Notes on the simply typed lambda calculus

Peter Aczel
Manchester University

June 16, 1998

Contents

1 Deduction **1-1**
1.1 Inference Systems . . . 1-1
1.1.1 The Definition . . . 1-1
1.1.2 Adding extra axioms . . . 1-2
1.1.3 Semantics for Inference Systems . . . 1-2
1.1.4 Formal Systems . . . 1-3
1.1.5 Rules of Inference . . . 1-3
1.2 Intuitionistic Implication . . . 1-4
1.2.1 A Hilbert-style formal system, H . . . 1-4
1.2.2 Natural Deduction . . . 1-5
1.2.3 Sequent Formulation, ND, of Natural Deduction . . . 1-7
1.2.4 Normal ND tree-proofs . . . 1-7
1.2.5 Sequent Calculus SC . . . 1-8
1.3 Intuitionistic Propositional Logic . . . 1-9
1.3.1 The Hilbert style formulation . . . 1-9
1.3.2 The Natural Deduction formulation . . . 1-9
1.3.3 The sequent formulation of Natural Deduction . . . 1-10
1.3.4 Sequent Calculus Formulation . . . 2-1

2 Untyped Lambda Calculus **2-1**
2.1 Preliminaries . . . 2-1
2.1.1 The notion of a function . . . 2-1
2.1.2 Examples of functions . . . 2-2
2.1.3 Functions as sets . . . 2-2
2.1.4 Multi-argument functions . . . 2-2
2.1.5 Currying . . . 2-3
2.1.6 The problem of variable clashes . . . 2-3
2.2 An Untyped Universe . . . 2-3
2.2.1 Some combinators in U . . . 2-4
2.2.2 Some laws of LC . . . 2-4
2.2.3 Problems with "$U = U^U$" . . . 2-5
2.2.4 A more general universe . . . 2-5
2.3 Syntax of LC . . . 2-5
2.3.1 Terms . . . 2-5
2.3.2 Construction Trees . . . 2-6
2.3.3 The de Bruijn terms . . . 2-7

2.3.4 Substitution 2-8
2.4 Deduction 2-8
2.5 Combinatory Logic 2-9
2.5.1 CL and its translation into $(w\lambda\beta)$ 2-10
2.5.2 A translation of $(w\lambda\beta)$ into CL 2-10
2.5.3 The converse of Proposition 6 2-10
2.5.4 The equivalence between $CL + (ext)$ and $(\lambda\beta\eta)$ 2-11
2.5.5 The equivalence between $CL + (wext)$ and $(\lambda\beta)$ 2-11
2.5.6 Summary 2-12
2.6 The Church-Rosser Theorem 2-12
2.6.1 The reduction relation 2-12
2.6.2 Proof of the Theorem for $(\lambda\beta)$ 2-13
2.7 Normalisation 3-1

3 Simply Typed Lambda Calculus 3-2
3.1 The Simple Type Theory STT 3-2
3.2 The variant STT' 3-3
3.2.1 Standard Set Theoretical Models of STT' 3-4
3.3 Standard Term Models of STT 3-4
3.4 The 'Normal Relation' method for STT 3-6
3.5 Girard's method 3-8
3.6 Another method for strong normalisation 3-8

Introduction

The simply typed lambda calculus, of these notes, has types built up from atomic types using the function type operation that forms a new type $A \to B$ from types A,B. The calculus can be viewed as a refined version of the purely implicational fragment of intuitionistic logic. The refinement consists in using terms of the untyped lambda calculus to represent formal derivations of the logic.

These notes consist of three sections, the last one being on the simply typed lambda calculus. In that section I focus mainly on the Curry style version for function types, I call STT, that consists of rules for typing terms of the untyped lambda calculus. The middle section is on the untyped lambda calculus itself while the first section is mainly on the three standard styles of inference system for intuitionistic implicational logic; the Hilbert style, the Natural deduction style and the Sequent calculus style.

The notes have been based on earlier notes for part of an M.Sc. course on type theory that I gave at Manchester University in the spring of 1997. Those notes were made available as 'working material' for the lectures on constructive type theory that I gave at the Summer school. The lectures aimed to give a presentation of the ideas of Martin-Löf's type theory using his 'meaning explanations' to justify the rules of inference. For this topic I suggest the references [6, 7, 8], where further references may be found. After the Summer School I became dissatisfied with the approach that I had taken, but did not have enough time to work out an approach that I was satisfied with. With the agreement of the editors I have prepared these notes for the proceedings.

The main purpose of the notes is to act as a tutorial introduction to the three topics it treats and their relationships with each other. The novice reader is advised to read the notes in conjunction with the use of a more thorough text such as [10] or [3], which have more detailed discussions and reference lists than are available here.

Almost all the material and its organisation is fairly standard, so that the expert will find little that is new here. But here are some distinctive features of my presentation.

1. In 1.1 I introduce a precise notion of *inference system* which is then used throughout the notes.

2. The untyped lambda calculus is motivated set theoretically by postulating an infinite set U that is equal to the set U^U of all unary functions on U. Of course this is impossible in classical axiomatic set theory. But I believe that it is consistent with an intuitionistic set theory in which non-well-founded sets are allowed, so that reasoning with such a set is not so unreasonable, at least for motivational purposes.

3. I use the method of proof introduced in [9] to prove the Church-Rosser theorem.

4. I set up a general framework for giving normalisation and strong normalisation proofs. This seems to me to be worthwhile, as there are now many proofs explicit or implicit in the literature. As there are now many type theories for which normalisation proofs can be found, and no doubt more to come, I think that efforts to systematise normalisation proofs for the simplest type theory will be useful.

1 Deduction

1.1 Inference Systems

1.1.1 The Definition

An **inference system** consists of

- a set of **statements**, $S, S', \ldots$
- a set of **steps**

$$\frac{S_1 \cdots S_n}{S}$$

with $n \geq 0$ **premisses** $S_1, \ldots, S_n$ and **conclusion** S, these being statements. When $n = 0$ then S is an **axiom**.

Given an inference system a **tree-proof** is an upward growing finite tree, labelled with statements, such that at each node, labelled with a statement S there is a step with conclusion S whose premisses label the children of the node. A tree-proof is a **tree-proof of** S if the statement S labels the root of the tree. If S has a tree-proof in an inference system Σ then S is a **theorem of** Σ, written $\Sigma \vdash S$, or just $\vdash S$ when Σ is understood from the context.

Exercises 1.1

1. *Show that the theorems of an inference system form the smallest set X of statements such that, for each step, if the premisses are in X then so is the conclusion.*

2. *Show that the set of tree-proofs of an inference system form the smallest set Y of finite trees labelled with statements such that for each step*

$$\frac{S_1 \cdots S_n}{S}$$

if $\tau_1, \ldots, \tau_n$ are trees in Y whose roots are labelled with $S_1, \ldots, S_n$ respectively then τ is a tree in Y, where τ is the finite tree whose root is labelled with S that has the trees $\tau_1, \ldots, \tau_n$ as immediate subtrees.

3. [**Linear Proofs**]
Given an inference system a **linear proof of a statement** *S is a finite sequence $S_1, \ldots, S_m$ of statements, with $S_m = S$, such that for $i = 1, \ldots, m$ there is a step*

$$\frac{S_{i_1} \cdots S_{i_n}}{S_i}$$

with $1 \leq i_1, \ldots, i_n < i$.

Show that a statement is a theorem of the inference system iff it has a linear proof.

1.1.2 Adding extra axioms

If $S_1, \ldots, S_n$ are statements of an inference system Σ then we write $\Sigma, S_1, \ldots, S_n$ for the inference system obtained from Σ by adding new axioms $S_1, \ldots S_n$; i.e. there are new steps

$$\frac{}{S_1} \quad \cdots \quad \frac{}{S_n}.$$

If Σ is understood then we just write

$$S_1, \ldots, S_n \vdash S$$

for $\Sigma, S_1, \ldots, S_n \vdash S$.

1.1.3 Semantics for Inference Systems

Often an inference system will have a **semantics**. For our purposes we will take a semantics for an inference system to be a collection of **interpretations**, I, each specifying the **correct** (or **true**) statements of the interpretation. We write $I \models S$ if S is a correct statement of I. An interpretation is **sound** if, for every step, if the premisses are correct then so is the conclusion. A semantics is sound if every interpretation is sound.

Given a semantics for an inference system, we write

$$S_1, \ldots S_n \models S$$

if for every interpretation in which $S_1, \ldots, S_n$ are correct the statement S is also correct. The semantics is **complete** if

$$S_1, \ldots S_n \models S \text{ implies } S_1, \ldots S_n \vdash S$$

and is **weakly complete** if

$$\models S \text{ implies } \vdash S.$$

Note that we can formulate a notion of strong completeness by defining '$X \vdash S$' and '$X \models S$', where X is a possibly infinite set of statements.

Exercise 1.2 *Show that if a semantics is sound then*

$$S_1, \ldots S_n \vdash S \textit{ implies } S_1, \ldots S_n \models S.$$

1.1.4 Formal Systems

A **formal system** is an inference system for which there is a data type T such that the following holds.

- Every statement can be represented by a value in T.
- There is an algorithm to determine whether or not a value in T represents a statement.
- There is an algorithm to determine whether or not a pair of values of T represent the same statement.
- there is an algorithm to determine whether or not a configuration

$$\frac{v_1 \cdots v_n}{v}$$

  of values in T represents a step of the inference system.

Note the following facts

- Given a formal system there is an algorithm to determine whether or not a finite tree of values of the data type represents a proof-tree or not.
- If Σ is a formal system then so is $\Sigma, S_1, \ldots, S_n$.

1.1.5 Rules of Inference

Usually the steps of an inference sytem are given by **rules**, each rule determining a set of steps called the **instances** of the rule. Often each rule is given **schematically** as a scheme; i.e. a configuration

$$\frac{\theta_1 \cdots \theta_n}{\theta}$$

of expressions involving **metavariables** that can be substituted for. The instances of the rule are then obtained by suitably substituting for the metavariables. What is meant by a suitable substitution has to be specified by an implicit or explicit **side condition** of the scheme. An **axiom scheme** is a special case of a schematically given rule in which all instances are steps having no premisses, so that the conclusion of each instance is an axiom.

Often a formal system is given by finitely many schematic rules, each having a syntactically specified matching algorithm for deciding whether or not a configuration of data values is an instance of the rule; i.e. can be obtained from the scheme by a suitable substitution for the metavariables.

1.2 Intuitionistic Implication

We assume given a set of **atomic formulae** $A_0, ...$ The **formulae** are generated from the atomic formulae using the rule

$$\frac{A_1 \quad A_2}{(A_1 \to A_2)}$$

i.e. the rule that if A_1, A_2 are formulae then so is $(A_1 \to A_2)$.

Abbreviation Conventions

1. Leave out outermost parentheses.
2. Associate to the right.

so, for example

$$A_1 \to A_2 \to A_3 \to A_4$$

abbreviates

$$(A_1 \to (A_2 \to (A_3 \to A_4))).$$

1.2.1 A Hilbert-style formal system, H

We describe a formal system, H, whose statements are the formulae defined above. There are two axiom schemes **(K), (S)** and the rule of inference **(MP)** of *Modus Ponens*

Axiom Schemes

(K) $A \to B \to A$.

