Bachelorthesis: Fundamental mathematics

Lambda calculus

Kobe Wullaert

0.1 Abstract

In this thesis we are going to study the semantics of the (untyped) lambda calculus and in particular we show a characterization of the theory corresponding to Scott's model. Lambda calculus is a formal way to define functions and describe their equality, this lays the foundation of the functional programming paradigm. The semantics of a language is like a representation theorem, it gives a meaning to the language (more specifically to each term). There are different sorts of semantics, the one that we will discuss is the denotational/mathematical semantics. We are going to represent functions and their equality as a mathematical structure (model). From a mathematical viewpoint this is very interesting because it is analogous to logic, where we also describe equality between terms, but instead of abstract functions, terms represent truth values.
Most of this thesis is based on the book [3], which is a leading book about the untyped lambda calculus.

0.2 Introduction

In the first chapter we introduce all concepts of the untyped λ-calculus, so we introduce different axioms that are used in the theory, some properties about terms like solvability, some special terms which lead to combinatory logic, an algebraic theory that corresponds to the λ-theory. We also define some properties about theories like Hilbert-Post complete theories, which are maximal theories such that not every term is in relation with every other. In the end of the chapter we introduce and investigate a specific theory K which will turn out to be the answer to the main question of this thesis: what is the theory of Scott's model? Then in chapter 2, we introduce Bohm-trees corresponding to terms, this tree is like a parse tree, but with the exception that before setting up the parse tree, the term is replaced by a minimal (equal) term called the head normal-form. Then in chapter 3, everything we defined earlier, will come together. We first develop the notion of a λ-model, but before doing this we first introduce combinatory logic. Then we introduce other models like the Term and syntactical model and we show that these are indeed λ-models. The term model is the most intuitive model (corresponding directly to the theory) and we need it to show that an equality in a λ-theory holds if and only if it holds in every λ-model which is a very powerfull theorem. The syntactical model is later used to show that Scott's model D_∞ is indeed a model. But before we can define D_∞ we need the notion of complete partial orders (cpo's) and projective limits (on cpo's). At last we prove that the theory K^* is indeed the theory of D_∞.
Although the λ-calculus is mainly used in theoretical computer science, this was not the reason why it was introduced. Alonzo Church is the founder of the calculus with the goal to create a new foundation of mathematics (so a new way of defining logic) and to investigate the notion of a function. Combinatory logic was developed for the same reason by eliminating bound variables and replacing them by combinators, but was researched (at the beginning) indepen-

dantly. The founder of combinatory logic is Moses Schonfinkel, but was further developped by Haskell Curry.
The theorem that motivates the statement that the lambda calculus is another foundation of mathematics is given by the Curry-Howard isomorphism, also called the Curry-Howard correspondance. This statement gives an equivalence between the different types of lambda calculi and different types of logic. For instance the simply-typed lambda calculus corresponds to the minimal propositional logic. More generally, without the explicit use of the λ-calculus, is that proof systems are equivalent to the models of computation. More information about the Curry-Howard correspondance is given in [14].
The reason why the λ-calculus is so important nowadays in computation is because of the Church-Turing hypothesis. This states that the Turing machine (which is a mathematical model of a computer consisting of a memory tape that can be changed) is as powerfull as a real computer. In other words this is stated as: Every computation can be performed by a turing machine. The lambda calculus is what is called turing complete, this means that the λ-calculus can generate any turing machine and so every computable function can be abstracted to a λ-function. For the interested reader there is an appendix added where some basic fundamentals of a programming language are defined in terms of the lambda-calculus, to give an intuition of the lambda calculus as a programming language and thus that it is turing complete.

Contents

Chapter 1

Theories

1.1 The lambda theory

The λ-theory is a formal language of equality representing a more abstract way to define functions and how they behave, i.e. we have terms (in our case they represent functions or input to functions) and equality between those terms together with notions like application. This is a general way to define both the composition of functions as applying an argument to a function. In particular we look at the untyped λ-calculus, this means that a function can take any sort of input. In mathematics a function is always from an object in a set/category to another object in (maybe) another set/category, while in the λ-calculus it doesn't matter where you start or end.
Before we give the definition of the λ-theory, we must introduce lambda terms. These will represent the abstract functions. There are 3 sort of terms: variables, functions and application of terms. The variable is very natural, the meaning of application is explained further but can be seen as composition of functions. A mathematician usually writes a function as $f(x) = ...$ or $x \mapsto ....$ As the λ-calculus tries to formalize it more abstractly another notation is used: $\lambda x.....$ So a function $f(x) = 5 * x$ is represented as $\lambda x.(5 * x)$.

Definition 1. *The set of λ-terms over a set Var, is denoted by Δ and is defined as follows:*

- $x \in Var \implies x \in \Delta$
- *Abstraction:* $x \in Var, M \in \Delta \implies (\lambda x.M) \in \Delta$
- *Application:* $M, N \in \Delta \implies (MN) \in \Delta$

where Var is a set with elements named variables.

The set of variables can be uncountable, but in most definitions it will be countable since a function usually has a finite or countable number of variables. Variables are mostly called $x, y, x_1, x_2, ...$ The terms we just defined are the functions.

Notation 1.
- *If a term has multiple input variables $x_1, ..., x_n$, then instead of writing $\lambda x_1.(\lambda x_2.(...\lambda x_n)...)$ we write: $\lambda x_1 x_2 ... x_n$.*
- *The application is always left-associative when no brackets are included, so $MNP = ((MN)P)$*

The following example shows that application is in general not associative.

Example 1. *Let $M \equiv \lambda xy.x, N \equiv P \equiv \lambda x.x$, then:*

$$(MN)P = (\lambda yx.x)(\lambda x.x) = \lambda x.x$$
$$M(NP) = (\lambda xy.x)(\lambda x.x) = \lambda y.(\lambda x.x) = \lambda yx.x.$$

As $(MN)P$ is a 'function' with 1 input variable and $M(NP)$ is a 'function' with 2 input variables they aren't the same.

1.1.1 Free variables

As we will see, before we can properly define the λ-theory, we need the concept of a free variable. A variable x is free in a term M if it isn't in a scope λx in M, more formally:

Definition 2. *The set of free variables of a term M, denoted by $FV(M)$, is defined as follows:*

- $FV(x) = x$
- $FV(\lambda x.M) = FV(M) - \{x\}$
- $FV(MN) = FV(M) \cup FV(N)$

Definition 3. *A term $M \in \Delta$ is closed if it has no free variables, i.e.*

$$FV(M) = \emptyset$$

The set of closed lambda-terms is denoted by Δ^0. A closed term is sometimes called a combinator.

Example 2.
- *$\lambda x.xy$ isn't closed since $FV(\lambda x.xy) = \{y\}$, so y is free and x is bounded*
- *$\lambda xy.xyx$ is closed, both x and y are bounded*
- *If we look at $x\lambda x.x$, x is both bounded and free. Although this is valid notation, if we encouter something like this we will replace an x by another variable.*

1.1.2 Theory of lambda

Now we formally define the equational theory: the λ-theory, discuss the substitution of terms and introduce some special λ-terms with a gap, called contexts.

Remark 1. *By a(n equational) theory over the term-set Δ, we mean a set together with an equivalence relation, which in our case is written as $=$ and if $\mathbb{T}_1, \mathbb{T}_2$ are theories, we denote $\mathbb{T}_1 + \mathbb{T}_2$ as the theory containing all the equalities that can be derived from using equalities from both theories. More formally we mean that the equivalence theory of $\mathbb{T}_1 + \mathbb{T}_2$ has as relation the minimal equivalence relation that contains both relations corresponding to $\mathbb{T}_1$ and $\mathbb{T}_2$.*

Definition 4. *A λ-theory (over Δ) is an equational theory consisting of the following axioms:*

$$M[x := N] = (\lambda x.M)N \tag{1.1}$$
$$\lambda x.M = \lambda y.M[x := y] \quad \text{if } y \notin FV(M) \tag{1.2}$$
$$M = N \implies LM = LN \tag{1.3}$$
$$M = N \implies ML = LN \tag{1.4}$$
$$M = N \implies \lambda x.M = \lambda x.N \tag{1.5}$$

with $M, N, L \in \Delta, x, y \in Var$ and where the square brackets, in $M[x := N]$, means the substitution of x by N in M which is defined as follows:

- $x[x := N] = x$
- $y[x := N] = y$ *if* $y \neq x$
- $(\lambda y.M_1)[x := N] = \lambda y.(M_1[x := N])$ *if* $y \notin FV(M)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$

The first axiom is called the α-conversion. It states that when you apply a term N to a term M with a bound variable x (as outer variable), it actually means that we replace x by N in M. So application can mean composition or applying an input to a function. For example:

- Let $f = \lambda x.(x^2)$ and $y = 4$, then $fy = (\lambda x.x^2)4 = 4^2 = 16 = f(4)$.
- Let $f = \lambda x.(x{+}5)$ and $g = \lambda y.(y{+}3)$, then $fg = (\lambda x.(x{+}5))(\lambda y.(y{+}3)) = (\lambda y.y + 3) + 5 = \lambda y.(y + 8) = f \circ g$

Notice that in the lambda calculus as we defined it, there are no notions of squared and addition of numbers. This can be defined properly but we won't do this, but this is the way how you look at application.
The functions $f(x) = x^3 + 2$ and $f(y) = y^3 + 2$ are actually the same, but they have a different syntax. The second axiom, called β-conversion, ensures that they are the same.
The third condition says when functions are equal and the same argument is

applied to both of them, their output will produce the same output. It is like $f(x)$ and $f(y)$ in the previous example. The fourth condition states the same but now functions and arguments are swapped. So when arguments are equal and we apply them to the same function, both of them also give the same output. The last condition states that when given equal terms, when we turn them into functions, their functions are also equal. An example of this is the following:

Example 3. *Let $M = (x+y)^2$ and $N = x^2 + y^2 + 2xy$. Both of them are equal and if we define functions $f = \lambda x.((x+y)^2)$ and $g = \lambda x.x^2 + y^2 + 2xy$, then by the last axiom those are also equal and if we apply it another time we have that $\lambda xy.(x+y)^2 = \lambda xy.(x^2 + y^2 + 2xy)$.*

The following examples of application and substitution are given in terms of only the λ-terms instead of more concrete functions $f : \mathbb{R}^n \to \mathbb{R}^n$ like given in previous examples:

Example 4.
- $((\lambda xy.xyy)[x := N])[y := M] = (\lambda y.(Nxx))[y := M] = NMM$
- $(\lambda x.xx)[x := \lambda y.y] = (\lambda y.y)(\lambda y.y)$

Notice that the third condition in the substitution needs that y is bounded. If we don't, this will allow the following result:

Example 5. *We will show that every 2 terms are equal:*

Proof. Define $F \equiv \lambda xy.yx$, then $FMN = ((\lambda x(\lambda y.yx))M)N = (\lambda y.yM)N = NM$. Thus in particular for $M = y, N = x$ we have $Fxy = yx$, but $Fyx = ((\lambda x(\lambda y.yx))y)x = (\lambda y.yy)x = xx$. And so we have $xx = yx$, from this equality we can derive any equation. If $xx = yx$, then the application with another term should be equal, if we apply $\lambda pq.p$ to both sides we have:

$$x = (\lambda pq.p)xx = (\lambda pq.p)yx = y$$

This is then true for all $x, y \in Var$ which clearly can't. □

Remark 2. *Notice that we defined "a λ-theory". The term "the λ-calculus" or "the theory λ" is used for the minimal λ-theory (over a set of terms).*

We now have defined the equality of terms. These rules can also be used to give a proof whether terms are the same. The equality actually means semantic equality (the mean the same), but there is also syntactic equality (they look complete the same). This we denote by $M \equiv N$. An example is $\lambda x.(\lambda y.(xy)) \equiv \lambda xy.(xy)$. We introduce this to define certain terms, so it's purely notational.
The next definition of convertibility is a common notation found in books, we therefore introduce it, but we will only use the notation when terms are convertible.

Definition 5.
- *$M, N \in \Delta$ are convertible if the axioms of the λ-theory can derive their equality. If so, we write $\lambda \vdash M = N$ as is done in logic.*
- *If a λ-theory $\mathbb{T}$ derives the equality $M = N$ we denote it by $\mathbb{T} \vdash M = N$.*

As the application isn't associative, it is also not commutative:

Remark 3. *The application need not be commutative:*

Proof. Take $M \equiv \lambda x.xy$ and $N \equiv \lambda z.(\lambda u.uz)$, then we have:

$$\begin{aligned} MN &= (\lambda z.(\lambda u.uz))y = \lambda u.uy \\ NM &= \lambda u.(u(\lambda x.xy)) \end{aligned}$$

These aren't equal, apply $P \in \Delta$ to both sides:

$$\begin{aligned} MNP = &\quad (\lambda u.uy)P = &\quad Py \\ NMP = &\quad (\lambda u.(u(\lambda x.xy)))P = &\quad P.(\lambda x.xy) \end{aligned}$$

□

The next proposition says that substitutions don't always commute.

Theorem 1. *("Substitution lemma") Take $M, N, L \in \Delta$, $x \neq y$ such that $x \notin FV(L)$, then:*

$$M[x := N][y := L] = M[y := L][x = N[y := L]]$$

Proof. We do this by induction on the structure of M:

- **M is a variable**: If $M \equiv x$, then both sides equal $N[y := L]$, if $M \equiv y$n then both equal L. If $M \equiv z$ with $z \notin \{x, y\}$, then both sides equal z
- **M is an abstraction**: Let $M = \lambda z.M_1$, then we have:

  $M[x := N][y := L] \equiv (\lambda z.M_1)[x := N][y := L] \equiv \lambda z.(M_1[x := N][y := L])$

  Now apply the induction hypothesis on $(M_1[x := N][y := L])$, thus we have:

  $$\begin{aligned} M[x := N][y := L] &\equiv \lambda z.(M_1[y := L][x = N[y := L]]) \\ &\equiv (\lambda z.M_1)[y := L][x = N[y := L]]) = M[y := L][x = N[y := L]] \end{aligned}$$

- **M is an application**: Let $M = (M_1 M_2)$, then we have:

$$(M_1M_2)[x := N][y := L] \equiv (M_1[x := N][y := L])(M_2[x := N][y := L])$$

By the induction hypothesis we have:

$$\begin{aligned} (M_1M_2)[x := N][y := L] &\equiv (M_1[y := L][x = N[y := L]])(M_2[y := L][x = N[y := L]]) \\ &\equiv (M_1M_2)[y := L][x = N[y := L]] \end{aligned}$$

□

Thus it isn't always commutative because it can happen that more terms have the same bounded variables and thus we must first substitute them, but on the other hand the equality of terms is preserved under substitution:

Theorem 2.
- $M = M' \implies M[x := N] = M'[x := N]$
- $N = N' \implies M[x := N] = M[x := N']$
- $M = M', N = N' \implies M[x := N] = M'[x := N']$

Proof.
- $M = M' \implies \lambda x.M = \lambda x.M' \implies (\lambda x.M)N = (\lambda x.M')N \implies M[x := N] = M'[x := N]$
- $N = N' \implies (\lambda x.M)N = (\lambda x.M)N' \implies M[x := N] = M[x := N']$
- Since $M = M' \implies \lambda x.M = \lambda x.M'$ and $N = N'$ we have $(\lambda x.M)N = (\lambda x.M')N'$, this is than $M[x := N] = M'[x := N']$

□

Notice that the second theorem doesn't imply:

$$N = N' \implies \lambda x.x(\lambda y.N) = \lambda x.x(\lambda y.N')$$

The reason for this is that we can't define the term

$$M \equiv \lambda x.x(\lambda y.)$$

since this isn't valid syntax, but a way to solve this is to introduce a context. Informally this is a term with a gap we can't fill.

