T$ of sort $(C, D))$

$\hat{T} = \gamma \circ (\hat{T}_0 \times \cdots \times \hat{T}_{n-1}) \circ \sigma$

where $\sigma : [A_0 \to B_0] \times \cdots \times [A_{n-1} \to B_{n-1}] \to [A \to B]$ is the CCC-morphism which corresponds to the CCC-operator $\sigma : (A_0, B_0) \cdots (A_{n-1}, B_{n-1}) \to (A, B)$, for example, for composition

$$
\begin{array}{ccc}
[A \to C] \times A & \xrightarrow{ev_{A,C}} & C \\
\downarrow 1_{[A \to C]} \times 1_A & & \\
\end{array}
\quad
\begin{array}{ccc}
[A \to B] \times [B \to C] \times A & \cong & [B \to C] \times [A \to B] \times A \\
\downarrow [1_{[A \to B]} \times ev_{A,B}] & & \\
\end{array}
$$

The transition is the well-known way to understand a cartesian closed category $C$ as a $C$-enriched category (Kelly, 1982).

For a proof of the proposition again the products have to be chosen suitably. Then the proof proceeds straightforward along the lines of that of Theorem 2.9. As pointed out, cartesian closed categories (as models of the theory of cartesian closure) may be seen as theories with higher types, extending the idea to conceive equational theories as Lawvere theories. Similarly, the typed $\Lambda$-calculus allows us to express theories with higher types by adding constants and axioms. These so-called $\Lambda$-theories are more concrete in the sense that one has an explicit term representation while cartesian closed categories abstract from a specific representation (as Lawvere theories opposed to a definition of equational theories in logic). Our result links the theory of cartesian closure to typed $\Lambda$-calculus, hence it seems to be reasonable to use the result to compare higher type theories as cartesian closed categories with higher type theories as $\Lambda$-theories.
2.11. DEFINITIONS. A $A$-theory $T$ consists of

- a $A$ TYPE-algebra $D$,
- a $D \times D$-sorted set $C = (C_{(d,d')} \mid d, d' \in D)$ of constants, and
- a $D \times D$-sorted set $E = (E_{(d,d')} \subseteq A^T_{(d,d')} \times A^T_{(d,d')} \mid d, d' \in D)$ of equations such that $E = E^+$, where

$A^T$-terms are defined by adding $C_{(d,d')} \subseteq A^T_{(d,d')}$, $d, d' \in D$, as a rule for term construction, and

$(M, N) \in E^+$ if $A^T(D) \rightarrow M = N$ with $A^T(D)$ being the calculus enriched by the constants and the equations.

A homomorphism $h: T \rightarrow T'$ of $A$-theories consists of

- a homomorphism $h: D \rightarrow D'$ of $A$ TYPE-algebras, and
- sorted sets $(h_d: A^T_{d} \rightarrow A^T_{h(d)} \mid d \in D), (h_{d,d'}: A^T_{(d,d')} \rightarrow A^T_{(h(d),h(d'))} \mid d, d' \in D)$ of functions, such that the $A$-structure is preserved and $h(M) = h(N)$ if $M = N$.

$A$-structure is preserved if

$$h_{(x,d)}(x \cdot M) = h(x) \cdot h_d(M) \quad \text{for} \quad M \in A_{(d,d')},$$

$$h_d(M'N) = h_{(d,d')}(M)'h_d(N),$$

$$h_{x \rightarrow d}(\lambda x \cdot M) = \lambda h(x) \cdot h_d(M), \quad h_{d}(MN) = h_{d \rightarrow d}(M)'h_d(N),$$

$$h_{x:d} = x:h(d), \quad h(\langle \rangle) = \langle \rangle,$$

$$h_{d_0 \times \cdots \times d_{n-1}}(\langle M_0, \ldots, M_{n-1} \rangle) = \langle h_{d_0}(M_0), \ldots, h_{d_{n-1}}(M_{n-1}) \rangle.$$

where

$$h(\langle x_0:d_0, \ldots, x_{n-1}:d_{n-1} \rangle) = \langle x_0:h(d_0), \ldots, x_{n-1}:h(d_{n-1}) \rangle.$$

The categories of $A$-theories (resp. $A(D)$-theories) (with a fixed $A$ TYPE-algebra $D$) are denoted by $A$-TH (resp. $A(D)$-TH).

It is not too surprising that

2.12. PROPOSITION. $A$-TH $\simeq$ $\text{CCC}^b$ and $\Lambda(D)$-TH $\simeq$ $\text{CCC}(D)^b$, where $\text{CCC}^b$ ($\text{CCC}(D)^b$) is the category of $\text{CCC}$-algebras (with a fixed algebra $D$ of objects).

Proof. Given $C \in \text{CCC}^b$ we construct a $A$-theory $T^C$ with $A$ TYPE-algebra $D^C$ being the algebra of objects of $C$, with constants $(C[d, d'] \mid d, d' \in D)$ and equations $M = N \in E^C$ if $[M]_C = [N]_C$, where $[M]_C$ is obtained by adding $[f] = f$ for $f: d \rightarrow d' \in C$ to the translation given above and interpreting $[M]$ in $C$. $E_C = E^C$ by 2.9 using $\#f\# = f$. 
A CCC-morphism $F: C \to C'$ induces a homomorphism $h^F: T^C \to T^{C'}$ of $\mathcal{A}$-theories by $h^F_{d,d'}(f) = F(f)$ for $f: d \to d' \in C$ (as $\mathcal{A}$ TYPE-homomorphism the object part of the functor must be used). The only property to be checked is $h^F(M) = h^F(N)$ if $M = N$.

It is a straightforward induction to check $F([M]_C) = [h^F(M)]_C$, for instance, $F([MN]_C) = [h^F(MN)]_C$:

\[
F([MN]_C) = F([\Phi(M) \cup \Phi(N) \cdot \langle \Phi(M), \Phi(N) \rangle]_C \circ ([M]_C \times [N]_C) \circ \text{ev}_{d,d'})
\]
\[
= [\Phi(M) \cup \Phi(N) \cdot \langle \Phi(M), \Phi(N) \rangle]_C \circ F([M]_C)
\]
\[
\times F([N]_C) \circ \text{ev}_{F(d),F(d')}
\]

(Substitution morphisms are preserved by product preserving functors.)

\[
= [h^F(M) h^F(N)]_C.
\]

But then

\[
M = N \Rightarrow [M]_C = [N]_C \Rightarrow F([M]_C) = F([N]_C) = [h^F(M)]_C = [h^F(N)]_C \Rightarrow h^F(M) = h^F(N).
\]

Vice versa, let $T$ be a $\mathcal{A}$-theory, a category $C^T$ is defined where the object structure is the $\mathcal{A}$ TYPE-algebra $D$ of $T$, homsets are $C^T[d,d'] := \{ [M] | M \in A_{d,d}' \}$ and composition is $[M] \circ [N] = \langle x : A \to N'(M'x) \rangle$. If we analogously use the definition of $\# \, \# \, \#$ to express the operators of cartesian closure it immediately follows from 2.5 that $C^T$ is a cartesian closed category. The mappings $(h_{d,d'}: A_{d,d}' \to A_{d,d}' | d, d' \in D)$ of a $\mathcal{A}$-theory homomorphism induce a cartesian closed functor $F^h: C^T \to C^T$ (use the equality $h(\#M\#) = \#F^h(M)\#$, which is proved by an easy induction).

These data define an equivalence of categories as the definitions ensure compatibility of the structures.

The whole development suggest the

2.13. DEFINITION. A $\mathcal{A}$-theory is called a concrete higher type theory, a CCC-model is called an abstract higher type theory.

Remark. Free cartesian closed categories over a given set of operators can be constructed (add the operators to the $\mathcal{A}$-calculus and use Proposition 2.12, freeness of this theory is easily shown). This proves a theorem stated in (Parsaye–Ghomi, 1981). In contrast to our proceeding,
Parsaye-Ghomi constructs a free cartesian closed category over a given signature category. Then the equivalence to an extended $\lambda$-calculus depends on the statement that the construction of free cartesian closed categories over a category preserves product and exponentiation structure on objects (Parsaye-Ghomi, 1981, p. 89) which may be doubted.