(S) $(A \to B \to C) \to (A \to B) \to A \to C$.

(MP) $\dfrac{A \to B \quad A}{B}$.

Each axiom scheme represents infinitely many instances; e.g. for each pair of formulae A, B the formula $A \to B \to A$ is an instance of $(Ax1)$.

A tree-proof of $A \to A$: Let A be any formula and let B be $A \to A$.

$$\dfrac{\dfrac{\dfrac{}{(A \to B \to A) \to (A \to B) \to B}(S) \quad \dfrac{}{A \to B \to A}(K)}{(A \to B) \to B}(MP) \quad \dfrac{}{A \to B}(K)}{B}(MP)$$

Note the conventions that we use to make the tree-proof more readable. Each leaf is given the name of the axiom scheme it is an instance of and each application of (MP) has been named.

A linear proof of $A \to A$: Again let A be any formula and let B be $A \to A$. Here is a linear proof consisting of five numbered formulae.

1.	$(A \to B \to A) \to (A \to B) \to B$	(S)
2.	$A \to B \to A$	(K)
3.	$(A \to B) \to B$	$MP(1,2)$
4.	$A \to B$	(K)
5.	B	$MP(3,4)$

Note some obvious conventions used on the right hand side of each line to justify that line. Note also that both the tree-proof and the linear proof are schematic - they apply uniformly to any formula A.

Exercise 1.3 (Deduction Theorem) *Show that if $A_1, \ldots A_n, A \vdash B$ then $A_1, \ldots, A_n \vdash A \to B$. [This is a standard result whose proof method can be found in many text-books on logic.]*

1.2.2 Natural Deduction

The *natural* way to prove an implication $A \to B$ is to assume A and try to deduce B, making use of the assumption A, when needed. Once B has been successfully deduced from A then we can infer $A \to B$. Notice that although A was assumed, when trying to deduce B the assumption of A is dropped once the inference step has been made to get $A \to B$. This kind of inference step involves the **discharge** of the assumption A. This is a new idea, that is not part of the apparatus of inference systems we have been using so far. Nevertheless there is a convenient way to modify the notion of a tree-proof so as to allow for the discharge of assumptions.

In an inference system Σ a tree-proof from assumptions $S_1, \ldots, S_n$ is simply a tree-proof in the inference system $\Sigma, S_1, \ldots, S_n$ so that, at the leaves of the tree any of the assumptions $S_1, \ldots, S_n$ may appear as well as any axioms of Σ. In natural deduction tree-proofs we allow the possibility of discharging an assumption at an inference step. We will do this by labelling the occurrences of the assumption being discharged and at the same time labelling the inference step where the discharge happens with the same label. Of course the same label should not be used for different examples of assumption discharge in the same tree.

We can now describe the two rules for constructing ND tree-proofs for Intuitionistic Implication.

The **Introduction Rule for Implication**, abbreviated $(\to I)$, states that given an ND tree-proof τ of a formula B we can form an ND tree-proof of $A \to B$ having $A \to B$ at the root and having the tree τ as its only immediate subtree. But any undischarged occurrences of A as an assumption in τ can become discharged in the new tree.

The **Elimination Rule for Implication**, abbreviated $(\to E)$, is simply the familiar modus ponens rule that we have called (MP). These rules are written schematically as follows.

$$(\to I) \quad \frac{\begin{matrix}[A] \\ B\end{matrix}}{A \to B} \qquad (\to E) \quad \frac{A \to B \;\; A}{B}$$

Note that in $(\to I)$ the assumption A has been placed in square brackets above B to indicate that it may be assumed in the proof of B, but then discharged at the inference step.

Examples of ND tree-proofs We give ND tree-proofs of the axioms of our Hilbert-style formal system for Intuitionistic Implication.

(K)

$$\dfrac{\dfrac{\dfrac{}{A}\,1}{B \to A}\,(\to I)}{A \to B \to A}\,(\to I)[1]$$

(S)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \to B \to C}\,1 \quad \dfrac{}{A}\,3}{B \to C}\,(\to E) \quad \dfrac{\dfrac{}{A \to B}\,2 \quad \dfrac{}{A}\,3}{B}\,(\to E)}{C}\,(\to E)}{A \to C}\,(\to I)[3]}{(A \to B) \to A \to C}\,(\to I)[2]}{(A \to B \to C) \to (A \to B) \to A \to C}\,(\to I)[1]$$

Note the convention for labelling a discharged assumption with a natural number that also labels the step where the assumption gets discharged.

Examples of ND linear proofs Here are two linear ND proofs, where we have used some obvious conventions to indicate where an assumption is made and where it gets discharged.

(K)

1.	A	(ass)
2.	B	(ass)
3.	$B \to A$	$(\to I)[2](1)$
4.	$A \to B \to A$	$(\to I)[1](3)$

(S)

1.	$A \to B \to C$	(ass)
2.	$A \to B$	(ass)
3.	A	(ass)
4.	$B \to C$	$(\to E)(1,3)$
5.	B	$(\to E)(2,3)$
6.	C	$(\to E)(4,5)$
7.	$A \to C$	$(\to I)[3](6)$
8.	$(A \to B) \to A \to C$	$(\to I)[2](7)$
9.	$(A \to B \to C) \to (A \to B) \to A \to C$	$(\to I)[1](8)$

Proofs of $A \to A$

$$\dfrac{\dfrac{}{A}\,1}{A \to A}\,(\to I)[1]$$

1.	A	(ass)
2.	$A \to A$	$(\to I)[1](1)$

1.2.3 Sequent Formulation, ND, of Natural Deduction

In this formulation of natural deduction the use of a special kind of tree-proof is replaced by the use of **sequents** as the statements of a formal system having standard Hilbert-style rules of inference. A **sequent** has the form

$$\Gamma \ \vdash\!\blacktriangleright \ A$$

where Γ is a finite sequence $A_1, \ldots, A_n$ of formulae and A is a formula.
Note: In the literature other symbols are often used, instead of $\vdash\!\blacktriangleright$; e.g. '$\vdash$' or '$\Rightarrow$'. But these symbols are often also used in other ways, leading to ambiguity.

The formal system is specified by the axiom scheme (ass) and the two rules of inference $(\to I)$ and $(\to E)$.

$$(ass) \quad \Gamma \ \vdash\!\blacktriangleright \ A \quad (A \text{ in } \Gamma)$$

$$(\to I) \quad \frac{\Gamma, A \ \vdash\!\blacktriangleright \ B}{\Gamma \ \vdash\!\blacktriangleright \ A \to B}$$

$$(\to E) \quad \frac{\Gamma \ \vdash\!\blacktriangleright \ A \to B \qquad \Gamma \ \vdash\!\blacktriangleright \ A}{\Gamma \ \vdash\!\blacktriangleright \ B}$$

Write $\Gamma \ \vdash\!\blacktriangleright \ _{ND} A$ if $\Gamma \ \vdash\!\blacktriangleright \ A$ is a theorem of this formal system; i.e. if $ND \vdash \Gamma \ \vdash\!\blacktriangleright \ A$.

Theorem 1.4 $\Gamma \ \vdash\!\blacktriangleright \ _{ND} A$ *iff there is an* ND *tree-proof of* A*, whose undischarged assumptions are all in* Γ*.*

Write $\Gamma \vdash_H A$ if $\Gamma \vdash A$ in the Hilbert style inference system for Implication with the axiom schemes (K), (S) and the rule (MP).

Theorem 1.5 $\Gamma \ \vdash\!\blacktriangleright \ _{ND} A$ *iff* $\Gamma \vdash_H A$.

The result in the following exercise expresses that the structural rules of weakening, contraction and permutation are all admissible rules of ND.

Exercise 1.6 *Show that if every formula in* Γ *also occurs in* Γ' *then*

$$\Gamma \ \vdash\!\blacktriangleright \ _{ND} A \quad \textit{implies} \quad \Gamma' \ \vdash\!\blacktriangleright \ _{ND} A.$$

1.2.4 Normal ND tree-proofs

A **redex** in an ND tree-proof is an occurrence of an instance of the rule $(\to E)$ whose left hand premiss is the conclusion of an instance of the rule $(\to I)$; i.e. a part of the tree of the form

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B}\,(\to I) \qquad \begin{array}{c}\vdots \\ A\end{array}}{B}\,(\to E)$$
$$\vdots$$

An ND tree-proof is **normal** if it contains no redex. Note that a redex is a detour and it is natural to consider 'simplifying' a redex by replacing it by its **contractum**,

which is the ND tree-proof obtained from the ND tree-proof of B, appearing in the redex, by replacing all the occurrences of A, as an assumption that gets discharged in the first premiss of the redex, by the ND tree-proof of A occuring in the second premiss of the redex. The result may be pictured

$$\begin{array}{c} \vdots \\ A \\ \vdots \\ B \end{array}$$

Theorem 1.7 *If* $\Gamma \vdash_{ND} A$ *then there is a normal* ND *tree-proof of* A *whose undischarged assumptions are all in* Γ.

Proof Idea: Given an ND tree-proof, if it is not already normal then choose a redex and replace it by its contractum. Do this repeatedly, until no more redexes remain. It is not obvious that this procedure eventually terminates successfully. Nevertheless it turns out that however the redexes to be contracted are chosen the procedure will indeed terminate. This is the **strong normalisation property** for Natural Deduction. In fact it turns out to be easier to show that a particular systematic strategy for choosing redexes leads to termination. This is the (weak) **normalisation property**. We will return to complete this proof later, when we discuss the simply typed lambda calculus.

1.2.5 Sequent Calculus SC

This formal system uses the same sequents as statements that are used in ND, but has slightly different rules. First of all there is the structural rule

$$\frac{\Gamma \ \mapsto A}{\Gamma' \ \mapsto A}$$

with the side condition that the same formulae appear in Γ as are in Γ'. The effect of having this rule is that the order that assumptions appear in the sequence Γ is irrelevent, as is the number of times the same formula appears in the sequence; i.e. Γ can be treated as a *set* of assumptions. The remaining rules are

$$(ass) \quad \Gamma \ \mapsto A \quad (A \text{ in } \Gamma)$$

$$(\to R) \quad \frac{\Gamma, A \ \mapsto B}{\Gamma \ \mapsto A \to B}$$

$$(\to L) \quad \frac{\Gamma \ \mapsto A \qquad \Gamma, B \ \mapsto C}{\Gamma, A \to B \ \mapsto C}$$

$$(cut) \quad \frac{\Gamma, A \ \mapsto C \qquad \Gamma \ \mapsto A}{\Gamma \ \mapsto C}$$

We write $\Gamma \ \mapsto_{SC} A$ if $SC \vdash \Gamma \ \mapsto A$.

Theorem 1.8 $\Gamma \ \mapsto_{SC} A$ *iff* $\Gamma \ \mapsto_{ND} A$.

Let SC^- be SC with the cut rule left out.

Theorem 1.9 *If* $\Gamma \ \mapsto_{SC} A$ *then* $\Gamma \ \mapsto_{SC^-} A$.