Definition 6. *A context $C[]$ is defined as:*

- *x is a context*

  is a context
- *if $C_1[]$ and $C_2[]$ are contexts, then $\lambda x.C_1[]$ and $C_1[]C_2[]$ are contexts.*

Definition 7. *Let $C[]$ be a context and $M \in \Delta$ a term. Then $C[M]$ is the result of placing M in the gaps. We write $C[] \in \Delta$ to denote it is a context over Δ.*

Example 6. *Define $C[] \equiv \lambda x.(\lambda y.[])$ and $M = xy$. Then we have $C[M] = \lambda x.(\lambda y.M) = \lambda x.(\lambda y.xy)$*

Notice that in the example the variables suddenly became bounded while before they were free.

Theorem 3. *Let $C[]$ be a context. Then $N = N' \implies C[N] = C[N']$*

Proof. We do this by induction on the structure of the context:

- If $C[] = x$, then $C[N] = x = C[N']$

- If $C[] = []$, then $C[N] = N = N' = C[N']$
- If $C[] = \lambda x.C_1[]$, then $C[N] = \lambda x.C_1[N] \overset{\text{I.H.}}{=} \lambda x.C_1[N'] = C[N']$
- If $C[] = C_1[]C_2[]$, then $C[N] = C_1[N]C_2[N] \overset{\text{I.H.}}{=} C_1[N']C_2[N'] = C[N]$

□

Notice that the converse of this theorem is trivially not true because we can take $C[]$ a variable that isn't used in N or N'.

1.1.3 Equality of theories

Since λ-theories are (mathematical) structures we can define an isomorphism between them. Let $\mathbb{T}_1, \mathbb{T}_2$ be theories with term-sets Δ_1, Δ_2.

Definition 8. *$\mathbb{T}_1, \mathbb{T}_2$ are equivalent if there are mappings*

$$()_1 : \Delta_1 \to \Delta_2 : M \mapsto (M)_1 \equiv M_1$$
$$()_2 : \Delta_2 \to \Delta_1 : M \mapsto (M)_2 \equiv M_2$$

such that: $\mathbb{T}_i \vdash M = N \iff \mathbb{T}_j \vdash M_j = N_j$ where $M_{i,j} \equiv (M_i)_j$ and $M, N \in \Delta_i$ with $\{i, j\} = \{1, 2\}$.

1.1.4 Combinators

We now introduce some special λ-terms (closed terms) which will be used later to discuss models, but they also have the property that every closed term can be generated by the application of (some of) these combinators.

Definition 9.
- $\boldsymbol{I} = \lambda x.x$
- $\boldsymbol{K} = \lambda xy.x$
- $\boldsymbol{S} = \lambda xyz.xz(yz)$
- $\Omega = (\lambda x.xx)(\lambda x.xx)$
- $\boldsymbol{1} = \lambda xy.xy$

Notice that $I = SKK$ because $SKKN = KN(KN) = N$ for any term $N \in \Delta$.

Definition 10. *The set of terms generated by $\delta \subset \Delta$, denoted as δ^+, is the least set $\delta \subset \delta^+$ such that it is closed under application, i.e. $\forall M, N \in \delta^+ : MN \in \delta^+$.*

Definition 11. *A collection of terms $\delta \subset \Delta$ is a basis for $\mu \subset \Delta$ if every term in μ has an equivalent term in δ^+, i.e. $\forall M \in \mu : \exists N \in \delta^+ : N = M$.*

Definition 12. *$\delta \in \Delta$ is a basis if δ is a basis for the closed terms Δ^0.*

Here is an interesting result of the combinators K and S:

Theorem 4. *$\{K, S\}$ is a basis*

Proof. We only give a sketch of the proof by giving an algorithm. Since we only look at closed terms, we only need to consider terms of the form $\lambda x.M$. Notice that the product of closed terms is again a closed term, but if we apply β-reduction we also have something of the form $\lambda x.M$ so it is the only case that needs to be checked.

- for $x \notin FV(M)$ we have: $\lambda x.M = KM$
- $\lambda x.MN = \lambda x.(((\lambda x.M)x)((\lambda x.N)x)) = S(\lambda x.M)(\lambda x.N)$

□

Notice that the proof is a constructive proof, i.e. it gives a way to compute the "factorization" of a term in $\{S, K\}$:

Example 7. *Take $M = \lambda xy.xyx$.*

Proof. We first work out $\lambda y.xyx$:

$$\begin{aligned} \lambda y.xyx &= S(\lambda y.xy)(\lambda y.x) \\ &= S(S(\lambda y.x)(\lambda y.y))(Kx) \\ &= S(S(Kx)I)(Kx) \end{aligned}$$

We can't work this further out because a.t.m. x isn't bound, so we now look at M:

$$\begin{aligned} \lambda xy.xyx &= \lambda x.(S(S(Kx)I)(Kx)) \\ &= S(\lambda x.S(S(Kx)I))(\lambda x.Kx) \\ &= S(\lambda x.S(S(Kx)I))(S(\lambda x.K)(\lambda x.x)) \\ &= S(\lambda x.S(S(Kx)I))(S(KK)I) \end{aligned}$$

We now simplify $\lambda x.S(S(Kx)I)$:

$$\begin{aligned} \lambda x.S(S(Kx)I) &= S(\lambda x.S)(\lambda x.S(Kx)I) \\ &= S(KS)(S(\lambda x.S(Kx))(\lambda x.I)) \\ &= S(KS)(S(\lambda x.S(Kx))(KI)) \\ &= S(KS)(S(S(\lambda x.S)(\lambda x.Kx))(KI)) \\ &= S(KS)(S(S(KS)(S(KK)I))(KI)) \end{aligned}$$

And thus our final result is:

$$S(S(KS)(S(S(KS)(S(KK)I))(KI)))(S(KK)I)$$

□

1.1.5 Extensionality

In algebra (group theory) the cancelling law, by the use of inverses, is very much used. We now define an equivalent notion for λ-theories, this is called extensionality. The reason this is axiom will be introduced is because of the fact that a function can be defined, instead of a formula, by fixing all the function values.

Definition 13.

- *A λ-theory is called extensional if*

$$(\forall x \notin FV(MN) : Mx = Nx) \implies M = N.$$

- *The theory $\lambda + ext$ is the λ-theory extended with this rule.*
- *A λ-theory has included the η-rule if*

$$\forall x \notin FV(MN) : \lambda x.Mx = M.$$

- *The theory λ_η is the λ-theory extended with this rule.*

Although η and ext are different, they can both prove the other one:

Theorem 5. *("Curry") The theories $\lambda + ext$ and λ_η are equivalent*

Proof. If $x \notin FV(M)$, then by extensionality we have $\lambda x.Mx = \lambda M$. Therefore $\lambda + ext \vdash \eta$: $(\lambda x.Mx)x = Mx$
For $x \notin FV(MN)$ we have $Mx = Nx \implies \lambda x.Mx = \lambda x.Nx$ then by η we have $M = N$. Therefore $\lambda_\eta \vdash ext$. □

1.1.6 Consistency

A trivial example of a lambda theory is a theory such that all closed terms are equivalent. This is of course not interesting and not usefull. A theory which doesn't satisfy this property is called consistent:

Definition 14. *A theory $\mathbb{T}$ (a set of equations) is called consistent if it doesn't prove every closed equation. If the theory is consistent then we denote it by $Con(\mathbb{T})$. If $\mathbb{T}$ is a set of axioms and we extend λ by it, write $Con(\mathbb{T})$ for $Con(\lambda + \mathbb{T})$.*

When discussing normal-forms, we show that λ_η is consistent and notice that by adding another axiom, we introduce more equalities. Thus if λ_η is consistent, then λ is certainly consistent since an extension of it is.

Sometimes a theory $\mathbb{T}$ is consistent, but if one equality $M = N$ is added, it can happen that $\mathbb{T} + M = N$ becomes inconsistent. We then say that M and N are incompatible and we write $M\#N$. An example of this case:

Example 8. *Assume $\lambda + K = S$. Thus for $X, Y, Z \in \Delta$ we have:*

$$KXYZ = SXYZ \implies XZ = XZ(YZ)$$

Take $X \equiv Z \equiv I$, we then have $I = YI = Y$. Therefore

$$\lambda + K = S \vdash M = I = N$$

for all terms M, N. Thus $\lambda + K = S$ isn't consistent.

1.1.7 Lattice of theories

The set of lambda theories λT (over a set of terms Δ) can be natural ordered such that it forms a complete lattice. This lattice is very large, so there are many λ-theories, but we won't prove this. In [11] it is shown that a specific class of λ-theories already has cardinality $2^{\mathbb{N}}$.

Theorem 6. *The operation $\wedge$ defined by $\mathbb{T}_1 \wedge \mathbb{T}_2 := T_1 \cap T_2$ is a infimum.*

Theorem 7. *The operation $\vee$ defined by $\mathbb{T}_1 \vee \mathbb{T}_2 := T_1 + T_2$ is a supremum.*

This showed that the set indeed forms a lattice and it is indeed a complete one because the union or intersection of (arbitrary number of) subsets is well-defined.

The minimal lambda theory, the bottom element in λT, is usually denoted by λ_β (in this thesis, as mentioned earlier it is just denoted λ) and the top element is the (inconsistent) theory that equates all terms. Although the top element isn't interesting, the bottom is. A longstanding open problem is whether or not their exists an extensional model of the λ-theory λ_β. In the paper [1] some properties about this lattice are found.

1.1.8 Subterms

We now introduce subterms to define a normal-form of a term: A term N is a subterm of M if N appears in M.

Definition 15. *The set of subterms of M, denoted by $Sub(M)$, is defined as follows:*

- $Sub(x) = \{x\}$
- $Sub(\lambda x.N_1) = Sub(N_1) \cap \{\lambda x.N_1\}$
- $Sub(N_1N_2) = Sub(N1) \cap Sub(N2) \cap \{N_1N_2\}$

Example 9.

- *$\lambda y.y$ is a subterm of $\lambda x.(M.(\lambda y.y))$*
- *$\lambda y.y$ isn't a subterm of $\lambda xy.x$*

1.1.9 Normal-form

The normal-form of a term is an equivalent term in which no further computing can be performed, so it is like a minimal term.

Definition 16. *We say that a term is in β-normal form (β-nf or nf) if it has no subterm $(\lambda x.R)S$. And a term M has a β-nf if there exists a term N in nf such that $N = M$.*

Recall the β-reduction: $M[x := N] = (\lambda x.M)N$. So if a term is in nf, it means that all possible substitutions are applied.

When working in λ_η we also can define:

Definition 17. *We say that a term is in $\beta\eta$-normal form ($\beta\eta$-nf) if it has no subterm $(\lambda x.R)S$ or $(\lambda x.Rx)$ with $x \notin FV(R)$.*

Like nf it is based on the η-reduction: $\lambda x.Mx = M$ for $x \notin FV(MN)$.

Example 10. *Examples of β-nf*

- *KI has a nf, because $KI = (\lambda xy.x)(\lambda x.x) = \lambda y.(\lambda x.x) = \lambda y.I$ and $\lambda y.I$ is in nf. KI itself isn't in nf, set $R = \lambda y.x$ and $S = \lambda x.x$.*
- *$\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ doesn't have a nf because when applying the β-reduction we always get back Ω so we can repeat the β-reduction infinitly many times and it will never be in a normal form.*

Example 11. *Examples of $\beta\eta$-nf*

- *K and S are $\beta\eta$-nf*
- *$\lambda x.x(\lambda z.xz)$ is not in $\beta\eta$-nf but has as $\beta\eta$-nf: $\lambda x.xx$*

The following proposition links both normal forms:

Theorem 8. *("Curry") M has a $\beta\eta$-nf if and only if M has a β-nf*

Theorem 9. *If M, N are different β-nf's (resp. $\beta\eta$-nf's) then $\lambda \not\vdash M = N$ (resp. $\lambda_\eta \not\vdash M = N$)*

Now we can easily proof that λ and $\lambda\eta$ are consistent.

Proof. We have $\lambda_\eta \not\vdash K = S$ because their $\beta\eta$-nf's are different and so $\lambda\eta$ is consistent. And since every $\beta\eta$-nf is a β-nf, we repeat the argument for λ_η. □

Another normal form is the head normal form (=hnf):

Definition 18. *A term M is a hnf if $M \equiv \lambda x_1 ... x_n \cdot xM_1 ... M_m$ and a term M' has a hnf if there is a hnf M such that $M = M'$.*

1.1.10 Solvability

A solvable term is a term that has an inverse for the application:

Definition 19.
- *A closed term $M \in \Delta^0$ is solvable if there are terms $N_1, ..., N_n \in \Delta$ such that $MN_1 \cdots N_n = I$.*
- *A term $M \in \Delta$, with $FV(M) = \{x\}$ is called solvable if $\lambda x.M$ is solvable.*
- *A term that isn't solvable is called unsolvable.*

Notice that in the definition solvable is defined such that there can be multiple terms N_i. If we would say that there is exists only 1 term N such that $MN = I$ then N wouldn't have a nf if for example $N \equiv \lambda xyz.xyz$ because when only applied 1 term only the x would be substituted but not the other 2 variables and a function of multiple arguments can't equal the identity function of 1 argument but if we allow $n > 1$ then we have:

$$\begin{aligned}((\lambda xyz.xyz)(\lambda x.x))\,(\lambda y.y) &= (\lambda yz.((\lambda x.x)yz))(\lambda y.y)\\ &= (\lambda yz.(yz)(\lambda y.y)\\ &= (\lambda z.((\lambda y.y)z)\\ &= \lambda z.z\end{aligned}$$

Example 12.
- *S is solvable since $SIII = IIII = I$*
- *Ω is unsolvable since if we apply $N_1, ..., N_n$ to Ω we can't simply it further. The reason is because of non-associativity and that Ω it is not of the form $\lambda x.N$, thus we can't use β-reduction.*

The following theorem gives a nice characterization of solvability:

Theorem 10. *("Wadsworth") M is solvable iff M has a hnf*

The following notion of equivalence is used later in the text:

Definition 20. *Let M, N be hnf's, i.e.:*

$$M \equiv \lambda x_1 ... x_n \cdot y M_1 ... M_m$$
$$N \equiv \lambda x_1 ... x_n^{'} \cdot y^{'} M_1 ... M_m^{'}$$

then M and N are equivalent, denoted by $M \sim N$, if

$$y \equiv y^{'}, n - m = n^{'} - m^{'}$$

Arbitrary terms M, N are equivalent if both are unsolvable or both are solvable and their hnf's are equivalent.

1.1.11 Hilbert-Post complete

We now define the notion of a Hilbert-Post (HP) complete theory, that is a maximally consistent theory:

Definition 21. *An equational theory $\mathbb{T}$ is called Hilbert-Post (HP) complete if for every term we have: $\mathbb{T} \vdash M = N \iff Con(\mathbb{T} + M = N)$.*

So an equational theory is maximally consistent if every equality of terms is added when it doesn't make the theory inconsistent.

