Models of Untyped λ-calculus
Course Semantics and Domain Theory, NWI-IMC011

Herman Geuvers

1 Untyped λ-calculus

We briefly recapitulate notions from the untyped λ-calculus. See [2] for a brief didactic introduction to untyped λ calculus and [1] for a full overview.

Definition 1.1. The terms of the untyped λ-calculus, Λ, are defined as follows.

$$\Lambda ::= \text{Var} \mid (\Lambda\,\Lambda) \mid (\lambda\text{Var}.\Lambda)$$

We omit brackets in the way of [1], by letting them associate to the left in applications, so $M\,N\,P$ denotes $(M\,N)\,P$. Furthermore, we omit the top brackets around a λ-abstraction and we compress a sequence of abstractions, so $\lambda x\,y\,z.x$ denotes $(\lambda x.(\lambda y.(\lambda z.x)))$. The λ binds the variable x in M in the term $\lambda x.M$ and we work *modulo α-conversion* also called *modulo renaming of bound variables*. So we identify $\lambda x.x$ with $\lambda y.y$ for which we write $\lambda x.x \equiv \lambda y.y$. Similarly we have $\lambda x\,y.x \equiv \lambda y\,x.y$. We write $\text{FV}(M)$ for the set of *free variables* of M, and we say that M is *closed* in case $\text{FV}(M) = \emptyset$. The substitution N for x inside M is denoted by $M[x := N]$ and it is done with the usual care, renaming bound variables before replacing x by N, if needed. For example we have $(\lambda x.y\,x)[y := \lambda z.x] \equiv (\lambda v.y\,v)[y := \lambda z.x] \equiv (\lambda v.(\lambda z.x)\,v)$.

Examples are the well-known combinators: $\mathbf{I} := \lambda x.x$, $\mathbf{K} := \lambda x\,y.x$ $\mathbf{S} := \lambda x\,y\,z.x\,z(y\,z)$. Here are some more well-known untyped λ-terms: $\mathbf{K}_* := \lambda x\,y.y$, $\omega := \lambda x.x\,x$, $\Omega := \omega\,\omega$.

The way to encode natural numbers as closed terms is as follows.

$$c_n := \lambda f.\lambda x.f^n(x)$$

where

$$f^n(x) \text{ denotes } \underbrace{f(\ldots f(f}_{n \text{ times } f}\; x))$$

So $c_2 := \lambda f.\lambda x.f(f\,x)$. These are also known as the *Church numerals*. A Church numeral c_n denotes the n-times iteration: it can be seen as a "higher order function" that takes a function f and returns the n-times iteration of f.

Definition 1.2. The notions of one-step β-reduction, $\longrightarrow_\beta$, multiple-step β-reduction, $\twoheadrightarrow_\beta$, and β-equality, $=_\beta$ are defined as follows.

$$\begin{array}{rcl}
(\lambda x.M)\,N & \longrightarrow_\beta & M[x := N] \\
M \longrightarrow_\beta N & \Rightarrow & M\,P \longrightarrow_\beta N\,P \\
M \longrightarrow_\beta N & \Rightarrow & P\,M \longrightarrow_\beta P\,N \\
M \longrightarrow_\beta N & \Rightarrow & \lambda x.M \longrightarrow_\beta \lambda x.N
\end{array}$$

We define $\twoheadrightarrow_\beta$ as the transitive reflexive closure of $\longrightarrow_\beta$ and $=_\beta$ as the transitive reflexive symmetric closure of $\longrightarrow_\beta$.

A term M is in *normal form* if there is no N with $M \longrightarrow_\beta N$. A term M is *normalizing* if there is an N with $M \twoheadrightarrow_\beta N$ and N in normal form.

Example 1.3. We show a computation with the Church numeral c_2: we apply it to the identity **I**.

$$\begin{aligned} \lambda z.c_2\,\mathbf{I}\,z &\equiv \lambda z.(\lambda f.\lambda x.f(f\,x))\mathbf{I}\,z \\ &\longrightarrow_\beta \lambda z.(\lambda x.\mathbf{I}(\mathbf{I}\,x))z \\ &\longrightarrow_\beta \lambda z.\mathbf{I}(\mathbf{I}\,z) \\ &\longrightarrow_\beta \lambda z.\mathbf{I}\,z \\ &\longrightarrow_\beta \lambda z.z \equiv \mathbf{I} \end{aligned}$$

In general there are many *redexes* within a term that can be reduced. (A redex is a sub-term of the form $(\lambda x.M)\,N$.) The Church-Rosser theorem states that one can always converge. See [1] for a proof.

Theorem 1.4 (Church-Rosser)**.** If $M \twoheadrightarrow_\beta P_1$ and $M \twoheadrightarrow_\beta P_2$, then there is a term Q with $P_1 \twoheadrightarrow_\beta Q$ and $P_2 \twoheadrightarrow_\beta Q$.

As corollaries of the Church Rosser Theorem one also has the following.

- If $M =_\beta N$, then there is a Q with $M \twoheadrightarrow_\beta Q$ and $N \twoheadrightarrow_\beta Q$.
- (Uniqueness of Normal Forms, UN) If $M =_\beta N_1$ and $M =_\beta N_2$, with N_1, N_2 in normal form, then $N_1 \equiv N_2$.

Often we want to fix a certain method for reducing terms, or we only want to contract redexes of a certain shape. This can be observed in the following example.

Examples 1.5. Define the **S** combinator as follows.

$$\mathbf{S} := \lambda x.\lambda y.\lambda z.x\,z(y\,z)$$

Then

$$\mathbf{S}\,\mathbf{K}\,\mathbf{I} \longrightarrow_\beta (\lambda y.\lambda z.\mathbf{K}\,z(y\,z))\mathbf{I}$$

There are several ways of reducing this term further:

$$\begin{aligned} (\lambda y.\lambda z.\mathbf{K}\,z(y\,z))\mathbf{I} &\qquad \text{is a redex} \\ \mathbf{K}\,z &\qquad \text{is a redex} \end{aligned}$$

$$\begin{aligned} (\lambda y.\lambda z.\mathbf{K}\,z(y\,z))\mathbf{I} &\longrightarrow_\beta \lambda z.\mathbf{K}\,z(\mathbf{I}\,z) \\ &\equiv \lambda z.(\lambda p\,q.p)\,z(\mathbf{I}\,z) \\ &\longrightarrow_\beta \lambda z.(\lambda q.z)\,(\mathbf{I}\,z) \\ \textit{Call by Value} \qquad &\longrightarrow_\beta \lambda z.(\lambda q.z)z \\ &\longrightarrow_\beta \lambda z.z \end{aligned}$$

But also

$$\begin{aligned} (\lambda y.\lambda z.\mathbf{K}\,z(y\,z))\mathbf{I} &\equiv (\lambda y.\lambda z.(\lambda p\,q.p)\,z(y\,z))\mathbf{I} \\ &\longrightarrow_\beta (\lambda y.\lambda z.(\lambda q.z)(y\,z))\mathbf{I} \\ &\longrightarrow_\beta \lambda z.(\lambda q.z)\,(\mathbf{I}\,z) \\ \textit{Call by Name} \qquad &\longrightarrow_\beta \lambda z.z \end{aligned}$$

In the previous example we have seen that the term $\lambda z.(\lambda q.z)\,(\mathbf{I}\,z)$ can be reduced in several ways. *Call-by-name* is the ordinary β-reduction, where one can contract any β-redex. In *call-by-value*, one is only allowed to reduce $(\lambda x.M)N$ if N is a *value*, where a value is an abstraction term or a variable ([7]). So to reduce a term of the form $(\lambda x.M)((\lambda y.N)P)$ "call-by-value", we first have to contract $(\lambda y.N)P$. Call-by-value restricts the redexes that are allowed to be contracted, but it does not prescribe which is the next redex to contract. More restrictive variations of β-reduction are obtained by defining a *reduction strategy* which is a recipe that describes for every term which redex to contract. Well-known reduction strategies are *left-most outermost* or *right-most innermost*. To understand these notions it should be observed that redexes can be *contained in another*, e.g. in $(\lambda x.M)((\lambda y.N)P)$ or in $(\lambda x.(\lambda y.N)P)Q$, but they can also be *disjoint*, in which case there's always one to the left of the other. Other reduction strategies select a set of redexes and contract these simultaneously (a notion that should be defined first of course). For example, it is possible to define the simultaneous contraction of all redexes in a term, which is usually called a *complete development*. We don't go into the theory of reduction strategies or developments here, but refer to the literature [1].

We can also have *infinite reductions*. The simplest infinite reduction is the following *loop*:

$$\Omega \longrightarrow_\beta \Omega$$

A term that doesn't loop but whose reduction path contains infinitely many different terms is obtained by putting $\omega_3 := \lambda x.x\,x\,x$, $\Omega_3 := \omega_3\omega_3$. Then:

$$\Omega_3 \longrightarrow_\beta \omega_3\,\omega_3\,\omega_3 \longrightarrow_\beta \omega_3\,\omega_3\,\omega_3\,\omega_3 \longrightarrow_\beta \dots$$

The untyped λ-calculus was defined by Church [4] and proposed as a system to capture the notion of *mechanic computation*, for which Turing proposed the notion of Turing machine. An important property of the untyped λ-calculus is that it is *Turing complete*, which was proved by Turing in 1936, see [5]. The power of Λ lies in the fact that you can *solve recursive equations*.

A recursive equation is a question of the following kind:

- Is there a term M such that
$$M\,x =_\beta x\,M\,x?$$
- Is there a term M such that
$$M\,x =_\beta \mathsf{if}\ (\mathsf{Zero}\,x)\ \mathsf{then}\ 1\ \mathsf{else}\ \mathsf{Mult}\,x\,(M\,(\mathsf{Pred}\,x))?$$

So, we have two expressions on either side of the $=_\beta$ sign, both containing an unknown M and we want to know whether a solution for M exists.