This is the **Cut Elimination Theorem** for this sequent calculus. It is also called Gentzen's **Hauptsatz**. Gentzen gave an algorithm for systematically removing cuts from a tree-proof in SC. By theorems 1.5 and 1.7 it suffices to show that if there is a normal ND tree-proof of A whose undischarged assumptions are all in Γ then $\Gamma \vdash\!\!\blacktriangleright_{SC^-} A$. This is not hard to do by induction on the size of the normal ND tree-proof.

Note that Natural Deduction and the Sequent Calculus are closely related approaches to deduction in which normalisation of ND tree-proofs corresponds to cut-elimination. We will prefer to focus on the natural deduction approach.

Exercise 1.10 *Prove theorems 1.4,1.5,1.8 and 1.9 using theorem 1.7 in the proof of theorem 1.9.*

1.3 Intuitionistic Propositional Logic

In this section we extend the purely implicational logic of the previous section by adding the other logical constants of intuitionistic propositional logic; i.e. we add the two binary connectives $\wedge$ and $\vee$ for conjunction and disjunction and the special constant $\bot$ for absurdity. The other propositional constants $\leftrightarrow$ and $\neg$ of bi-implication and negation can be defined in the standard way. So every non-atomic formula has one of the forms

$$(A_1 \to A_2), \quad (A_1 \wedge A_2), \quad (A_1 \vee A_2), \quad \bot$$

and we use the definitions

$$(A_1 \leftrightarrow A_2) \;=\; (A_1 \to A_2) \wedge (A_2 \to A_1)$$

and

$$\neg A \;=\; (A \to \bot).$$

1.3.1 The Hilbert style formulation

We keep the axiom schemes and rule of inference we had for implication and add the axiom schemes

$$\begin{array}{ll}
(Ax3) & A_1 \to A_2 \to (A_1 \wedge A_2) \\
(Ax4) & (A_1 \wedge A_2) \to A_1 \\
(Ax5) & (A_1 \wedge A_2) \to A_2 \\
(Ax6) & A_1 \to (A_1 \vee A_2) \\
(Ax7) & A_2 \to (A_1 \vee A_2) \\
(Ax8) & (A_1 \to C) \to (A_2 \to C) \to (A_1 \vee A_2) \to C \\
(Ax9) & \bot \to A
\end{array}$$

1.3.2 The Natural Deduction formulation

We first give the schemes for the new forms of ND tree-proofs.

$$(\wedge I)\ \frac{A_1 \quad A_2}{A_1 \wedge A_2} \qquad (\wedge E1)\ \frac{A_1 \wedge A_2}{A_1} \qquad (\wedge E2)\ \frac{A_1 \wedge A_2}{A_2}$$

$$(\vee I1)\ \frac{A_1}{A_1 \vee A_2} \qquad (\vee I2)\ \frac{A_2}{A_1 \vee A_2} \qquad (\vee E)\ \frac{A_1 \vee A_2 \qquad \begin{array}{c}[A_1]\\ C\end{array} \qquad \begin{array}{c}[A_2]\\ C\end{array}}{C}$$

$$(\bot\ E)\ \frac{\bot}{C}$$

Note that $(\vee E)$ involves the discharge of assumption occurrences in the ND tree-proofs of the second and third premisses.

The notion of a **normal** ND tree-proof extends to the full intuitionistic propositional logic after we add the new forms of **redex** for $\wedge$ and $\vee$:

$$\dfrac{\dfrac{\begin{array}{cc}\vdots & \vdots\\ A_1 & A_2\end{array}}{A_1 \wedge A_2}}{A_i}\qquad\qquad \dfrac{\begin{array}{ccc}\vdots & [A_1] & [A_2]\\ A_i & \vdots & \vdots\\ A_1 \vee A_2 & C & C\end{array}}{C}$$

where $i = 1, 2$. Their **contracta** are

$$\begin{array}{ccc}\vdots & & \vdots\\ A_i & \text{and} & A_i\\ \vdots & & \vdots\\ & & C\\ & & \vdots\end{array}$$

respectively. With these definitions theorem 3 of section 2 and its methods of proof carry over to full intuitionistic propositional logic.

1.3.3 The sequent formulation of Natural Deduction

We give the new rules of inference in the sequent formulation.

$$(\wedge I)\ \frac{\Gamma \mapsto A_1 \quad \Gamma \mapsto A_2}{\Gamma \mapsto A_1 \wedge A_2}$$

$$(\wedge E1)\ \frac{\Gamma \mapsto A_1 \wedge A_2}{\Gamma \mapsto A_1} \qquad (\wedge E2)\ \frac{\Gamma \mapsto A_1 \wedge A_2}{\Gamma \mapsto A_2}$$

$$(\vee I1)\ \frac{\Gamma \mapsto A_1}{\Gamma \mapsto A_1 \vee A_2} \qquad (\vee I2)\ \frac{\Gamma \mapsto A_2}{\Gamma \mapsto A_1 \vee A_2}$$

$$(\vee E)\ \frac{\Gamma \mapsto A_1 \vee A_2 \quad \Gamma, A_1 \mapsto C \quad \Gamma, A_2 \mapsto C}{\Gamma \mapsto C}$$

$$(\bot\, E)\ \frac{\Gamma \mapsto \bot}{\Gamma \mapsto C}$$

1.3.4 Sequent Calculus Formulation

$$(\wedge R)\ \frac{\Gamma \ \vdash\!\blacktriangleright\ A_1 \qquad \Gamma \ \vdash\!\blacktriangleright\ A_2}{\Gamma \ \vdash\!\blacktriangleright\ A_1 \wedge A_2}$$

$$(\wedge L1)\ \frac{\Gamma, A_1 \ \vdash\!\blacktriangleright\ C}{\Gamma, (A_1 \wedge A_2) \ \vdash\!\blacktriangleright\ C} \qquad (\wedge L2)\ \frac{\Gamma, A_2 \ \vdash\!\blacktriangleright\ C}{\Gamma, (A_1 \wedge A_2) \ \vdash\!\blacktriangleright\ C}$$

$$(\vee R1)\ \frac{\Gamma \ \vdash\!\blacktriangleright\ A_1}{\Gamma \ \vdash\!\blacktriangleright\ A_1 \vee A_2} \qquad (\vee R2)\ \frac{\Gamma \ \vdash\!\blacktriangleright\ A_2}{\Gamma \ \vdash\!\blacktriangleright\ A_1 \vee A_2}$$

$$(\vee L)\ \frac{\Gamma, A_1 \ \vdash\!\blacktriangleright\ C \qquad \Gamma, A_2 \ \vdash\!\blacktriangleright\ C}{\Gamma, (A_1 \vee A_2) \ \vdash\!\blacktriangleright\ C}$$

$$(\bot\, L)\ \ \Gamma, \bot \vdash\!\blacktriangleright\ C$$

Exercise 1.11 *Extend your proofs of theorems 1.4,1.5,1.8 and 1.9 of section 2 to the full intuitionistic propositional logic.*

2 Untyped Lambda Calculus

2.1 Preliminaries

2.1.1 The notion of a function

Let A, B be sets. As usual we write $f : A \to B$ if f **is a function from** A **to** B. So we have the rule of **function application**:

$$\frac{f : A \to B \qquad a \in A}{f(a) \in B}$$

Functions are treated **extensionally** so that functions $f, g : A \to B$ are equal if they have the same value on every argument; i.e. if $f, g : A \to B$ then

$$f = g \iff (\forall x \in A)\ f(x) = g(x).$$

A function $f : A \to B$ can be defined by a definition of the form

$$f(x) = \cdots x \cdots \text{ for } x \in A.$$

where '$\cdots x \cdots$' is an expression involving the variable x that denotes a value in the set B whenever x is assigned a value in the set A. It is convenient to write

$$(\lambda x \in A) \cdots x \cdots$$

for the unique function $f : A \to B$ with the above defining equation. So we have the rule of function abstraction:

$$\frac{\begin{array}{c}[x \in A] \\ \cdots x \cdots \in B\end{array}}{(\lambda x \in A) \cdots x \cdots \ :\ A \to B}$$

2.1.2 Examples of functions

Let A, B, C be sets.

Identity Let $I_A = (\lambda x \in A)x$. Then $I_A : A \to A$ is the unique function defined by

$$I_A(x) = x \text{ for } x \in A.$$

Constant Let $K_{A,b} = (\lambda x \in A)b$, where $b \in B$. Then $K_{A,b} : A \to B$ is the unique function defined by

$$K_{A,b}(x) = b \text{ for } x \in A.$$

Composition Let $(g \circ f) = (\lambda x \in A)(g(f(x)))$ where $f : A \to B$ and $g : B \to C$. Then $(g \circ f) : A \to C$ is the unique function defined by

$$(g \circ f)(x) = g(f(x)) \text{ for } x \in A.$$

2.1.3 Functions as sets

Let A, B be sets. The **cartesian product** $A \times B$ is the set of all ordered pairs (a, b) with $a \in A$ and $b \in B$. So we have the rule

$$\frac{a \in A \quad b \in B}{(a, b) \in A \times B}$$

If $f : A \to B$ then its **graph** is the set G of all pairs $(a, f(a))$ with $a \in A$. The set G is a subset of $A \times B$ such that

$$(\forall x \in A)(\exists! y \in B) \ \ (x, y) \in G.$$

Conversely for any subset G of $A \times B$ that satisfies the above condition we can define a function $f : A \to B$ with the defining equation

$$f(x) = \text{ the unique } y \in B \text{ such that } (x, y) \in G.$$

In axiomatic set theory functions are identified with their graphs.

2.1.4 Multi-argument functions

Let $A_1, \ldots, A_n, B$ be sets. The cartesian product $A_1 \times \cdots \times A_n$ is the set of all ordered n-tuples $(a_1, \ldots, a_n)$ with $a_1 \in A_1, \ldots, a_n \in A_n$. If $f : A_1 \times \cdots \times A_n \to B$ then we have the rule

$$\frac{a_1 \in A_1 \ \cdots a_n \in A_n}{f(a_1, \ldots, a_n) \in B}$$

If $\cdots$ is an expression for a value in B when variables $x_1, \ldots, x_n$ that may occur in $\cdots$ are assigned values in $A_1, \ldots, A_n$ respectively then

$$(\lambda(x_1, \ldots, x_n) \in A_1 \times \cdots \times A_n) \ \ \cdots$$

is the unique function $f : A_1 \times \cdots \times A_n \to B$ with the defining equation

$$f(x_1, \ldots, x_n) = \ \cdots.$$

2.1.5 Currying

Let A, B, C be sets. We write C^B for the set of all the functions $B \to C$. If $f : A \times B \to C$ then $g : A \to C^B$ where

$$g = (\lambda x \in A)(\lambda y \in B) f(x, y).$$

We call g the **curried** version of f. This term is named after the American logician Haskell B Curry, who was the main developer of *Combinatory Logic*. His name has also been used for the functional programming language *Haskell*. The function f can be recaptured from g by a process called **uncurrying**:

$$f = (\lambda(x, y) \in A \times B) g(x)(y)$$

2.1.6 The problem of variable clashes

Let A be a set and let $f : A \times A \to A$. The curried version $g : A \to A^A$ has defining equation

$$(1) \quad g(x) = (\lambda y \in A) f(x, y) \text{ for } x \in A.$$

We could use 'z' instead of 'y':-

$$(2) \quad g(x) = (\lambda z \in A) f(x, z) \text{ for } x \in A.$$

But we cannot use 'x' instead of 'y':-

$$(3) \quad g'(x) = (\lambda x \in A) f(x, x) \text{ for } x \in A.$$

The function g' defined in (3) is generally quite different to the function g defined in (1) or in (2). In (3) the variable 'x' occurs free on the left hand side while it only occurs bound in the lambda expression on the right hand side so that g' is a constant function.