3. SPECIFICATIONS AND THEORIES WITH HIGHER TYPES

Our concept of specifications with higher types is closely related to the concept suggested by Parsaye-Ghomi (1981). The difference from standard specifications is the use of higher types as sorts. In our approach the operators are polymorphic because of the two-level mechanism. In extension to (Parsaye-Ghomi, 1981) type structure may be added. We again compare cartesian closure and $\lambda$-calculus.

3.1. DEFINITION. An algebraic specification \( \text{TYPE} = \text{A TYPE} + (\emptyset, T\Sigma, TE) \) is called a type specification, the operators of \( T\Sigma \) are (additional) type constructors.

Remarks. (i) As \( A \text{ TYPE} \) is a fixed part in any type specification, we only declare the additional data using the keyword “types”:

- “sums” types ops \( \_ + \_ : \text{type type} \to \text{type} \)
- “universal object”
  - types ops \( U : \to \text{type} \)
  - eqns \( U = U \to U \)
- “natural numbers”
  - types ops \( \_ + \_ : \text{type type} \to \text{type} \)
  - nat : \to \text{type}
  - eqns \( \text{nat} = 1 + \text{nat} \).

(ii) Heterogoneous type specifications are possible as well, allowing to deal with several categories. We only consider homogeneous type specifications in this paper.

3.2. DEFINITION. A higher type signature \( HSIG \) consists of a type signature \( \Sigma \text{ TYPE} \) and a set \( H\Sigma \) of higher type operators of the form

\[ \sigma : t_0 \ldots t_{n-1} \to t_n \quad n > 0 \]

with \( t_i \in T_{\Sigma \text{ TYPE}}(X), i \in \mathbb{N} \), being terms over a suitable set \( X \) of type variables (\( \sigma : \to t \) is used for \( \sigma : 1 \to t \) for short):
A. Cartesian Closure.

spec CCC[HSIG] is

CCC with

objects ops \{ \omega: \text{type}^n \to \text{type} | \omega \in T \Sigma_n \}

var A, B: \text{type}

in morphisms ops \{ \sigma: (t_0 \times \cdots \times t_{n-1}, t_n) | \sigma: t_0 \times \cdots \times t_{n-1} \to t_n \in H \Sigma \}

B. A-calculus. Use \Sigma \text{TYPE} as signature of types, and extend the
term forming rules by

\[ \sigma: t_0 \times \cdots \times t_{n-1} \to t_n \in A \langle t_0 \times \cdots \times t_{n-1}, t_n \rangle \]

\( A[\text{HSIG}] \) refers to this calculus.

We have the choice of using either the language of cartesian closure or
that of \( A \)-calculus to specify higher type equations. A specification of a
fixpoint operator may demonstrate the two styles:

\begin{itemize}
  \item \( C^3 \) spec FIXPOINT is
      \begin{align*}
      \text{var } A: & \text{type} \\
      \text{in ops } & Y: [B \to B] \to B \\
      \text{eqns } & \langle 1_{B \to B}, Y \rangle \circ \text{ev}_{B,B} = Y
      \end{align*}
  \item \( A \) spec FIXPOINT is
      \begin{align*}
      \text{var } B: & \text{type} \\
      \text{in ops } & Y: [B \to B] \to B \\
      \text{eqns } & \langle f: [B \to B], f(Y'f) = \langle f: [B \to B] \rangle \cdot Y'f
      \end{align*}
\end{itemize}

("\( C^3 \)" and "\( A \)" refer to the language used).

The \( C^3 \)-specification is in a rigorous combinatory style, while the \( A \)-
specification appears to be more readable. With a slight modification

\begin{itemize}
  \item \( A \) spec FIXPOINT is
      \begin{align*}
      \text{var } B: & \text{type} \\
      \text{in ops } & Y: [B \to B] \to B \\
      \text{var } f: [B \to B] \text{ in} \\
      \text{eqns } & f(Y'f) = Y'f
      \end{align*}
\end{itemize}

even reminds of the standard style of specification (implicity stating that
the terms are bound by the free variables occurring in an equation).

3.3. Definition. (i) A \textit{(higher type) \( A \)-specification} \( A \) SPEC consists of
a type specification \( \text{TYPE} \), a higher type signature \( \text{HSIG} = (\Sigma \text{TYPE}, H\Sigma) \) and a (finite) set \( \text{AE} \) of \( \Lambda \)-equations of the form

\[
M = N
\]

with \( M, N \in A[\text{HSIG}]_{(\ell, \ell)} \) with \( \ell, \ell' \in T^{\Sigma \text{TYPE}}(X) \) for a suitable set of type variables \( (M, N) \) are \( \Lambda \)-functions).

(ii) A (higher type) \( C^3 \)-specification \( C^3\text{SPEC} \) consists of a type specification \( \text{TYPE} \), a higher type signature \( \text{HSIG} = (\Sigma \text{TYPE}, H\Sigma) \) and a (finite) set \( C^3\text{E} \) of \( C^3 \)-equations of the form

\[
T = T'
\]

with \( T, T' \) being \( \text{CCC}[\text{HSIG}](D) \)-terms of the same sort where \( D = T^{\Sigma \text{TYPE}}(X) \) for a suitable set \( X \) of type variables.

(iii) We use \( \text{CCC}[C^3\text{SPEC}] \) and \( A[A\text{SPEC}] \) to denote the two-level specification obtained by adding the respective equations to \( \text{CCC}[\text{HSIG}] \) and \( A[\text{HSIG}] \).

3.4. EXAMPLES. (i) Any algebraic specification \( \text{SPEC} = (S, \Sigma, E) \) extends to a higher type specification. The \( C^3 \)-(resp. \( \Lambda \)-) structure only adds the higher type structure:

\[
A \text{ spec NAT is}
\]

\[
\text{types ops nat: } \rightarrow \text{ type}
\]

\[
\text{in ops 0: } \rightarrow \text{ nat}
\]

\[
\text{suc: nat } \rightarrow \text{ nat}
\]

\[
\text{add: nat nat } \rightarrow \text{ nat}
\]

\[
\text{var m, n: nat in}
\]

\[
\text{eqns add'}(m, 0')() = m
\]

\[
\text{add'}(m, \text{suc'}n) = \text{suc'}(\text{add'}(m, n))
\]

With the additional convention that

\[
\sigma(M_0, \ldots, M_{n-1}) := \sigma'(M_0, \ldots, M_{n-1})
\]

we obtain the standard notation.

\[
C^3 \text{ spec NAT is}
\]

\[
\text{types ops nat: } \rightarrow \text{ type}
\]

\[
\text{in ops 0: } \rightarrow \text{ nat}
\]

\[
\text{suc: nat } \rightarrow \text{ nat}
\]

\[
\text{add: nat nat } \rightarrow \text{ nat}
\]

\[
\text{eqns } \langle 1_{\text{nat}}, 0 \rangle_{\text{nat}} \circ \text{add} = 1_{\text{nat}}
\]

\[
\langle \text{p}_{\text{nat, nat}}, \text{q}_{\text{nat, nat}} \circ \text{suc} \rangle \circ \text{add} = \text{add} \circ \text{suc}
\]
(ii) We introduce coproducts (we shall substantiate the implicit statement below):

\( A \) spec COPRODUCT is

- types \( \_, + \_, \_ : \text{type} \rightarrow \text{type} \)
- var \( A, B, C : \text{type} \)
- in ops \( u_{A,B} : A \rightarrow A + B \)
  \( v_{A,B} : B \rightarrow A + B \)

\[ [\_, -, -] : [A \rightarrow C][B \rightarrow C] \rightarrow [A + B \rightarrow C] \]

\( \text{var} \ f : [A \rightarrow C], \ g : [B \rightarrow C], \ h : [A + B \rightarrow C] \)
\( \text{eqns } \lambda x : A \cdot [f, g](u_{A,B} \cdot x) = f \)
\( \lambda y : B \cdot [f, g](v_{A,B} \cdot y) = g \)
\( [\lambda x : A \cdot h(u_{A,B} \cdot x), \lambda y : B \cdot h(v_{A,B} \cdot y)] = h \) (**)

Coproducts in the \( C^3 \)-idiom are more cumbersome to define (reasons will be discussed below):

\( C^3 \) spec COPRODUCT is

- types \( \_, + \_, \_ : \text{type} \rightarrow \text{type} \)
- var \( A, B, C : \text{type} \)
- in ops \( u_{A,B} : [A \rightarrow A + B] \)
  \( v_{A,B} : [B \rightarrow A + B] \)
\( [\_, -, -] : [A \rightarrow C][B \rightarrow C] \rightarrow [A + B \rightarrow C] \)

\( \text{eqns } <u_{A,B}, [\_, -, -]> \circ \circ \circ \circ = p_{[A \rightarrow C][B \rightarrow C]} \)
\( <v_{A,B}, [\_, -, -]> \circ \circ \circ \circ = q_{[A \rightarrow C][B \rightarrow C]} \)
\( <<u_{A,B}, 1_{[A + B \rightarrow C]}> \circ \circ \circ \circ , <v_{A,B}, 1_{[A + B \rightarrow C]}> \circ \circ \circ \circ \circ > \)
\( \circ [\_, -, -] = 1_{[A + B \rightarrow C]} \)

where \( \_, \_ : [A \rightarrow B][B \rightarrow C] \rightarrow [A \rightarrow C] \) is the "enriched" composition (compare Remark 2.10(ii)).