The answer is: **yes**, if we can rewrite the equation to one of the form

$$M \quad =_\beta \quad \boxed{\dots M \dots} \tag{1}$$

Note that this is possible for the equations written above. For example the first equation is solved by a term M that satisfies $M =_\beta \lambda x.x\,M\,x$.

That we can solve equation of the form (1) is because every term in the λ-calculus has a *fixed point*. Even more: we have a *fixed point combinator*.

Definition 1.6.
- The term M is a *fixed point* of the term P if $P\,M =_\beta M$.
- The term $\mathbf{Y}$ is a *fixed point combinator* if for every term P, $\mathbf{Y}\,P$ is a fixed point of P, that is if
$$P\,(\mathbf{Y}\,P) =_\beta \mathbf{Y}\,P.$$

In the λ-calculus we have various fixed point combinators, of which the $\mathbf{Y}$-combinator is the most well-known one: $\mathbf{Y} := \lambda f.(\lambda x.f(x\,x))(\lambda x.f(x\,x))$.

Exercise 1.7. Verify that the $\mathbf{Y}$ as defined above is a fixed point combinator: $P\,(\mathbf{Y}\,P) =_\beta \mathbf{Y}\,P$ for every λ-term P.
Verify that $\Theta := (\lambda x\,y.y(x\,x\,y))(\lambda x\,y.y(x\,x\,y))$ is also a fixed point combinator that is even *reducing*: $\Theta\,P \twoheadrightarrow_\beta P\,(\Theta\,P)$ for every λ-term P.

The existence of fixed-points is the key to the power of the λ-calculus. But we also need natural numbers and booleans to be able to write programs. We have already seen the *Church numerals*:

$$c_n := \lambda f.\lambda x.f\,\underbrace{f(\ldots f(f}_{n \text{ times } f}\;x))$$

The successor is easy to define for these numerals: $\mathsf{Suc} := \lambda n.\lambda f\,x.f(n\,f\,x)$. Addition can also be defined quite easily, but if we are lazy we can also use the fixed-point combinator. We want to solve

$$\mathsf{Add}\,n\,m \quad = \quad \mathsf{if}\,(\mathsf{Zero}\,n)\,\mathsf{then}\,m\,\mathsf{else}\,\mathsf{Add}\,(\mathsf{Pred}\,n)\,m \tag{2}$$

where Pred is the predecessor function, Zero is a test for zero and $\mathsf{if}\ldots\mathsf{then}\ldots\mathsf{else}$ is a case distinction on booleans. The booleans can be defined by

$$\begin{aligned}\mathsf{true} &:= \mathbf{K} := \lambda x\,y.x\\ \mathsf{false} &:= \mathbf{K}_* := \lambda x\,y.y\\ \mathsf{if}\,b\,\mathsf{then}\,P\,\mathsf{else}\,Q &:= b\,P\,Q.\end{aligned}$$

Exercise 1.8.

1. Verify that the booleans behave as expected: $\mathsf{if}\,\mathsf{true}\,\mathsf{then}\,P\,\mathsf{else}\,Q =_\beta P$ and $\mathsf{if}\,\mathsf{false}\,\mathsf{then}\,P\,\mathsf{else}\,Q =_\beta Q$.
2. Define a test-for-zero Zero on the Church numerals: $\mathsf{Zero}\,c_0 =_\beta \mathsf{true}$ and $\mathsf{Zero}\,c_{n+1} =_\beta \mathsf{false}$. (Defining the predecessor is remarkably tricky!)

We can now solve (2) using the fixed point combinator by taking

$$\mathsf{Add} := \mathbf{Y}(\lambda a\,n\,m.\mathsf{if}\,(\mathsf{Zero}\,n)\,\mathsf{then}\,m\,\mathsf{else}\,a\,(\mathsf{Pred}\,n)\,m),$$

because if we write H for $\lambda a\,n\,m.\mathsf{if}\,(\mathsf{Zero}\,n)\,\mathsf{then}\,m\,\mathsf{else}\,a\,(\mathsf{Pred}\,n)\,m$, we see that

$$\begin{aligned}\mathsf{Add}\,n\,m &=_\beta H\,\mathsf{Add}\,n\,m\\ &=_\beta \mathsf{if}\,(\mathsf{Zero}\,n)\,\mathsf{then}\,m\,\mathsf{else}\,\mathsf{Add}\,(\mathsf{Pred}\,n)\,m)\end{aligned}$$

Exercise 1.9.

1. Give a term Add that represents addition that does not use the fixed point combinator (so Add should be a term in normal form.)
2. Give a term M such that for all terms N, we have $M\,N =_\beta N\,M\,N$.
3. Give a term F such that $F\,x =_\beta \mathsf{if}\,(\mathsf{Zero}\,x)\,\mathsf{then}\,1\,\mathsf{else}\,\mathsf{Mult}\,x\,(F\,(\mathsf{Pred}\,x))$. (So F represents the faculty function. You may assume that Mult represents multiplication and Pred represents predecessor.

Apart from the natural numbers and booleans, it is not difficult to find encodings of other data, like lists and trees.

Definition 1.10. A *λ-theory* is a set of equations between λ-terms that is closed under the following rules.

$$\frac{}{(\lambda x.M)N = M[x := N]}\,(\beta) \qquad \frac{M = N \qquad P = Q}{M\,P = N\,Q}\,(\text{app}) \qquad \frac{M = N}{\lambda x.M = \lambda x.N}\,(\xi)$$

$$\frac{}{M = M}\,(\text{refl}) \qquad \frac{M = N \qquad N = P}{M = P}\,(\text{trans}) \qquad \frac{M = N}{N = M}\,(\text{sym})$$

The theory λ_β is defined as the minimal λ-theory, so the equations derivable using only the six rules above. One can extend this theory by adding more equations, or more rules, for example $\lambda_{\beta\eta}$ is the theory one obtains by adding the η-rule:

$$\frac{}{\lambda x.M\,x = M}\,(\eta) \text{ if } x \notin \mathrm{FV}(M)$$

If E is a set of equations between λ-terms, $\mathrm{Th}(E)$ is the λ-theory that takes the equations of E as additional axioms and uses the rules of λ_β.

Note that $M = N$ in λ_β if and only if $M =_\beta N$: the minimal λ-theory is just the well-known β-equality. Later, when we have defined a notion of model, we will define the *theory of a model* as the set of equations that are true in that model. This is always a λ-theory and it is an interesting question what theory it is, or put differently: which new equations hold in the model? We can also start from the other side and think about adding equations to λ_β and see if there is a model in which these equations hold. An important question is which equations can be added "safely", in the sense that they don't make the theory inconsistent.

Definition 1.11. A λ-theory T is *inconsistent* in case $T = \{M = N \mid M, N \in \Lambda\}$, that is: all terms are equal in T. A λ-theory is *consistent* in case it is not inconsistent.

There are all kinds of equations one could add to λ_β, and wonder if the theory becomes inconsistent. Some examples:

1. $\mathbf{K} = \mathbf{K}_*$
2. $\mathbf{K} = \mathbf{I}$
3. $\mathbf{K} = \Omega$
4. $c_1 = \mathbf{I}$
5. $c_0 = c_1$
6. $c_n = c_m$ for some $n, m \in \mathbb{N}$ with $n \neq m$
7. $\{M = N \mid M$ and N don't have a normal form $\}$
8. Add a new constant δ to Λ and the equations $\delta\, M\, N = \mathsf{true}$ (for $M, N \in \Lambda$ with $M =_\beta N$) and $\delta\, M\, N = \mathsf{false}$ (for $M, N \in \Lambda$ with $M \neq_\beta N$).
9. Add a new constant ε to Λ and the equations $\varepsilon\,(M\,N) = M$ and $\varepsilon\, P = P$ in case P is not an application.

Some of these (sets of) equations make the theory inconsistent. This is shown by proving that $M = N$ holds if we add the equation(s), which is usually not difficult (but a nice puzzle). Proving consistency of a set E is harder and is usually done by defining a model in which all equations of E hold, but e.g. $\mathbf{K} = \mathbf{K}_*$ does not.

Example 1.12. The first two equations above yield an inconsistent λ-theory. (That is: $\mathrm{Th}(E)$ is inconsistent for these equations.)

1. Suppose $\mathbf{K} = \mathbf{K}_*$. Then we have, for all M, N,
$$M = \mathbf{K}\, M\, N = \mathbf{K}_*\, M\, N = N.$$
2. Suppose $\mathbf{K} = \mathbf{I}$. Then we have, for all M, N,
$$M = \mathbf{K}\, M\, N = \mathbf{I}\, M\, N = \mathbf{K}\,\mathbf{I}\,\mathbf{I}\, M\, N = \mathbf{I}\,\mathbf{K}\,\mathbf{I}\, M\, N = \mathbf{K}\,\mathbf{I}\, M\, N = \mathbf{I}\, N = N.$$

One motivation for studying the consistency of λ-theories is that one might want to add equations to λ-calculus, or add constants with new equations. Then we don't want these new equations to make the theory trivial (i.e. equate everything). Another reason is that we want to capture *undefinedness* in λ-calculus: when do we consider a term M to be "undefined". Of course, this is not a formal notion inside λ-calculus, but we have an intuitive grasp of what it should be: M is undefined in case we cannot use it in any way to produce meaningful output. Later in

Section 3 we will come to a somewhat more refined analysis of "undefined", but for now we can observe as a criterion that it should be safe to equate all undefined terms, because if we can't do anything with M and we can't do anything with N, it should be safe to add $M = N$. This makes that we have to conclude that "M does not have a normal form" is not a proper definition of M is undefined", because we have the following Lemma.