Example 13. *λ is clearly not HP, since otherwise we would have that all λ-theories (over the same term set) are equivalent. This isn't the case because the minimal λ-theory λ doesn't satisfies the extensionality axiom by definition.*

Lemma 1. *The union $\mathbb{T}$ of a chain $\mathbb{T}_1 \subset \mathbb{T}_2 \subset ...$ of consistent theories, is again consistent*

Proof. In logic there is also a notion of consistency and this is a standard proof in it. You assume it isn't consistent, thus we can derive a contradiction $M = N$. This derivation only consists of a finite steps/formulas. Thus the set of those formulas is finite. Thus there is a $\mathbb{T}_i$ such that those formulas are included in it. But since $\mathbb{T}_i$ is consistent, this is a contradiction. □

Theorem 11. *Every λ-theory $\mathbb{T}$ can be extended to a HP-complete λ-theory*

Proof. Set $\mathbb{P} = \{\mathbb{T}_0 \supseteq \mathbb{T} | con(\mathbb{T}_0)\}$. This set isn't empty because $\mathbb{T} \in \mathbb{P}$. Take a chain $T_0 \subseteq T_1 \subseteq ...$ in $\mathbb{P}$. As **LT** is a complete lattice there exists an upperbound $K = \bigcup_{i \in \mathbb{N}} K_i$. Clearly $\mathbb{T} \subseteq K$ and thus we only have to check that K is consistent, but this follows from previous lemma. Since $K \in \mathbb{P}$ we can thus apply Zorn's lemma and therefore $\mathbb{P}$ has a maximal element $\mathbb{T}_1$. We claim this is HP-complete because if this isn't the case, then there exists an equality $M = N$ such that $(\mathbb{T}_1 + [M = N])^+$ is consistent, but this contradicts the maximality of $\mathbb{T}_1 \in \mathbb{P}$. □

1.1.12 Labelled terms

A variant on the terms is called the labelled λ-terms, this we don't need but there is a subset that we will use: $\Delta\ \bot$. The only difference is that to the alphabet the constant $\bot$ and is also a term on itself. The reduction rules added to the 'standard' rules are:

- $\bot\ M = \bot$
- $\lambda x.\ \bot = \bot$
- $\bot\ [x := N] = \bot$

We introduce this because once we introduce the Scott-model, some nice properties follow and this also gives the possibility to create 'empty' nodes on the Bohm tree

1.2 The theory K

In this section we describe the theory K and its unique HP-complete extension K^*. The reason why we study this is because it turns out to be the theory corresponding to Scott's model.

The theory K consists of the minimal lambda theory where alle unsolvable (closed terms) are equal, these terms represents the notion of undefined.

Definition 22.
- $K_0 = \{M = N : M, N \in \Delta^0 \ \textit{unsolvable}\}$
- $K = K_0^+$

If a theory contains K, we say it is sensible.

Definition 23. *We define K^* to be:*

$$K^* = \{M = N | M, N \in \Delta^0, M \sim_s N\}$$

where $M \sim_s N$ is true if for every context C we have

$$C[M] \textit{ solvable} \iff C[N] \textit{ solvable.}$$

The following lemma proves that K^* is extensional:

Lemma 2. $K_\eta^* \vdash M = N \implies M \sim_s N$

Proof. We prove this by induction on the length of the proof of $M = N$, with this technique we mean the following:
Assume $K_\eta^* \vdash M = N$, then there exist a derivation using the axioms of λ, or they are equal if $C[M]$ is solvable which is equivalent to saying that $C[N]$ is solvable. Then induction is done on the number of steps (derivation rules) needed to prove this. For example, $\lambda \vdash yz = ((\lambda x.x)y)z$, this is proven by first applying the β-reduction to get $y = (\lambda x.x)y$ and continuing with the rule $M = N \implies ML = NL$. So this proof consists of 2 steps. The base case consists of first showing that the statement holds for terms that are equal by definition, because they need zero steps to show it. Then, in the induction step we look at the last axiom/step needed to show it. Afterwards we apply the induction hypothesis on the proof needed to show the previous derivation.

- If $M = N$ is an axiom in λ_η, then once $C[M]$ is solvable, there exists $x_1, ..., x_n \in Var$ and $P_1, ..., P_m \in \Delta$ such that

  $$(\lambda x_1 ... x_n \cdot C[M])P_1 ... P_m = \mathbf{I},$$

  thus we also have $(\lambda x_1 ... x_n \cdot C[N])P_1 ... P_m =_\eta \mathbf{I}$. Therefore $C[N]$ is $\beta\eta$-solvable. But solvability and $\beta\eta$-solvability are equivalent, thus $C[N]$ is solvable.

- If $M = N$ is an axiom in K^* then we are immediatly done by definition of K^*.

- Assume the last step uses the axiom $M = N \implies \lambda x.M = \lambda x.N$. Then by induction we have $M \sim_s N$, thus in particular for contexts of the form $C[\lambda x.[]]$.
- Assume the last step uses the axiom $M = N \implies LM = LN$. Then by induction we have again $M \sim_s N$. Thus also for contexts $C[([]L)]$ and $C[(L[])]$.
- If transitivity was last used, so $(M = L, L = N) \implies M = N$ then by induction $M \sim_s L$ and $L \sim_s N$ and thus $C[M]$ solvable $\iff$ $C[L]$ solvable $\iff$ $C[N]$ solvable.

□

Corollary 1. *K^* is extensional*

Proof. As K^* is a λ-theory, we have that $\lambda + ext = \lambda_\eta$. But by the previous lemma we have $K^*_\eta \subseteq K^*$. And the converse is always true, so $K^*_\eta = K^*$. □

Theorem 12. *K^* is consistent*

Proof. Since $\mathbf{I}$ is solvable and Ω is unsolvable, we are done by the previous lemma. □

The following shows that K^* is the maximal consistent theory consisting of closed terms that contains K:

Theorem 13. $Con(K + M = N) \implies M \sim_S N$

Proof. Assume $M \not\sim_S N$, then there exists a context $C[]$ such that:

$$(\lambda x_1 ... x_n \cdot C[M])(P_1 ... P_n) = \mathbf{I}$$
$$(\lambda x_1 ... x_n \cdot C[N])(P_1 ... P_n) \neq \mathbf{I}$$

But if we assume that $M = N$ then $C[N] = C[M]$ and thus

$$(\lambda x_1 ... x_n \cdot C[M])(P_1 ... P_n) = (\lambda x_1 ... x_n \cdot C[N])(P_1 ... P_n),$$

thus $\mathbf{I}$ is equal to an unsolvable term and this can only happen when the theory $K + M = N$ is inconsistent. □

Corollary 2. $K \subset K^*$

Proof. This is a direct consequence of the previous theorem, because if we take $M = N$ in K, then $Con(K + M = N) = Con(K)$ and K is consistent because $\mathbf{K}$ and $\mathbf{S}$ are both solvable and λ doesn't derives it, so $K \nvdash \mathbf{K} = \mathbf{S}$. □

Corollary 3. *A consistent theory $\mathbb{T}$, consisting of closed terms, that extends K, has K^* as an extension*

Proof. This is exactly what the previous theorem says. For every $M = N$, that $\mathbb{T}$ derives, we have $Con(K + M = N)$ because $K + M = N \subset \mathbb{T}$ and $\mathbb{T}$ is consistent. Thus for every equation $M = N$ we have that $M \sim_s N$ by the previous theorem, thus it is by definition in K^*. □

Corollary 4. *K^* is the unique HP-complete λ-theory extending K.*

Proof. K^* is consistent and by the previous corrolary every other consistent theory containing K is a part of K^*. □

Chapter 2

Bohm trees

In this chapter we introduce trees and in particular bohm trees, these represent in a sense a parse tree of a term and will be usefull in studying the theory of Scott's model. We don't go too deep in this as this isn't the main part of this thesis but we need it. More information is found in [3].

2.1 Trees

A tree can be formally defined as a set of finite sequences together with a relation between these sequences.

Definition 24. *The set of sequences, denoted as Seq, is defined as:*

$$Seq := \{< n_1, ..., n_k > | k, n_1, ..., n_k \in \mathbb{N}\}$$

The empy sequence is denoted as <> and the length of a sequence $\alpha =< n_1, ..., n_k > \in Seq$ *is denoted by* $lh(\alpha) = k$. *The concatenation of sequences* $\alpha =< n_1, ..., n_k >, \beta =< m_1, ..., m_l >$ *is* $\alpha * \beta =< n_1, ..., n_k, m_1, ..., m_l >$.
The ordering on sequences is defined as: $\alpha \leq \beta$ *iff* $k \leq n$ *and* $n_i = m_i$ *for* $0 < i \leq k$ *with* α, β *defined as above.*

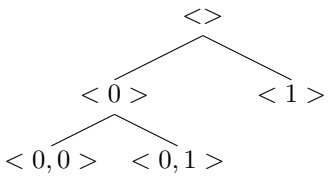
Definition 25. *A tree is a set* $A \subset Seq$ *such that:*

- $\alpha \in A, \beta \leq \alpha \implies \beta \in A$
- $\alpha * < n + 1 > \in A \implies \alpha * < n > \in A$

If a sequence α *is in* A *we call* α *a node of the tree.*

The idea behind this definition is that every sequence represents a node of the tree and if the length of a sequence is k, the corresponding node is at depth k. Although a tree doesn't consists of only nodes, but also with connections so that we know which nodes are children of a certain node, called the parent. This relation is enforced since there is a partial order on Seq and thus on A. If $\alpha \leq \beta$ and $lh(\alpha) < lh(\beta)$, we have that β is a child of α.
An example of a tree is given below:

Example 14. *If A consists of $<>, < 0 >, < 1 >, < 0, 0 >, < 0, 1 >$, then the tree is visualised as:*



As a tree without any labels on the nodes isn't very interesting we introduce:

Definition 26. *Let Σ be a set of symbols. A Σ-labelled tree is a partial map $\phi : Seq \to \Sigma$ such that:*

$$T_\phi = \{\alpha \in Seq : \phi(\alpha) \text{ is defined}\}$$

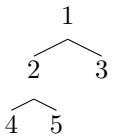
is a tree. We call T_ϕ the underlying (naked) tree. $\phi(\alpha)$ is called the label at node α.

Note that ϕ is partially defined since we only want labels on nodes that are included in the tree. To visualise this labelled tree we represent the nodes not by the sequence, but by the image of the sequence.

Example 15. *Let $\Sigma = \{1, ..., 5\}$ and let the tree A consists of sequences $<>, < 1 >, < 2 >, < 1, 1 >, < 1, 2 >$. Define ϕ as follows:*

- $\phi(<>) = 1$,
- $\phi(< 0 >) = 2$,
- $\phi(< 1 >) = 3$,
- $\phi(< 0, 0 >) = 4$,
- $\phi(< 0, 1 >) = 5$.

Then the Σ-labelled tree looks like:



For the remaining part of this chapter, fix $\mathbb{T}$ a λ-theory. We now define the Bohm tree:

Definition 27. *Take*

$$\Sigma = \{\bot\} \cup \{\lambda x_1 ... x_n \cdot y | x_1, ..., x_n, y \text{ variables}\}$$

The Bohm tree of a λ-term M is the Σ-labelled tree such that:

- *If M is unsolvable:*

$$BT(M)(<>) = \bot$$
$$BT(M)(< k > *\alpha) \text{ is undefined.}$$

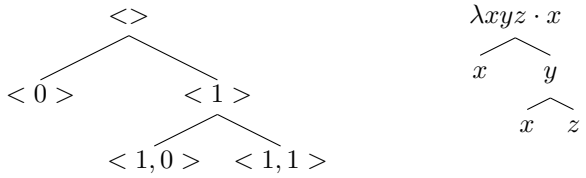
- *If M is solvable with principal hnf $\lambda x_1...x_n \cdot yM_0...M_{m-1}$:*

$$BT(M)(<>) = \lambda x_1...x_n \cdot y$$

$$BT(M)(< k > *\alpha) = \begin{cases} BT(M_k)(\alpha) & \text{if } k < m \\ \text{undefined} & \text{if } k \geq m \end{cases}$$

So for every unsolvable term, its Bohm tree is the tree with only 1 node with label $\bot$.

Example 16. *Let $M = \lambda xyz \cdot xx(yxz)$, then its corresponding tree and Bohm tree are:*



If terms are equal, we expect them to have the same Bohm tree:

Theorem 14. $M = N \implies BT(M) = BT(N)$

Proof. Suppose $M = N$ and let α be a sequence. We show by induction on $lh(\alpha)$ that $BT(M)(\alpha) = BT(N)(\alpha)$. We are then done since if the nodes with their labels match (also undefined nodes) then the functions are equal.
Suppose $lh(\alpha) = 0$. Thus $\alpha =<>$. If M is unsolvable, then N is to. Thus by definition both labels are $\bot$. If both are solvable then both their top node are labelled as $\lambda x_1...x_n \cdot y$ if we write M and N as:

$$\begin{aligned} M &= \lambda x_1...x_n \cdot yM_0...M_{m-1} \\ N &= \lambda x_1...x_n \cdot yN_0...N_{m-1} \end{aligned}$$

with $M_k = N_k$ for $k < m$. Now suppose $lh(\alpha) > 0$, thus we can write α as $< k > *\alpha'$. If M is unsolvable then both nodes are undefined. If both are solvable, take their hnf's as in the lemma. If $k \geq m$, then both nodes are undefined. If $k < m$, we have:

$$BT(M)(\alpha) = BT(M)(< k > *\alpha') = BT(M_k)(\alpha')$$
$$BT(N)(\alpha) = BT(N)(< k > *\alpha') = BT(N_k)(\alpha')$$

And since $lh(\alpha') < lh(\alpha)$ we are done by induction. □

Instead of looking at a tree with a Σ-labelled tree, we can also insert at each node the number of successors:

Definition 28. *Let Σ be a set of symbols. A Σ-labelled tree is a partial map $\phi : Seq \to \Sigma \times \mathbb{N}$ such that:*

- *If $\phi(\sigma)$ is defined and if $\tau \leq \sigma$, then $\phi(\tau)$ is defined.*
- *If $\phi(\sigma) =< a, n >$ then for every $k \geq n$ we have that $\phi(\sigma * < k >)$ is undefined.*

We call $T_\phi = \{<>\} \cup \{\sigma \in Seq | \sigma = \sigma' * < k >, \phi(\sigma') =< a, n >, k < n\}$ the underlying (naked) tree.

From now on when we talk about Bohm trees, we mean effective Bohm trees:

Definition 29. *The effective Bohm tree of a term M, denoted by $BT^e(M)$, is the partially Σ_1 labelled tree defined as follows:*

- *If M is unsolvable:*

$$BT^e(M)(\sigma) \text{ is undefined, } \forall \sigma.$$

- *If M is solvable with principal hnf $\lambda x_1 ... x_n \cdot y M_0 ... M_{m-1}$:*

$$BT^e(M)(<>) =< \lambda x_1 ... x_n \cdot y, m >$$

$$BT^e(M)(< k > *\sigma) = \begin{cases} BT^e(M_k)(\alpha) & \text{if } k < m \\ \text{undefined} & \text{if } k \geq m \end{cases}$$

Note that the only differences between $BT(M)$ and $BT^e(M)$, are:

- The label $\perp$ is replaced by an empty label
- To each node we add an extra number.

Thus $BT(M)$ and $BT^e(M)$ are isomorphic trees. From now on we only work with effective Bohm trees instead of Bohm trees. We therefore (always) write $BT(M) := BT^e(M)$ and use Bohm tree for effective Bohm tree.
Sometimes, the following notion is needed:

Definition 30.

- *A Bohm-like tree is a partially Σ-labelled tree as in the definition of the bohm tree.*
- *The set of Bohm-like trees is denoted by $\mathfrak{B}$*

Notice that every Bohm tree is a Bohm-like tree.
We introduce certain subtrees:

Definition 31. *Let $A \in \mathfrak{B}$. For $k \in \mathbb{N}$ we define:*

$$A^k(\alpha) = \begin{cases} A(\alpha) & \text{if } lh(\alpha) < k \\ \text{undefined} & \text{else} \end{cases}$$

To each (finite) Bohm-like tree we can associate a term M_A (or sometimes denoted $M(A)$) as follows:

Definition 32.
- *If* $A = \bot$, *take* $M_A \equiv \Omega$.
- *If* $A = \lambda x_1, ..., x_n \cdot y$, *take* $M_A \equiv A = \lambda x_1, ..., x_n \cdot y$.
- *If* $A = \lambda x_1, ..., x_n \cdot y$, *take* $M_A \equiv A = \lambda x_1, ..., x_n \cdot y M_{A_1} ... M_{A_k}$.

Corollary 5. *For every finite* $A \in \mathfrak{B}$ *we have* $BT(M(A)) = A$.

We use following notation:

Notation 2.
- $BT^k(M) \equiv (BT(M))^k$
- $M^{(k)} \equiv M(BT^k(M))$

Theorem 15. $BT(M^{(k)}) = BT^k(M)$

Proof.

$$BT(M^{(k)}) = BT(M(BT^k(M))) = BT^k(M)$$

□

2.2 Relations on Bohm-like trees

In this chapter we define some relations between Bohm-like trees. The definitions introduced here are characterizations and are defined different in [3], there they make use of infinite η-expansions.