If $h : A \to A$ let $k : A \to A$ be given by

$$k(y) = g(h(y)) \text{ for } y \in A.$$

Using (1) to expand the right hand side we get:-

$$k(y) = (\lambda y \in A) f(h(y), y) \text{ for } y \in A,$$

which is wrong! Instead we can use (2) to get

$$k(y) = (\lambda z \in A) f(h(y), z) \text{ for } y \in A.$$

Moral: Before making a textual substitution you must first suitably relabel bound variables so as to avoid variable clashes. This same problem of variable clashes is a familiar feature of quantifiers in the predicate calculus.

2.2 An Untyped Universe

The (untyped) **Lambda Calculus** (LC) is a calculus of 'functions' where it 'makes sense' to 'apply' anything to anything. To motivate LC we postulate a non-trivial set U (so U should have more than one element) such that $U = U^U$. This will be our universe of functions. For $a, b \in U$ we write $a \cdot b$ or usually just ab for the result of applying the function $a : U \to U$ to $b \in U$. We use the following abbreviations:-

$ab_1b_2 \cdots b_n$	for	$\cdots((ab_1)b_2)\cdots b_n$
$\lambda x. \cdots$	for	$(\lambda x \in U) \cdots$
$\lambda x_1 \cdots x_n. \cdots$	for	$\lambda x_1. \cdots \lambda x_n. \cdots$

Note that if $f : U^n \to U$ and $a = \lambda x_1 \cdots x_n . f(x_1, \ldots, x_n)$ then $a \in U$ and, for $x_1, \ldots, x_n \in U$

$$f(x_1, \ldots, x_n) = a x_1 \cdots x_n.$$

So $f = (\lambda(x_1, \ldots, x_n) \in U^n) a x_1 \cdots x_n$. We call a the **curried** version of f.

2.2.1 Some combinators in U

$$\begin{array}{ll} I = \lambda x.x, & Ia = a \\ K = \lambda xy.x, & Kab = a \\ B = \lambda xyz.x(yz), & Babc = a(bc) \\ C = \lambda xyz.xzy, & Cabc = acb \\ S = \lambda xyz.xz(yz), & Sabc = ac(bc) \end{array}$$

The above λ-expressions are examples of **combinators**. Following their definitions are their defining equations, where a, b, c are arbitrary elements of U. There can be equations connecting the combinators. For example we have the following two results.

Proposition 2.1 $I = SKK$.

Proof: Observe that for any $x \in U$

$$SKKx = Kx(Kx) = x.$$

But I is the unique element of U such that $Ix = x$ for all $x \in U$. It follows that $SKK = I$.

Proposition 2.2 $C = S(BBS)(KK)$.

Proof: Observe that for any $x, y, z \in U$

$$\begin{array}{ll} S(BBS)(KK)xyz & = BBSx(KKx)yz \\ & = B(Sx)(KKx)yz \\ & = B(Sx)Kyz \\ & = Sx(Ky)z \\ & = xz(Kyz) \\ & = xzy \\ & = xyz. \end{array}$$

So we get the result.

Combinatory Logic (CL) is concerned with the combinators such as I, K, B, C, S and equations involving them.

2.2.2 Some laws of LC

$$(\beta) \qquad (\lambda x. \cdots x \cdots)a = \cdots a \cdots$$

$$(\eta) \qquad \lambda x.(ax) = a$$

$$(ext) \qquad \frac{[x \in A] \quad ax = bx}{a = b}$$

Exercise 2.3 *Let $V = CB(SII)$. Given $a \in U$ let $d = Va$ and let $c = dd$. Show that $c = ac$. Show that $c = Ya$.*

We call c a **fixed point** of a and we call Y a **fixed point combinator**.

2.2.3 Problems with "$U = U^U$"

In the standard axiomatic set theory ZFC it is inconsistent to have a non-trivial set U such that $U = U^U$. But it is probably relatively consistent in the variant of ZFC that uses intuitionistic logic instead of clasical logic and drops the Foundation Axiom. We give two proofs of the inconsistency.

First Contradiction in ZFC Suppose that $a \in U \subseteq U^U$ and let $b = a(a)$. Then $(a, b) \in a$. So, as $(a, b) = \{\{a\}, \{a, b\}\}$,

$$a \in \{a\} \in (a, b) \in a$$

contradicting the Foundation Axiom.

Second Contradiction in ZFC This proof uses cardinal numbers. Let $n > 1$ be the cardinality of U. If $U = U^U$ then $n = n^n$. But by Cantor's theorem $2^n > n$ so that $n^n \geq 2^n > n$, contradicting $n = n^n$.

Note that the reasoning in the first proof is constructive. But a closer examination of the cardinality argument in the second proof will show that the second proof is not constructive.

2.2.4 A more general universe

This time we assume that U is a non-trivial set with functions $F : U \to U^U$ and $G : U^U \to U$ such that $F \circ G$ is the identity function on U^U. We can define

$$\begin{array}{lll} ab & = F(a)(b) & \text{for } a, b \in U \\ \lambda x.f(x) & = G(f) & \text{for } f \in U^U \end{array}$$

We still have

$$(\beta) \quad (\lambda x.f(x))a = f(a).$$

But only have weak versions of (η) and (ext). Let

$$U' = \{G(f) \mid f \in U^U\}.$$

Then we have

$(w\eta)$ If $a \in U'$ then $a = \lambda x.(ax)$.

$(wext)$ If $a, b \in U'$ and $ax = bx$ for all $x \in U$ then $a = b$.

When $U' = U$ then we do get (η) and (ext) and F is a bijection $F : U \cong U^U$, with inverse G.

2.3 Syntax of LC

2.3.1 Terms

We assume given a set const of constants and an infinite set var of variables. The **terms** of LC are built up from the constants and variables using application and lambda abstraction. So the terms M are given by the BNF style grammar

$$M ::= \ x \mid c \mid (MM) \mid (\lambda x.M)$$

where c is used for constants and x for variables. Alternatively the set of terms is inductiveley defined using the following rules.

1. Every constant is a term and every variable is a term.
2. If M_1, M_2 are terms then so is $(M_1 M_2)$.
3. If x is a variable and M is a term then $(\lambda x.M)$ is a term.

Notational Conventions

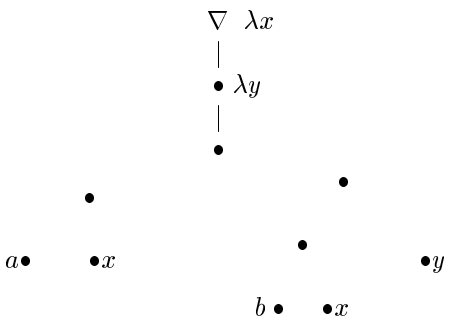
- Leave off outermost parentheses from terms.
- $MM_1M_2\cdots M_n$ abbreviates $((\cdots((MM_1)M_2)\cdots)M_n)$
- $\lambda x_1\cdots x_n.M$ abbreviates $(\lambda x_1.(\cdots.(\lambda x_n.M)\cdots))$.

Example: $\lambda xy.(ax(bxy))$ abbreviates

$$(\lambda x.(\lambda y.((ax)((bx)y))).$$

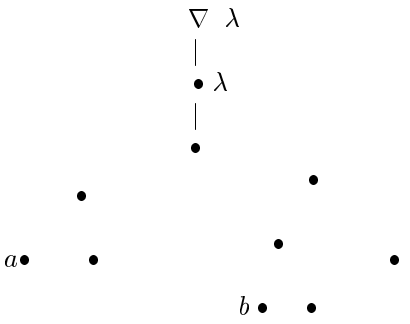
2.3.2 Construction Trees

Each term is constructed in a unique way using the rules and so has an associated **construction tree**. For example the term $(\lambda x.(\lambda y.((ax)((bx)y)))$ has the tree



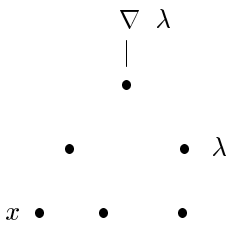
Note that ∇ indicates explicitly the root of the downward growing tree. Each leaf is labelled with a variable or constant.

A variable leaf $\bullet y$ is **bound** if there is a λ-node $\bullet\lambda y$ in the path from the leaf to the root of the tree. Otherwise the leaf is free. If a leaf $\bullet y$ is bound then the first λ-node $\bullet\lambda y$, in the path from the leaf to the root, is called the **binder** of the leaf. The only formal purpose of bound variables is to specify the binders of bound leaf nodes. So, instead of using construction trees with bound variables we may use **construction graphs** with a pointer up from each bound leaf nodes to their binder. The occurrences of the bound variables can then be removed. So, for example, the previous tree has the following graph.



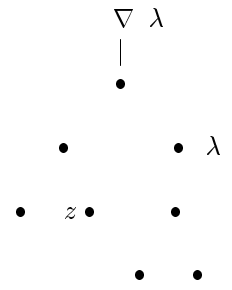
This is the construction graph of $(\lambda x.(\lambda y.((ax)((bx)y)))$.

As another example here is the construction graph of $\lambda y.(xy(\lambda y.y))$.



The two terms $\lambda z.(xz(\lambda x.x))$ and $\lambda v.(xv(\lambda x.x))$ have the same construction graph.

Such terms are said to be α-**convertible**. Here is yet another example of a construction graph and a pair of α-convertible terms for it.



1. $\lambda x.(xz(\lambda y.(xy)))$
2. $\lambda w.(wz(\lambda x.(wx)))$

The term 2 can be obtained from the term 1 by simultaneously relabelling x as w and y as x. Next we give three alternative notations for the graph.

Bourbaki $\lambda(\Box z(\lambda(\Box\Box)))$

de Bruijn $\lambda(1z(\lambda(21)))$

dual de Bruijn $\lambda(1z(\lambda(12)))$

The **de Bruijn index** of a bound leaf in a construction tree is the number of λ-nodes occuring in the path from the leaf up to and including the binder of the leaf node. The **dual de Bruijn index** is instead the number of λ-nodes in the path from and including the binder up to the root.

The **(dual) de Bruijn term** associated with a term is obtained by replacing each variable occuring at a bound leaf by the (dual) de Bruijn index of the leaf and then erasing the variable from each binder.

2.3.3 The de Bruijn terms

We first define the de Bruijn **pre-terms** using the BNF-style grammar equation

$$M ::= c \mid x \mid (MM) \mid (\lambda M) \mid 1 \mid 2 \mid 3 \mid \cdots.$$

For each variable x and each pre-term M let $(\lambda x.M)$ be the pre-term obtained from (λM) by simultaneously making the replacements

$$\begin{array}{c} x \to 1 \\ 1 \to 2 \\ 2 \to 3 \\ \vdots \end{array}$$

Example: If M is $\lambda(1z(\lambda(12)))$ then $\lambda z.M$ is $\lambda(\lambda(21(\lambda(23))))$.