Lemma 1.13. Of the equations above, the following are inconsistent: (1), (2), (5), (6), (7), (8), (9). The following are consistent: (3), (4).

Proof. We have just seen the proof for (1) and (2). For the others: (5), (6), (7) and (8) are exercises and (3) is a consequence of deep technical result and can be found e.g. in [1]. For the consistency of (4): $c_1 = \lambda f.\lambda x.f\,x =_\eta \lambda f.f = \mathbf{I}$ and $\lambda_{\beta\eta}$ is certainly a consistent theory. For the inconsistency of (9), suppose we have added such an ε. Then $\mathbf{K} = \varepsilon\,\mathbf{K} = \varepsilon\,(\mathbf{I}\,\mathbf{K}) = \mathbf{I}$, so we have $\mathbf{K} = \mathbf{I}$ which is inconsistent. Note that any term ε that "can take apart an application" is inconsistent, no matter what it does when applied to a non-application, because, e.g. $\mathbf{K} = \varepsilon\,(\mathbf{K}\,\mathbf{I}) = \varepsilon\,(\mathbf{I}\,(\mathbf{K}\,\mathbf{I})) = \mathbf{I}$. $\square$

2 Models of untyped λ calculus

In this section we present models of untyped λ-calculus as they were defined by Scott [10], Plotkin [8] and Engeler [6]. In our presentation we follow [3], which gives a very broad overview of models of untyped λ-calculus and also shows how they have been used to prove properties about the λ-calculus itself. Another great source for models is of course [1].

2.1 What we need for a model

In a model we should in any case have a binary application operator, so one usually starts from an applicative structure.

Definition 2.1. An *applicative structure* is a tuple $\langle D, \cdot\rangle$ with $\cdot : D \times D \to D$. Given an applicative structure, there is a map F from D to $D \to D$ given by $F(d)(x) := d \cdot x$.

The binary operation is sometimes omitted and then one write $d\,x$ for $d \cdot x$. As in λ-calculus, omitted brackets associate to the left, so $d_1\,d_2\,d_3$ denotes $(d_1 \cdot d_2) \cdot d_3$.

There are many applicative structures, like $\langle \mathbb{N}, +\rangle$, $\langle \mathbb{R}, *\rangle$, $\langle \wp(X), \cup\rangle$, but only very few of them give rise to a model of λ-calculus. (As a matter of fact, applicative structures with a commutative operation do not.) To interpret λ-calculus, we need to be able to view a $d \in D$ as a function from D to D (which is done via F), but also we need to be able to view a function from D to D as an element of D, because we will need e.g. to view $\boldsymbol{\lambda} x \in D.x$ as an element of D. So we would like to have an inverse of F:

$$\begin{tikzcd} D \arrow[r, bend left, "F"] \arrow[r, phantom, "\simeq"] & (D \to D) \arrow[l, bend left, "G"] \end{tikzcd}$$

with $G \circ F = \mathrm{Id}_D$ and $F \circ G = \mathrm{Id}_{D \to D}$. Given an applicative structure with such maps F and G we can define an interpretation of untyped λ-calculus in D as in Figure 1.

The problem with the definition in Figure 1 is that there is no isomorphism between D and $D \to D$. (Unless of course $|D| = 1$, but then all λ-terms are interpreted as the same element, so the theory of the model is inconsistent.) So we have to replace $D \to D$ by a suitable subset of the full function space. This may well be possible, because we don't need all functions: we need at leas the one that are given by "left application of an element from D", the functions of the shape $\boldsymbol{\lambda} x \in D.d \cdot x$.

Letting Val = Var $\to D$, we define $[\![-]\!] : \Lambda \to \text{Val} \to D$ as follows. For $\rho \in \text{Var} \to D$ we define $[\![M]\!]_\rho$ by induction on M as follows.

$$\begin{array}{rcll} [\![x]\!]_\rho & := & \rho(x) & \\ [\![PQ]\!]_\rho & := & F([\![P]\!]_\rho)([\![Q]\!]_\rho) & (= [\![P]\!]_\rho \cdot [\![Q]\!]_\rho) \\ [\![\lambda x.P]\!]_\rho & := & G(\boldsymbol{\lambda} d \in D.[\![P]\!]_{\rho[x \mapsto d]}), & \end{array}$$

where $\rho[x \mapsto d]$ is ρ with the value for x updated to d.

Figure 1: Interpreting untyped λ-terms

Definition 2.2. Given an applicative structure $\langle D, \cdot \rangle$, we say that $f : D \to D$ is *representable* if there is a $d \in D$ with $f(x) = d \cdot x$ for all x. We define

$$R(D) := \{f : D \to D \mid f \text{ is representable}\}.$$

We see that $|R(D)| \leq |D|$, so these sets could be isomorphic.

Definition 2.3 (Model, first definition)**.** A *model of untyped λ-calculus* (also *λ-model*) is a tuple $\langle D, \cdot, G \rangle$ with $\langle D, \cdot \rangle$ an applicative structure and

1. $G : R(D) \to D$ with $F \circ G = \text{Id}_{R(D)}$ (where $F(d)(x) := d \cdot x$),
2. $[\![-]\!]$ of Figure 1 is well-defined.

It is not obvious that the two conditions in the definition can be satisfied, and therefore it is far from obvious that models exist at all. So this is a very preliminary definition. For the second requirement, the crucial point is to show that $\boldsymbol{\lambda} d \in D.[\![P]\!]_{\rho[x \mapsto d]}$ is in $R(D)$ for every P and ρ. This amounts to a kind of *combinatory completeness* of the model, stating that each term-with-free-variables over $\langle D, \cdot \rangle$ can be represented as a function (from D to D) that is an element of $R(D)$.

Remark 2.4. It should be noted that the notion of λ-model in Definition 2.3 is different from what is defined as a λ-model in e.g. [1]. In fact the notion above is closer to what is called a "syntactic λ-model" in [1]. As these are course notes, meant to introduce the ideas and intuitions behind models of untyped λ-calculus, and to actually construct one or two, we will not take the time and space to give further comparisons with the literature. See the references for further reading.

The first condition in the definition states that $R(D)$ is a so called "retract" of D and it implies that $|R(D)| \leq |D|$. This is crucial to ensure that the interpretation of the λ-calculus in a λ model is *sound*, that is: the β-equality between terms implies equality in the model. Or, phrased differently: the equational theory of a λ-model is a λ-theory.

Definition 2.5. For $\mathcal{M} := \langle D, \cdot, G \rangle$a λ-model, we define the *theory of* $\mathcal{M}$, notation Th($\mathcal{M}$) as follows.

$$\text{Th}(\mathcal{M}) := \{N = P \mid \mathcal{M} \models N = P\},$$

where $\mathcal{M} \models N = P$ denotes $\forall \rho : \text{Var} \to D\, ([\![N]\!]_\rho = [\![P]\!]_\rho)$.

Proposition 2.6. For $\mathcal{M} := \langle D, \cdot, G \rangle$ a λ-model, the interpretation $[\![-]\!]_\rho : \Lambda \to D$ is sound, that is, for all $N, P \in \Lambda$ and for all ρ we have

$$N =_\beta P \Longrightarrow [\![N]\!]_\rho = [\![P]\!]_\rho.$$

As a consequence, Th($\mathcal{M}$) is a λ-theory.

Proof. By induction on the derivation of $N =_\beta P$. The only interesting case to check is the β-rule. We have

$$\begin{array}{rcll}
[\![(\lambda x.N)P]\!]_\rho & = & F(G(\boldsymbol{\lambda} d \in D.[\![N]\!]_{\rho[x\mapsto d]}))([\![P]\!]_\rho) & \\
& = & (\boldsymbol{\lambda} d \in D.[\![N]\!]_{\rho[x\mapsto d]})([\![P]\!]_\rho) & \text{by } F \circ G = \mathrm{Id} \\
& = & [\![N]\!]_{\rho[x\mapsto [\![P]\!]_\rho]} & \\
& = & [\![N[x := P]]\!]_\rho d & \text{by Substitution Lemma 2.7.}
\end{array}$$

□

So, as always we need a Substitution Lemma, to show that "substitution and interpretation commute".

Lemma 2.7 (Substitution Lemma)**.** For $\mathcal{M} := \langle D, \cdot, G\rangle$ a λ-model, ρ a valuation, $N, P \in \Lambda$, we have

$$[\![N]\!]_{\rho[x\mapsto [\![P]\!]_\rho]} = [\![N[x := P]]\!]_\rho.$$

Proof. By induction on N. □

2.2 Complete Lattices

In the previous section we have given a first definition of model without constructing one and even without showing that they exist. Scott (see [10]) was the first to notice that the semantics of a formal system describing computation, be it a programming language, (partial) recursive functions or λ-calculus, should be given in terms of *ordered structures*, where the ordering represents the degree of *definedness* or the *information content* of the mathematical object. This gives rise to domain theory and cpos (complete partial orders) and has been developed further by Scott, Plotkin (see [8]) and many others. The simplest setting that is good enough for untyped λ-calculus is provided by *complete lattices*, which are a specific type of cpos.

Definition 2.8. A partial order $\langle D, \sqsubseteq\rangle$ is a *complete lattice* if all subsets have a least upperbound (lub). To be precise, we have $\forall X \subseteq D(\bigsqcup X$ exists), where $\bigsqcup X$ satisfies the following properties.

(lub$_1$) $\forall x \in X(x \sqsubseteq \bigsqcup X)$.

(lub$_2$) $\forall y \in D(\forall x \in X(x \sqsubseteq y)) \to \bigsqcup X \sqsubseteq y$.

In a partial order, lubs –if they exist– are unique , so we are justified to use the notation $\bigsqcup X$ for *the* least upperbound of X, which has as defining properties (lub$_1$) and (lub$_2$). A complete lattice is also a complete partial order (cpo), so all of the theory about cpos applies to complete lattices.