(iii) **Recursive domains** can be specified by introduction of a type and a recursive type equation. Universal object:

\[ \text{spec UNIVERSAL is} \]

- types \( \text{ops } U : \rightarrow \text{type} \)
- eqns \( U = [U \rightarrow U] \)

This allows the interpretation of the untyped \( \lambda \)-calculus by

\[ I(x) = x : U, \quad I(\lambda x : M) = \lambda x : U \cdot I(M), \quad I(MN) = I(M) \cdot I(N). \]

Self-application is possible as \( (\lambda x : U \cdot M)(\lambda x : U \cdot M) \) is well defined. Clearly, the interpretation satisfies the axioms of the \( \lambda - \beta - \eta \)-calculus. Observe that \( A[\text{UNIVERSAL}] \) is not isomorphic to the \( \lambda - \beta - \eta \)-calculus because of the product structure (which is not surjective pairing in the sense of
(Barendregt, 1981) as not necessarily $U = U \times U$ in the TYPE-algebra). Hence $\lambda - \beta - \eta$-calculus is only a fragment of $A[\text{UNIVERSAL}]$-calculus.

Another way to introduce additional structure on types uses retracts explicitly:

\[ A \text{ spec } \text{UNIVERSAL}^\beta \text{ is } \]
\[ \text{types ops } U : \rightarrow \text{type} \]
\[ \text{in ops } c : [U \rightarrow U] \rightarrow U \]
\[ d : U \rightarrow [U \rightarrow U] \]
\[ \text{var } f : [U \rightarrow U] \]
\[ \text{eqns } d'(c'f) = f \]

We now interpret untyped $\lambda$-calculus by

\[ I(x) = x : U, \quad I(\lambda x \cdot M) = c' \lambda x : U \cdot I(M), \quad I(MN) = (d'I(M)) I(N). \]

Self-application again is possible, and the $\beta$-axiom is satisfied but not necessarily the $\eta$-axiom

\[ I(\lambda x \cdot (Mx)) = c'(\lambda x : U \cdot (d'M)x) = c'(d'M). \]

Other recursive data structures can be introduced in the same way

\[ A(C^3) \text{ spec NATBINTREE is } \]
\[ \text{NAT with SUMS with } \]
\[ \text{types ops } \text{tree} : \rightarrow \text{type} \]
\[ \text{eqns } \text{tree} = \text{nat} + \text{tree} \times \text{nat} \times \text{nat} \]

We even can specify parameterized recursive data structures

\[ A(C^3) \text{ spec BINTREE is } \]
\[ \text{SUMS with } \]
\[ \text{types ops } \text{tree} : \text{type} \rightarrow \text{type} \]
\[ \text{var } A : \text{type} \text{ in } \]
\[ \text{eqns } \text{tree}(A) = A + \text{tree}(A) \times A \times A \]

which defines tree structures of arbitrary height.

Remark. The specification mechanism does not allow to make a general statement like “all systems of type equations have a solution.” Such a statement is possible if hierarchical specifications of level greater than two are considered:
Assume that there is a third level of specification for which \texttt{SUPER-TYPE} plays the part of \texttt{type}. Then we can specify

\begin{verbatim}
A spec \(YC^3C^3\) is
  supertype ops \texttt{type}: \rightarrow \texttt{SUPERTYPE}
  var \(AA: \texttt{SUPERTYPE}\)
  in types  ops \(YY: [AA \rightarrow AA] \rightarrow AA\)
  var \(F: [AA \rightarrow AA]\)
  eqns \(YY'F = f(YY'F)\)
  in \texttt{SUMS} with
  \texttt{FIXPOINT} (replacing \texttt{type} by \(AA\))
\end{verbatim}

where \texttt{SUMS} is the specification obtained from \texttt{COPRODUCT} omitting the unicity axiom (**).

Extending the two-level approach in the obvious way, models of this specification are small cartesian closed categories, on each level products being monoids, such that each of the categories has a fixpoint operator and that the lower level categories have a sum. Construction of an initial model should give a cartesian closed category generated by the cartesian closed category \(C\) which has objects being formal solutions to recursive domain equations, and all recursive procedures over the base functions induced by the domain equations as morphisms. A hint how to confirm non-triviality of such a model may be the following consideration: Let \(C^p\) be the full subcategory of cpo's containing all cpo's which are a solution to a set of domain equations involving the functors 1, \(-\times-\), \(-+\) (sum), \(-\rightarrow\). Iterating this construction to functor and product categories of the categories achieved, the structure should be rich enough to interpret \(YC^3C^3\) (observe that domain equations in functor and product categories are computed pointwise).

It should be stressed that such structures are out of the scope of this paper, nevertheless we believe that they are of some interest. Again not too surprising, both the specification styles turn out to be equivalent in the following sense

3.5. Definition. The translations of \(A\)-terms and CCC-terms are extended to the specified operators by \(#\sigma\# = \sigma\), \([\sigma] = \sigma\) for \(\sigma: t_0 \cdots t_{n-1} \rightarrow t_n \in H\Sigma\). Let \(CCC[A\ SPEC]\) (resp. \(A[C^3\ SPEC]\)) denote the two-level specifications obtained by adding the translated equations, i.e.,

\begin{verbatim}
CCC[A\ SPEC] = CCC[HSIG] + \{(\([M]\), \([N]\)) | (M, N) \in AE\}
A[C^3\ SPEC] = A[HSIG] + \{(\#T\#, \#T'\#) | (T, T') \in C^3E\}.
\end{verbatim}
3.6. THEOREM. The theories $A[A \text{SPEC}](D)$ and $\text{CCC}[A \text{SPEC}](D)$ (resp. $\text{CCC}[C^3 \text{SPEC}](D)$ and $A[C^3 \text{SPEC}](D)$) are equivalent for all TYPE-algebras $D$.

This is a straightforward consequence of Theorem 2.9.

3.7. DEFINITION. A $A[A \text{SPEC}]$-theory is a $A$-theory $T$ such that

- the type algebra of $T$ is a TYPE-algebra
- $A[A \text{SPEC}](D)$ is contained in $T$ in the sense that all $A[A \text{SPEC}](D)$-terms are $T$-terms and that all $A[A \text{SPEC}](D)$-equations hold in $T$.

A homomorphism of $A[A \text{SPEC}]$-theories is a homomorphism $h: T \to T'$ of $A$-theories such that

$$h(I(t_0) \times \cdots \times I(t_{n-1}))(\sigma: I(t_0) \cdots I(t_{n-1}) \to I(t_n)) = \sigma: h(I(t_0)) \cdots h(I(t_{n-1})) \to h(I(t_n))$$

for $\sigma: t_0 \cdots t_{n-1} \to t_n \in H\Sigma$, $I: T_{\Sigma \text{TYPE}}(X) \to D^T \in \Sigma \text{TYPE}^b$, and $h: D^T \to D'^T \in \Sigma \text{TYPE}^b$. ("$h$ preserves the SPEC-structure") and similarly for $A[C^3 \text{SPEC}]$. The categories are denoted by $A[A \text{SPEC}]$-TH and $A[C^3 \text{SPEC}]$-TH.