Definition 33. *Let* $A, B \in \mathfrak{B}$.

- $A \subseteq_k B \iff$ *at depth k, the subtree of A is a subtree of B.*
- $A =_k B \iff$ *their subtrees are equal to atleast depth k.*
- $A^\eta \subseteq B \iff \forall k \in \mathbb{N} : \exists A' : [A' \to_\eta A, A' \subseteq_k B]$
- $A^\eta \subseteq^\eta B \iff \exists A', B' : [A' \to_\eta A, B' \to_\eta B, A' \subseteq_k B']$
- $A \leq_\eta B \iff \forall k \in \mathbb{N} : \exists A' : [A' \to_\eta A, A' =_k B]$

When dealing with terms we usually want to look at their Bohm-trees, so the following notation is introduced:

Definition 34. *Let* $M, N \in \Delta$

- $M \lesssim_\eta N \iff BT(M) \leq_\eta BT(N)$

Definition 35. $P \in \Delta \bot$ *is an approximation normal form (anf) for* $M \in \Delta \bot$ *if:*

- $BT(P) \subseteq BT(M)$
- P *is* $\beta \bot$*-nf*

The set of all anf's for $M \in \Delta \bot$ *is denoted by* $\mathfrak{C}(M)$

2.3 Bohm transformations

Bohm transformations are functions such that they correspond to a context and will be helpfull in proving a result about solvability as it gives a way of constructing unsolvable terms when applied to a Bohm-transformation. Proofs in this section are only given in great steps since some are long proofs and the results and definitions introduced here are only used to give 1 proof. The reader is refered to [3] to see the details.

Definition 36.

- *A solving transformation is a mapping $f : \Delta \to \Delta$ such that one of the following holds:*
  1. $\exists x : \forall M : f(M) = Px$
  2. $\exists x, N : \forall M : f(M) = P[x := N]$
- *A bohm-transformation is a finite composition of solving transformations*

Let π be a bohm-transformation and we write M^π for $\pi(M)$.

Theorem 16. *For every π there is a context $C_\pi[]$ such that for every M we have: $M^\pi = C_\pi[M]$*

Proof. By induction on the structure of π:

- $M^\pi = Mx$, set $C_\pi[] = []x$
- $M^\pi = M[x := N]$, set $C_\pi[] = (\lambda x.[])N$
- $M^\pi = \pi_1 \circ \pi_2$, set $C_\pi[] = C_{\pi_1}[C_{\pi_2}[]]$

□

Definition 37. *Let $\mathfrak{F} \subset \Delta$. Then π is called $\alpha - \mathfrak{F}$-faithfull if for every terms $M, N \in \mathfrak{F}$ we have:*

$$[M =_\alpha N \iff M^\pi \sim N^\pi] \,\&\, [BT(M)(\alpha) \text{ defined} \iff M^\pi \text{ solvable}]$$

Definition 38. *Let $\mathfrak{F} \subset \Delta$. Then $\mathfrak{F}$ agrees up to α if for every $M, N \in \mathfrak{F}$ we have that their bohm-trees have the same nodes under α, i.e.*

$$\forall \beta < \alpha : BT(M)(\beta) = BT(N)(\beta)$$

Theorem 17. *If $\mathfrak{F}$ agrees along α then there exists a pi that is $\alpha - \mathfrak{F}$-faithfull*

Proof. This is proved by induction on $lh(\alpha)$. For the base case and if M, N are unsolvable the identity is choosen, otherwise there is a π_0 constructed such that it the image of the terms also agrees up to α and they also have the same label at α. By these conditions (and one extra) it follows that we know how the terms under π_0 look like. Then a new π_1 is introduced and then the induction hypothesis (on $\pi_1 \circ \pi_0$) says that there is a π_2 and then $\pi = \pi_2 \circ \pi_1\pi_0$ proves the result. □

Theorem 18. *If M is solvable and $M \not\sim N$ then forall other terms $P \in \Delta$ there is a π such that $M^\pi = P$ and N^π is unsolvable*

Proof. By Wadsworth we can write $M = \lambda x_1 ... x_n \cdot y M_1 ... M_m$. Since $M \not\sim N$ we assume N is unsolvable. Then $\pi(Q) = Q x_1 ... x_n [y := \lambda a_1 ... a_m \cdot P]$ does the trick. □

Chapter 3

Models

In this chapter we study the notion of models, we introduce a couple of definitions of models and we discuss Scott's model. We first introduce lambda models and syntactical models, we show that to every λ-theory we can associate a λ-model, called the term model. And then we introduce an order-theoretical concept called a cpo and we look at the projective limits on cpo's, because Scott's model D_∞ will be a projective limit (and even a direct limit but we don't include this). And we finally show that K^*, the theory introduced chapter 2.2 of this thesis, is indeed the theory corresponding to D_∞.
An equational theory says whether certain terms are equal, but not what they mean. A model of a theory has the purpose of describing what it means. Therefore a model can also be seen as a representation theorem of a lambda-calculus.

3.1 Combinatory logic and algebra's

3.1.1 Combinatory logic

There is a variety of definitions of models introduced over the years, usually the first definition is by the use of combinatory algebra's. When other models are defined we prove that we can associate a "correct" combinatory algebra, but before defining the combinatory algebra, we look at combinatory logic (=CL). This is an equational theory that is equivalent with the λ-theory, when added a couple of extra axioms.

Definition 39. *The CL-terms, denoted by $\mathfrak{C}$, is defined as:*

- $x \in Var \implies x \in \mathfrak{C}$
- $\boldsymbol{K}, \boldsymbol{S} \in \mathfrak{C}$
- $M, N \in C \implies (MN) \in \mathfrak{C}$

The theory **CL** is then defined as:

Definition 40. *The theory **CL** consists of following axioms:*

$$\begin{aligned} \boldsymbol{K}MN &= M \\ \boldsymbol{S}MNL &= ML(NL) \\ M = N &\implies LM = LN \\ M = N &\implies ML = LN \end{aligned}$$

Again with "=" an equivalence relation and we write $CL \vdash M = N$ if the theory of CL derives their equality.

Remark 4. *As with the lambda calculus, if we don't write any brackets we assume left associativity.*

The definition of closed terms, free/bounded variables are the same as in the case of a λ-theory.

Lemma 3. *Let $\boldsymbol{I} \equiv \boldsymbol{SKK}$, then $\boldsymbol{I}P = P$ for all $P \in \mathfrak{C}$*

Proof. $\mathbf{I}P = \mathbf{SKK}P = \mathbf{K}P(\mathbf{K}P) = P$ □

As you may have noticed, the CL-terms only has application defined, but not abstraction. We don't need this since we can show that lambda abstraction can be simulated by using only **K** and **S**.

Definition 41. *Let $x \in Var$ and $P \in \mathfrak{C}$. Then we define $\lambda^* x.P$ as:*

- $\lambda^* x.x = \boldsymbol{I}$
- $\lambda^* x.P = \boldsymbol{K}P$ *if* $x \notin FV(P)$
- $\lambda^* x.PQ = \boldsymbol{S}(\lambda^* x.P)((\lambda^* x.Q))$

Notice that we could have expected this since **K** and **S** are a basis for the (closed) terms. We can now also relate the λ and CL-terms:

Definition 42. *We define $CL : \Delta \to \mathfrak{C}$ and $\lambda : \mathfrak{C} \to \Delta$ by:*

$$\begin{aligned} x_{CL} &= x & x_\lambda &= x \\ c_{CL} &= c & c_\lambda &= c \\ (MN)_{CL} &= M_{CL}N_{CL} & (MN)_\lambda &= M_\lambda N_\lambda \\ (\lambda x.M)_{Cl} &= \lambda^* x.M_{CL} & \boldsymbol{K}_\lambda &= \lambda xy.x \\ & & \boldsymbol{S}_\lambda &= \lambda xyz.xz(yz) \end{aligned}$$

with $M_{CL} = CL(M)$ and $M_\lambda = \lambda(M)$

We now define the set of axioms $\mathbf{A}_\beta$ such that λ and $\mathbf{CL}+\mathbf{A}_\beta$ are equivalent:

Definition 43. ***A**$_\beta$ consists of following axioms:*

- $\boldsymbol{K} = \lambda^* xy.x$

- $\boldsymbol{S} = \lambda^* xyz.xz(yz)$
- $\lambda^* xy.\boldsymbol{S}(\boldsymbol{K}x)(\boldsymbol{K}y) = \lambda^* xy.\boldsymbol{K}(xy)$
- $\lambda^* xy.\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{KK})x)y = \lambda^* xyz.xy$
- $\lambda^* xyz.\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{KS})x)y)z = \lambda^* xyz.\boldsymbol{S}(\boldsymbol{S}xz)(\boldsymbol{S}yz)$

Before proving the equivalence of λ en $\mathbf{CL} + \mathbf{A}_\beta$ we first need some intermediate result:

Lemma 4. $(\lambda^* x.N)_\lambda = \lambda x.N_\lambda$

Proof. We do this by induction on the structure of N. The base cases (variables and constant) are trivially true. So the only case left is when N is the application of 2 terms. Let $N \equiv N_1 N_2$ then $(\lambda^* x.N)_\lambda = (\lambda^* x.N_1 N_2)_\lambda = (S(\lambda^* x.N_1)(\lambda^* x.N_2))_\lambda$. The induction hypothesis then gives us:

$$\begin{aligned}(\lambda^* x.N)_\lambda &= S_\lambda(\lambda x.N_{1,\lambda})(\lambda x.N_{2,\lambda}) \\ &= \lambda z.\,(((\lambda x.N_{1,\lambda})z)((\lambda x.N_{2,\lambda})z))\end{aligned}$$

By definition of substitution we have:

$$\begin{aligned}\lambda z.\,(((\lambda x.N_{1,\lambda})z)((\lambda x.N_{2,\lambda})z)) &= \lambda z.\,(N_{1,\lambda}[x := z]N_{2,\lambda}[x := z]) \\ &= \lambda z.\,(N_{1,\lambda}N_{2,\lambda}[x := z]) \\ &= \lambda x.N_{1,\lambda}N_{2,\lambda} = \lambda x.(N_1 N_2)_\lambda\end{aligned}$$

□

Since the theory **CL** only has application and some special elements we can naturally extend it to an algebraic structure, the following theorem shows that we can therefore view a λ-theory as such algebraic structure:

Theorem 19. *The theories λ and $\boldsymbol{CL} + \boldsymbol{A}_\beta$ are equivalent*

Proof. The mappings that define the equivalence are choosen to be λ and CL.

- To show $\lambda \vdash M_{CL,\lambda} = M$: This is done by induction on the structure of M, the base cases are trivial since if we apply first λ and then CL we get the same as we started with and those are always equal. And for the application it follows immediatly from the induction hypothesis. For $M = \lambda x.N$ we have:

  $$M = \lambda x.N \to^{CL} \lambda^* x.N_{CL} \to^{\lambda} \lambda x.N_{CL,\lambda}$$

  Thus by the induction hypothesis we have: $\lambda \vdash N_{CL,\lambda} = N$. Thus by the last equation of the lambda axioms we are done.

- To show $CL + \mathbf{A}_\beta \vdash M_{\lambda,CL} = M$: This is done by induction on the structure of M, the base cases are trivial since if we apply first λ and then CL we get the same as we started with and those are always equal. And for the application it follows immediatly from the induction hypothesis. For the constants $\mathbf{K}, \mathbf{S}$ we have:

  $$K \to^{\lambda} \lambda xy.x = \lambda x(\lambda y.x) \to^{CL} \lambda^* x.(\lambda y.x)_{CL} = \lambda^* x.\lambda^* y.(x)_{CL} = \lambda^* x.\lambda^* y.x = \lambda^* xy.x$$

  And by the first equation in $\mathbf{A}_\beta$ those are equal. For $\mathbf{S}$ we do the same and then use the second equation in $\mathbf{A}_\beta$.

- To show: $\lambda \vdash M = N \implies CL + A_\beta \vdash M_{CL} = N_{CL}$. Since we have to check every axiom in λ, this is quite a lot of (non-interesting) computation. The reader is therefore refered to the BOOK.

- To show: $\lambda \vdash P_\lambda = Q_\lambda \implies CL + A_\beta \vdash P = Q$. This follows from previous point:

  $$\lambda \vdash P_\lambda = Q_\lambda \implies CL + A_\beta \vdash P_{\lambda,CL} = Q_{\lambda,CL}$$

  But by the second point we then have:

  $$CL + A_\beta \vdash P = P_{\lambda,CL} = Q_{\lambda,CL} = Q$$

- To show: $CL + A_\beta \vdash P = Q \implies \lambda \vdash P_\lambda = Q_\lambda$. We also refer you to the book.

- To show: $CL + A_\beta \vdash M_{CL} = N_{CL} \implies \lambda \vdash M = N$. This is the same as the fourth point with the use of the first and previous point.

□

We first defined λ-calculus and then **CL**, but sometimes it is done in the other order like in [5].

3.1.2 Combinatory algebra's

Definition 44. *An applicative structure is a set together with a binary operation,* $\mathfrak{A} = (X, \cdot)$.

Instead of writing $x \in X$ we usually write $x \in \mathfrak{A}$.

Definition 45. *A combinatory algebra is an applicative structure* $\mathfrak{A} = (X, \cdot)$ *such that there are elements* $k, s \in \mathfrak{A}$ *such that* $kxy = x$ *and* $sxyz = xz(yz)$.

Remark 5. *As with combinators (in the term-sense) we have an identity element i=skk.*

Although we call it a algebra, it doesn't have the nice properties of algebras:

Remark 6. *Let* $\mathfrak{A}$ *be a combinatory algebra that isn't trivial, then:*

- $\mathfrak{A}$ *isn't commutative*
- $\mathfrak{A}$ *isn't associative*
- $\mathfrak{A}$ *isn't finite*

Proof.
- If $\mathfrak{A}$ would be commutative, then $ki = ik = k$, thus $a = kab = kiab = ib = b$ for all $a, b \in A$
- If $\mathfrak{A}$ would be associative, then $(ki)i = k(ii)$, thus $i = ii = k(ii)a = ((ki)ia) = \lambda x.(\lambda y.x)i)ia = (\lambda y.i)ia = ia = a$ for all $a \in \mathfrak{A}$
- Take $k_0 = k$ and $k_{i+1} = kk_i$ and these are all distinct.

□

Since we want to define models (in terms of combinatory algebra's), we need to be able to associate a set of terms to $\mathfrak{A}$:

Definition 46. *The set of terms corresponding to $\mathfrak{A} = (X, \cdot)$, denoted by $\mathbb{T}(\mathfrak{A})$, is inductively defined as:*

- *We introduce variables $v_0, v_1, ...$ and add them to $\mathbb{T}(\mathfrak{A})$, i.e. $\forall i : v_i \in \mathbb{T}(\mathfrak{A})$*
- *To every element in $\mathfrak{A}$ we associate a constant c_a in $\mathbb{T}(\mathfrak{A})$*
- *The application of terms is a term, i.e. $\forall M, N \in \mathbb{T}(\mathfrak{A}) : (MN) \in \mathbb{T}(\mathfrak{A})$*

If $\mathfrak{A}$ is a combinatory algebra, we extend $\mathbb{T}(\mathfrak{A})$ with $\boldsymbol{K}$ and $\boldsymbol{S}$.

Definition 47. *Let $\mathfrak{A}$ be a combinatory algebra. Then $\Delta(\mathfrak{A})$ is the set of λ-terms such that the constants c_a, with $a \in \mathfrak{A}$, are added.*

In the previous section we defined maps λ, CL between λ-terms and CL-terms. We define the same maps between $\Delta(\mathfrak{A})$ and $\mathbb{T}(\mathfrak{A})$.