Example 2.9.
- The powerset is a complete lattice ordered by set-inclusion: For A a set, $\langle \wp(A), \subseteq\rangle$ is a complete lattice. If $Y \subseteq \wp(A)$, we define $\bigsqcup Y := \bigcup Y$, which is indeed the least upperbound of Y.
- Our usual flat domains are not complete lattices, but if we add a top element they become complete lattices. See $\mathbb{B}_\bot^\top$ and C in Figure 2.
- $\langle \mathbb{N}, \leq\rangle$ is not a complete lattice because $\mathbb{N}$ has no lub, but if we add a top element ω it is a complete lattice which is often called Ω. See Figure 2.

Proposition 2.10. In a complete lattice, every subset X has a *greatest lower bound*, the glb of X, denoted by $\bigsqcap X$. The defining properties for $\bigsqcap X$ are

(glb$_1$) $\forall x \in X(\bigsqcap X \sqsubseteq x)$.

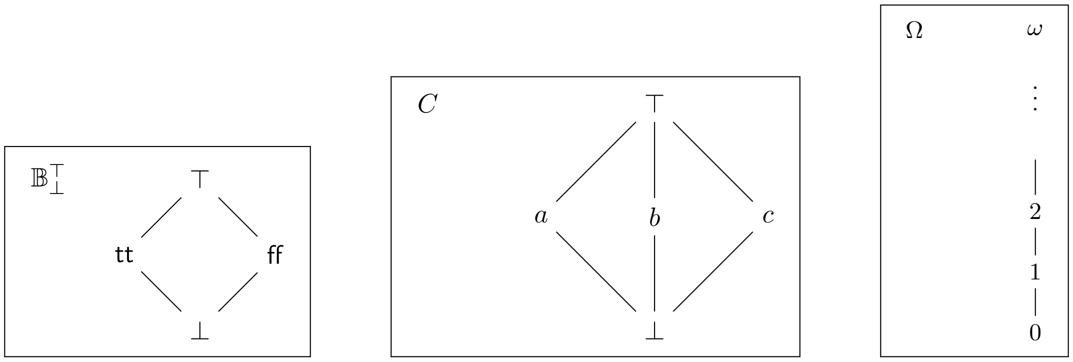


Figure 2: Some complete lattices

(glb$_2$) $\forall y \in D(\forall x \in X(y \sqsubseteq x)) \to y \sqsubseteq \bigsqcap X$.

Proof. Exercise. □

Like with least upper bounds, greatest lower bounds are unique in a partial order (if they exist), so the notation $\bigsqcap X$ denotes the unique glb of X. Complete lattices are lattices and we can define the lub of two elements $d, e \in D$, also called the *join* of d and e, by $d \sqcup e := \bigsqcup\{d, e\}$. Similarly we can define the *meet* of d and e as the glb, by $d \sqcap e := \bigsqcap\{d, e\}$. Their defining properties are

$$\begin{array}{rcl@{\qquad\qquad}rcl}
x & \sqsubseteq & x \sqcup y & x \sqcap y & \sqsubseteq & x \\
y & \sqsubseteq & x \sqcup y & x \sqcap y & \sqsubseteq & y \\
x \sqsubseteq z \wedge y \sqsubseteq z & \Longrightarrow & x \sqcup y \sqsubseteq z & z \sqsubseteq x \wedge z \sqsubseteq y & \Longrightarrow & z \sqsubseteq x \sqcap y
\end{array}$$

It should be noted that complete lattices are not necessarily *distributive*. Distributivity says that $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Distributivity does not hold in the example complete lattice (C) of Figure 2. There $a \sqcap (b \sqcup c) = a \neq \perp = (a \sqcap b) \sqcup (a \sqcap c)$. The *absorption law* holds in complete lattices: $x \sqcup (x \sqcap y) = x$. (Exercise.) Furthermore, complete lattices have a top element ($\bigsqcup D$) and a bottom element ($\bigsqcup \emptyset$).

Definition 2.11. For $\langle D, \sqsubseteq\rangle$ a complete lattice, we say that $X \subseteq D$ is *directed* if $X \neq \emptyset$ and $\forall x, y \in X\, \exists z \in X\, (x \sqsubseteq z \wedge y \sqsubseteq z)$.

For $\langle E, \sqsubseteq\rangle$ also a complete lattice and a function $f : D \to E$, we say that

1. f is *monotone* if $\forall x, y \in D\, (x \sqsubseteq y \to f(x) \sqsubseteq f(y))$.
2. f is *continuous* if it *preserves lubs of directed sets*: for all $X \subseteq D$ with X directed, we have $f(\bigsqcup X) = \bigsqcup f(X)$.
   (Here, $f(X) = \{f(x) \mid x \in X\}$.)

Proposition 2.12. For complete lattices, continuity implies monotonicity but not the other way around.

Proof. Let $f : D \to E$ be continuous and suppose $x \leq y$. Then $\{x, y\}$ is a directed set, so we have $f(y) = f(\bigsqcup\{x, y\}) = \bigsqcup\{f(x), f(y)\}$, so $f(x) \sqsubseteq f(y)$.

A counterexample for the reverse direction is given by considering $f : \Omega \to \Omega$ (see Figure 2) with $f(n) = 0$ (for $n \in \mathbb{N}$) and $f(\omega) = \omega$. This f is monotone, but not continuous. □

Much of the theory for cpos can be restated (and proved) directly for complete lattices. Some of the results apply directly, as a complete lattice is also a cpo, while for other results the proof has to be re-checked to see if it also works with the slightly adapted notions. We therefore state the following without proofs.

Proposition 2.13. If $\langle D, \sqsubseteq_D\rangle$ and $\langle E, \sqsubseteq_E\rangle$ are complete lattices then $\langle D\times E, \sqsubseteq_{D\times E}\rangle$ and $\langle [D \to E], \sqsubseteq_{D\to E}\rangle$ are complete lattices where

- $D \times E := \{(d, e) \mid d \in D, e \in E\}$, $(d, e) \sqsubseteq_{D\times E} (d', e')$ if $d \sqsubseteq_D d'$ and $e \sqsubseteq_E e'$.
- $D \to E := \{f : D \to E \mid f$ is continuous$\}$, $f \sqsubseteq_{D\to E} g$ if $\forall d \in D. f(d) \sqsubseteq_E g(d)$.

Proposition 2.14. For $\langle D, \sqsubseteq_D\rangle$, $\langle E, \sqsubseteq_E\rangle$ and $\langle F, \sqsubseteq_F\rangle$ complete lattices we have the following.

- $\pi_1 : D \times E \to D$ and $\pi_2 : D \times E \to E$ are continuous and for $f_1 : F \to D$ and $f_2 : F \to E$ continuous, the *pairing function* $<f_1, f_2> : F \to D \times E$ is continuous.
- $f : D \times E \to F$ is continuous (monotone) if and only f is continuous (monotone) in each of its arguments.
- If $f : D \to E$ and $g : E \to F$ are continuous (monotone), then $g \circ f : D \to F$ is continuous (monotone).
- $\mathrm{Eval} : ([D \to E] \times D) \to E$ defined by $\mathrm{Eval}(f, d) := f(d)$ is continuous.
- $\mathrm{Curr} : [D \times E \to F] \to (D \to (E \to F))$ defined by $\mathrm{Curr}(f)(d)(e) := f(d, e)$ is continuous. (That is: for f and d given, $\mathrm{Curr}(f)(d) : E \to F$ is continuous, for f given, $\mathrm{Curr}(f) : D \to (E \to F)$ is continuous, and Curr itself is continuous.)

Proposition 2.15 (Least fixed point)**.** For $\langle D, \sqsubseteq\rangle$ a complete lattice and $f : D \to D$ continuous, f has a *least fixed point* $\mathsf{lfp}(f)$ given by

$$\mathsf{lfp} := \bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}\}.$$

Like for cpos, it can also be shown that lfp is a continuous function from $[D \to D]$ to D.

Using the framework of complete lattices we can get a model of untyped λ-calculus if we can find a complete lattice D with $D \simeq [D \to D]$. As a matter of fact, we can do with a little less.

Definition 2.16. A complete lattice $\langle D, \sqsubseteq\rangle$ is called *reflexive* if $[D \to D]$ is a *retract* of D (notation $[D \to D] < D$), that is: there are continuous maps $F : D \to [D \to D]$ and $G : [D \to D] \to D$ such that $F \circ G = \mathrm{Id}_{[D\to D]}$.

We could take a more general (abstract) perspective and talk about reflexive objects in cartesian closed categories, but the plan is to present models of untyped λ calculus using only domain theory, so we will not use categorical terminology. See e.g. [1] for a categorical description. A reflexive complete lattice now gives rise to a model of the untyped λ-calculus.

Theorem 2.17. A reflexive complete lattice yields a model of the untyped λ-calculus as defined in Definition 2.3.

Proof. So we have $\mathcal{M} = \langle D, \sqsubseteq\rangle$ with continuous maps $F : D \to [D \to D]$ and $G : [D \to D] \to D$ such that $F \circ G = \mathrm{Id}_{[D\to D]}$. Of Definition 2.3 we only have to check the second item: that $[\![-]\!]$ of Figure 1 is well-defined. This follows from the property that $\boldsymbol{\lambda} d \in D.[\![P]\!]_{\rho[x\mapsto d]}$ as defined in Figure 1 is continuous. That is the statement of Lemma 2.18 below. □

Lemma 2.18. Let $\mathcal{M} = \langle D, \sqsubseteq\rangle$ be a complete lattice with continuous maps $F : D \to [D \to D]$ and $G : [D \to D] \to D$ such that $F \circ G = \mathrm{Id}_{[D\to D]}$.