We conclude from Proposition 2.12 and the definition

3.8. PROPOSITION. (i) $A[A \text{SPEC}]$-TH $\sim \text{CCC}[\text{SPEC}]^b$, $A[C^3 \text{SPEC}]$-TH $\sim \text{CCC}[C^3 \text{SPEC}]^b$.

(ii) $A[A \text{SPEC}](T_{\text{TYPE}})$ ($A[C^3 \text{SPEC}](T_{\text{TYPE}})$) are initial algebras in the corresponding categories, $T_{\text{TYPE}}$ being the initial TYPE-algebra.

This equivalence of specification styles justifies

3.9. DEFINITION. A specification with higher types $H\text{SPEC}$ is either a $A$- or a $C^3$-specification. $Th[H\text{SPEC}] = A[H\text{SPEC}](T_{\text{TYPE}})$ is called the concrete higher type theory presented by $H\text{SPEC}$. The isomorphism class of $C^T[H\text{SPEC}]$ is called the abstract higher type theory presented by $H\text{SPEC}$.

While the results state an equivalence of the $C^3$- and the $A$-idiom, examples such as Example 3.4(ii) demonstrate that specifications are written with greater ease in the $A$-style. Apparently variables are more suited to express the kind of structures being specifiable. We discuss the example of
coproducts to understand that the difficulties are inherent to the $C^3$-language. Coproducts can be introduced via the two-level specification

$$\text{spec BICCC is}$$

objects ops \(\_ + \_\): type \(\rightarrow\) type

var \(A, B, C: \text{type}\) in

morphisms ops \(u_{A,B}': (A, A + B)\)

\(v_{A,B}': (B, A + B)\)

\([\_ , \_]': (A, C)(B, C) \rightarrow (A + B, C)\)

var \(f: (A, C), g: (B, C), h: (A + B, C)\) in

eqns \(u_{A,B} \circ f = f\)

\(v_{A,B} \circ g = g\)

\([u_{A,B} \circ h, v_{A,B} \circ h] = h\)

(Models are small cartesian closed categories with binary coproducts). In comparison, $C^3$COPRODUCT uses function space objects to express the arity of the operators $\langle[\_, \_\_\_\_]: [A \rightarrow C][B \rightarrow C] \rightarrow [A + B \rightarrow C]\rangle$.

In fact $\langle[\_, \_\_\_\_]\rangle$ is the enriched version (Kelly, 1982) of $\langle[\_, \_\_\_\_]\rangle'$:

$$[A + B \rightarrow C] \times (A + B) \overset{\text{ev}_{A + B, C}}{\longrightarrow} C$$

$$[\_, \_\_\_\_] \times \langle[\_, \_\_\_\_]\rangle \overset{\gamma}{\cong} ([A \rightarrow C] \times [B \rightarrow C] \times (A + B)) + ([A \rightarrow C] \times [B \rightarrow C] \times A) + ([A \rightarrow C] \times [B \rightarrow C] \times B)$$

where \(p, p'\) are the obvious projections and the isomorphism \(\gamma\) is due to the fact that a functor $\_ \times A$ preserves colimits in a cartesian closed category. Similarly $u_{A,B}$, $v_{A,B}$ are the enriched versions of $u_{A,B}'$, $v_{A,B}'$ as well as the defining equations in $C^3$COPRODUCT are rephrased from the equations of the two-level specification of coproducts.

The example suggests the more general proceeding to generalize higher type specifications by including operators on function domains, i.e., operators

$$\sigma: (A_0, B_0) \cdots (A_{n-1}, B_{n-1}) \rightarrow (A_n, B_n) \quad (*)$$

and equations involving free variables over function domains. Such a specification method is more powerful but it has the disadvantage that functional completeness cannot be guaranteed (Lambek, 1974), where functional completeness states (in terms of the $A$-calculus) that for any
term \( M \) with a free occurrence of a variable \( x \) there exists a closed term \( N \) such that \( M = N^\prime x \) or \( M = Nx \). If we take the specification

\[
\text{CC with}
\]

\[
\text{objects ops } s: \rightarrow \text{type in}
\]

\[
\text{morphisms ops } \sigma: (1, s) \rightarrow (1, s)
\]

the initial algebra cannot be functionally complete as the term \( \sigma(f) \) cannot be abstracted (simply as variables of type \( f: (A, B) \) cannot be bound in \( \lambda \)-calculus, for a more elaborate counter example compare Lambek, 1974).

We conclude that specifications including operators of the form \( * \) not necessarily yield higher type theories as cartesian closed categories are functionally complete (Lambek, 1974).

4. MODELS FOR THEORIES WITH HIGHER TYPES

Given a theory with higher types, one naturally asks for a suitable model theory, prominent questions being those for existence of free models and for completeness of theories. Parsaye–Ghomi (1981) introduces a notion of models with higher types, and states that both questions can be answered positively. We follow his lines, but choose a different, but equivalent approach. Our results are contradictory to that in (Parsaye–Ghomi, 1981) as existence of initial models and completeness of theories can only be established in certain cases.

Our notion of models of theories with higher types is defined as follows:

4.1. Definition. Given a category \( C \), an object \( A \) of \( C \) is called a generator if for all morphisms \( g \neq h: B \rightarrow C \) there exists a morphism \( f: A \rightarrow B \) such that \( f \circ g \neq f \circ h \). If \( C \) has a terminal object which is generator, \( C \) is called concrete.

4.2. Remark. Any concrete category \( C \) may be viewed as a category of sets and functions with objects \( C[1, A] \) and morphisms \( C[1, f]: C[1, A] \rightarrow C[1, B], a \rightarrow a \circ f \) (concreteness guarantees the isomorphism), but concrete categories are easier to handle technically.

4.3. Definition. Let \( T \) be a abstract higher type theory (we shall henceforth only consider abstract higher type theory, i.e., CCC-models Definition 2.13). The category of \( T \)-models, denoted by \( T\text{-mod} \), has objects \( (A, F: T \rightarrow A) \) with \( A \) being a concrete cartesian closed category, and \( F: T \rightarrow A \) being a cartesian closed functor which is the identity on objects.
Given $T$-models $F: T \to A$, $F': T \to A'$, a family $h = (h_B: A[1, B] \to A'[1, B] \mid B \in T)$ is called a homomorphism of $T$-models if for all $f: B \to C \in T$ the diagram

$$A[1, B] \xrightarrow{h_B} A'[1, B]$$

commutes.

A functor $F: C \to C'$ between cartesian closed categories is called cartesian closed if it preserves finite products and if the canonically induced morphisms $\Phi_{A,B}: F([A \to B]) \to [F(A) \to F(B)]$ are isomorphisms where

$$[F(A) \to F(B)] \times F(A) \cong F(B)$$

4.4. Remark. (i) An equivalent, functorial definition is to state that $T$-models are weakly cartesian closed functors $F: T \to \text{Set}$, i.e., the $\Phi_{A,B}$'s are injective, with natural transformations as homomorphisms.

(ii) The definition of “algebras” with higher type (Parsaye-Ghomi, 1981) coincide with our notion of $T$-models. For a proof basically Remark 4.2 and the above remark may be used.

4.5. Examples. (i) Consider the specification $\text{NAT}$ given in Example 3.4(i). Let $I(\text{nat}) = \mathbb{N}_0$ and $I(1) = 1$, $I(d \times d') = I(d) \times I(d')$, $I([d \to d']) = [I(d) \to I(d')]$, and $\mathbb{N}_0[\mathbb{N}_0] \times \mathbb{N}_0 = \text{Set}(\mathbb{N}_0, \mathbb{N}_0)$ for $d, d' \in T_{\text{TYPE[NAT]}}$ with composition of functions. $\mathbb{N}_0$ defines a $Th[\text{NAT}]$-model with the canonical choice of operations.