Let Var be the set of these variables:

Definition 48. *A valuation in $\mathfrak{A}$ is a map $\rho : Var \to \mathfrak{A}$. The interpretation of a term in $\mathbb{T}(A)$ under such ρ is defined as:*

$$\begin{aligned} [\![x]\!]^{\mathfrak{A}}_{\rho} &= \rho(x) \\ [\![c_a]\!]^{\mathfrak{A}}_{\rho} &= a \\ [\![MN]\!]^{\mathfrak{A}}_{\rho} &= [\![M]\!]^{\mathfrak{A}}_{\rho}[\![N]\!]^{\mathfrak{A}}_{\rho} \end{aligned}$$

For $M \in \Delta(\mathfrak{A})$ we define the interpretation as

$$[\![M]\!]_{\rho} = [\![M_{CL}]\!]_{\rho}$$

The following definition defines the equality on elements:

Definition 49. $\mathfrak{A} \vDash M = N$ *if $[\![M]\!]^{\mathfrak{A}}_{\rho} = [\![N]\!]^{\mathfrak{A}}_{\rho}$ is true for every valuation ρ.*

Definition 50. *A combinatory algebra $\mathfrak{A}$ is weakly extensional if for $M, N \in \mathbb{T}(\mathfrak{A})$ we have $\mathfrak{A} \vDash M = N \implies \lambda x.M = \lambda x.N$ for all $x \in Var$.*

We now have terms and equality between them, so we can define the theory corresponding to it:

Definition 51. *The theory of a combinatory algebra $\mathfrak{A}$ is:*

$$Th(\mathfrak{A}) := \{M = N | \mathfrak{A} \vDash M = N, M, N \in \mathbb{T}(\mathfrak{A})^0\}$$

Definition 52. *A λ-algebra is a combinatory algebra $\mathfrak{A}$ such that for all $M, N \in \mathbb{T}(\mathfrak{A})$ we have:*

$$\lambda \vdash M_\lambda = N_\lambda \implies \mathfrak{A} \vDash M = N$$

Theorem 20. *A combinatory algebra $\mathfrak{A}$ is a λ-algebra iff*

- $\lambda \vdash M = N \implies \mathfrak{A} \vDash M = N$ *for all* $M, N \in \Delta(\mathfrak{A})$
- $\mathfrak{A} \vDash \boldsymbol{K}_{\lambda,CL} = \boldsymbol{K}$
- $\mathfrak{A} \vDash \boldsymbol{S}_{\lambda,CL} = \boldsymbol{S}$

Proof.

- **Only if**: We show $\forall M \in \Delta(\mathfrak{A}) : \lambda \vdash M = M_{CL,\lambda}$, Because then we have:

$$\begin{aligned} \lambda \vdash M = N &\implies \lambda M_{CL,\lambda} = N_{CL,\lambda} \\ &\implies \mathfrak{A} \vDash M_{CL} = N_{CL} \\ &\implies \mathfrak{A} \vDash M = N \end{aligned}$$

  and we showed the first point. So we show $M = M_{CL,\lambda}$ by induction on the structure of M. As usual the base case and application are trivial, the only one we show is that of abstraction. So let $M \equiv \lambda x.N$.

$$M_{CL,\lambda} = (\lambda x.N)_{CL,\lambda} = (\lambda^* x.N)_\lambda$$

  So it remains to show that $(\lambda^* x.N)_\lambda = \lambda x.N_\lambda$. We have proven this already as a lemma to show the equivalence of λ and $\mathbf{CL} + \mathbf{A}_\beta$ and the proof is exactly the same.
  Instead of showing $\mathfrak{A} \vDash \mathbf{K}_{\lambda,CL} = \mathbf{K}$ (and the same for $\mathbf{S}$) we show this for an arbritrary CL-term A. Since $\lambda \vdash M = M_{CL,\lambda}$ for M a λ-term and A_λ is a λ-term we have $\lambda \vdash A_\lambda = A_{\lambda,CL,\lambda}$ then the result follows as $\mathfrak{A}$ is a λ-algebra.

- **If**: As with the case **Only if**, we show an intermediate result $\forall A \in \mathbb{T}(\mathfrak{A}) : \mathfrak{A} \vDash A_{CL,\lambda} = A$ because then we have:

$$\begin{aligned} \lambda \vdash A_\lambda = B_\lambda &\implies \mathfrak{A} \vDash A_\lambda = B_\lambda \\ &\implies \mathfrak{A} \vDash A = A_{\lambda,CL} = B_{\lambda,CL} = B \end{aligned}$$

  We show $\mathfrak{A} \vDash A_{CL,\lambda} = A$ by induction on the structure of A. The base cases are trivial as is the application. Let $A \equiv \lambda x.B$.

□

Definition 53. *A λ-algebra is called a λ-model if the Meyer-Scott axiom holds in $\mathfrak{A}$:*

$$(\forall x \in \mathfrak{A} : ax = bx) \implies \boldsymbol{1}a = \boldsymbol{1}b$$

where $\boldsymbol{1} = \boldsymbol{S(KI)}$

Theorem 21. *A λ-algebra $\mathfrak{A}$ is a λ-model iff $\mathfrak{A}$ is weakly extensional*

Proof.

- **If**: Since $\mathfrak{A}$ is weakly extensional, we have:

  $$(\forall x : ax = bx) \implies \lambda x.ax = \lambda x.bx$$

  And we have:

  $$\mathbf{1{=}S(KI)} = \lambda yz.\mathbf{KI}z(yz) = \lambda yz.I(yz) = \lambda yz.yz$$

  Thus we have

  $$\mathbf{1}a = \lambda x.ax = \lambda x.bx = \mathbf{1}b$$

- **Only if**: If $A = B$ then $\forall x : (\lambda x.A)x = A = B = (\lambda x.B)x$. As this is for every x, we have by the Meyer-Scott that $\mathbf{1}(\lambda x.A) = \mathbf{1}(\lambda x.B)$. But we also have:

  $$\mathbf{1}(\lambda x.A) = (\lambda yz.yz)(\lambda x.A) = \lambda z(\lambda x.A)z = \lambda x.A$$

□

Theorem 22. *A λ-algebra $\mathfrak{A}$ is extensional iff $\mathfrak{A}$ is weakly extensional and satisfies $\boldsymbol{I} = \boldsymbol{1}$*

Proof.

- **Only if:** Assume $\mathfrak{A}$ is extensional and let $\mathfrak{A} \vDash M = N$, thus $(\lambda x.M)x = (\lambda x.N)x$, but by extensionality we then have $\lambda x.N = \lambda x.M$. To show $\mathbf{I} = \mathbf{1}$ we use $\mathbf{1}xy = (\lambda pq.pq)xy = xy = (\mathbf{I}x)y = \mathbf{I}xy$. Thus by extensionality we conclude $\mathbf{1} = \mathbf{I}$.

- **If:** By the previous theorem, we have that since $\mathfrak{A}$ is weakly extensional, it is a λ-model. Thus $(\forall x : ax = bx) \implies \mathbf{1}a = \mathbf{1}b$. But because $\mathbf{1} = \mathbf{I}$ we have that $\mathbf{1}a = \mathbf{I}a = a$ and also for b, thus we are done.

□

The following theorem shows that there is a connection between theories and models:

Theorem 23. *$\lambda \vdash M = N$ iff $\mathfrak{A} \vDash M = N$ for all λ-models $\mathfrak{A}$*

Proof. This is proved in the following section. □

Definition 54. *A homomorphism between λ-algebras $\mathfrak{A}_1, \mathfrak{A}_2$ is a map $\phi : \mathfrak{A}_1 \to \mathfrak{A}_2$ such that:*

- $\phi(x \cdot y) = \phi(x) \cdot \phi(y)$

- $\phi(k) = k$
- $\phi(s) = s$

Theorem 24. *The category of lambda theories* $\boldsymbol{LT}$ *is equivalent with the category of lambda algebras* $\boldsymbol{LA}$.

Proof. This is proved in section 2.5 of [13]. □

3.2 Term model

Given a λ-theory $\mathbb{T}$ we can construct a λ-model by using the fact that every *lambda*-theory corresponds to a theory in CL. But there is also a trivial way of associating a model to $\mathbb{T}$, this is called the term model. Although at first sight the term-model isn't very interesting, it makes completeness of the lambda calculus easy to prove and is therefore important.

Definition 55. *The term model of* $\mathbb{T}$ *is* $\mathfrak{A}(\mathbb{T}) = (\frac{\Delta}{=_{\mathbb{T}}}, \cdot, [\boldsymbol{K}], [\boldsymbol{S}])$*, where:*

- $M =_{\mathbb{T}} N \iff \mathbb{T} \vdash M = N$
- $\frac{\Delta}{=_{\mathbb{T}}}$ *contain the equivalence classes*
- $[\boldsymbol{x}]$ *is the equivalence class of* $x \in \Delta$

Theorem 25. $\mathfrak{A}(\mathbb{T})$ *is a combinatory algebra*

Proof. This is clear since the only conditions that need to be checked are the definitions of **K,S**. □

Lemma 5. *For* $M \in \Delta$ *with* $FV(M) = \{x_1, ..., x_n\}$ *we have:*

$$[\![M]\!]_\rho = [M[x_1...x_n = P_1...P_n]]_{\mathbb{T}}$$

where $[P_i]_{\mathbb{T}} = \rho(x_i)$

Proof. We first show this for $A \in \mathbb{T}$ by induction on the structure of A, the RHS then becomes A_λ instead of A:

- $A \equiv x_i$:

$$[\![A]\!] = [\rho(x_i)]$$
$$[A_\lambda[x_1...x_n = P_1...P_n]] = [x_i[x_i = P_i]] = [\rho(x_i)]$$

- $A \equiv c_a$ a constant with $a \in \mathfrak{A}$:

$$[\![A]\!] = [\![c_a]\!] = [a]$$
$$[A_\lambda[x_1...x_n = P_1...P_n]] = [(c_a)_\lambda]$$

  And both are clearly in the same equivalence class.

- $A \equiv BC$:

$$\begin{aligned}[\![A]\!] = [\![BC]\!] = [\![B]\!][\![C]\!] &= [B_\lambda[x_1...x_n = P_1...P_n]]\,[C_\lambda[y_1...y_m = Q_1...Q_n]] \\ &= [BC_\lambda[x_1...y_m = P_1...P_m]] \\ &= [A_\lambda[x_1...y_m = P_1...P_m]]\end{aligned}$$

So now we have proved it for CL-terms. By definition of interpretation for Δ we have:

$$[\![M]\!] = [\![M_{CL}]\!]$$

And by previous step this equals:

$$[M_{CL,\lambda}[x_1...x_n = P_1...P_n]]$$

But $\mathbb{T} \vdash M = M_{CL,\lambda}$ and thus we are done. □

Theorem 26. $\mathbb{T} \vdash M = N \iff \mathfrak{A} = \mathfrak{A}(\mathbb{T}) \vDash M = N$

Proof. • **Only if**:

$$\begin{aligned}\mathbb{T} \vdash M = N &\implies \forall P_1...P_n : \mathbb{T} \vdash M[x_1...x_n := P_1...P_n] = N[x_1...x_n := P_1...P_n] \\ &\implies \forall P_1...P_n : [M[x_1...x_n := P_1...P_n]] = [N[x_1...x_n := P_1...P_n]]\end{aligned}$$

By applying the previous lemma we then have:

$$\forall \rho : [\![M]\!] = [\![N]\!] \tag{3.1}$$

This means that the term model derives it.

- **If**: As the model derives $M = N$, then in particular for $\rho_0(x) = [x]_{\mathbb{T}}$ they are equal under their intepretation. We also have

$$[\![M]\!]_{\rho_0} = [M]_{\mathbb{T}}$$

because of the previous lemma and thus we have:

$$[M]_{\mathbb{T}} = [\![M]\!]_{\rho_0} = [\![N]\!]_{\rho_0} = [N]_{\mathbb{T}}$$

But that is the definition of equality in $\mathbb{T}$ and thus we are done.

□

We are now ready to prove the following:

Theorem 27. $\mathfrak{A}(\mathbb{T})$ *is a* λ*-algebra*

Proof. We prove this by the characterization of a λ-algebra so we only have to show:

$$\mathbb{T} \vdash \mathbf{K}_{\lambda,CL} = \mathbf{K}$$

and also for $\mathbf{S}$. □

Theorem 28. $\mathfrak{A} := \mathfrak{A}(\mathbb{T})$ *is a* λ*-model*

Proof. Assume $\mathfrak{A}$ derives $ax = bx$ for every x. By definition of the term-model we can write $a = [M], b = [N], x = [z]$, thus

$$\mathfrak{A} \vDash [Mz] = [M][z] = [N][z] = [Nz].$$

Since

$$\mathbb{T} \vdash M = N \iff \mathfrak{A} = \mathfrak{A}(\mathbb{T}) \vDash M = N$$

we have: $\mathbb{T} \vdash Mz = Nz$. And since $\mathfrak{A}$ is a λ-theory we have $\lambda z.Mz = \lambda z.Nz$, thus:

$$\mathfrak{A} \vdash \mathbf{1}M = \lambda z.Mz = \lambda z.Nz = \mathbf{1}N. \tag{3.2}$$

Thus by the the same equation and by definition of N, M we can go back to the term-model:

$$\mathfrak{A} \vDash \mathbf{1}a = \mathbf{1}b \tag{3.3}$$

□

Theorem 29. $\mathfrak{A}(\mathbb{T})$ *is extensional iff* $\mathbb{T} \vdash \boldsymbol{ext}$

Proof. We use the same approach as the proof of previous theorem. The equality in the term-model is equivalent to equality in the theory and thus we can easily switch between them. □

Theorem 30. *Every* λ*-algebra* $\mathfrak{A}$ *can be embedded into a* λ*-model*

Proof. The approach goes as follows: we write $\mathfrak{A}$ as a closed-term model and clearly a closed-term model can be embedded into its open-term model which is a λ-model.
Remember that $Th(\mathfrak{A})$ consist of the equality of the closed CL-terms, we define $Th(\bar{\mathfrak{A}})$ as the set containing the equality of closed λ-terms. We denote $\mathfrak{A}^0(\mathbb{T})$ as the term-model that contains only equality of closed CL-terms. We then have that

$$\mathfrak{A}^0(Th(\mathfrak{A}))$$

is isomorphic with the substructure of $\mathfrak{A}$ generated by k and s (this is thus the image of the closed terms when mapped into $\mathfrak{A}$). Now define

$$Th(\bar{\mathfrak{A}}) = \{M = N | M, N \in \mathbb{T}(\mathfrak{A}), M, N \text{ closed}, \mathfrak{A} \vDash M = N\}.$$

Then $\mathfrak{A} \cong \mathfrak{A}^0(Th(\bar{\mathfrak{A}})) \subset \mathfrak{A}(Th(\bar{\mathfrak{A}}))$ □

3.3 Syntactical models

A λ-model is defined to be an applicative structure with some axioms regarding combinators, now we are going to define an equivalent notion of a λ-theory and -model using the interpretation, instead of using combinators.

Definition 56. *Let $\mathfrak{A} = (X, \cdot)$ be an applicative structure. A syntactical interpretation in $\mathfrak{A}$, denoted as $[\![]\!]_\rho$, corresponding to a valuation ρ, is defined as:*

- $[\![x]\!]_\rho = \rho(x)$
- $[\![c_a]\!]_\rho = a$
- $[\![MN]\!]_\rho = [\![M]\!]_\rho [\![N]\!]_\rho$
- $[\![\lambda x.M]\!]_\rho \cdot a = [\![M]\!]_{\rho(x:=a)}$
- $[\forall x \in FV(M) : \rho_1(x) = \rho_2(x)] \implies [\![M]\!]_{\rho_1} = [\![M]\!]_{\rho_2}$

We call $\mathfrak{A} = (X, \cdot, [\![]\!])$ as syntactical applicative structure.