The interpretation $[\![-]\!]$ of Figure 1 is well-defined and for every $P \in \Lambda$ and $\rho \in \mathrm{Val}$, the function $\boldsymbol{\lambda} d \in D.[\![P]\!]_{\rho[x\mapsto d]}$ is continuous.

Proof. We prove the two properties simultaneously by induction on P.

- $P = x$. Then $\boldsymbol{\lambda} d \in D.[\![x]\!]_{\rho[x\mapsto d]}$ is just $\boldsymbol{\lambda} d \in D.d$ which is continuous.
- $P = y$. Then $\boldsymbol{\lambda} d \in D.[\![y]\!]_{\rho[x\mapsto d]}$ is just $\boldsymbol{\lambda} d \in D.\rho(y)$, a constant function, which is continuous.

- $P = NQ$. Then $[\![NQ]\!]_\rho = F([\![N]\!]_\rho)([\![Q]\!]_\rho)$ is well-defined. Furthermore, $\boldsymbol{\lambda} d \in D.F([\![N]\!]_{\rho[x\mapsto d]}) : D \to [D \to D]$ is continuous, because it is $F \circ \boldsymbol{\lambda} d \in D.[\![N]\!]_{\rho[x\mapsto d]}$, a composition of continuous functions. So $\boldsymbol{\lambda} d \in D.F([\![N]\!]_{\rho[x\mapsto d]})([\![Q]\!]_{\rho[x\mapsto d]})$ is continuous, because it is $\mathrm{Eval} \circ <\boldsymbol{\lambda} d \in D.F([\![N]\!]_{\rho[x\mapsto d]}), \boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[x\mapsto d]}>$, which is a composition of continuous functions. (See Proposition 2.14.)
- $P = \lambda y.Q$. Then $[\![\lambda y.Q]\!]_\rho = G(\boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[y\mapsto d]})$ is well-defined, because we know (by induction) that $\boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[y\mapsto d]}$ is continuous. By induction, we also know that, for every $d \in D$, $\boldsymbol{\lambda} e \in D.[\![Q]\!]_{\rho[y\mapsto d, x\mapsto e]}$ is continuous. So $\boldsymbol{\lambda}(d,e) \in D \times D.[\![Q]\!]_{\rho[y\mapsto d, x\mapsto e]}$ is continuous (because it is continuous in each argument separately). So by Currying: $\boldsymbol{\lambda} e \in D.\boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[y\mapsto d, x\mapsto e]}$ is continuous. We can conclude that $\boldsymbol{\lambda} e \in D.G(\boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[y\mapsto d, x\mapsto e]})$ is continuous, because it is just $G \circ \boldsymbol{\lambda} e \in D.\boldsymbol{\lambda} d \in D.[\![Q]\!]_{\rho[y\mapsto d, x\mapsto e]}$.

□

2.3 Graph models

The idea is now to consider a complete lattice of the form $\langle \wp(D), \subseteq\rangle$ for some set D and consider the continuous functions from $\wp(D)$ to $\wp(D)$. We are looking for a set D such that $[\wp(D) \to \wp(D)]$ is a retract of $\wp(D)$, that is: we have $F : \wp(D) \to [\wp(D) \to \wp(D)]$ and $G : [\wp(D) \to \wp(D)] \to \wp(D)$ such that $F \circ G = \mathrm{Id}_{[\wp(D)\to\wp(D)]}$. We can identify a function f with its "graph", the set of pairs $\{(x, f(x) \mid x \in D\}$ and for continuous function between power-sets these graphs can be represented quite concisely as a so called "trace".

Definition 2.19. For $f : \wp(D) \to \wp(D)$ continuous, we define the *trace of* f, $\mathrm{Tr}(f)$, as follows.

$$\mathrm{Tr}(f) := \{(\alpha, a) \mid \alpha \text{ finite and } a \in f(\alpha)\}.$$

To fix notation, we will use $\alpha, \beta, \gamma, \ldots$ for finite subsets and $a, b, c, \ldots$ for elements. To express that a subset α of X is finite, we write $\alpha \subseteq_{\mathrm{fin}} X$. We denote the set of *finite subsets* of a set X by $\wp_{\mathrm{fin}}(X)$.

Clearly, the trace of $f : \wp(D) \to \wp(D)$ gives partial information about f: for finite sets α it tells us what $f(\alpha)$ is by telling us which are the elements a such that $a \in f(\alpha)$. As a matter of fact, for continuous f, the trace fully determines f.

Lemma 2.20. A continuous function $f : \wp(D) \to \wp(D)$ is completely determined by its trace $\mathrm{Tr}(f)$. More precisely, we have, for $X \in \wp(D)$,

$$f(X) = \{a \mid \exists \alpha \subseteq_{\mathrm{fin}} X\, ((\alpha, a) \in \mathrm{Tr}(f))\}.$$

Furthermore, Tr is a continuous injective function from $[\wp(D) \to \wp(D)]$ to $\wp(\wp_{\mathrm{fin}}(D) \times D)$.

Proof. The proof of the first part relies on continuity of f. It is an exercise. For the second part, we note that (exercise)

$$f \sqsubseteq g \iff \mathrm{Tr}(f) \subseteq \mathrm{Tr}(g),$$

so Tr is injective. For $Z \subseteq [\wp(D) \to \wp(D)]$ a directed set, we have

$$\begin{aligned}
\mathrm{Tr}(\bigsqcup Z) &= \{(\alpha, a) \mid \alpha \subseteq_{\mathrm{fin}} D \wedge a \in \bigsqcup Z(\alpha)\}\\
&= \{(\alpha, a) \mid \alpha \subseteq_{\mathrm{fin}} D \wedge a \in \bigcup_{f\in Z} f(\alpha)\}\\
&= \{(\alpha, a) \mid \alpha \subseteq_{\mathrm{fin}} D \wedge \exists f \in Z (a \in f(\alpha))\}\\
&= \bigcup_{f\in Z} \{(\alpha, a) \mid \alpha \subseteq_{\mathrm{fin}} D \wedge a \in f(\alpha)\}\\
&= \bigcup_{f\in Z} \mathrm{Tr}(f)
\end{aligned}$$

□

$$\begin{tikzcd}
\wp(D) \arrow[rr, bend left, "F"] & \simeq & {[\wp(D) \to \wp(D)]} \arrow[dl, bend left, "\mathrm{Tr}"] \\
& \wp(\wp_{\mathrm{fin}}(D) \times D) \arrow[ul, bend left, "\mathsf{i}"] &
\end{tikzcd}$$

Figure 3: Graph model: the general picture

Coming back to the diagram in the beginning of Section 2.1, immediately after Definition 2.1, we can observe that we have made some progress, because we have a continuous injection Tr from $[\wp(D) \to \wp(D)]$ to $\wp(\wp_{\mathrm{fin}}(D) \times D)$. It now suffices to construct a specific D and an injection i from $\wp(\wp_{\mathrm{fin}}(D) \times D)$ to $\wp(D)$. The situation is summarized in Figure 3.

Of course, we also have to define the inverse map, F, but that is simpler.

Definition 2.21. Given the continuous injection $\mathrm{Tr} : [\wp(D) \to \wp(D)] \to \wp(\wp_{\mathrm{fin}}(D) \times D)$ of Definition 2.19, if we have an injection i from $\wp(\wp_{\mathrm{fin}}(D) \times D)$ to $\wp(D)$ we can define $F : \wp(D) \to [\wp(D) \to \wp(D)]$ as follows. For $X, Y \subseteq D$ we define

$$F(X)(Y) := \{a \mid \exists \alpha \subseteq_{\mathrm{fin}} Y (\mathsf{i}(\alpha, a) \in X)\}.$$

So, we view X as a function from $\wp(D)$ to $\wp(D)$ by looking at the (codes of) pairs (α, a) it contains. Each such pair gives partial information about the function, because it says that for α as input, we have that a is an element of the output. So if we apply this function to Y we need to look which a occur in the output of X for some input $\alpha \subseteq_{\mathrm{fin}} Y$.

Lemma 2.22. For the function F, defined in Definition 2.21, we have that $F(X)$ is continuous for every $X \subseteq D$ and F is continuous.

Proof. The proof is by writing out the definitions and some set-theoretic reasoning. To give the idea we prove that for $P, Q \subseteq \wp(D)$, directed, we have $F(\bigcup P)(\bigcup Q) = \bigcup_{X \in P, Y \in Q} F(X)(Y)$. So let $P, Q \subseteq \wp(D)$.

$$\begin{aligned}
F(\bigcup P)(\bigcup Q) &= \{a \mid \exists \alpha \subseteq_{\mathrm{fin}} \bigcup Q\ (\mathsf{i}(\alpha, a) \in \bigcup P)\} \\
&\overset{(*)}{=} \{a \mid \exists X \in P\, \exists Y \in Q\, \exists \alpha \subseteq_{\mathrm{fin}} Y (\mathsf{i}(\alpha, a) \in X)\} \\
&= \bigcup_{X \in P, Y \in Q} \{a \mid \exists \alpha \subseteq_{\mathrm{fin}} Y (\mathsf{i}(\alpha, a) \in X)\} \\
&= \bigcup_{X \in P, Y \in Q} F(X)(Y).
\end{aligned}$$

Here, $(*)$ uses the fact that if a finite set α is in $\bigcup Q$, then each element a_i of α is in some $Y_i \in Q$, so –because Q is directed– there is an $X \in Q$ that contains all a_i. □

We now collect all this together in the notion of a graph model.

Definition 2.23. A *graph model* is a pair $\langle D, \mathsf{i}\rangle$, where $\mathsf{i} : \wp(\wp_{\mathrm{fin}}(D) \times D) \to \wp(D)$ is injective.