(ii) Any $Th[\text{UNIVERSAL}]$-model $C$ defines a $\lambda$-model $M_C$ in the sense of (Barendregt, 1981):

Use that $\lambda$-models are equivalent to $\lambda$-families (Barendregt, 1981, 5.4.5). One can prove that

$$X := C[1, U]$$

$$F_n := \{C[1, f] \mid f: U^n \to U \in C\}$$

$$\Box(\_): F_1 \to X, \quad C[1, f] \to C[1, "f" \circ \_]$$

$$\_ \cdot \_ : C[1, d \times 1_{U \circ \text{ev}_{U, U}}] : X \times X \to X$$

$$\lambda \lambda_n: F_{n+1} \to F_n, \quad C[1, g] \to C[1, A(g)]$$
AXEL POIGNE

is a $\lambda$-model, where

\[
\begin{array}{ccc}
[U \to U] \times U & \xrightarrow{ev_{U,U}} & U \\
\downarrow \rho_x \\
1 \times U & \xrightarrow{f} & U \\
\end{array}
\quad \quad
\begin{array}{ccc}
[U \to U] \times U & \xrightarrow{ev_{U,U}} & U \\
\downarrow \rho_x \\
U \times U & \xrightarrow{g} & U \\
\end{array}
\]

The proof is a straightforward, but tedious computation. Concreteness guarantees that the above defines mappings. For $TH[UNIVERSAL]$-models the $\eta$-axiom holds as well. The idea is similar to that in Obtulowicz and Wiweger (1978) for universal objects with $U = [U \to U]$.

At the moment the connection between $Th[UNIVERSAL]$-mod and $TH[universal^p]$-mod and the respective categories of $\lambda$-calculi is not clear. We conjecture that any $\lambda$-model can be freely extended to a $Th[UNIVERSE^p]$-model.

If we translate the results of (Parsae–Ghomi, 1981) to our framework, it is stated that $T[1, \_]: T \to T^h$ is an initial $T$-model where

4.6. **Definition.** $T^h$ is the hom-category of $T$ with the same objects as $T$ and morphisms $T^h[B, C] := \{T[1, f] \mid f: B \to C \in T\}$ and composition of functions, with $T[1, f]: T[1, B] \to T[1, C], b \mapsto b \circ f$.

4.7. **Proposition.** $(T^h, T[1, \_])$ is an initial $T$-model iff $T$ is concrete.

Proof. Concreteness of $T$ implies that $T \simeq T^h$ as $f \neq g: A \to B$ iff $T[1, f] \neq T[1, g]$. Hence $(T^h, T[1, \_])$ is a $T$-model and trivially initial. Assume that $(T^h, T[1, \_])$ is initial. Let $f \neq g: A \to B \in T$. By abstraction $"f" \neq "g,"$ where

\[
\begin{array}{ccc}
[A \to B] \times A & \xrightarrow{ev_{A,B}} & A \\
\downarrow \rho_x \\
1 \times A & \xrightarrow{f} & A \\
\end{array}
\]

Then $T[1, "f"] \neq T[1, "g"]$ and $T[1, f] \neq T[1, g]$. $(T[1, \_])$ preserves cartesian closure by assumption, hence $T[1, "f"] = "T[1, f"]"$. As $T[1, f], T[1, g]$ are functions there exists a $a: 1 \to A \in T$ such that $a \circ f \neq a \circ g$.

Remark. (i) $(T^h, T[1, \_])$ is the canonical candidate for an initial $T$-
MODELS WITH HIGHER TYPES

model. The equivalent functorial T-model (Remark 4.4) is the hom-functor $T[1,-]: T \to \text{Set}$. Due to the Yoneda lemma (MacLane, 1972), there exists a unique natural transformation $\iota: T[1,-] \to F$ to any functor $F: T \to \text{Set}$ preserving the terminal object. But $T[1,-]$ is weakly cartesian closed iff T is concrete, or—in terms of T-models—$T^h$ is cartesian closed only if T is concrete. The basic observation is that $f \neq g: A \to B \in T$ must imply $T[1,f] \neq T[1,g]$ to obtain cartesian closure of $T^h$.

4.8. PROPOSITION. There are higher type theories which are not concrete.

Proof. Let

spec SIMPLE is

- types $s: \to \text{type}$
- in ops $a: \to s$
  $f, g: s \to s$
- eqns $a \circ f = a \circ g = a$

The generated A-calculus is finite Church–Rosser if conversion is replaced by reduction, and if the rewrite rules $f'a \to a$, $g'a \to a$ are added. A proof for this immediately follows by adapting the proof of (Gandy, 1980) for typed $\lambda$-calculi with products. The rewrite rules do not cause particular problems as the reduction and the rewriting commutes trivially (compare Voss, 1983). But then normal forms exist and $f, g: s \to s$ are in normal form. Hence $f \neq g$ in $T := Th[\text{SIMPLE}]$. Again using the existence of normal forms, one can prove that $T[1,s] = \{a\}$. We conclude that $T[1,f] = T[1,g]$ (the use of the A-calculus is justified by the results given above).

The example SIMPLE—as well as NAT where, for instance, $\lambda\langle x: \text{nat}, y: \text{nat}\rangle \cdot \text{add}'\langle x, y\rangle \neq \lambda\langle x: \text{nat}, y: \text{nat}\rangle \cdot \text{add}'\langle y, x\rangle$, but the underlying “functions” are equal—demonstrates that the distinction between theories and models is that between intension and extension; morphisms may be identified as “functions” on the categorical “elements” but they are different as “elements” of a function space object. The observation stimulates the idea that a distinction between intensional and extensional models may be appropriate. Intensional model will be called algebras.

4.9. DEFINITION. (i) A category $C$ with finite products is called weakly cartesian closed if for all objects $B, C \in C$ there exists an object $[B \to C]$
and a morphism \( ev_{B,C} : [B \to C] \times B \to C \) such that for any morphism \( f : A \times B \to C \) there exists a morphism \( A(f) : A \to [B \to C] \) such that

\[
[B \to C] \times B \xrightarrow{ev_{B,C}} C \\
A(f) \times 1_B \downarrow \quad \downarrow f \\
A \times B
\]

commutes (compared to cartesian closure the unicity axiom is missing).

(ii) Given a higher type theory \( T \), \((A, F : T \to A)\) is called a **T-algebra** if \( A \) is a concrete weakly cartesian closed category with the same objects as \( T \) and \( F : T \to A \) is a functor which preserves finite products and weak cartesian closure (i.e., \( F([B \to C]) = [F(B) \to F(C)] \) and \( F(ev_{B,C}) = ev_{F(B),F(C)} \)), and which is the identity on objects. With homomorphisms as defined in Definition 4.3 this defines a category \( T\text{-alg} \).

**Remark.** \( T\text{-mod} \) is a full subcategory of \( T\text{-alg} \).

**Examples.** Any \( \text{Th}[\text{UNIVERSE}] \) algebra defines a \( \lambda \)-algebra in the sense of (Barendregt, 1981) using the characterization via combinatory algebras (Barendregt, 1981, 5.4.12):

\[
X := A[1, U] \\
_\cdot_ := A[1, d \times 1_U \circ ev_{U,U}] : X \times X \to X \\
K := A(A(p_{U,U}) \circ c) \circ c \\
S := A(A(A(A(p_{U,U} \circ d \times ev_{U,U} \circ ev_{U,U}) \circ c) \circ c) \circ c
\]

\((p_i : U \times U \times U \to U \text{ projections})\). A proof is tedious. Again the relationship between the different categories of algebras is not settled yet.

4.10. **Definition.** Let \( C \) be a cartesian closed category, and \( A \) be an object of \( C \). Then \( C_A \) is the category with the same objects as \( C \) and morphisms \( C_A[B, C] := C[A \times B, C] \). Composition is defined by \( f^A \circ g^A := \langle p_{A,B}, f \rangle \circ g \) for \( f : A \times B \to C, \ g : A \times C \to D \) (Lambek, 1974), where \( f^A : A \times B \to C \) is used to distinguish \( f : A \times B \to C \) as an element of \( C_A \).