Let $\mathfrak{A}$ be a syntactical applicative structure, we denote:

$$\mathfrak{A} \vDash M = N \iff \forall \rho : [\![M]\!]_\rho = [\![N]\!]_\rho$$

Definition 57.

- *$\mathfrak{A}$ is a syntactical λ-algebra if*

$$\lambda \vdash M = N \implies \mathfrak{A} \vDash M = N \tag{3.4}$$

- *$\mathfrak{A}$ is a syntactical λ-model if forall x we have*

$$\forall a : [\![M]\!]_{\rho(x:=a)} = [\![N]\!]_{\rho(x:=a)} \implies \mathfrak{A} \vDash \lambda x.M = \lambda x.N \tag{3.5}$$

Definition 58. *A homomorphism between syntactical λ-algebras $\mathfrak{A}_1, \mathfrak{A}_2$ is a map $\phi : \mathfrak{A}_1 \to \mathfrak{A}_2$ such that for all $M \in \Delta(A_1)$ one has:*

$$\phi[\![M]\!]_\rho = [\![\phi M]\!]_{\phi \circ \rho}$$

The following proposition shows that every syntactical λ-model is indeed a λ-algebra:

Theorem 31. *If $\lambda \vdash M = N$, then for every syntactical λ-model $\mathfrak{A}$ we have:*

$$\mathfrak{A} \vDash M = N$$

Proof. Assume $\lambda \vdash M = N$, we prove this by showing that the interpretation of both sides of the axioms in the λ-theory remain equal. The only non-direct is the α-conversion, $\lambda \vdash (\lambda x.M)N = M[x := N]$:

$$[\![(\lambda x.M)N]\!]_\rho = [\![\lambda x.M]\!]_\rho [\![N]\!]_\rho = [\![M]\!]_{\rho(x:=[\![N]\!]_\rho)} \tag{3.6}$$

So we need $[\![M]\!]_{\rho(x:=[\![N]\!]_\rho)} = [\![M[x := N]]\!]_\rho$. This is true, but for a proof I refer you to page 103 in [3] as this are a lot of calculations. □

To every syntactical λ-algebra we can easily associate a λ-algebra and vice versa, but it is actually more general, the 'natural conversion' is actually an isomorphism that shows that the category of λ-algebras is isomorphic to the category of syntactical λ-algebras and the models coincide.

3.4 Scott's model

In this section we will discuss Scott's model D_∞ of the untyped lambda calculus, the first non-trivial model. It will be a reflexive object defined using projective (or inverse) limit and we show that it's theory is independent of the starting object. We will use cpo's to define D_∞ but there are more ways to do this, it is also done using domains, lattices. Scott himself did the construction with the use of lattices but later is was generalised. We use cpo's since it is the most abstract order-theoretic way to do Scott's construction.

3.4.1 Complete partial orders

Let $D = (D, \leq), D_1, D_2, ...$ be posets.

Definition 59. *A non-empty subset $X \subset D$ is directed if every 2 elements in X have an upperbound in X, i.e. $\forall x, y \in X : \exists z \in X : (x \leq z, y \leq z)$.*

Definition 60. *D is a complete partial order (cpo) if:*

- *There is a least element in D, i.e. $\exists \perp \in D : \forall x \in D : \perp \leq x$*
- *The supremum of every directed set X exists in D, this is denoted by $\bigvee X$*

We shall always denote the least element by $\perp$ and it is called the bottom.

Definition 61. *A map $f : D_1 \to D_2$ is continuous if for every directed subset $X \subset D_1$ we have:*

$$f(\bigvee X) = \bigvee f(X)$$

The reason why it is called continuous, is because on every cpo we can define a topology, called the Scott-topology. A map is continuous iff it is continuous w.r.t. the Scott-topology.

The following propositions show that the category of cpo's with continuous maps is cartesian closed:

Theorem 32. *The cartesian product $D_1 \times D_2$ is a cpo with partial order:*

$$(x_1, x_2) \leq (y_1, y_2) \iff x_1 \leq y_1, x_2 \leq y_2$$

Proof. Take $X \subset D_1 \times D_2$ directed and define:

$$X_1 = \{x_1 \in D_1 : \exists x_2 \in D_2, (x_1, x_2) \in X\}$$
$$X_2 = \{x_2 \in D_2 : \exists x_1 \in D_1, (x_1, x_2) \in X\}$$

Notice that since X is directed, both X_1 and X_2 are directed, thus $\bigvee X = (\bigvee X_1, \bigvee X_2)$ is a supremum for X.
Denote $\perp_1$ (resp. $\perp_2$) for the bottom element of D_1 (resp. D_2), thus $\perp = (\perp_1, \perp_2)$ is a bottom element of $D_1 \times D_2$ □

Definition 62. *The (continuous) function space between D_1 and D_2 is*

$$[D_1 \to D_2] = \{f : D_1 \to D_2 |\ f \text{ continuous}\}$$

we order this pointwise:

$$f \leq g \iff \forall x \in D_1 : f(x) \leq g(x)$$

Theorem 33. *The function space is a cpo*

Proof. Take $F \subset [D_1 \to D_2]$ directed and define:

$$\bigvee F : D_1 \to D_2 : x \mapsto \bigvee \{f(x) | f \in F\}$$

We have to show that $\bigvee F$ is continuous so that $\bigvee F$ is indeed again in the function space $[D_1 \to D_2]$. The bottom element is the function $x \mapsto \perp_2$.
Take $X \subset D_1$ directed, we have to show that $\bigvee F(\bigvee X) = \bigvee \bigvee F(X)$:

$$\bigvee F(\bigvee X) = \bigvee \{f(\bigvee X) | f \in F\}$$

As f itself is continuous this is equal to:

$$\bigvee \{\bigvee f(X) | f \in F\} = \bigvee \bigvee F(X)$$

□

We now give some usefull properties about continuity:

Theorem 34. *Let $f : D_1 \times D_2 \to D_3$. Then f is continuous iff f is continuous in each argument, i.e. $\forall x_1 \in D_1, x_2 \in D_2 : f(x, x_2), f(x_1, x)$ are continuous*

Proof.
- **Only if**: Assume $f(x, \cdot) : D_2 \to D_3$ and $f(\cdot, x) : D_1 \to D_3$ are continuous. Let $X = X_1 \times X_2 \subset D_1 \times D_2$ be directed. Since

$$\bigvee X = (\bigvee X_1, \bigvee X_2)$$

we have:

$$\begin{aligned} f(\bigvee X) = \quad & f(\bigvee X_1, \bigvee X_2) && = \bigvee_{x_1 \in X_1} f(x_1, \bigvee X_2) \\ = \quad & \bigvee_{x_1 \in X_1} \bigvee_{x_2 \in X_2} f(x_1, x_2) && = \bigvee_{(x_1, x_2) \in X} f(x_1, x_2). \end{aligned}$$

- **If**: Assume $f : D_1 \times D_2 \to D_3$ is continuous. Let $X_1 \subset D_1$ be directed, then $X_1 \times \{x\}$ is directed. Thus by continuity of f we have:

$$\begin{aligned} f(\bigvee X_1, x) = & \quad f(\bigvee X_1, \bigvee\{x\}) & = f(\bigvee(X_1 \times \{x\})) \\ = & \quad \bigvee_{(x_1,x)\in X_1\times\{x\}} f(x_1, x) & = \bigvee_{x_1\in X_1} \bigvee_{x\in\{x\}} f(x_1, x) \\ = & \quad \bigvee_{x_1\in X_1} f(x_1, x). \end{aligned}$$

  For a directed set $X_2 \subset D_2$ it is the same.

□

Both application and abstraction are continuous:

Theorem 35. *Define* $App : [D_1 \to D_2] \times D_1 \to D_2 : (f, x) \mapsto f(x)$*, then App is continuous*

Proof. By the previous theorem we have to show that it is continuous in each argument:

- Fix $f \in [D_1 \to D_2]$. We have to show that $\lambda x.f(x)$ is continuous, but this follows since $f = \lambda x.f(x)$ and f is in the function space and thus continuous
- Fix $x \in D_1$. We have to show that $h = \lambda f.f(x)$ is continuous. Take $F \subset [D_1 \to D_2]$ directed, then

$$h(\bigvee F) = (\lambda f.f(x))\bigvee F = \bigvee F(x)$$

  By the definition of the supremum in the function space we have:

$$\bigvee F(x) = \bigvee\{f(x) : f \in F\} = \bigvee\{h(f) : f \in F\} = \bigvee h(F)$$

□

We now prove that abstraction is continuous:

Theorem 36. *Let* $f : D_1 \times D_2 \to D_3$ *be a map. Define* $\hat{f} : D_1 \to (D_2 \to D_3)$ *such that* $\hat{f}(x) : D_2 \to D_3 : y \mapsto f(x, y)$*. Then the following are equivalent:*

1. *f is continuous*
2. *$\hat{f}$ is continuous and $\hat{f}(x)$ is continuous.*

Proof.

- Assume f is continuous and let $X_1 \subset D_1$ be directed. Then

$$f(\bigvee X_1, y) = f(\bigvee(X_1 \times \{y\})) = \bigvee_{(x_1,y)\in X_1\times\{y\}} f(x_1, y)$$

for $y \in X_2$. Thus we can write $\hat{f}$ as

$$\hat{f}(\bigvee X_1) : D_2 \to D_3 : y \mapsto \bigvee_{(x_1,y)\in X_1\times\{y\}} f(x_1, y) = \bigvee_{x_1\in X_1} f(x_1, y).$$

By the definition of the supremum in a function space this equals:

$$\bigvee_{x_1\in X_1} \hat{f}(x_1) : D_2 \to D_3 : y \mapsto f(x_1, y).$$

Therefore $\hat{f}$ is continuous. We now show that $\hat{f}(x) : D_2 \to D_3$ is continuous. Let $X_2 \subset D_2$ be directed. Then:

$$\begin{aligned}
\hat{f}(x)(\bigvee X_2) = & \quad f(x, \bigvee X_2) & = f(\bigvee(\{x\} \times X_2)) \\
= & \quad \bigvee_{(x,x_2)\in\{x\}\times X_2} f(x, x_2) & = \bigvee_{x\in\{x\}} \bigvee_{x_2\in X_2} f(x, x_2) \\
= & \quad \bigvee_{x_2\in X_2} f(x, x_2) & = \bigvee_{x_2\in X_2} \hat{f}(x)(x_2).
\end{aligned}$$

- Assume $\hat{f}$ and $\hat{f}(x)$ are continuous and let $X_1 \times X_2 \subset D_1 \times D_2$ be directed, we have to show that f is continuous in each argument. We first fix $x_2 \in D_2$ and show that $f(\cdot, x_2) : D_1 \to D_3$ is continuous. Since $\hat{f}$ is continuous we have:

$$f(\bigvee X_1, x_2) = (\hat{f}(\bigvee X_1))(x_2) = (\bigvee_{x_1\in X_1} \hat{f}(x_1))(x_2)$$

By the definition of the supremum we have that

$$\bigvee_{x_1\in X_1} \hat{f}(x_1) : D_2 \to D_3 : y \mapsto f(x_1, y)$$

equals:

$$D_2 \to D_3 : y \mapsto \bigvee_{x_1\in X_1} f(x_1, y).$$

Thus

$$(\bigvee_{x_1\in X_1} \hat{f}(x_1))(x_2) = \bigvee_{x_1\in X_1} f(x_1, x_2).$$

We now fix $x_1 \in D_1$ and show that $f(x_1, \cdot) : D_2 \to D_3$ is continuous. Since $\hat{f}(x)$ is continuous we have:

$$f(x_1, \bigvee X_2) = (\hat{f}(x_1))(\bigvee X_2) = \bigvee_{x_2\in X_2} (\hat{f}(x_1))(x_2).$$

As above, the claim

$$\bigvee_{x_2\in X_2} \hat{f}(x_1)(x_2) = \bigvee_{x_2\in X_2} f(x_1, x_2)$$

also follows from the definition of the supremum.

□

3.4.2 Reflexive cpo's as models

A reflexive cpo is a cpo such that the function space, up to isomorphism, is a subset of the cpo itself:

Definition 63. *A cpo D is reflexive if there exist continuous maps*

$$F : D \to [D \to D], \quad G : [D \to D] \to D$$

such that $F \circ G = Id_{[D \to D]}$

It will turn out that Scott's model is a reflexive cpo and it is even stronger, its function space is (instead of a subset) isomorphic to the cpo.

Let D be a reflexive cpo via F, G. We now define the application and interpretation:

Definition 64.

- *For $x, y \in D$ set $x \cdot y = F(x)(y)$*
- *Let ρ be a continuous valuation in D. Define $[\![]\!] = [\![]\!]_\rho : \Delta \to D$ as follows:*

$$\begin{aligned} [\![x]\!] &= \rho(x) \\ [\![c_a]\!] &= a \\ [\![MN]\!] &= [\![M]\!] \cdot [\![N]\!] \\ [\![\lambda x.M]\!] &= G(\lambda d.[\![M]\!]_{\rho(x:=d)}) \end{aligned}$$

We can also extend this to $\Delta \perp$ by setting $[\![\perp]\!] = \perp$.

The following lemma says that $[\![\lambda x.M]\!]$ is well-defined since the function G only takes continuous functions as input:

Lemma 6. *$\lambda d.[\![M]\!]_{\rho(x:=d)}$ is continuous*

Proof. We prove this by induction on the structure of M.

- If $M \equiv x$ then $\lambda d.[\![M]\!]_{\rho(x:=d)} = \lambda d.[\![x]\!]_{\rho(x:=d)} = \lambda d.\rho(d) = \rho$.
- If $M \equiv y \not\equiv x$ then $\lambda d.[\![M]\!]_{\rho(x:=d)} = \lambda d.[\![y]\!]_{\rho(x:=d)} = \lambda d.\rho(y)$ and since this is a constant function we are done.
- If $M \equiv N_1 N_2$ then $\lambda d.[\![M]\!]_{\rho(x:=d)} = \lambda d.[\![N_1]\!]_{\rho(x:=d)} \lambda d.[\![N_2]\!]_{\rho(x:=d)}$, by the induction hypothesis both of them are continuous and since the application is continuous we are done.
- If $M \equiv \lambda y.P$ then

  $$[\![M]\!]_{\rho(x:=d)} = [\![\lambda y.P]\!]_{\rho(x:=d)} = G(\lambda e.[\![P]\!]_{\rho(x:=d)(y:=e)}).$$

  If we define $f(d, e) := [\![P]\!]_{\rho(x:=d)(y:=e)}$, we can apply the induction hypothesis on f. Thus f is continuous and since both abtraction and G are continuous the statement is proven because the composition is continuous.

□

We now prove that to every reflexive object we can define a λ-model:

Theorem 37. $\mathfrak{A} = (D, \cdot, [\![]\!])$ *is a syntactic* λ*-model*

Proof. The first three conditions are included in the definition of $[\![]\!]_\rho$ so we only have to check $[\![\lambda x.P]\!]_\rho \cdot a = [\![P]\!]_{\rho(x:=a)}$ in order to be a syntactical applicative structure.

$$[\![\lambda x.P]\!]_\rho \cdot a = G(\lambda d.[\![P]\!]_{\rho(x:=d)}) \cdot a$$

As the application is $x \cdot y := F(x)(y)$ this equals:

$$F(G(\lambda d.[\![P]\!]_{\rho(x:=d)}))(a)$$

And $(F \circ G) = Id$ thus we have:

$$(\lambda d.[\![P]\!]_{\rho(x:=d)})(a) = [\![P]\!]_{\rho(x:=a)}$$

Now we show that it is a syntactical λ-model:

$$\begin{array}{lll} \forall d : [\![M]\!]_{\rho(x:=d)} = [\![N]\!]_{\rho(x:=d)} & \Longrightarrow & \lambda d.[\![M]\!]_{\rho(x:=d)} = \lambda d.[\![N]\!]_{\rho(x:=d)} \\ & \Longrightarrow & G(\lambda d.[\![M]\!]_{\rho(x:=d)}) = G(\lambda d.[\![N]\!]_{\rho(x:=d)}) \\ & \Longrightarrow & [\![\lambda x.M]\!]_\rho = [\![\lambda x.N]\!]_\rho \end{array}$$

□

Theorem 38. $\mathfrak{A}$ *defined as above is extensional iff* $D \cong [D \to D]$ *via* F, G*. So if* D *is extensional it means that* $G \circ F = Id_D$*.*

Proof.