A graph model yields a λ-model (as in Definition 2.3) if we define $G := \mathsf{i} \circ \mathrm{Tr} : [\wp(D) \to \wp(D)] \to \wp(D)$ (Definition 2.19 and Figure 3) and $F : \wp(D) \to [\wp(D) \to \wp(D)]$ (Definition 2.21) as follows.

$$\begin{aligned}
G &:= \{\mathsf{i}(\alpha, a) \mid \alpha \subseteq_{\mathrm{fin}} D \wedge a \in f(\alpha)\}, \\
F(X)(Y) &:= \{a \mid \exists \alpha \subseteq_{\mathrm{fin}} Y (\mathsf{i}(\alpha, a) \in X)\}.
\end{aligned}$$

A simple way of defining the injection i (it is actually a bijection) is by taking $D = \mathbb{N}$ and using well-known bijections between $\mathbb{N}$ and finite subsets of $\mathbb{N}$ and between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$. If we write finite sets as ordered sequences $\{n_1, \ldots, n_k\}$ we can encode them as natural numbers by $e(\{n_1, \ldots, n_k\}) = \Sigma_{i=1}^{k} 2^{n_i}$, and $e(\emptyset) = 0$. The inverse of e is defined by writing n as a binary number and then "reading off" from this sequence, in reverse order, which number is in the set. (For example, $11 = 2^3 + 2^1 + 2^0$, giving 1011, producing the set $\{0, 1, 3\}$.) Then we can define, for $X \subseteq \wp_{\text{fin}}(\mathbb{N}) \times \mathbb{N}$,

$$\mathsf{i}(X) := \{<e(\alpha), a> \mid (\alpha, a) \in X\}$$

where $<n, m> := \frac{(n+m)(n+m+1)}{2} + n$ is the well-known bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$. This amounts to the model P_ω, independently defined by Plotkin [8] and Scott (see [9], but the definition occurs in earlier notes by Scott).

Definition 2.24. The model P_ω of Plotkin and Scott is the graph-model defined by taking $D = \mathbb{N}$ and $\mathsf{i} : \wp(\wp_{\text{fin}}(\mathbb{N}) \times \mathbb{N}) \to \wp(\mathbb{N})$ as just defined.

The model P_ω is about the first concrete model of untyped λ-calculus that has been defined. Scott had already defined other versions, now known as D_∞, but P_ω seems to be the first that appeared in print. The theory of P_ω, $\text{Th}(P_\omega)$ is interesting to study, as it will equate more terms than just the β-equal one, so we can observe which equalities can be safely added to λ-calculus without making the theory inconsistent. It might be that $\text{Th}(P_\omega)$ is sensitive to the precise injection from $\wp(\wp_{\text{fin}}(\mathbb{N}) \times \mathbb{N})$ to $\wp(\mathbb{N})$ that we choose, so the theory of P_ω is not necessarily the easiest to study. We now introduce *Engeler's model*, often called D_A, but we will call it $\mathcal{E}$, following [3] (which also gives an overview of other models and their theories). This model has been defined by Engeler [6], based on ideas of Scott [10].

The idea is to construct D in such a way that i is just the identity. This avoids the need for computing with actual encodings when we want to determine the interpretation of a term, or if we want to determine if two terms have the same interpretation. So we should let D consist of pairs (α, a) where $\alpha \subseteq_{\text{fin}} D$ and $a \in D$.

Definition 2.25. Given a nonempty set A, we define D_A by induction using the following rules.

$$\frac{a \in A}{a \in D_A} \qquad\qquad \frac{\alpha \subseteq_{\text{fin}} D_A \quad a \in D_A}{(\alpha, a) \in D_A}$$

An alternative is to define D_i for $i \in \mathbb{N}$ by $D_0 := A$ and $D_{i+1} := D_i \cup \{(\alpha, a) \mid \alpha \subseteq_{\text{fin}} D_i, a \in D_i\}$. Then define $D_A := \bigcup_{i \in \mathbb{N}} D_i$.

The *rank* of an element $d \in D_A$, $\mathsf{rk}(d)$, is the smallest i such that $d \in D_i$. The rank of $\alpha \subseteq_{\text{fin}} D_A$ is the maximum of the ranks of the elements of α, and 0 if $\alpha = \emptyset$.

An alternative way of defining the rank is

$$\begin{aligned} \mathsf{rk}(a) &:= 0 \text{ for } a \in A, \\ \mathsf{rk}((\alpha, a)) &:= 1 + \max(\mathsf{rk}(a), \max\{\mathsf{rk}(b) \mid b \in \alpha\}). \end{aligned}$$

Definition 2.26. The model $\mathcal{E}$ of Engeler is the graph-model defined by taking $D = D_A$ and $\mathsf{i} : \wp(\wp_{\text{fin}}(D_A) \times D_A) \to D_A$ the identity.

3 The theory of $\mathcal{E}$

We will now take a closer look into the theory of the Engeler model, $\text{Th}(\mathcal{E})$. It is one of the few theories of models that is very well described and understood. It turns out that $\text{Th}(\mathcal{E}) = \text{Th}(P_\omega)$. We can easily compute the interpretations of some λ-terms if we just follow the definition that we concretize in Figure 4.

Example 3.1. We give some interpretations of λ-terms in $\mathcal{E}$.

For $\rho \in \text{Var} \to \wp(D_A)$, the interpretation $[\![M]\!]_\rho$ in $\wp(D_A)$ is inductively defined as follows.

$$\begin{array}{rcl}
[\![x]\!]_\rho & := & \rho(x) \\
[\![PQ]\!]_\rho & := & F([\![P]\!]_\rho)([\![Q]\!]_\rho) \\
 & = & \{a \mid \exists \alpha \subseteq_{\text{fin}} [\![Q]\!]_\rho\, ((\alpha, a) \in [\![P]\!]_\rho)\} \\
[\![\lambda x.P]\!]_\rho & := & G(\boldsymbol{\lambda} d \in D.[\![P]\!]_{\rho[x \mapsto d]}) \\
 & = & \{(\alpha, a) \mid \alpha \subseteq_{\text{fin}} D \wedge a \in [\![P]\!]_{\rho[x\mapsto \alpha]}\}.
\end{array}$$

Figure 4: Interpreting untyped λ-terms in $\mathcal{E}$

1. $[\![\mathbf{I}]\!]_\rho = \{(\alpha, a) \mid a \in \alpha\}$.
2. $[\![\mathbf{K}]\!]_\rho = \{(\alpha, (\beta, a)) \mid a \in \alpha\}$.
3. $[\![\mathbf{II}]\!]_\rho = \{b \mid \exists \beta \subseteq_{\text{fin}} [\![\mathbf{I}]\!]_\rho\, ((\beta, b) \in [\![\mathbf{I}]\!]_\rho)\} = \{b \mid \exists \beta \subseteq_{\text{fin}} [\![\mathbf{I}]\!]_\rho\, (b \in \beta)\} = \{b \mid b \in [\![\mathbf{I}]\!]_\rho\} = [\![\mathbf{I}]\!]_\rho$.
4. $[\![\lambda x_1 \ldots . x_n.M]\!]_\rho = \{(\beta_1, (\ldots(\beta_n, b)\ldots)) \mid b \in [\![M]\!]_{\rho[x_1 \mapsto \beta_1, \ldots, x_n \mapsto \beta_n]}\}$.
5. $[\![\Omega]\!]_\rho = \emptyset$. This is not so easy to prove and left as an exercise. It uses the fact that D_A is inductively defined, and so every element in D_A has a rank.

In Section 1 we have already touched upon the question of what makes a λ-term "undefined" and we pointed out that –no matter what we choose as definition– it should be the case that we can equate all undefined terms and arrive at a consistent theory. So "not having a normal form" is not a good definition of "undefined" in λ-calculus, as the theory that equates all terms that don't have a normal form is inconsistent.

It turns out that "not having a *head-normal form*" is a better definition of "undefined", as this captures much better when a term cannot be used in any way. We will define this notion and show that Th($\mathcal{E}$) equates all terms that don't have head-normal form. We already know that Th($\mathcal{E}$) is consistent (because, e.g. $[\![\mathbf{I}]\!]_\rho \neq [\![\mathbf{K}]\!]_\rho$). The theory of $\mathcal{E}$ is precisely charachterized by *Böhm tree equality*, a notion we will also define, and we will indicate how the proof proceeds of

$$\mathcal{E} \models M = N \Leftrightarrow \mathsf{BT}(M) = \mathsf{BT}(N).$$

Lemma 3.2. Every λ-term M is of one of the following two forms

1. $M \equiv \lambda x_1 \ldots . x_n.y\, P_1 \, \ldots \, P_k$ with $n \geq 0$, $k \geq 0$.
2. $M \equiv \lambda x_1 \ldots . x_n.(\lambda y.N)\, P_1 \, \ldots \, P_k$ with $n \geq 0$, $k \geq 1$.

Proof. By induction on M.

- $M \equiv y$, a variable is of the first form
- $M \equiv \lambda x.M'$, then, if M' is of the first form, then M is as well, and if M' is of the second form, then M is as well.
- $M \equiv M'\, M''$, then, if M' is a variable, M is of the first form, if M' is a λ-abstraction, then M is of the second form, if M' is an application, then M is of the first form in case M' is of the first form with $k = 0$ and otherwise it is of the second form.

□

The two forms in the Lemma do not overlap, so any λ-term is of either one of the two forms.

Definition 3.3. If M is of the form $M \equiv \lambda x_1.\dots x_n.y\, P_1\, \dots\, P_k$ (with $n \geq 0$, $k \geq 0$), we say that M is in *head-normal form*, and has *head variable* y.

If M is of the form $M \equiv \lambda x_1.\dots x_n.(\lambda y.N)\, P_1\, \dots\, P_k$ (with $n \geq 0$, $k \geq 1$), we say that $(\lambda y.N)\, P_1$ is the *head-redex* of M.