Note that we are here preferring to take the dual de Bruijn approach. The set of **de Bruijn terms** is now defined to be the set of pre-terms inductively defined using the same rules as we used earlier to define the terms of LC, except now the lambda abstraction rule has to be understood in the sense we have just explained. So now there is no such thing as a bound variable in a de Bruijn term, only perhaps in our notation to refer to it. Instead the numerical indices are used.

From now on we will use the de Bruijn terms. For each pre-term M let $\mathsf{var}(M)$ be the set of variables that occur in M. Also let $M\{y/x\}$ be the result of replacing all occurrences of x in M by y.

Proposition 2.4 *$\lambda x.M = \lambda y.(M\{y/x\})$ if $y \notin \mathsf{var}(M)$.*

Proposition 2.5 *If (λN) is a (de Bruijn) term and $x \notin \mathsf{var}(N)$ then there is a unique term M such that $(\lambda N) = (\lambda x.M)$.*

2.3.4 Substitution

Given a variable y and a term N the function $M \mapsto M[N/y]$ from terms to terms is the unique function $(\ )'$ such that

$$\begin{cases} c' & = c \\ y' & = N \\ x' & = x \quad \text{if } x \neq y \\ (M_1M_2)' & = M_1'M_2' \\ (\lambda x.M)' & = (\lambda x.M') \quad \text{if } x \notin \mathsf{var}(N) \cup \{y\} \end{cases}$$

Proposition 2.6 *The function $M \mapsto M[N/y]$ is well defined.*

Note that this is not literal substitution as indices in N get shifted up 1 for each λ above it in the construction tree of N.

Lemma 2.7 (The Substitution Lemma)

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

if $x \neq y$ and $x \notin \mathsf{var}(L)$.

Proof: By induction on the structure of M.

2.4 Deduction

The **$\lambda\beta$-calculus** is the formal system, whose statements are equations between terms, having the following axiom schemes and rules of inference.

(1) $M = M$

(2) $\dfrac{M = N}{N = M}$

(3) $\dfrac{M = N \quad N = L}{M = L}$

(4) $\dfrac{M_1 = N_2 \quad M_2 = N_2}{M_1M_2 = N_1N_2}$

(ξ) $\dfrac{M = N}{\lambda x.M = \lambda x.N}$

(β) $(\lambda x.M)N = M[N/x]$

The $w\lambda\beta$-**calculus** is just like the $\lambda\beta$-calculus except that the rule (ξ) is left out. The $\lambda\beta\eta$-**calculus** is obtained from the $\lambda\beta$-calculus by adding the axiom scheme

(η) $\lambda x.(Mx) = M$, provided that $x \notin \mathsf{var}(M)$.

We will also consider the following rule.

(ext) $\dfrac{Mx = Nx}{M = N}$, provided that $x \notin \mathsf{var}(M) \cup \mathsf{var}(N)$.

There are weak versions $(w\eta)$ and $(wext)$ of (η) and (ext), where it is required that M, N have λ-form; i.e. are terms of the form $\lambda x.L$.

The Consistency Problem: Show that there are terms M, N such that $M = N$ is not a theorem of $(\lambda\beta\eta)$. See section 2.6.

Exercises 2.8

1. *Show that* $(\lambda\beta\eta) \equiv (w\lambda\beta) + (ext)$*; i.e. both sides have the same theorems.*
2. *Show that*
   (a) $(\lambda\beta) \vdash (w\eta)$,
   (b) $(\lambda\beta) \equiv (w\lambda\beta) + (wext)$.
3. *If* $Y = \lambda z.((\lambda x.z(xx))(\lambda x.z(xx)))$ *show that for any term* M
$$(w\lambda\beta) \vdash M(YM) = YM.$$
4. *If* S, K, I *are the terms given in section 2 show that*
$$(\lambda\beta) \vdash I = SKK.$$
   Can you show that
$$(w\lambda\beta) \vdash I = SKK?$$

2.5 Combinatory Logic

Very roughly, combinatory logic is an approach to the ideas of the lambda calculus that avoids lambda abstraction and bound variables. This means that substitution can be understood literally. The syntax of combinatory logic is given by the grammar

$$M ::= c \mid K \mid S \mid x \mid (MM)$$

So, instead of lambda abstraction there are the special symbols K, S. It turns out that in the formal system CL for combinatory logic, defined below, we can define a simulation of lambda abstraction so that $(w\lambda\beta)$ can be translated into CL. The main axioms for CL are the defining equations for K and S. There is an easy translation of CL into $(w\lambda\beta)$, so that these two formal systems are closely related. But the two translations are not exactly inverses of each other. Really $(w\lambda\beta)$ is a **conservative extension** of a subsystem that is equivalent to CL. The rule (ext) also makes sense for combinatory logic and we can show that $CL + (ext)$ is equivalent to $(\lambda\beta\eta)$. Also we can define a combinatory logic weakening $(wext)$ of (ext) and show that $CL + (wext)$ is equivalent to $(\lambda\beta)$.

2.5.1 CL and its translation into $(w\lambda\beta)$

The formal system CL consists of the axiom scheme and rules **(1)-(4)** from section 4, for equations between terms of combinatory logic, together with the axiom schemes

$$\begin{array}{lll}(K) & KMN & = M \\ (S) & SMNL & = ML(NL)\end{array}$$

where M, N, L are arbitrary combinatory logic terms. Below we will call such terms CL-**terms** in contrast to the terms of the lambda calculus which we will call LC-**terms**.

The translation $M \mapsto M_{LC}$ from CL-terms to LC terms is defined by structural recursion using the equations

$$\begin{cases} c_{LC} & = c \\ K_{LC} & = \lambda xy.x \\ S_{LC} & = \lambda xyz.xz(yz) \\ x_{LC} & = x \\ (MN)_{LC} & = M_{LC}N_{LC} \end{cases}$$

It is straightforward to prove the following result.

Proposition 2.9 $CL \vdash M = N \implies (w\lambda\beta) \vdash M_{LC} = N_{LC}$.

2.5.2 A translation of $(w\lambda\beta)$ into CL

In order to give such a translation we need to simulate lambda abstraction in CL. For each variable x we define the operation $M \mapsto \lambda^* x.M$ on CL-terms by structural recursion using the following equations.

$$\begin{cases} \lambda^* x.x & = I & (\text{where } I \text{ is } SKK) \\ \lambda^* x.M & = KM & (x \notin \mathsf{var}(M)) \\ \lambda^* x.(MN) & = S(\lambda^* x.M)(\lambda^* x.N) & (x \in \mathsf{var}(MN)) \end{cases}$$

Lemma 2.10 *For all CL-terms M, N*

$$CL \vdash (\lambda^* x.M)N = M[N/x].$$

The translation $M \mapsto M_{CL}$ of LC-terms into CL-terms is by structural recursion using the equations

$$\begin{cases} c_{CL} & = c \\ x_{CL} & = x \\ (MN)_{CL} & = M_{CL}N_{CL} \\ (\lambda x.M)_{CL} & = \lambda^* x.M_{CL} \end{cases}$$

Proposition 2.11 $(w\lambda\beta) \vdash M = N \implies CL \vdash M_{CL} = N_{CL}$.

2.5.3 The converse of Proposition 6

We have seen translations both ways between CL and $(w\lambda\beta)$. When extensionality is added to both formal systems then the translations carry over and are inverses of each other in a suitable sense, showing that $CL+(ext)$ is equivalent to $(w\lambda\beta)+(ext)$; i.e. $(\lambda\beta\eta)$. But the translations between CL and $(w\lambda\beta)$ are not quite inverses. By using a variant of λ^* we now show that '$\Longrightarrow$' in proposition 6 can be strengthened to '$\Longleftrightarrow$'. In the variant definition of λ^* we use the equation

$$\lambda^* x.(Mx) = M \quad (x \notin \mathsf{var}(M))$$

and only use the third equation in the previous definition of λ^* when the above equation does not apply.

Exercise 2.12 *Show, using this variant definition of λ^* when defining $M \mapsto M_{CL}$, that*

$$\begin{cases} (K_{LC})_{CL} & = K \\ (S_{LC})_{CL} & = S \end{cases}$$

and hence that

$$(M_{LC})_{CL} = M$$

for all CL-terms M. Note that these are syntactic identities and not equalities proved in CL.

Proposition 2.13

$$CL \vdash M = N \iff (w\lambda\beta) \vdash M_{LC} = N_{LC}$$

for all CL-terms M, N.

Proof: Note that Proposition 2.11 still holds when using the variant definition of λ^*. So if $(w\lambda\beta) \vdash M_{LC} = N_{LC}$ then $CL \vdash (M_{LC})_{CL} = (N_{LC})_{CL}$. So, by the exercise, $CL \vdash M = N$.

Exercise 2.14 *Show that $(w\lambda\beta)$ can be replaced by $(w\lambda\beta)+(\eta)$ in Proposition 2.11 and hence also in Proposition 2.13.*

2.5.4 The equivalence between $CL + (ext)$ and $(\lambda\beta\eta)$

Proposition 2.15

1. $CL + (ext) \vdash M = N \implies (\lambda\beta\eta) \vdash M_{LC} = N_{LC}$.
2. $(\lambda\beta\eta) \vdash M = N \implies CL + (ext) \vdash M_{CL} = N_{CL}$.
3. $CL + (ext) \vdash M = (M_{LC})_{CL}$ *for any CL-term M.*
4. $(\lambda\beta\eta) \vdash M = (M_{CL})_{LC}$ *for any LC-term M.*
5. *'$\implies$' in 1,2 can be replaced by '$\iff$'.*

2.5.5 The equivalence between $CL + (wext)$ and $(\lambda\beta)$

In combinatory logic the rule $(wext)$ is the weakening of the rule (ext) which requires that M, N are functional in the sense of the following definition.

Definition 2.16 *A CL-term is* **functional** *if it has one of the forms*

$$K, KM, S, SM, SMN.$$

Observe that, when using the original definition of λ^*, a CL-term $\lambda^* x.M$ is always functional so that M_{CL} is a functional term for any LC-term M that is in λ-form. But note that this observation does not work with the variant definition of λ^*!

Exercise 2.17 *Show that for each functional CL-term M there is a LC-term N in λ-form such that*

$$(\lambda\beta) \vdash M_{CL} = N.$$

Proposition 2.18 *The results of Proposition 2.15 hold when (ext) is replaced by $(wext)$ and $(\lambda\beta\eta)$ is replaced by $(\lambda\beta)$.*

2.5.6 Summary

The main relationships between combinatory logics and lambda calculi are summarised in the following diagram.

$$\begin{array}{ccccc} CL & \subseteq & CL+(wext) & \subseteq & CL+(ext) \\ \Downarrow\uparrow & & \Updownarrow & & \Updownarrow \\ w\lambda\beta & \subseteq & \lambda\beta & \subseteq & \lambda\beta\eta \end{array}$$

2.6 The Church-Rosser Theorem

The aim of this section is to prove the Church-Rosser theorem for the lambda calculus $(\lambda\beta)$. With some more work it can also be proved for the calculus $(\lambda\beta\eta)$. This result, for each calculus, has the immediate consequence that the calculus is consistent. Here, by the **consistency** of an equational calculus we mean the property that not all equations can be proved, or equivalently, that for distinct variables x, y the equation $(x = y)$ cannot be proved.