A prerequisite for the further development is the

4.11. **Theorem** (Lambek, 1974). (i) \( C_A \) is a cartesian closed category, and the functor \( E_A : C \to C_A, \ f \mapsto q_{A,B} \circ f \) for \( f : B \to C \in C \) preserves cartesian closure.
(ii) For any cartesian closed functor $F: C \to D$ and any morphism $a: 1 \to F(A) \in D$ there exists a unique functor $F[a]: C_A \to D$ such that

$$\begin{array}{c}
C \\ \downarrow F \\
D \\
\end{array} \xleftarrow{E_A} \xrightarrow{F[a]} \xrightarrow{\downarrow F[a]} C_A$$

commutes and $F[a](p^A_{A,1}) = a$. $F[a]$ is defined by $F[a](f^A) := \langle \langle b \circ a, 1_B \rangle \circ F(f) \rangle$ for $f: A \times B \to C \in C$.

(iii) $E_A$ is faithful if $C[1, A] \neq \emptyset$.

We have the following additional properties:

4.12. Proposition. $C_A$ is concrete if $C$ is concrete.

Proof. Let $f \neq g: A \to C \in C \subseteq C_A[A, B, C]$. By concreteness of $C$ there exists a $\langle a, b \rangle: 1 \to A \times B$ such that $\langle a, b \rangle \circ f \neq \langle a, b \rangle \circ g$. But then $\langle p_{A,1}, q_{A,1} \circ b \rangle \circ f \neq \langle p_{A,1}, q_{A,1} \circ b \rangle \circ g$ as $\langle a, 1_B \rangle \circ \langle p_{A,1}, q_{A,1} \circ b \rangle \circ f = \langle a, b \rangle \circ f$.

4.13. Proposition. Let $f \neq g: A \to C \in C$. Then $p^A_{A,1} \circ E_A(f) \neq p^A_{A,1} \circ E_A(g)$ in $C_A$.

Proof. As $p^A_{A,1} \circ E_A(f) = \langle p_{A,1}, p_{A,1} \rangle \circ q_{A,1} \circ f = p_{A,1} \circ f$, $p^A_{A,1} \circ E_A(f) = p^A_{A,1} \circ E_A(g)$ implies $f = g$.

4.14. Proposition. $(T^h_A, T_A[1, \_])$ is a $T$-algebra for all objects $A \in T$.

Proof. $1$ is a generator by Proposition 4.13. $T^h_A$ is a category with finite products as hom-functors preserve finite products. Choose $T_A[1, \text{ev}_{B, C}]$ and $T_A[1, A(f)]$ as weakly cartesian closed structure.

4.15. Proposition. $(T^h, T[1, \_])$ is an initial $T$-algebra.

Proof. Let $(A, F)$ be a $T$-algebra. We construct a homomorphism $i: T^h \to A$ by $i(T[1, b]) := F(b)$ with $b: \to B \in T$. It is straightforward to check that $i$ is the unique homomorphism.

Next we analyse completeness.

4.16. Definition. A higher type theory $T$ is ext- (int-) complete if

$$\forall f, g: A \to B \in T: [\forall (A, F) \in \text{T-mod} (\in \text{T-alg}): F(f) = F(g) \Rightarrow f = g].$$

We immediately conclude
4.17. **Proposition.** (i) **T** is ext-complete if **T** is concrete.

(ii) Every theory is int-complete.

**Proof.** (i) is obvious. For (ii) use \( T^h \) and observe that \( T[1, q_{A,A} \circ f] \neq T[1, q_{A,A} \circ g] \) for \( f \neq g: A \to B \in T \).

For the rest of the paper we are concerned with **T**-models. One may wonder if the results on existence of initial models and on ext-completeness of theories are exact. The answer is not so obvious because of some pathological cases (at least from the experience of set theory).

4.18. **Remark.** We conjecture that there exist higher type theories which (i) have initial models but are not ext-complete, (ii) are ext-complete but do have initial models, (iii) are neither ext-complete nor have initial models. So far we have not been able to construct examples which do not seem to be straightforward as questions such as completeness of typed \( \lambda \)-calculi with equations are involved (Friedman, 1975). We believe that such examples should use the

4.19. **Observation.** Any **T**-model \( (A, F) \) satisfies \( \forall f, g: A \to B: f \neq g \Rightarrow \langle f \rangle_A \neq \langle g \rangle_A \) is epi.

According to this conjecture neither completeness nor existence of initial algebras sufficiently characterize concrete theories but a combination may do.

4.20. **Observation.** A higher order theory \( T \) is concrete if it is ext-complete and has an initial model \( (I, I) \) such that \( I: T \to I \) is full.

**Proof.** Let \( f, g: A \to B \in T \). \( I(f) = I(g) \) implies \( F(f) = F(g) \) for all **T**-models \( (A, F) \) (use \( f = g \Leftrightarrow "f" = "g" \)), hence \( f = g \) in \( T \) by completeness. Thus \( I \) is faithful and an isomorphism by assumption, \( I \) being an identity on objects.

One immediately suspects that the functor \( I: T \to I \) must be full for an initial **T**-model \( (I, I) \), otherwise leaving freedom for interpretation. Unfortunately, we cannot imagine a straightforward proof because of the rather complex internal structure of concrete cartesian closed categories. The only method applicable seems to use functional completeness of cartesian closed categories as discussed in (Lambek, 1974).

We now proceed as follows: Given an ext-complete theory \( T \) such that an initial model exists, we construct a model \( T^c \) which is a faithful extension of the theory. The construction adds enough “indeterminates” by a transfinite construction (\( C_A \) may be viewed as “adding an indeterminate” \( x: 1 \to A \) to \( C \), Lambek, 1974). If the initial algebra is not “full” the initial homomorphism \( I \to T^c \) maps a morphism of \( I \) to a morphism added in the construction of \( T^c \). As indeterminates can be freely interpreted, this
morphism can be mapped to different morphisms of a suitable category. But then we have two different homomorphisms out of the initial model into the same model, being a contradiction to the assumption.

We first construct the model $T^c$.

4.21. Definition. Let $T$ be a higher type theory. An object $A$ of $T$ is called weakly initial if $T[1, A] = \emptyset$ and $|T[A, B]| \leq 1$ for all objects $B \in T$.

4.22. Lemma. $B \in T$ is weakly initial iff $T[1, B] = \emptyset$ and $|T[A \times B, C]| \leq 1$ for all $A, C \in T$.

Proof. $T[1, B] = \emptyset$ implies $C[1, A \times B] = \emptyset$. \[\]

4.23. Lemma. Let $T$ be a higher type theory, and let $C \in T$ be weakly initial. If $T[1, A] \neq \emptyset$ then $C$ is weakly initial in $T_A$.

Proof. $T_A[1, C] = T[A \times 1, C] = \emptyset$ as $f: A \times 1 \to C$ implies $\langle a, 1 \rangle \circ f: 1 \to C \in T$. $|T_A[A \times C, D]| = |T[A \times B \times C, D]| \leq 1$ for all $B, D \in T$ by assumption. \[\]

4.24. Lemma. (i) Let $(F_{\alpha, \beta}: T_{\alpha} \to T_{\beta} \mid \alpha, \beta < \lambda)$, $\lambda$ being a limit ordinal, be a transfinite chain (i.e., $F_{\alpha, \beta} \circ F_{\beta, \gamma} = F_{\alpha, \gamma}$) of theories and theory morphisms which are the identity on objects. Then the colimit $(T_{\lambda}, (F_{\alpha, \lambda}: T_{\alpha} \to T_{\lambda}))$ is given by

$$T_{\lambda}[A, B] := \bigcup_{\alpha < \lambda} T_{\alpha}[A, B]$$

with $f \sim g$ iff $\exists \gamma < \lambda: F_{\alpha, \gamma}(f) = F_{\beta, \gamma}(g)$ for $f: A \to B \in T_{\alpha}$, $g: A \to B \in T_{\beta}$. For definition of the operations use that for all $[f], [g] \in T_{\lambda}$, there exists an $\alpha < \lambda$ and $f', g' \in T_{\alpha}$ such that $[f'] = [f]$ and $[g'] = [g]$.

(ii) If $A$ is weakly initial in all $T_{\alpha}, \alpha < \lambda$, then $A$ is weakly initial in $T_{\lambda}$.

Proof. (i) As all $T_{\alpha}$ are CCC(D)-algebras, $D$ being the algebra of objects which is identical for all $T_{\alpha}$, we can apply the standard construction of filtered colimits (MacLane, 1972) which is given.