If $Q =_\beta M$ and M is in head normal form, we say that M *is a head-normal form of* Q, or Q *has head-normal form* M, often abbreviated to the hnf of M.

Example 3.4.

1. $\mathbf{I}$ and $\mathbf{K}$ are head-normal forms and in general, every normal form is a head-normal form.
2. $\Omega := (\lambda x.x\, x)\, (\lambda x.x\, x)$ is not in head-normal form, and it has no head-normal form. Similarly for $\Omega_3 := (\lambda x.x\, x\, x)\, (\lambda x.x\, x\, x)$.
3. $\mathbf{Y} := \lambda f.(\lambda x.f(x\, x))(\lambda x.f(x\, x))$ is not in head-normal form, but it has a head-normal form $\lambda f.f\, ((\lambda x.f(x\, x))(\lambda x.f(x\, x)))$.
4. $\mathbf{K}\, \mathbf{I}\, \mathbf{I}$ is not in head-normal form, but it has a head-normal form $\mathbf{I}$.
5. $M := \mathbf{S}\, \mathbf{I}\, \mathbf{I}$ is not in head-normal form, but $\lambda z.z\, (\mathbf{I}\, z)$ is a head-normal form of M and so is $\lambda z.z\, z$.

The head-normal form of a term (if it exists) is not unique, as the last example shows. It is in fact somewhat unique, in the sense that its structure is fixed, as the following Lemma states.

Lemma 3.5. If M has head-normal forms $\lambda x_1.\dots x_n.y\, P_1\, \dots\, P_k$ and $\lambda x_1.\dots x_m.z\, Q_1\, \dots\, Q_\ell$, then

- $m = n$
- $y = z$
- $k = \ell$ and for all $i \leq k$ we have $P_i =_\beta Q_i$.

Proof. Using the Church-Rosser Theorem: if $\lambda x_1.\dots x_n.y\, P_1\, \dots\, P_k =_\beta \lambda x_1.\dots x_m.z\, Q_1\, \dots\, Q_\ell$, then these terms have a common reduct, but reduction does not affect the number of leading abstractions, the head variable or the number of arguments of the head-variable. □

Proposition 3.6. For M a closed term: M has a head-normal form if and only if M is *solvable*, that is, there are terms $Q_1, \dots, Q_k$ (for some k) such that $M\, Q_1 \cdots Q_k =_\beta \mathbf{I}$.

Proof. From left to right: If $M =_\beta \lambda x_1.\dots x_n.y\, P_1\, \dots\, P_k$ and M is closed, then $y = x_i$ for some i, and we must have $n \geq 1$. So if we apply M to n arguments $Q_1, \dots, Q_n$ where we take $Q_i = \lambda z_1.\dots z_k.\mathbf{I}$ then $M\, Q_1, \dots, Q_n =_\beta \mathbf{I}$. From right to left, see [1]. □

The terminology "solvable" emphasizes that a term that has a hnf is meaningful and hence should never be considered "undefined". The terms that we should consider as "undefined" are the terms that we cannot "make use of" in any way, i.e. the terms M that we cannot produce $\mathbf{I}$ from by giving arguments. We will show that in $\mathcal{E}$ all unsolvable terms are interpreted as $\perp$ (which is $\emptyset$ in $\mathcal{E}$), and therefore they are all equated. More precisely, $\mathcal{E}$ equates terms that have the same *Böhm tree*.

Definition 3.7. For M a λ-term, the *Böhm tree* of M, notation $\mathsf{BT}(M)$ is defined as follows.

- If M has no hnf, $\mathsf{BT}(M) = \cdot$, an unlabeled node represented by a black dot.
- If $M =_\beta \lambda x_1.\dots x_n.y\, P_1\, \dots\, P_k$, then $\mathsf{BT}(M)$ is the tree

$$\begin{tikzcd}
 & \lambda x_1.\dots x_n.\, y \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
\mathsf{BT}(P_1) & \cdots & \mathsf{BT}(P_k)
\end{tikzcd}$$

a tree with top node labelled $\lambda x_1.\dots x_n.y$, and sub-trees $\mathsf{BT}(P_1), \mathsf{BT}(P_2), \dots, \mathsf{BT}(P_k)$.

Böhm trees are usually drawn as if the top node has label y, with the edges to the sub-tree starting from y and the part $\lambda x_1 . \dots x_n .$ acting as a affix to the node, so we will adhere to that habit. It should be noted that the definition of Böhm tree is not an inductive definition, but a co-inductive one. $\mathsf{BT}(M)$ can be infinite and to compute the BT of $\lambda x_1 . \dots x_n . y\, P_1 \, \dots \, P_k$ we will have to compute the hnf of P_i, which may be a term larger than M.

Example 3.8. We give some examples of Böm trees.

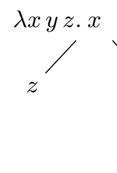
1. $\mathsf{BT}(\mathbf{I})$ and $\mathsf{BT}(\mathbf{K})$ are trees with only a top node labelled by the full term:

$$\mathsf{BT}(\mathbf{I}) = \quad \lambda x.\, x \qquad\qquad \mathsf{BT}(\mathbf{K}) = \quad \lambda x\, y.\, x$$

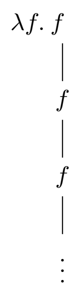
2. $\mathsf{BT}(\Omega)$, $\mathsf{BT}(\mathbf{S})$, and $\mathsf{BT}(\mathbf{Y})$ are as follows.

$$\mathsf{BT}(\Omega) = \quad \cdot$$

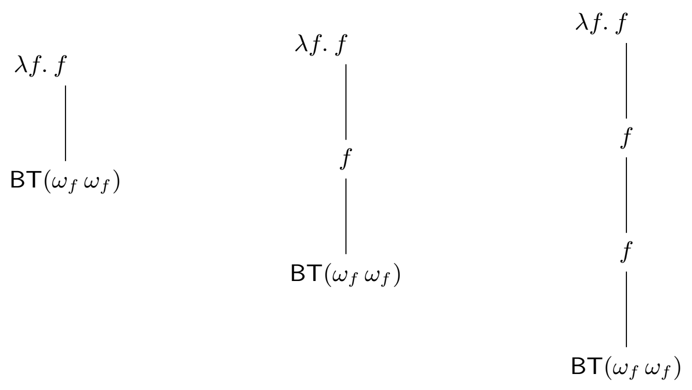
$\mathsf{BT}(\mathbf{S}) =$



$\mathsf{BT}(\mathbf{Y}) =$



The hnf of $\mathbf{Y}$ is $\lambda f . f\,((\lambda x . f(x\,x))(\lambda x . f(x\,x)))$ and if we write ω_f for $\lambda x . f(x\,x)$, we see that $\mathsf{BT}(\mathbf{Y})$ is the infinite tree generated as follows.



To generate the Böhm tree of a term M, one has to determine the hnf of M, say N and then determine the hnfs of sub-terms of N etcetera. As hnfs are not unique, this may at first glance seem like a process that depends on the actual hnf of M that we happen to pick. However, that is not the case: hnfs are not unique, but they are "unique enough", as we have seen in Lemma 3.5. We state this in the following Lemma.

Lemma 3.9. The notion of Böhm tree of a term M, $\mathsf{BT}(M)$ as defined in Definition 3.7 is well-defined and we have

$$M =_\beta N \Longrightarrow \mathsf{BT}(M) = \mathsf{BT}(N).$$

Proof. The statement in the Lemma follows from Lemma 3.5 in combination with the cases of Definition 3.7. □

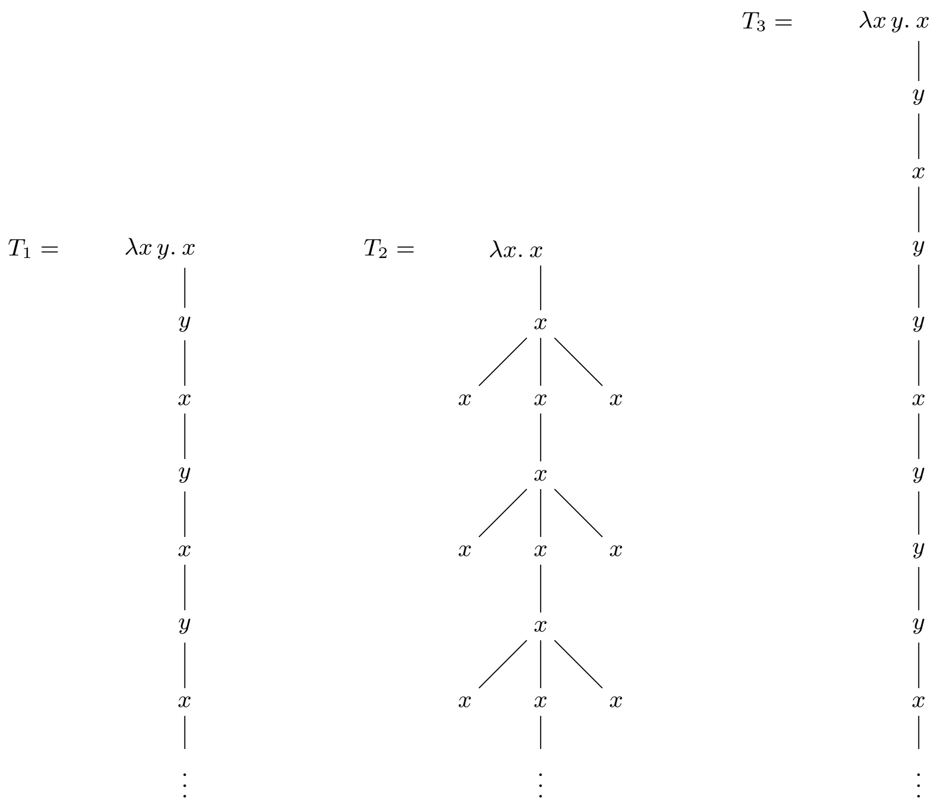
Böhm trees give a semantics of λ-terms and –as a consequence of Lemma 3.9, $\mathrm{Th}(\mathsf{BT}) := \{M = N \mid \mathsf{BT}(M) = \mathsf{BT}(N)\}$ is a λ-theory. Some simple equations that hold in $\mathrm{Th}(\mathsf{BT})$ are

$\Omega = \Omega_3$, because they both don't have a hnf, and hence their BT is the trivial one. Another simple observation is that (η) does not hold in Th(BT): just compare $\mathsf{BT}(y)$ and $\mathsf{BT}(\lambda x.y\,x)$.