2.6.1 The reduction relation

A **redex** for $(\lambda\beta)$ is a term of the form

$$(\lambda x.M)N$$

and its **contractum** is the term $M[N/x]$. These are the β-**redexes**. For $(\lambda\beta\eta)$ there are also the η-**redexes**; i.e. terms of the form

$$\lambda x.(Mx)$$

where $x \notin \mathsf{var}(M)$, having contractum M. If M is a redex then we write $M^\wedge$ for its contractum.

Exercise 2.19 *Show that N contr N' if and only if it can be proved using the following rules.*

1. *M contr $M^\wedge$ for each redex M.*
2. *If M contr M' then*
   - *$\lambda x.M$ contr $\lambda x.M'$,*
   - *(ML) contr $(M'L)$ and (LM) contr (LM') for each term L.*

The following lemma will be useful.

Lemma 2.20 *Let L be a term and let z be a variable. If M is a redex then so is $M[L/z]$ and $M^\wedge[L/z] = (M[L/z])^\wedge$.*

Proof: Let M be the redex $((\lambda x.N)M_0)$. By Proposition 2.4 we may assume that the variables x, z are distinct so that

$$M[L/z] = ((\lambda x.(N[L/z]))(M_0[L/z]))$$

and $M[L/z]$ is a redex. So, by the Substitution Lemma 2.7

$$\begin{array}{rl} M^\wedge[L/z] & = N[M_0/x][L/z] \\ & = N[L/z][M_0[L/z]/x] \\ & = (((\lambda x.(N[L/z]))(M_0[L/z])))^\wedge \\ & = M[L/z]^\wedge \end{array}$$

For either calculus, given terms M, N we write that M **contracts to** M', abbreviated

$$M \; contr \; M',$$

if M' can be obtained from M by replacing an occurrence of a redex in M by its contractum. Also we write that M **reduces to** M', abbreviated

$$M \; red \; M'$$

if M' can be obtained from M by a sequence of zero, one or more contractions

$$M \; contr \; \cdots \; contr \; M'.$$

So the reduction relation *red* on terms is the reflexive, transitive closure of the contraction relation *contr*. We can now state the theorem.

Theorem 2.21 (Church-Rosser) *For both $(\lambda\beta)$ and $(\lambda\beta\eta)$, if both M red N and M red K then there is a term L such that both N red L and K red L.*

Note: Call a relation R on a set A **confluent on** A if

$$xRy, z \Longrightarrow \exists w{\in}A \; (y, zRw),$$

where xRy, z abbreviates $(xRy \wedge xRz)$ and y, zRw abbreviates $(yRw \wedge zRw)$. Then the above Church-Rosser Theorem states that the reduction relation is confluent for both calculi.

Exercise 2.22 *Given a relation R on a set A let $\sim_R$ be the relation on A given by*

$$y \sim_R z \iff \exists w{\in}A \; (y, zRw).$$

Show that R is confluent iff $\sim_R$ is an equivalence relation. Use the Church-Rosser Theorem to show that for both calculi

$$M \sim_{red} N \iff \vdash M = N.$$

Hence show that each calculus is consistent.

2.6.2 Proof of the Theorem for $(\lambda\beta)$

We will define a relation $\rhd$ on terms and show that:-

I If $M \; contr \; M'$ then $M \rhd M'$.

II If $M \rhd M'$ then $M \; red \; M'$.

III $\rhd$ is confluent.

We can deduce the theorem from these three properties in the following way. Let $M \; red \; N, K$. This means that there are sequences of zero, one or more contractions

$$M \; contr \; \cdots \; contr \; N \text{ and } M \; contr \; \cdots \; contr \; K$$

so that, by I,

$$M \rhd \cdots \rhd N \text{ and } M \rhd \cdots \rhd K.$$

By repeated use of III we can fill in the following rectangle by working from the top left hand corner to the bottom right hand corner.

$$\begin{array}{ccccc} M & \rhd & \cdots & \rhd & N \\ \triangledown & & \cdots & & \triangledown \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \triangledown & & \cdots & & \triangledown \\ K & \rhd & \cdots & \rhd & L \end{array}$$

In this way we eventually get a term L such that

$$N \rhd \cdots \rhd L \text{ and } K \rhd \cdots \rhd L.$$

By II and the transitivity of *red* we get that N, K *red* L.

It remains to define the relation $\rhd$ and prove I,II,III.

Definition 2.23 *We define '$M \rhd M'$' by recursion on the structure of M. There are three cases, depending on the form of M.*

$$x \rhd M' \iff [x = M']$$

$$\lambda x.N \rhd M' \iff (\exists N')[N \rhd N' \text{ and } \lambda x.N' = M']$$

$$(M_1 M_2) \rhd M' \iff (\exists M_1')(\exists M_2')[M_1 \rhd M_1' \,\&\, M_2 \rhd M_2' \,\&(M' = (M_1' M_2')) \text{ or } M \text{ and } (M_1' M_2') \text{ are redexes and } M' = (M_1' M_2')^\wedge]$$

It is immediate from this definition that we have the following lemma.

Lemma 2.24

1. *$x \rhd x$.*
2. *If $N \rhd N'$ then $\lambda x.N \rhd \lambda x.N'$.*
3. *If $M_1 \rhd M_1'$ and $M_2 \rhd M_2'$ then $(M_1 M_2) \rhd (M_1' M_2')$ and if $(M_1 M_2)$ is a redex then so is $(M_1' M_2')$ and $(M_1 M_2) \rhd (M_1' M_2')^\wedge$.*

Exercises 2.25

1. *Show that, for any term M, $M \rhd M$ and if M is a redex then $M \rhd M^\wedge$. Hence prove I, using the previous lemma.*
2. *Prove II by induction on the structure of M.*

In the proof of III we will need the following result.

Lemma 2.26 *If $M \rhd M'$ and $L \rhd L'$ then $M[L/z] \rhd M'[L'/z]$.*

Proof: Let $M \rhd M'$ and $L \rhd L'$.

$M = z$**:** Then $M' = z$ so that

$$M[L/z] = L \rhd L' = M'[L'/z].$$

$M = y$ **with** $y \neq z$**:** Then $M[L/z] = y$ and $M' = y$ so that $M'[L'/z] = y$.

$$M[L/z] = y \rhd y = M'[L'/z].$$

$M = \lambda x.N$**:** By Proposition 2.4 we may assume that $M[L/z] = \lambda x.(N[L/z])$. As $M \rhd M'$, $M' = \lambda x.N'$ for some N' such that $N \rhd N'$. By the induction hypothesis $N[L/z] \rhd N'[L'/z]$ so that

$$M[L/z] = \lambda x.(N[L/z]) \rhd \lambda x.(N'[L'/z]) = M'[L'/z].$$

$M = (M_1 M_2)$**:** As $M \rhd M'$ there are M_1', M_2' such that $M_1 \rhd M_1'$ and $M_2 \rhd M_2'$ and either (i) or (ii) below.

(i) $M' = (M_1' M_2')$.

(ii) Both M and $(M_1'M_2')$ are redexes and $M' = (M_1'M_2')^\wedge$.

By the induction hypothesis

$$(*) \quad M_1[L/z] \rhd M_1'[L'/z] \text{ and } M_2[L/z] \rhd M_2'[L'/z]$$

If (i) then

$$\begin{aligned} M[L/z] &= ((M_1[L/z])(M_2[L/z])) \\ &\rhd ((M_1'[L'/z])(M_2'[L'/z])) \\ &= M'[L'/z] \end{aligned}$$

If (ii) then

$$\begin{aligned} M[L/z] &= ((M_1[L/z])(M_2[L/z])) \\ &\rhd ((M_1'[L'/z])(M_2'[L'/z]))^\wedge \\ &= (M_1'M_2')[L'/z]^\wedge \\ &= (M_1'M_2')^\wedge[L'/z] \quad \text{by Lemma 2.20} \\ &= M'[L'/z] \end{aligned}$$

■

Corollary 2.27 *If (M_1M_2) is a redex and $M_1 \rhd M_1'$, $M_1 \rhd M_2'$ then $(M_1'M_2')$ is a redex and $(M_1M_2)^\wedge \rhd (M_1'M_2')^\wedge$.*

Proof of III We must show that $\rhd$ is confluent. This is a consequence of the lemma below that uses the following definition. For if $M \rhd N, K$ then, by the lemma, $N, K \rhd M^*$ and we are done.

Definition 2.28 *We define M^*, for each term M, by recursion on the structure of M. There are three cases.*

$$\begin{aligned} x^* &= x \\ (\lambda x.N)^* &= \lambda x.N^* \\ (M_1M_2)^* &= \begin{cases} (M_1^*M_2^*) & \text{if } (M_1M_2) \text{ is not a redex} \\ (M_1^*M_2^*)^\wedge & \text{if } (M_1M_2) \text{ is a redex} \end{cases} \end{aligned}$$

Note that in the last equation if (M_1M_2) is a redex then so is $(M_1^*M_2^*)$.

Lemma 2.29 *If $M \rhd M'$ then $M' \rhd M^*$.*

Proof: By induction on the structure of M. Let $M \rhd M'$. There are three cases.

$M = x$**:** Then $M' = x \rhd x = M^*$.

$M = \lambda x.N$**:** Then $M' = \lambda x.N'$ for some N' such that $N \rhd N'$ and, by the induction hypothesis, $N' \rhd N^*$ so that

$$M' = \lambda x.N' \rhd \lambda x.N^* = M^*.$$

$M = (M_1M_2)$**:** Then there are M_1', M_2', with $M_1 \rhd M_1'$ and $M_2 \rhd M_2'$, such that either (i) or (ii) below.

(i) $M' = (M_1'M_2')$.

(ii) Both M and $(M_1'M_2')$ are redexes and $M' = (M_1'M_2')^\wedge$.

By the induction hypothesis $M_1' \rhd M_1^*$ and $M_2' \rhd M_2^*$. If M is not a redex then

$$M' = (M_1'M_2') \rhd (M_1^*M_2^*) = M^*.$$

It remains to consider the case when M is a redex. Then $(M_1^*M_2^*)$ is also a redex and $M^* = (M_1^*M_2^*)^\wedge$ so that $(M_1'M_2') \rhd M^*$. Also, as $(M_1'M_2')$ is a redex, $(M_1'M_2')^\wedge \rhd M^*$. If (i) above then $M' = (M_1'M_2')$ and if (ii) above then $M' = (M_1'M_2')^\wedge$. In either case $M' \rhd M^*$.

■

2.7 Normalisation

The definitions in this section apply to each of the calculi $(\lambda\beta)$ and $(\lambda\beta\eta)$. A term is **normal** if no subterm is a redex. If M *red* M' and M' is normal then we call M' a **normal form of** M. A term M is **(weakly) normalisable** if M has a normal form and is **strongly normalisable** if every contraction path

$$M \textit{ contr } M' \textit{ contr } M'' \cdots$$

is finite.