(ii) Trivially $T_{\lambda}[1, A] = \emptyset$ and $|T_{\lambda}[A, B]| \leq 1$ as $F_{\alpha, \beta}(f) = f': A \to B \in T_{\beta}$ for $f: A \to B \in T_{\alpha}$. \[\]

4.25. Definition. Let $T$ be a higher type theory, $\mathcal{X} := \{ x: A \mid x \in X, A \in T \text{ such that } T[1, A] \neq \emptyset \}$, where $X$ is a set with $|X| \geq |T^0| + \aleph_0$ ($T^0$ objects of $T$), and let $\gamma$ be the least ordinal such that $|\gamma| = |\mathcal{X}|$ with $i: \gamma \to \mathcal{X}$ being an isomorphism.
We construct a transfinite chain of higher type theories by

(a) \( T_0 := T \)
(b) \( T_x := T_\alpha, F_{\alpha, \beta} := E_\beta : T_\alpha \to T_\beta \), where \( \iota(\alpha) = \alpha : A \)
(c) \( (T_\lambda, (F_{\alpha, \lambda} : T_\alpha \to T_\lambda | \alpha < \lambda)) \) is the colimit of the chain \( (T_\alpha | \alpha < \lambda) \), for limit ordinals \( \lambda \leq \gamma \).

4.26. Theorem. Let \( T \) be a higher type theory such that \( A \in T \) is weakly initial if \( T[1, A] = \emptyset \). Then \( T' := T_\gamma \) is concrete and \( F := F_{0, \gamma} : T \to T_\gamma \) is faithful.

Proof. (i) All functors \( F_{\alpha, \beta} : T_\alpha \to T_\beta \), \( \alpha \leq \beta \leq \lambda \), are faithful:

(a) \( F_{\alpha, \beta} : T_\alpha \to T_\beta \) is faithful by Theorem 4.11(iii) as \( T_\alpha[1, A] \neq \emptyset \), where \( \iota(\alpha) = \alpha : A \).
(b) Let \( \alpha < \lambda \leq \gamma \), \( \lambda \) being a limit ordinal, \( f \neq g : A \to B \in T_\alpha \). By assumption the functors \( F_{\alpha, \beta} : T_\alpha \to T_\beta \), \( \alpha \leq \beta < \lambda \), are faithful, hence \([f] \neq [g]\) in \( T_\lambda \).
(ii) By Lemmas 4.23 and 4.24, \( A \) is weakly initial in \( T_\gamma \) if \( A \) is weakly initial in \( T \).
(iii) Assume that \( f \neq g : A \to B \in T_\gamma \). Then \( f' \neq g' : A \to B \in T_\gamma \) for some \( \alpha < \gamma \) where \( F_{\alpha, \gamma}(f') = f, F_{\alpha, \gamma}(g') = g \). Then there exists a least ordinal \( \beta \) such that \( \alpha < \beta \) and \( \iota(\beta) = \beta : A \). Otherwise \( \{x : A \mid x \in X\} \leq \alpha \), but \( \{x : A \mid x \in X\} \leq |X| + \aleph_0 \). Without restriction of generality let \( \beta = \alpha' \). Then \( p_{\alpha, 1}^A \circ F_{\alpha, \alpha}(f') \neq p_{\alpha, 1}^A \circ F_{\alpha, \alpha}(g') \) in \( T_\alpha \) (4.13), and \( F_{\alpha, \alpha}(p_{\alpha, 1}^A) \circ f \neq F_{\alpha, \alpha}(p_{\alpha, 1}^A) \circ g \).

The next step is to use \( (T_\gamma, F) \) to establish the contradiction sketched above. We need

4.27. Observation. Any cartesian closed functor \( G : A \to A' \) such that

\[
\begin{array}{ccc}
\text{T} & \xrightarrow{F} & \text{T} \\
\downarrow & & \downarrow \\
A & \xrightarrow{G} & A'
\end{array}
\]

and such that \( G \) is the identity on objects defines a homomorphism \( G : (A, F) \to (A', F') \) of T-models with \( G_b(b) := G(b) \) for \( b : 1 \to B \in A \). (For convenience we use the same notation for functor and corresponding homomorphism.)

Proof. \( A'[1, F'(f)](G_b(b)) = G_b(b) \circ F'(f) = G(b) \circ G(F(f)) = G(b \circ F(f)) = G_{C}(A[1, F(f)](b)) \) for \( f : B \to C \in T \).
Remark. In fact, the set of homomorphisms between $T$-models $(A, F)$, $(A', F')$ and the set of cartesian closed functors $G: A \to A'$ satisfying the above requirements are isomorphic. The proof is cumbersome as it depends on “enriched” versions of the structure.

4.28. **Lemma.** Let $T$ be a ext-complete higher type theory, $(I, I)$ an initial $T$-model. Then an object $A$ of $T$ is weakly initial if $T[1, A] = \emptyset$.

**Proof.** Let $T_f$ with $|T_f[A, B]| = 1$ if $T[A, B] \neq \emptyset$ is a concrete higher type theory. (The theory properties are computed via $T$.) By initiality there exists a homomorphism $F_f: I \to T_f$, hence $T[1, A] = \emptyset$ implies $I[1, A] = \emptyset$. Completeness ensures that $f \neq g: A \to B \in T$ implies $I(f) \neq I(g)$ in $I$ (4.20) and weak initiality of $A$ in $T$.

4.29. **Theorem.** A higher type theory is concrete iff it is ext-complete and has an initial model.

**Proof.** Let $T$ be an ext-complete higher type theory, $(I, I)$ an initial $T$-model. Assume that $T$ is not concrete and $I: T \to I$ is not faithful. Then there exists a $y: 1 \to A \in I$ such that

(i) $y: 1 \to A \notin \text{Im}(I)$

(ii) $y \circ I(f) \neq y \circ I(g)$ for some $f \neq g: A \to B \in T$

(iii) $x \circ f = x \circ g$ (in $T$) for all $x: 1 \to A$, $f, g: A \to B \in T$.

Moreover there exists a unique $T$-homomorphism

$$
\begin{array}{ccc}
I & \xrightarrow{I} & T \\
\downarrow & & \downarrow F \\
T & \xleftarrow{g} & T^c
\end{array}
$$

where $(T^c, F)$ is constructed as in Definition 4.25, being a model by Theorem 4.26. On the other hand, we use transfinite induction to define a cartesian closed functor

$$
\begin{array}{ccc}
T^c & \xrightarrow{H} & I \\
\downarrow & & \downarrow I \\
T & \xleftarrow{g} & T
\end{array}
$$
as follows: For any $B$ such that $I[1, B] \neq \varnothing \ (\Rightarrow T[1, B] \neq \varnothing)$ choose an element $x_B: 1 \to B \in I$. Let

(a) $H_0 := I: T \to I$

(b) $H_x := H_x[x_B]: T_x \to I$ where $\iota(x) = x: B \ ((H_{B | B \leq x})$ is so)

(c) $H_\lambda: T_\lambda \to I$ is uniquely induced from the cocone $(H_\alpha | \alpha < \lambda)$ for limit ordinals $\lambda \leq \gamma$.

By initiality $g \circ H = 1_1 (H$ is a homomorphism Observation 4.27). Then $g(y) \circ F(f) \neq g(y) \circ F(g)$ and $g(y) \notin \text{Im}(F) \ (g(y) \in I(F)$ then $y \circ f = y \circ g$). We conclude that there exists a $\beta < \gamma$ such that $F_{\beta, \gamma}(g(y)) \neq \varnothing$ but $F_{\beta', \gamma}(g(y)) = \varnothing$. Assume that $\iota(\beta') = x: B$. There exists a morphism $h: B \times 1 \to A \in T_{\beta}$ such that $h^B = F_{\beta, \gamma}(g(y))$ and $h \neq q_{B, 1} \circ a$ for all $a: 1 \to A \in T_{\beta}$ (otherwise $F_{\beta, \gamma}(g(y)) \neq \varnothing$, i.e., $g(y)$ depends on the indeterminate added by constructing $T_{\beta'}$. (Observe that all computations are on horn-sets of the form $T_\alpha[1, -].$)

Next we construct a homomorphisms (cartesian closed functor) $K: T^c \to T_{\beta}$ as follows:

(a) $K_\alpha := F_{\alpha, \gamma} \circ E_B$ for $\alpha \leq \beta$.