- Assume $G \circ F = Id$, then:

$$\begin{array}{lll} \forall e : de = d' e & \Longrightarrow & \forall e : F(d)(e) = F(d')(e) \\ & \Longrightarrow & F(d) = F(d') \\ & \Longrightarrow & d = GF(d) = GF(d') = d' \end{array}$$

- Assume $\mathfrak{A}$ is extensional and let $d \in D$ and $d' = G(F(d))$, then

$$d' e = F(d')(e) = F(G(F(d)))(e) = F(d)(e) = de$$

  Thus by extensionality we have $G(F(d)) = d' = d$.

□

Instead of only looking at reflexive objects in the category CPO whose objects are the complete partial orders and whose morphisms are the continuous maps, this can be generalised to reflexive objects in arbitrary cartesian closed categories.

3.4.3 Projective limit

The projective limit is taken on a projective system of cpo's:

Definition 65. *A projective system is a countable sequence $(D_n, f_n)_{n\in N}$ with $D_0, D_1, ...$ cpo's and $f_n \in [D_{n+1} \to D_n]$.*

Definition 66. *The projective limit of a projective system (D_n, f_n), denoted as $D_\infty = \varprojlim(D_n, f_n)$, is defined as*

$$D_\infty = \{(x_n)_n \in \Pi_{n\in N} D_n | \forall n \in N : f_n(x_{n+1}) = x_n\}$$

with pointwise ordering: $(x_n)_n \leq (y_n)_n \iff \forall n \in N : x_n \leq y_n$.

Theorem 39. *The projective limit is a cpo*

Proof. Take $X \subset \varprojlim D_n$ directed and define:

$$\bigvee X = \lambda n. \bigvee \{x_n | x \in X\}$$

Since X directed we have $\forall n : \{x_n | x \in X\}$ is directed. Denote the supremum as y_n we need to show that $(y_n)_{n\in N} \in D_\infty$, i.e. $f_n(y_{n+1}) = y_n$:

$$f_n(y_{n+1}) = f_n(\bigvee \{x_{n+1} | x \in X\}) = \bigvee f_n(\{x_{n+1} | x \in X\}) = \bigvee \{x_n | x \in X\} = y_n$$

Here we used the compatibility and continuity of f_n. □

3.4.4 Projections

Let D, D_1, D_2 be cpo's. A projection is a more general way of defining the notion of a sub cpo:

Definition 67. *A pair of mappings (ϕ, ψ) is a projection of D_2 on D_1 if*

- *$\phi : D_1 \to D_2$ and $\psi : D_2 \to D_1$ are continuous*
- *$\psi \circ \phi = Id_{D_1}$*
- *$\phi \circ \psi \leq Id_{D_2}$*

A given projection can be extended to a projection between the function spaces:

Theorem 40. *Let (ϕ, ψ) be a projection of D_2 on D_1. Then there exists a projection (ϕ^*, ψ^*) of $[D_2 \to D_2]$ on $[D_1 \to D_1]$ defined by:*

$$\phi^*(f) = \phi \circ f \circ \psi, \quad \psi^*(g) = \psi \circ g \circ \phi,$$

for $f \in [D_1 \to D_1]$ and $g \in [D_2 \to D_2]$

Proof. The continuity of ψ^* and ϕ^* follows since it is the composition of continuous maps. So we only have to show $\psi^* \circ \phi^* = Id$ and $\phi^* \circ \psi^* \leq Id$:

$$\begin{aligned} \phi^*(\psi^*(g)) &= \phi^*(\psi \circ g \circ \phi) \\ &= \phi \circ \psi \circ g \circ \phi \circ \psi \\ &\leq Id \circ g \circ Id \end{aligned}$$

$$\begin{aligned} \psi^*(\phi^*(f)) &= \psi^*(\phi \circ f \circ \psi) \\ &= \psi \circ \phi \circ f \circ \psi \circ \phi \\ &= Id \circ g \circ Id = g \end{aligned}$$

□

This lemma, and thus (ϕ^*, ψ^*), will be helpfull in constructing the model of Scott.

Example 17. *Define* $\phi_0 : D \to [D \to D]$ *and* $\psi_0 : [D \to D] \to D$ *by:*

$$\begin{aligned} \phi_0(x) &= \lambda y \in D.x \\ \psi_0(f) &= f(\bot) \end{aligned}$$

Then (ϕ_0, ψ_0) *is a projection of* $[D \to D]$ *on* D*, this projection is called the standard projection.*

Proof. We have to show that ϕ_0 and ψ_0 are continuous as well as $\phi_0(\psi_0(f)) \sqsubseteq f$ and $\psi_0(\phi_0(x)) = x$: Let $X \subseteq D$ be directed, then

$$\phi_0(\bigvee X) = \lambda y \in D. \bigvee X = \bigvee_{x \in X} \lambda y \in D.x = \bigvee_{x \in X} \phi_0(x)$$

Let $F \subset [D \to D]$ be directed, then

$$\psi_0(\bigvee F) = \bigvee F(\bot) = \bigvee_{f \in F} f(\bot) = \bigvee_{f \in F} \psi_0(f)$$

Thus both maps are continuous.

$$\psi_0(\phi_0(x)) = \psi_0(\lambda y \in D.x) = (\lambda y \in D.x)\ \bot = x$$

$$\phi_0(\psi_0(f)) = \phi_0(f(\bot)) = \lambda y \in D.f(\bot) \sqsubseteq \lambda y \in D.f(y) = f$$

□

3.4.5 Scott's model

Our goal is to construct a cpo D_∞ such that $D_\infty \cong [D_\infty \to D_\infty]$. We give an explicit construction, but notice that we actually want to solve an equation. In chapter 4 "domain equations" of the book [15], the approach we used is further generalised and if we change the function space, models of different λ-theories will be solutions.
Let D be a fixed cpo with (ϕ_0, ψ_0) the standard projection of $[D \to D]$ on D. Recursively define:

$$\begin{aligned} D_0 &= D \\ D_{n+1} &= [D_n \to D_n] \\ (\phi_{n+1}, \psi_{n+1}) &= (\phi_n^*, \psi_n^*) \end{aligned}$$

Remark 7. *Notice that the ψ_n and ϕ_n are well-defined since if we take $D^1 = D_n$ and $D^2 = [D_n \to D_n] = D_{n+1}$ in the lemma, we have that (ϕ_n^*, ψ_n^*) is a projection of D_{n+2} on D_{n+1}.*

Thus we have that $(D_n, \psi_n)_{n\in\mathbb{N}}$ is a projective system. We now set

$$D_\infty = \varprojlim (D_n, \psi_n).$$

We will show that $D_\infty \cong [D_\infty \to D_\infty]$ and is therefore a reflexive cpo. Thus we can conclude then that D_∞ is an extensional λ-model.

We first state (without proof) some results about elements in D_∞ and a definition that will be used in proofs later on, more specifically in showing the isomorphism of D_∞ and its function space:

Theorem 41. *Upto isomorphism we have:*

$$D = D_0 \subseteq D_1 \subseteq D_2 \cdots \subseteq D_\infty.$$

This follows from the fact that there exists a projection pair. Because of this there exists a map $\Phi_{n\infty} : D_n \to D_\infty$ that identifies each element in D_n with an element in D_∞.

Theorem 42. *Let $x \in D_\infty$, then:*

- $(x_n)_m = x_{min(n,m)}$,
- $n \leq m \implies x_n \sqsubseteq x_m \sqsubseteq x$,
- $x = \bigvee_n x_n$.

The first 2 properties follow from the fact that we can write an element in D_n in terms of an element in D_m and vice versa. The third property follows from the first two properties combined with the fact that for a given $x \in D_\infty$, the set $\{x_n\}_{n\in\mathbb{N}}$ is directed.

We define a binary operation on D_∞ which is actually the application on D_∞ when it is showed that it is a λ-model:

Definition 68. *Let $x, y \in D_\infty$, then $x \cdot y := \bigvee_n x_{n+1}(y)$.*

Thus notice that $x \cdot y = x(y)$.

Theorem 43. *The map $Ap : D_\infty \times D_\infty \to D_\infty : (x, y) \mapsto x \cdot y$ is continuous*

This follows because x_{n+1} and y_n are both continuous maps, as it are elements in a (continuous) function space

Other usefull properties are:

Theorem 44. *Let $x, y \in D_\infty$, then*

- $x_{n+1} \cdot y = x_{n+1} \cdot y_n = (x \cdot y_n)_n$
- $x_0 \cdot y = x_0 = (x \cdot \bot)_0$

This are easy calculations, where we use $D_n \subset D_{n+1}$.

Lemma 7. *Let $x, y \in D_\infty$, then $x \sqsubseteq y \iff \forall z \in D_\infty : x \cdot z \sqsubseteq y \cdot z$*

This follows from the previous theorem and that every continuous map is monotone.

Corollary 6. *Let $x, y \in D_\infty$, then $x = y \iff \forall z \in D_\infty : x \cdot z = y \cdot z$*

The proofs of previous theorems are worked out in the chapter 'Construction of models' in [3]

The following theorem is actually an analogue to the representation theorem of Riesz. Every function can be written as the application of 2 elements, the Riesz representation theorem says the same, but instead of application it uses the dot product:

Theorem 45. *The cpo's D_∞ and $[D_\infty \to D_\infty]$ are isomorphic*

Proof. Define $F : D_\infty \to [D_\infty \to D_\infty] : x \mapsto \lambda y.(x \cdot y)$. We first show this map is injective:
Assume $F(x) = F(z)$. Thus $\lambda y.(x \cdot y) = \lambda y.(z \cdot y)$, therefore $\forall y \in D_\infty : x \cdot y = z \cdot y$. The previous corrolary then implies that $x = z$.
To show that F is surjective we define for $f \in [D_\infty \to D_\infty]$ the element $\bar{f} = \bigvee_n (\lambda y \in D_n.(f(y))_n)$.
In order to be the surjective we have to show that for every $f \in [D_\infty \to D_\infty]$ we can write $f(y)$ as the application of 2 elements in D_∞ because $f = \lambda y.f(y)$.

We now show that $\forall y \in D_\infty : f(y) = \bar{f} \cdot y$:

$$\begin{aligned}
\bar{f} \cdot y &= \bigvee_m \bar{f}_{m+1}(y_m) = \bigvee_m (\bar{f} \cdot y_m)_m \\
&= \bigvee_m ((\bigvee_n (\lambda y \in D_n.(f(y))_n)) \cdot y_m)_m \\
&= \bigvee_{m,n} (((\lambda y \in D_n.(f(y))_n)) \cdot y_m)_m \\
&= \bigvee_m (((\lambda y \in D_m.(f(y))_m)) \cdot y_m)_m \\
&= \bigvee_m (f(y_m))_m = \bigvee_{k,l} (f(y_k))_l \\
&= \bigvee_k f(y_k) = f(y)
\end{aligned}$$

The only statement we still have to show, in order to prove that F is an isomorphism, is that F is continuous, but this follows from the continuity of the application $x \cdot y$ and of abstraction. □

Remark 8. *Although in a λ-model we only consider equalities, we can (in the case of Scott's model) also define $D_\infty \vDash M \sqsubseteq N$. Recall:*

$$D_\infty \vDash M = N \iff [\![M]\!]_\rho = [\![N]\!]_\rho, \forall \rho$$

and since $Im([\![]\!]_\rho) \subset D_\infty$ we can define:

$$D_\infty \vDash M \sqsubseteq N \iff [\![M]\!]_\rho \sqsubseteq [\![N]\!]_\rho, \forall \rho$$

3.4.6 Theory of Scott's model

In this (sub)section we will prove our main result which states that the theory of D_∞ is equivalent to K^*.

Lemma 8. $N \subset M \implies D_\infty \vDash M \sqsubseteq N$

Proof. We prove this by induction on the structure of M and we assume always that N is a proper subterm of M, otherwise $M \equiv N$ and the statement is then trivially true. If $M \equiv x$ or $M \equiv \bot$, then the statement is trivial. Let $M \equiv M_1 M_2$ and assume $N \subset M_1$. Thus we have:

$$[\![M]\!]_\rho = [\![M_1]\!]_\rho \cdot [\![M_1]\!]_\rho = Ap([\![M_1]\!]_\rho, [\![M_2]\!]_\rho).$$

Since Ap is continuous it is in particular monotonic. By induction we have $[\![N]\!]_\rho \sqsubseteq [\![M_1]\!]_\rho$, therefore:

$$[\![N M_2]\!]_\rho = [\![N]\!]_\rho \cdot [\![M_2]\!]_\rho \sqsubseteq [\![M_1]\!]_\rho \cdot [\![M_2]\!]_\rho = [\![M_1 M_2]\!]_\rho = [\![M]\!]_\rho$$

But $N \subset NM_2$, thus we can apply the induction hypothesis again and we have $[\![N]\!]_\rho \sqsubseteq [\![NM_2]\!]\rho$ and thus we conclude:

$$[\![N]\!]_\rho \sqsubseteq [\![NM_2]\!]_\rho \sqsubseteq [\![M]\!]_\rho$$

Let $M \equiv \lambda x.M_1$. The abstraction is also continuous and thus monotonic. Since $N \subset M$, we have $N \subset M_1$. Thus by induction we conclude $[\![N]\!]_\rho \sqsubseteq [\![M_1]\!]_\rho$. Therefore we have by the monotonicity of abstraction:

$$[\![N]\!]_\rho \sqsubseteq [\![M_1]\!]_\rho \sqsubseteq [\![\lambda x.M_1]\!]_\rho = [\![M]\!]_\rho$$

□

Lemma 9. *For every $N \in \mathfrak{C}(M)$ it follows that $D_\infty \vDash N \sqsubseteq M$*

Proof. Since $N \in \mathfrak{C}(M)$, $BT(N) \subset BT(M)$ we have that $N \equiv N(BT(N))$ is obtained by setting some subterms of $M(BT(M))$ equal to $\bot$. Thus

$$D_\infty \vDash N \sqsubseteq M(BT(M)) = M$$

□

Theorem 46. *("Approximation theorem") Let $M \in \Delta\ \bot$, then:*

$$D_\infty \vDash M = \bigvee \mathfrak{C}(M)$$

Proof. If we can show that $D_\infty \vDash M = M(BT(M))$ we are done, because by definition of $\mathfrak{C}(M)$ we have that $M(BT(M)) \in \mathfrak{C}(M)$ and this is cleary the supremum, thus we have then

$$D_\infty \vDash M = M(BT(M)) = \bigvee M(BT(M)).$$

Since $M(BT(M)) =_\beta M$, we have to show that equality in Scott's model is preserved when applying the β-rule, or more general that every λ-axiom is preserved. But since D_∞ is a λ-model this is true, so we are done. □

The following theorem is a more precise formulation of the approximation theorem:

Corollary 7. *For $M \in \Delta\ \bot$ we have:*

$$D_\infty \vDash M = \bigvee_k M^{[k]} \tag{3.7}$$

where $M^{[k]}$ is the term corresponding to the tree $BT^k(M)$.