Some Examples: Trivially all normal terms are normalisable and strongly normalisable. Examples of normal terms are

$$I = \lambda x.x,\ K = \lambda xy.x,\ S = \lambda xyz.xz(yz),\ \Omega_0 = \lambda x.(xx).$$

Note that $\Omega = \Omega_0\Omega_0$ is not normal. In fact Ω *contr* Ω so that Ω is not normalisable or strongly normalisable. Note that if $K' = KK\Omega$ then K' is not normal, but is normalisable as K' *contr* K. In contrast, as K' *contr*K', the term K' is not strongly normalisable.

Exercises 2.30

1. *Show that a term is normal in the $(\lambda\beta)$ calculus iff it has the form*

   $$\lambda x_1 \cdots x_n.aN_1 \cdots N_k$$

   where $n, k \geq 0$, $x_1, \ldots, x_n$ are variables, a is a constant or variable and $N_1, \ldots, N_k$ are normal terms. What happens in the $(\lambda\beta\eta)$ calculus?

2. *Show that every normalisable term has a unique normal form.*

3. *Show that every strongly normalisable term is normalisable.*

4. *Show that if M is strongly normalisable then the set of its contraction paths*

   $$M \textit{ contr } M' \textit{ contr } M'' \cdots$$

   is finite. [Hint: Use König's Lemma]

5. *Show that, for each of the two calculi, it is decidable whether or not an equation $(M = N)$ between normalisable terms M, N, is provable.*

3 Simply Typed Lambda Calculus

3.1 The Simple Type Theory *STT*

We assume given a set of **atomic types** and generate the types from the atomic types using the rule

If A_1, A_2 are types then so is $(A_1 \to A_2)$

The **typing judgments** have the form $M : A$ where M is a term of the untyped lambda calculus and A is a type. In particular a **variable declaration** is a typing judgment of the form $x : A$ where x is a variable. A sequence

$$x_1 : A_1, \ldots, x_n : A_n$$

of zero, one or more variable declarations, for a non-repeating list $x_1, \ldots, x_n$ of variables, is called a **context**.

The type theory STT is a formal system whose statements have the form

$$\Gamma \ \mapsto \blacktriangleright \ M : A$$

where Γ is a context and $M : A$ is a typing judgment. The axioms and rules of STT are given by the following schemes.

$$(ass) \qquad \Gamma \ \mapsto \blacktriangleright \ x : A \qquad\qquad (x : A \text{ in } \Gamma)$$

$$(appl) \qquad \frac{\Gamma \ \mapsto \blacktriangleright \ M : (B \to C) \qquad \Gamma \ \mapsto \blacktriangleright \ N : B}{\Gamma \ \mapsto \blacktriangleright \ (M\,N) : C}$$

$$(abstr) \qquad \frac{\Gamma, y : B \ \mapsto \blacktriangleright \ M : C}{\Gamma \ \mapsto \blacktriangleright \ \lambda y.M : (B \to C)}$$

Exercises 3.1

1. *In this exercise identify the (atomic) formulae of Intuitionistic Implicational Logic with the (atomic) types of STT. Let $x_1, x_2, \ldots$ be an infinite non-repeating list of variables. Show that*

$$A_1, \ldots, A_n \ \vdash\!\blacktriangleright_{ND} A \iff \exists M[x_1 : A_1, \ldots, x_n : A_n \ \vdash\!\blacktriangleright_{STT} M : A].$$

Moreover show that for all sequents $A_1, \ldots A_n \ \vdash\!\blacktriangleright A$ there is a one-one correspondence between the ND tree-proofs of $A_1, \ldots, A_n \ \vdash\!\blacktriangleright A$ and pairs consisting of a term M and an STT tree-proof of the sequent $x_1 : A_1, \ldots, x_n : A_n \ \vdash\!\blacktriangleright M : A$.

2. *In the following let Γ be a context $x_1 : A_1, \ldots, x_n : A_n$. Prove the following results.*

   (a) If $\Gamma \ \vdash\!\blacktriangleright_{STT} M : A$ then $\mathsf{var}(M) \subseteq \{x_1, \ldots, x_n\}$.

   (b) If Γ' is a context that includes every variable declaration of Γ then

$$\Gamma \ \vdash\!\blacktriangleright_{STT} M : A \implies \Gamma' \ \vdash\!\blacktriangleright_{STT} M : A.$$

   (c) If $\Delta \ \vdash\!\blacktriangleright_{STT} N_i : A_i$ for $i = 1, \ldots, n$ then

$$\Gamma \ \vdash\!\blacktriangleright_{STT} M : A \implies \Delta \ \vdash\!\blacktriangleright_{STT} M[N_1, \ldots N_n / x_1, \ldots x_n] : A.$$

   (d) If M red M' then

$$\Gamma \ \vdash\!\blacktriangleright_{STT} M : A \implies \Gamma \ \vdash\!\blacktriangleright_{STT} M' : A.$$

3.2 The variant STT'

The terms of STT' are like the untyped terms except that lambda abstractions have to include a typing of the variable being bound and so have the form $\lambda x : A.M$, where A is a type. The rules of STT' are just as the rules of STT except that the rule $(abstr)$ has to be replaced by the following rule.

$$(abstr)' \quad \frac{\Gamma, y : B \ \vdash\!\blacktriangleright M : C}{\Gamma \ \vdash\!\blacktriangleright \lambda y : B.M : (B \to C)}$$

Exercises 3.2

1. *If M is a term of STT' then let M^- be the untyped term obtained by erasing ': A' from each abstraction subterm $\lambda x : A.N$. Show that*

$$\Gamma \ \vdash\!\blacktriangleright_{STT} M : A \iff \exists M'[M = M'^- \text{ and } \Gamma \ \vdash\!\blacktriangleright_{STT'} M' : A].$$

2. *Show that if there is a type A such that $\Gamma \ \vdash\!\blacktriangleright_{STT'} M : A$ then there is a unique such A and for that A there is a unique tree-proof in STT' of $\Gamma \ \vdash\!\blacktriangleright M : A$.*

By the result of the second exercise we may write $type_\Gamma(M)$ for the unique type A such that $\Gamma \ \vdash\!\blacktriangleright_{STT'} M : A$, when it exists. Also note that if Γ is the context $x_1 : A_1, \ldots, x_n : A_n$ then the terms M of STT' such that $\Gamma \ \vdash\!\blacktriangleright_{STT'} M : A$ are in one-one correspondence with the ND tree-proofs of $A_1, \ldots, A_n \ \vdash\!\blacktriangleright A$. So these terms M can be viewed as formal expressions that represent the corresponding ND tree-proofs.

3.2.1 Standard Set Theoretical Models of STT'

Given an assignment of a set $[\![A_0]\!]$ to each atomic type A_0 we may define a **standard set theoretical model for** STT' in the following way. Define $[\![A]\!]$ for each type A by structural recursion using the equation

$$[\![A_1 \to A_2]\!] = [\![A_2]\!]^{[\![A_1]\!]},$$

where, for sets X, Y, Y^X is the set of all the functions from X to Y. Also, if Γ is the context $x_1 : A_1, \ldots, x_n : A_n$ then let

$$[\![\Gamma]\!] = [\![A_1]\!] \times \cdots \times [\![A_n]\!].$$

Now, by structural recursion on M, whenever $\Gamma \mapsto_{STT'} M : A$ we may assign a function

$$[\![M]\!]_\Gamma : [\![\Gamma]\!] \to [\![A]\!]$$

using the following equations, where we let $\vec{a} = (a_1, \ldots, a_n) \in [\![\Gamma]\!]$.

$$\begin{cases} [\![x_i]\!]_\Gamma(\vec{a}) & = a_i \qquad (i = 1, \ldots, n) \\ [\![(MN)]\!]_\Gamma(\vec{a}) & = [\![M]\!]_\Gamma(\vec{a})([\![N]\!]_\Gamma(\vec{a})) \\ [\![\lambda y : B.M]\!]_\Gamma(\vec{a}) & = (\boldsymbol{\lambda} b \in [\![B]\!])[\![M]\!]_{\Gamma, y:B}(\vec{a}, b) \end{cases}$$

where, in the last equation the right hand side is the function $f : [\![B]\!] \to [\![C]\!]$ such that for all $b \in [\![B]\!]$

$$f(b) = [\![M]\!]_{\Gamma, y:B}(\vec{a}, b).$$

Note that it would not be so easy to formulate a notion of set theoretical model for the type theory STT.

3.3 Standard Term Models of STT

The notion of a standard term model of STT will be a useful tool in describing a variety of constructions for proving that every STT-term is in a set S, for various sets S such as the set $\mathcal{N}$ of normalisable terms or the set $\mathcal{SN}$ of strongly normalisable terms.

Call an untyped term M an STT-term if $\Gamma \mapsto_{STT} M : A$ for some context Γ and some type A. In this section we introduce a useful notion of term model for STT. Let $\mathcal{T}$ be the set of untyped terms. For subsets X, Y of $\mathcal{T}$ let $X \to Y$ and $X \to^w Y$ be given by the following definitions.

$$\begin{array}{ll} X \to Y & = \{M \mid \forall N \in X \ (MN) \in Y\} \\ X \to^w Y & = \{\lambda x.M \mid \forall N \in X \ M[N/x] \in Y\} \end{array}$$

Definition 3.3 *A* **standard term model of** STT *is an assignment of a subset* $[\![A]\!]$ *of* $\mathcal{T}$ *to each type* A *such that for all types* A_1, A_2

$$[\![A_1]\!] \to^w [\![A_2]\!] \ \subseteq \ [\![A_1 \to A_2]\!] \ \subseteq \ [\![A_1]\!] \to [\![A_2]\!].$$

Given such a model let $x_1 : A_1, \ldots x_n : A_n \models M : A$ if

$$M[N_1, \ldots, N_n / x_1, \ldots, x_n] \in [\![A]\!]$$

for all $N_1 \in [\![A_1]\!], \ldots, N_n \in [\![A_n]\!]$.