We don't have to know the precise definition of a term to be able to compute its BT, as we only need to know its hnf. For example, if we know that M satisfies the equation.

It is also nice (or in any case a pleasant pass time) to draw a tree and ask if there is a λ-term that has that tree as its Böhm tree.

Example 3.10. We consider the following trees T_1, T_2, T_3 and construct terms M_1, M_2, M_3 such that $\mathsf{BT}(M_i) = T_i$ for $i = 1, 2, 3$. NB. in T_3, the number of y's between two x's is increasing.



- For T_1 we observe that we are looking for a term M_1 with $M_1 =_\beta \lambda x\,y.N_1$ where $N_1 =_\beta x\,(y\,N_1)$. So we take $N_1 := \mathbf{Y}(\lambda n.x\,(y\,n))$ and $M_1 := \lambda x\,y.N_1$.

- For T_2 we observe that we are looking for a term M_2 with $M_2 =_\beta \lambda x.N_2$ where $N_2 =_\beta x\,(x\,x\,N_2\,x)$. So we take $N_2 := \mathbf{Y}(\lambda n.x\,(x\,x\,n\,x))$ and $M_2 := \lambda x.N_2$.
(Alternatively, one could start from $M_2 =_\beta \lambda x.x\,N_2$ with $N_2 =_\beta x\,x\,(x\,N_2)\,x$, then take $N_2 := \mathbf{Y}(\lambda n.x\,x\,(x\,n)\,x)$ and $M_2 := \lambda x.x\,N_2$.)

- For T_3 we observe that we are looking for a term M_3 with $M_3 =_\beta \lambda x\,y.N_3$ where N_3 has to be parameterized over a natural number that represents the number of ys in between two xs. This can be done by defining $N\,c_n$ in a clever way. We want the following: $N\,c_n =_\beta x\,(c_n\,y\,(N\,c_{n+1}))$ because this produces $x\,(y\,(\ldots(y\,(N\,c_{n+1}))\ldots))$, with $n\times$ a y. So we take $N := \mathbf{Y}(\lambda n\,c.x\,(c\,y\,(n\,(\mathsf{Suc}\,c))))$, where Suc is the successor function on the Church numerals, see Section 1. Now we take $M := \lambda x\,y.N\,c_1$.

We now develop some more theory towards proving that

$$\mathsf{BT}(M) = \mathsf{BT}(N) \iff \mathcal{E} \models M = N.$$

A full proof of this result can be found in [1] and in [3] (using a different technique). Here we just study the implication from left to right, and we give the "Approximation Theorem" (with out full proof) from which the left to right implication follows immediately. It has independent interest as it defines an ordering on untyped λ-terms, indicating how we can view Λ as a partially ordered set. To clarify the picture we extend Λ with a $\perp$-term.

Definition 3.11. We define $\Lambda\perp$ as Λ extended with a constant $\perp$, where we add the following reduction rules

$$\begin{aligned} \lambda x.\perp &\longrightarrow_{\perp} \perp, \\ \perp M &\longrightarrow_{\perp} \perp. \end{aligned}$$

The extension of (β) reduction with this new ($\perp$) reduction will be called $\beta\perp$-reduction, and it extends immediately to multiple step reduction and $\beta\perp$-equality. We also have the usual notion of $\beta\perp$-normal form.

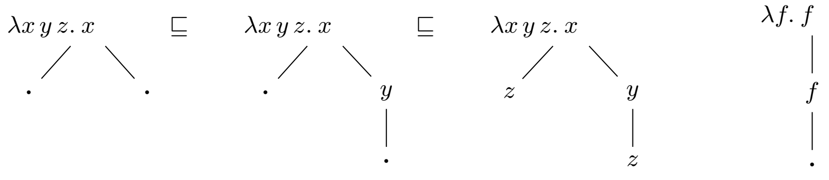
The interpretation in graph models is extended to include $\perp$ by defining $[\![\perp]\!]_\rho := \emptyset$.

The interpretation of $\Lambda\perp$-terms as Böhm trees is immediate by letting $\mathsf{BT}(\perp) := \cdot$.

For the semantics and for Böhm trees, $\perp$ acts just like Ω. In graph models, $[\![\lambda x.\perp]\!]_\rho = \emptyset$ and $[\![\perp N]\!]_\rho = \emptyset$, so graph models are sound for $=_{\beta\perp}$: if $M =_{\beta\perp} N$, then $[\![M]\!]_\rho = [\![N]\!]_\rho$.

The ordering on λ-terms will be inherited from the ordering on Böhm trees, which is defined in the obvious way: a tree T_1 is included in T_2 when some of the sub-trees in T_2 have been replaced by a $\cdot$ to obtain T_1. We write $T_1 \sqsubseteq T_2$ for this sub-tree relation.

Example 3.12. We see two Böhm trees that are sub-trees of $\mathsf{BT}(\mathbf{S})$ of Example 3.8, as indicated in the figure below. The rightmost Böhm tree is a sub-tree of $\mathsf{BT}(\mathbf{Y})$.



Definition 3.13. Let $M, N \in \Lambda\perp$.

1. We say that N *approximates* M, notation $N \sqsubseteq M$, if $\mathsf{BT}(N) \sqsubseteq \mathsf{BT}(M)$.
2. We say that N is an *approximate normal form* of M, or N is an anf of M, if
   - N is a $\beta\perp$-normal form, and
   - $N \sqsubseteq M$.
3. $\mathcal{A}(M) := \{N \in \Lambda\perp \mid N \text{ is an anf of } M\}$.

Remark 3.14. The ordering on $\Lambda\perp$ is reflexive, transitive and anti-symmetric, so it is a partial order. Using the Scott topology on Böhm trees, one can also define a topology on $\Lambda\perp$. We will not pursue that here, but see [1].

Example 3.15. 1. $\mathcal{A}(\mathbf{Y}) = \{\perp, \lambda f.\,f\perp, \lambda f.\,f^2\perp, \lambda f.\,f^3\perp, \ldots\}$

2. $\mathcal{A}(\mathbf{S}) = \{\perp, \lambda x\,y\,z.\,x\perp\perp, \lambda x\,y\,z.\,x\,y\perp, \lambda x\,y\,z.\,x\perp(x\perp), \lambda x\,y\,z.\,x\,y\,(x\perp), \lambda x\,y\,z.\,x\perp(x\,z),$ $\lambda x\,y\,z.\,x\,y\,(x\,z)\}$.
   Note that, e.g. $\lambda x\,y\,z.\perp$ and $\lambda x\,y\,z.\perp y\,(x\perp)$ are not in $\mathcal{A}(\mathbf{S})$, because they are not $\beta\perp$-normal.

We now state, without proof, the Approximation Theorem, from which one side of the equivalence between Böhm tree equality and equality in $\mathcal{E}$ follows. A proof of the Theorem can be found, e.g. in [1].

Theorem 3.16 (Approximation Theorem)**.** For $M \in \Lambda\bot$, we have, for all $\rho \in \text{Val}$,

$$[\![M]\!]_\rho = \bigcup\{[\![N]\!]_\rho \mid N \in \mathcal{A}(M)\},$$

where $[\![-]\!]_\rho$ is the interpretation in Engeler's model $\mathcal{E}$.

Corollary 3.17. For all $M, N \in \Lambda$,

$$\mathsf{BT}(M) = \mathsf{BT}(N) \Longrightarrow \mathcal{E} \models M = N.$$

Proof. Immediately from the Approximation Theorem: If $\mathsf{BT}(M) = \mathsf{BT}(N)$, then $\mathcal{A}(M) = \mathcal{A}(N)$, so $[\![M]\!]_\rho = [\![N]\!]_\rho$ for all ρ. □

References

[1] H. Barendregt. *The lambda calculus, its syntax and semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, second, revised edition, 1984.

[2] H. Barendregt and E. Barendsen. Introduction to lambda calculus. Revised edition, `https://www.cs.ru.nl/ herman/onderwijs/provingwithCA/lambda.pdf`, 2000.

[3] Chantal Berline. From computation to foundations via functions and application: The λ-calculus and its webbed models. *Theor. Comput. Sci.*, 249(1):81–161, 2000.

[4] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 55:56–68, 1940.

[5] M. Davis, editor. *The Undecidable, Basic Papers on Undecidable Propositions, Unsolvable Problems And Computable Functions.* Raven Press, New York, 1965.

[6] Erwin Engeler. Algebras and combinators. *Algebra universalis*, 13(1):389–392, 1981.

[7] G.D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125 – 159, 1975.

[8] Gordon Plotkin. A set-theoretical definition of application. Memorandum MIP–R–95, School of Artificial Intelligence, University of Edinburgh, 32pp, 1972, An extended version is at `https://era.ed.ac.uk/handle/1842/206`, 1972.

[9] Dana Scott. Data types as lattices. *SIAM Journal on Computing*, 5(3):522–587, 1976.

[10] Dana Scott. Lambda calculus: some models, some philosophy. In *Studies in Logic and the Foundations of Mathematics*, volume 101, pages 223–265. Elsevier, 1980.