Proof. Let $N \in \mathfrak{C}(M)$. Take k such that every node in $BT(N)$ have a depth less than k. Thus $N \sqsubseteq M^{[k]}$ and since $M^{[k]} \in \mathfrak{C}(M)$. Thus $D_\infty \vDash M^{[k]} \sqsubseteq M$, therefore the result follows from the approximation theorem. □

Lemma 10. *Let $M \in \Delta$. Then M is unsolvable iff $D_\infty \vDash M = \bot$.*

Proof.
- **Only if:** If M unsolvable, then $\mathfrak{C}(M)$ only contains $\bot$. But the approximation theorem tells us that $D_\infty \vDash M = \bigvee \mathfrak{C}(M)$. Thus $D_\infty \vDash M = \bot$
- **If:** Assume M is solvable while $D_\infty \vDash M = \bot$. As M is solvable their exists $x_1, ..., x_n \in Var, N \in \Delta : D_\infty \vDash \mathbf{I} = (\lambda x_1, ..., x_n \cdot M)N$. Thus:

$$D_\infty \vDash \mathbf{I} = (\lambda x_1, ..., x_n \cdot \bot)N = \bot .$$

Therefore we have that D_∞ is inconsistent since $x = x\mathbf{I} = x \ \bot = \bot$ for all $x \in D_\infty$. But we know that D_∞ is consistent, so M is unsolvable.

□

Theorem 47. *Let $M, N \in \Delta$. Then*

$$D_\infty \vDash M \sqsubseteq N \implies \forall C[] : (C[M] \textit{ solvable} \implies C[N] \textit{ solvable})$$

Proof. Suppose $D_\infty \vDash M \sqsubseteq N$. Take $C[] \in \Delta$. If $C[N]$ is unsolvable then $C[M] \sqsubseteq C[N]$. But by the previous lemma $C[N] = \bot$. Thus $C[M] = \bot$, but then the lemma tells us that $C[M]$ is unsolvable. □

Theorem 48. $\forall C[] : (C[M] \textit{ solvable} \implies C[N] \textit{ solvable}) \implies BT(M)^\eta \subseteq^\eta BT(N)$

Proof. If $BT(M)^\eta \not\subseteq^\eta BT(N)$ then there is a $\alpha \in Seq$ such that $BT(M)(\alpha)$ is defined and $BT(M)(\alpha) \neq BT(N)(\alpha)$. Take α minimal and thus $\mathfrak{F} = \{M, N\}$ agrees along α. Thus there is a bohm-transformation π which is $\alpha - \mathfrak{F}$-faithfull. Since the faitfullness, $BT(M)(\alpha) \neq BT(N)(\alpha)$ and $BT(M)(\alpha)$ is defined we have that M^π is solvable and $M \not\sim N$. Thus there is a bohm-transformation π' such that $M^{\pi'}$ solvable and $N^{\pi'}$ unsolvable but then there is a context $C_{\pi'}[]$ such that $C_{\pi'}[M]$ solvable and $C_{\pi'}[N]$ unsolvable. □

Let $M, N \in \Delta \ \bot$.

Lemma 11. $M \precsim N \implies D_\infty \vDash M \sqsubseteq N$

Proof. Since $M \precsim N \implies \forall k : M^{[k]} \precsim N$ and $M^{[k]} \in \mathfrak{C}(N)$:

$$\forall k : D_\infty \vDash M^{[k]} \sqsubseteq N \implies D_\infty \vDash M = \bigvee_k M^{[k]} \sqsubseteq N$$

□

Lemma 12. $M \lesssim_\eta N \implies D_\infty \vDash M \sqsubseteq N$

Proof. We first show the statement for M in $\beta \ \bot$-nf by induction on the structure of M:

1. $M \equiv x$: Let $N \equiv \lambda a_1...a_n \cdot xM_1...M_n$, by induction on k we show that $D_\infty \vDash x_k \sqsubseteq N$ and the result follows as $x = \bigvee_k x_k$: If $k = 0$ then

$$\begin{aligned} x_0 &\sqsubseteq \lambda a_1...a_n \cdot x_0 a_1...a_n \\ &\sqsubseteq \lambda a_1...a_n \cdot x_0 \perp ... \perp \\ &\sqsubseteq \lambda a_1...a_n \cdot xM_1...M_n = M \end{aligned}$$

   Assume by induction that the theorem is true for $i = 0...k$, then:

$$\begin{aligned} x_{k+1} &\sqsubseteq \lambda a_1...a_n \cdot x_{k+1} a_1...a_n \\ &\sqsubseteq \lambda a_1...a_n \cdot x_{k+1}(a_1)_k(a_2)_{k-1}...(a_n)_{k+1-n} \end{aligned}$$

   Since $x \lesssim_\eta M$, we have $a_i \lesssim_\eta M_i$ for $i = 1...n$, thus by induction we have

$$D_\infty \vDash (a_i)_{k-1+i} \sqsubseteq M_i.$$

   Therefore we have:

$$D_\infty \vDash x_{k+1} \sqsubseteq \lambda a_1...a_n \cdot xM_1...M_n = M.$$

2. $M \equiv \lambda x_1...x_m \cdot yM_1...M_n$: Since $M \lesssim_\eta N$, it follows that

$$N = \lambda x_1...x_m a_1...a_k \cdot yN_1...N_n L_1...L_l$$

   such that $M_1...M_n \lesssim_\eta N_1...N_n$ and $a_1...a_k \lesssim_\eta L_1...L_l$. The induction hypothesis then says that $M_1...M_n \sqsubseteq N_1...N_n$ and the base case implies that $a_1...a_k \sqsubseteq L_1...L_l$

Now the statement follows for every N: □

Lemma 13. $M \lesssim_\eta N \implies D_\infty \vDash M = N$

Proof. As the previous lemma implies $D_\infty \vDash M \sqsubseteq N$ we only have to show $D_\infty \vDash N \sqsubseteq M$.

$$\begin{aligned} M \lesssim_\eta N &\implies \forall k : \exists M_0 : M_0 \to_\eta M, M_0^{(k)} = N^{(k)} \\ &\implies \forall k : D_\infty \vDash N^{(k)} = M_0^{(k)} \sqsubseteq M_0 = M \\ &\implies D_\infty \vDash N = \bigvee_k N^{(k)} \sqsubseteq M \end{aligned}$$

□

Theorem 49. $BT(M)^\eta \subseteq^\eta BT(N) \implies D_\infty \vDash M \sqsubseteq N$.

Proof. Since $M^\eta \sqsubseteq_\approx^\eta$ there are M', N' such that $M \lesssim_\eta M' \sqsubseteq_\approx N'_\eta \gtrsim N$. By the previous lemma we have:

- $M \lesssim_\eta M' \implies D_\infty \vDash M = M'$ (also for N and N')

- $M' \underset{\sim}{\sqsubseteq} N' \implies D_\infty \vDash M' \sqsubseteq N'$

Thus $D_\infty \vDash M = M' \sqsubseteq N' = N$ □

The following theorem shows the main result:

Theorem 50. $D_\infty \vDash M \sqsubseteq N \iff \forall C[] : (C[M]$ *solvable* $\implies C[N]$ *solvable*)

Proof. If we combine previous propositions we get:

$$\begin{array}{lll} D_\infty \vDash M \sqsubseteq N & \implies & \forall C[] : (C[M] \text{ solvable} \implies C[N] \text{ solvable}) \\ & \implies & BT(M)^\eta \subseteq^\eta BT(N) \\ & \implies & D_\infty \vDash M \sqsubseteq N \end{array}$$

□

Corollary 8. *("Characterization theorem for D_∞")*

$$D_\infty \vDash M = N \iff K^* \vdash M = N$$

Proof. By the previous theorema we have:

$$\begin{array}{l} D_\infty \vDash M \sqsubseteq N \iff \forall C[] : (C[M] \text{ solvable} \implies C[N] \text{ solvable}) \\ D_\infty \vDash N \sqsubseteq M \iff \forall C[] : (C[N] \text{ solvable} \implies C[M] \text{ solvable}) \end{array}$$

And we have: $D_\infty \vDash M = N \iff D_\infty \vDash M \sqsubseteq N, D_\infty \vDash N \sqsubseteq M$. □

Thus by the characterization theorem we know that although D was a random cpo, the language of the Scott-model is independent of the choice of D!

Chapter 4

Conclusion

In this thesis we have worked our way up to study Scott's model. The main result that we encountered is the fact that its language is:

- Independent of the starting cpo
- A maximal consistent theory

The study of the lambda calculus is a very rich and relatively young subject. A lot of different concepts in mathematics are used, we used notions from

- Order theory: cpo's
- (Algebraic) logic: combinatory logic

In research there are connections with topology (tree and visser-topologies)[7][10], category theory (reflexive objects in cartesian closed categories generally[8] and in more concrete ones like vector spaces [4], ...), sheaves (on topological spaces)[2] and so on. We didn't really use algebraic concepts like rings, but polynomial rings are used in papers to prove properties about λ-algebras[13]. What we did is also a specific application of a more general theory called 'Model theory'.

Appendices

Appendix A

Lambda calculus and programming languages

In this chapter we show a little bit of intuition why the lambda calculus can represent the computable functions by showing how some of the basics of a programming language are implemented. We won't work out everything because it are all easy computations, but by showing some definitions and some calculations we can get a feeling. Examples and more operations can be found in [6] and [12].

A.1 Boolean logic

At first sight when defining the values true and false, it might seen random but when defining the (logical) operators and/or it will all work out well:

Definition 69.
- $\boldsymbol{T} = \lambda xy.x$
- $\boldsymbol{F} = \lambda xy.y$

As you might have noticed true is actually the combinator $\boldsymbol{K}$*.*

There are different ways to define the logical operators but we use the following convention:

Definition 70.
- $AND = \lambda ab.aba$
- $OR = \lambda ab.aab$

As we expect we only have that AND is true when both arguments are true:

Theorem 51.
- $AND\boldsymbol{TT} = \boldsymbol{T}$
- $AND\boldsymbol{TF} = \boldsymbol{F}$
- $AND\boldsymbol{FT} = \boldsymbol{F}$

- $AND\boldsymbol{FF} = \boldsymbol{F}$

Proof.

$$AND\mathbf{TT} = (\lambda ab.aba)(\lambda xy.x)(\lambda xy.x) = (\lambda xy.x)(\lambda xy.x)(\lambda xy.x) = \lambda xy.x = \mathbf{T}$$

$$AND\mathbf{FT} = (\lambda ab.aba)(\lambda xy.y)(\lambda xy.x) = (\lambda xy.y)(\lambda xy.x)(\lambda xy.y) = \lambda xy.y = \mathbf{F}$$

The others are analogue. □

Also the expected values for the OF operation will give the expected values.

A.2 Conditional statements

We can also define $IfThenElse$:

Definition 71. $IfThenElse = \lambda x.x$

Remark 9. *Notice that although* $\boldsymbol{T}$ *or* $\boldsymbol{F}$ *are the only usefull/important inputs, it is also defined for other lambda terms but then it doesn't make (any) sense.*

Theorem 52.
- $IfThenElse\,\boldsymbol{T}MN = M$
- $IfThenElse\,\boldsymbol{T}MN = N$

Proof.

$$IfThenElse\mathbf{T}MN = (\lambda x.x)\mathbf{T}MN) = \mathbf{T}MN = (\lambda xy.x)MN = M$$

$$IfThenElse\mathbf{F}MN = (\lambda x.x)\mathbf{F}MN) = \mathbf{F}MN = (\lambda xy.y)MN = N$$

□

A.3 Numbers and arithmetic

Another important aspect are the numbers:

Definition 72. *The term corresponding to* $n \in \mathbb{N}$*, denoted by* $\bar{n}$*, is defined as:*

$$\bar{n} = \lambda fx.f^n x$$

where $f^n x = f(...(f(x))...)$ *with* f *repeated* n *times. The term* $\bar{n}$ *is called the* n*-th Church Numeral.*

To make sure this definition is how we would expect we have following definitions and theorems:

Definition 73.
- *The successor function:* $\boldsymbol{succ} = \lambda nfx.f(nfx)$
- *The add function:* $\boldsymbol{add} = \lambda nmfx.nf(mfx)$

- *The multiplication function:* $\boldsymbol{mult} = \lambda nmfx.n(mf)$

Theorem 53.
- $\boldsymbol{succ}\bar{n} = \lambda nfx.f(nfx))\lambda fx.f^n x = \lambda fx.f((\lambda fx.f^n x)fx) = \lambda fx.f(f^n x) = \lambda fx.f^{n+1}x = n \bar{+} 1$
- $\boldsymbol{add}\bar{n}\bar{m} = \lambda fx.\bar{n}f(\bar{m}fx) = (\lambda fx.\bar{n}f)((\lambda fx.f^m x)fx) = \lambda fx.\bar{n}f(f^m x) = \lambda fx.(\lambda fx.f^n x)f(f^m x) = \lambda fx.f^n(f^m(x)) = \lambda fx.f^{n+m}x = n \bar{+} m$

Analogue we have that $\boldsymbol{mult}\bar{n}\bar{m} = n\bar{m}$.

We can also define the exponential function, predicate function and so on. A slightly harder example is that of the factorial function: $\mathbf{fact}\bar{n} = \bar{n}!$.

Definition 74. $\boldsymbol{fact}\bar{n} = \boldsymbol{IfThenElse}(\boldsymbol{IsZero}(\bar{n}))(\bar{1})(\boldsymbol{mult}\bar{n}(\boldsymbol{fact}(\boldsymbol{pred}\bar{n})))$

So notice here that we recursively defined it and if we would work it out on a example this can be pretty long. To see a (almost complete) computation of the factorial of 2, I refer you to page 23 in the notes [12]. Also notice that we defined the term by using the term itself, but this problem can be solved by defining it as follows:

$$\begin{aligned} \mathbf{fact\_rec} &= \lambda fn.\mathbf{IfThenElse}(\mathbf{IsZero}(\bar{n}))(\bar{1})(\mathbf{mult}\bar{n}(f(\mathbf{pred}\bar{n}))) \\ \mathbf{fact} &= \lambda n.\mathbf{fact\_rec}\mathbf{fact\_rec}\bar{n} \end{aligned}$$

The book [9] is a book that, probably, gives all essentials of a programming language using lambda calculus, including datastructures and so on, in a non-mathematical way.
Thus although it is very hard to write a program using the lambda calculus, it seems possible.

Bibliography

[1] Stefania Lusin Antonino Salibra. *The lattice of lambda-theories*. 2003.

[2] Steve Awodey. *Topological Representation of the Lambda calculus*. 1999.

[3] H.P. Barendregt. *The Lambda Calculus, Its syntax and semantics, revised edition*. Vol. Studies in logic and the foundation of mathematics. North-Holland, 1984.

[4] Steve Zdancewic Benoit Valiron. *Modeling Simply-Typed Lambda Calculi in the Category of Finite Vector Spaces*. 2014.

[5] *Course slides of The power of lambda calculus and types*. `http://www.cs.ioc.ee/ewscs/2016/geuvers/geuvers-slides-lecture1.pdf`.

[6] Manuel Eberl. *The untyped lambda calculus*. `https://www21.in.tum.de/~eberlm//lambda_paper.pdf`.

[7] Silvia Ghilezan. *Full intersection types and topologies in lambda calculus*. 2000.

[8] Giulio Manzonetto. *Models and theories of lambda calculus*. 2009.

[9] Greg Michaelson. *an introduction to functional programming through lambda calculus*. 1988.

[10] Fer-Jan de Vries Paula Severi. *Continuity and Discontinuity in Lambda Calculus*. 2005.

[11] Antonino Salibra. *A continuum of theories of lambda calculus without semantics*. `https://pdfs.semanticscholar.org/348b/4275200514bb295e4cba47879a7f9cdc6891.pdf`.

[12] Peter Selinger. *Lecture notes on the Lambda Calculus*. `https://www.irif.fr/~mellies/mpri/mpri-ens/biblio/Selinger-Lambda-Calculus-Notes.pdf`.

[13] Peter Selinger. *The Lambda Calculus is Algebraic*. `https://www.mscs.dal.ca/~selinger/papers/combinatory.pdf`.

[14] Morten Heine Sorensen. *Lectures on the Curry-Howard Isomorphism*. Vol. Studies in logic and the foundation of mathematics. Elsevier, 2006.

[15] Viggo Stoltenberg-Hansen. *Mathematical theory of domains*. Cambridge University Press, 2008.