(b) $K_{\beta'} := K_{\beta'[p_{\alpha, 1}^B]} (p_{\alpha, 1}^B: 1 \to B \in T_{\beta})$

(c) $K_{\zeta} := K_{\zeta[z]}$ for $\beta' \leq \zeta$ and for some $z: 1 \to C \in T_{\beta}$, where $\iota(\zeta') = x: C$ (observe that $T^c[1, C] \neq \varnothing$ implies $T_{\beta}[1, C] \neq \varnothing$).

(d) $K_\lambda$ is induced by colimit properties for limit ordinal $\lambda \leq \gamma$.

We now compute:

(i) $E_B(g(y)) = q_{B, 1} \circ g(y)$ in $T^c$.

(ii) $K(g(y)) = K_{\beta'}(h^B) = K_{\beta'[p_{\alpha, 1}^B]}(h^B)$

$= \langle \langle \gamma \circ p_{\alpha, 1}^B, 1 \gamma \rangle \circ K_{\beta}(h)$

$= \langle \langle \gamma \circ p_{\alpha, 1}^B, 1 \gamma \rangle \circ E_B(h)$ (use that $F_{\beta, \gamma}$ is faithful)

$= \langle \langle \gamma \circ p_{\alpha, 1}^B, 1 \gamma \rangle \circ (q_{B, 1} \circ h)^B$.

Interpreted in $T^c$ we obtain $K(g(y)) = h$. From faithfulness of $F_{\beta, \gamma}$ we conclude $h \neq q_{B, 1} \circ g(y)$. Thus $g \circ E_B \neq g \circ K: I \to T_{\beta}$ in contradiction to initiality of $I$ ($T_{\beta}$ is concrete by Observation 4.19). But then $I: T \to I$ must be full what in combination with Definition 4.16 proves the theorem. 

Remark. It should not be too surprising that weakly initial objects cause some problems as initial objects cannot have an element in a cartesian closed category; $x: 1 \to 0$ implies that $A \simeq I$ for all objects $A$. 


4.30. **COROLLARY.** For a higher type theory $T$ with initial algebra $(I, I)$ the functor $I: T \to I$ is full.

**Proof.** Apply image factorization (in $\text{CCC}(T^0)$)

\[
\begin{array}{ccc}
T & \xrightarrow{I} & I \\
\downarrow & & \downarrow \\
\text{Im}(I) & & \\
\end{array}
\]

We use faithfulness of $\text{Im}(I) \to I$ to apply the proof of Theorem 4.29 to $\text{Im}(I)$ observing that completeness is only used to establish faithfulness. But $\text{Im}(I)\text{-mod} \simeq T\text{-mod}$. □

The remaining problem is to characterize higher type theories which are complete (resp. have initial algebras).

4.31. **DEFINITION.** An object $A$ of a higher type theory $T$ is called safe if $A$ is weakly initial or $\langle \rangle_A: A \to 1$ is epi.

**Remark.** Any object $A$ is safe if $T[1, A] \neq \emptyset$.

4.32. **LEMMA.** $E_A: T \to T_A$ is faithful if $\langle \rangle_A: A \to 1$ is epi.

**Proof.** As $- \times B$ is a left adjoint, as left adjoints preserve colimits, and as any epi is uniquely determined by a pushout square (Herrlich–Strecker, 1974) $\langle \rangle_A \times 1_B: A \times B \to 1 \times B$ is epi for all objects $B$. But $q_{A,B} = \langle \rangle_A \times 1_B \circ q_{1,B}$. □

4.33. **PROPOSITION.** A higher type theory is complete if all objects are safe.

**Proof.** Apply the argument of Theorem 4.26, but modify the construction by choosing $\mathcal{X} := \{x: A | x \in X, A \in T \text{ such that } \langle \rangle_A: A \to 1 \text{ is epi} \}$ as the set of indeterminates to be added. □

Proposition 4.33 gives us a sharp characterization of theories which have a functionally free algebra in the sense that there exists an algebra $(A, F)$ such that $F$ is faithful. It seems quite unlikely to characterize complete theories further except that one can state

**Remark.** If $T$ is complete then $q_{A,A} \circ f \neq q_{A,A} \circ g$ for $f \neq g: A \to B \in T$ with $A$ being not safe.

The lack of a sharper characterization results from the fact that factorization may strongly depend on the order indeterminates are added. We
conclude the section with a remark on the existence of free T-algebras and T-models. Given a higher type theory T, forgetful functors
\[ U: T\text{-alg} \to \text{Set}^S \quad V: T\text{-mod} \to \text{Set}^S \]
are defined by \( U(A, F) := (A[1, B] \mid B \in S) \), \( V(A, F) := (A[1, B] \mid B \in S) \), where \( S := \{B \in T \mid B \text{ not weakly initial in } T\} \).

4.34. Proposition. (i) \( U \) has a left adjoint.

(ii) \( V \) has a left adjoint if an initial T-algebra exists.

Proof. Given a S-ordered set \( Y \) adjoin the indeterminates by a (transfinite) construction along the lines of Definition 4.25. Observe that the construction of Proposition 4.17 preserves weak cartesian closure.

5. Conclusion and Outlook

We have looked for a basis of a theory of abstract data types which include the use of higher types. The approach appears to be natural because of the equivalence of the "algebraic" theory of cartesian closure and the "operational" \( A \)-calculus. While cartesian closure stresses the structural aspect, the \( A \)-calculus provides some intuition about how to compute with higher type structures. The \( A \)-calculus turns out to be very helpful to prove properties of a specification as several results such as normalization, Church–Rosser theorems, etc., are available in the literature. We have for instance used the \( A \)-calculus to investigate the composition of implementations of abstract data types in (Poigné and Voss, 1985), where proofs essentially depend on operational properties of the \( A \)-calculus.

While the question of higher type theories seems to be quite settled, the situation is less satisfactory for models: Algebras behave nicely with regard to completeness and existence of free algebras but loose structural properties as cartesian closure, while completeness and existence of free structures does not hold for models in general. If we compare the advantages and disadvantages, we believe that algebras are more suited for a theory of data type specifications. The loss of cartesian closure may even be desirable: Assume that we have a specification with two operators of the same arity for which rather different axioms hold, but which cannot be distinguished as functions if applied to terms. One may ask if the operators have the same "meaning." Certainly, they are equal from an extensional point of view, but from a pragmatic point of view they may be considered as different as they may be used in quite different contexts (in practice they may be implemented in different ways for instance). In general, we believe that
models with higher types

the use of functions in abstract data type theory is intensional, hence a
model theory based on algebras seems to be justified. Nevertheless models
are interesting for theoretical purposes.

A line of future research should extend the equivalence of categorical
structures and \( \lambda \)-calculi to more complicated structures such as Mar-
tin–Löf type theory (Martin–Löf, 1979), second order typed \( \lambda \)-calculus
(Reynolds, 1974), or the theory of constructions of (Coquand and Huet,
1984).

For a methodology of specifications with higher types our work merely
provides elements of a mathematical foundation leaving open all problems
of modularization techniques which are to be developed in subsequent
papers.

acknowledgments

I would like to thank Samson Abramsky for his help when preparing the text, and the
referees for valuable comments.

received August 1984; accepted May 13, 1985

references

Burstall, R. M., and Goguen, J. A. (1980), Semantics of CLEAR, a specification language,
Coquand, T., and Huet, G. (1984), A theory of constructions, preprint; presented at the Inter-
national Symposium on Semantics of Data Types at Sophia-Antipolis.
Chalmers University, Göteborg.
Parameterized data types in algebraic specification languages, in “Proc. Int. Colloq.
Automata Lang. and Programming 1980, Nordwijkerhout, Lecture Notes in Computer
Friedman, H. (1975), Equality between functionals, in “Logic Colloquium” ((R. Parikh, Ed.),
York/Berlin.
Gandy, R. (1980), Strong normalization of typed \( \lambda \)-calculus, in “To H. B. Curry: Essays on
Combinatory Logic, Lambda Calculus, and Formalism” (P. J. Seldin and J. R. Hindley,


GUTTAG, J. V. (1976), Abstract data types and the development of data structures, SIGPLAN Notices 8.