Theorem 3.4 (Soundness) *Given a standard term model of* STT

$$\Gamma \mapsto_{STT} M : A \implies \Gamma \models M : A.$$

Proof: This is by a straightforward induction on the size of an STT tree-proof of $\Gamma \vdash\!\blacktriangleright M : A$. The base case, for the axioms (ass), is trivial. The induction step for the rule ($appl$) uses

$$[\![B \to C]\!] \subseteq [\![B]\!] \to [\![C]\!].$$

The induction step for the rule ($abstr$) uses

$$[\![B]\!] \to^w [\![C]\!] \subseteq [\![B \to C]\!].$$

But some care is needed to relabel the variable y, given an instance of the rule

$$\frac{\Gamma, y : B \vdash\!\blacktriangleright M : C}{\Gamma \vdash\!\blacktriangleright \lambda y.M : (B \to C)}.$$

The induction hypothesis is $\Gamma, y : B \models M : C$. If Γ is the context $x_1 : A_1, \ldots, x_n : A_n$ then this means that for all $N_1 \in [\![A_1]\!], \ldots, N_n \in [\![A_n]\!]$

$$(*) \qquad (\forall N \in [\![B]\!])\ M[N_1, \ldots, N_n, N/x_1, \ldots, x_n, y] \in [\![C]\!].$$

We can rewrite $(*)$ as

$$(\forall N \in [\![B]\!]) M'[N/y'] \in [\![C]\!]$$

where

$$M' = M[y'/y][N_1, \ldots, N_n/x_1, \ldots, x_n],$$

provided that y' is a fresh variable (i.e. distinct from each of $x_1, \ldots, x_n, y$ and not occurring in any of $M, N_1, \ldots, N_n, N$). So we get from (*) that

$$\lambda y'.M' \in [\![B]\!] \to^w [\![C]\!] \subseteq [\![B \to C]\!].$$

Now observe that

$$\lambda y'.M' = (\lambda y.M)[N_1, \ldots, N_n/x_1, \ldots, x_n].$$

It follows that for all $N_1 \in [\![A_1]\!], \ldots, N_n \in [\![A_n]\!]$

$$(\lambda y.M)[N_1, \ldots, N_n/x_1, \ldots, x_n]) \in [\![B \to C]\!]$$

so that we have $\Gamma \models \lambda y.M : (B \to C)$ as required.

■

Definition 3.5 *A non-empty collection $\mathcal{C}$ of sets of terms is defined to be an STT-*__collection__ *if there is a binary operation m on $\mathcal{C}$ such that for all $X, Y \in \mathcal{C}$*

$$X \to^w Y \subseteq m(X, Y) \subseteq X \to Y.$$

We call such an opeartion m an __implication operation__ *for $\mathcal{C}$. Call an STT-collection $\mathcal{C}$* __variable rich__ *if every variable is in every set in $\mathcal{C}$.*

Theorem 3.6 *Let S be a set of terms such that there is a variable rich STT-collection of subsets of S. Then every STT-term is in S.*

Proof: Let $\mathcal{C}$ be a variable rich STT-collection of subsets of S and choose any $X_0 \in \mathcal{C}$. We may define a standard term model of STT as follows. We recursively define $[\![A]\!]$ for all types A by letting $[\![A_0]\!] = X_0$ for each atomic type A_0 and letting

$$[\![A_1 \to A_2]\!] = m([\![A_1]\!], [\![A_2]\!]).$$

Now if M is an STT-term then there is a context Γ and a type A such that $\Gamma \vdash\!\blacktriangleright {}_{STT} M : A$. So, by the previous theorem, $\Gamma \models M : A$. If Γ is the context $x_1 : A_1, \ldots, x_n : A_n$ then $M \in [\![A]\!]$, as $x_1 \in [\![A_1]\!], \ldots, x_n \in [\![A_n]\!]$, and hence $M \in S$.

■

3.4 The 'Normal Relation' method for STT

We will define the notion of a 'regular' set of terms S and the notion of a 'normal relation' for a set of terms S and prove the result that if a regular set S has a normal relation then we can define a simple variable rich STT-collection of subsets of S so that every STT-term is in S. We will apply this result to both the sets $\mathcal{N}$ and $\mathcal{SN}$. In this way we cover a variety of constructions that give normalisation proofs for STT.

For each set S of terms let S^* be the set of terms that have the form

$$xN_1 \cdots N_k$$

where $x \in Var$, $k \geq 0$ and $N_1, \ldots, N_k \in S$. Observe that we always have $S^* \subseteq (S \to S^*)$. We define S to be a **regular set** if $S^* \subseteq S$ and also whenever $(Mx) \in S$, with $x \notin \mathsf{var}(M)$ then $M \in S$.

Exercise 3.7 *Show that both the set $\mathcal{N}$ of normalisable terms and the set $\mathcal{SN}$ of strongly normalisable terms are regular.*

Lemma 3.8 *If S is a regular set then*

$$S^* \subseteq X, Y \subseteq S \implies S^* \subseteq (X \to Y) \subseteq S.$$

Proof: Let S be a regular set. First observe that $(S^* \to S) \subseteq S$. For if $M \in (S^* \to S)$ then choose a variable x that is not in $\mathsf{var}(M)$. Then, as $x \in S^*$, $(Mx) \in S$ so that $M \in S$, as S is regular. Now let $S^* \subseteq X, Y \subseteq S$. Then, as the binary operation $\to$ on sets of terms is antimonotone in its first argument and monotone in its second,

$$S^* \subseteq (S \to S^*) \subseteq (X \to Y) \subseteq (S^* \to S) \subseteq S.$$

■

Let S be a set of terms. A relation $\mathcal{R}$ on terms is defined to be S-**invariant** if

$$M \mathcal{R} M' \implies (MN) \mathcal{R} (M'N)$$

for all $N \in S$. Note that the reduction relation red is always S-invariant.

Lemma 3.9 *Let X, Y be sets of terms. If $\mathcal{R}$ is an X-invariant relation then*

$$Y \text{ is } \mathcal{R}\text{-closed} \implies (X \to Y) \text{ is } \mathcal{R}\text{-closed}.$$

Proof: Let $\mathcal{R}$ be an X-invariant relation and let Y be $\mathcal{R}$-closed. Given $M \mathcal{R} M' \in (X \to Y)$ we must show that $M \in (X \to Y)$. So let $N \in X$. Then $(MN) \mathcal{R} (M'N) \in Y$, as $\mathcal{R}$ is X-invariant, so that $(MN) \in Y$, as Y is $\mathcal{R}$-closed.

■

Call a redex $((\lambda x.M)N)$ an S-**redex** if $N \in S$. Let $\mathcal{R}_S$ be the relation on terms

$$\mathcal{R}_S = \{(M, M^\wedge) \mid M \text{ is an } S\text{-redex}\}.$$

Given a relation $\mathcal{R}$ on terms, S is $\mathcal{R}$-**closed** if, whenever $M \mathcal{R} M'$,

$$M' \in S \implies M \in S.$$

Lemma 3.10 *If X, Y are sets of terms such that Y is $\mathcal{R}_X$-closed then $(X \to^w Y) \subseteq (X \to Y)$.*

Proof: Let Y be $\mathcal{R}_X$-closed and let $\lambda x.M \in (X \to^w Y)$. If $N \in X$ then $L = ((\lambda x.M)N)$ is an X-redex, so that $L\mathcal{R}_X L^\wedge$. But $L^\wedge = M[N/x] \in Y$ so that $L \in Y$ as required.

■

We call a relation $\mathcal{R}$ on terms **normal** for a set S of terms if the following conditions hold.

1. $\mathcal{R}_S \subseteq \mathcal{R}$,
2. $\mathcal{R}$ is S-invariant,
3. S is $\mathcal{R}$-closed.

Call an STT-collection $\mathcal{C}$ **simple** if $(X \to Y) \in \mathcal{C}$ for all $X, Y \in \mathcal{C}$. It follows that $m_{max}(X,Y) = (X \to Y)$ defines an implication operation for $\mathcal{C}$, the maximum possible one.

Theorem 3.11 *If S is a regular set having a normal relation $\mathcal{R}$ then the collection*

$$\mathcal{C}_\mathcal{R}(S) = \{X \mid X \text{ is } \mathcal{R}\text{-closed and } S^* \subseteq X \subseteq S\}$$

is a simple variable rich STT-collection of subsets of S.

Proof: As S is $\mathcal{R}$-closed, $S \in \mathcal{C}_\mathcal{R}(S)$, so that $\mathcal{C}_\mathcal{R}(S)$ is non-empty. For $X, Y \in \mathcal{C}_\mathcal{R}(S)$ let $m_{max}(X,Y) = (X \to Y)$. By Lemma 3.8, $S^* \subseteq m_{max}(X,Y) \subseteq S$. As $X \subseteq S$, $\mathcal{R}$ is X-invariant, so that, by Lemma 3.9, $m_{max}(X,Y)$ is $\mathcal{R}$-closed, as Y is $\mathcal{R}$-closed. Thus $m_{max}(X,Y) \in \mathcal{C}_\mathcal{R}(S)$. Finally, as $X \subseteq S$, $\mathcal{R}_X \subseteq \mathcal{R}_S \subseteq \mathcal{R}$ so that Y is $\mathcal{R}_X$-closed and hence, by Lemma 3.10,

$$(X \to^w Y) \subseteq m_{max}(X,Y) = (X \to Y).$$

Thus $\mathcal{C}_\mathcal{R}(S)$ is a simple STT-collection. It is variable rich because $Var \subseteq S^* \subseteq X$ for any $X \in \mathcal{C}_\mathcal{R}(S)$.

■

Note: Another possible implication operation for $\mathcal{C}_\mathcal{R}(S)$ is given by

$$m_{min}(X,Y) = (S^* \cup (X \to^w Y))^\mathcal{R},$$

where, for any set X of terms, $X^\mathcal{R}$ is the smallest $\mathcal{R}$-closed set that includes X. It is the set of those terms M such that there is an $\mathcal{R}$-path

$$M\mathcal{R}M^{(1)}\mathcal{R}M^{(2)}\mathcal{R}\cdots\mathcal{R}M^{(n)} \in X$$

with $n \geq 0$.

Corollary 3.12 *If S is a regular set that has a normal relation then every STT-term is in S.*

Given any relation on terms $\mathcal{R}$, we may form the smallest S-invariant relation, $\mathcal{R}^S$, that includes $\mathcal{R}$. In fact it consists of all pairs $(MN_1\cdots N_k, M'N_1\cdots N_k)$ such that $M\mathcal{R}M'$ and $N_1, \ldots N_k, \in S$ with $k \geq 0$.

Exercises 3.13 *If S is a regular set show that*

1. *If S has a normal relation then $(\mathcal{R}_S)^S$ is its smallest normal relation.*
2. *S has a normal relation if and only if S is $(\mathcal{R}_S)^S$-closed.*
3. *If $\mathcal{R}$ is a normal relation for S and S' is a regular subset of S that is $\mathcal{R}$-closed then $\mathcal{R}$ is also a normal relation for S'*

Theorem 3.14

1. *red is a normal relation for the regular set $\mathcal{N}$.*
2. *$(\mathcal{R}_{\mathcal{N}})^S$ is a normal relation for $\mathcal{N}$ whenever $\mathcal{N} \subseteq S$.*
3. *$(\mathcal{R}_{\mathcal{SN}})^S$ is a normal relation for the regular set $\mathcal{SN}$ whenever $\mathcal{SN} \subseteq S$.*

By applying the corollary to this result we get several constructions that give the normalisation theorem for STT, some of them also giving the stronger result for strong normalisation.

Exercise 3.15 *The* **left-most redex** *of a non-normal term is that redex that starts furthest to the left among all redexes. The* **left-most contraction** *is the contraction of the left-most redex. A* **left-most reduction** *is a sequence of left-most contractions. Let $\mathcal{LN}$ be the set of those terms having a left-most reduction to normal form. Show that $\mathcal{LN}$ is regular and has a normal relation.*

3.5 Girard's method

Girard's 'candidate de reducibilité' method for proving strong normalisation involves a slightly different simple variable rich STT-collection of subsets of $\mathcal{SN}$ than those of the form $\mathcal{C}_{\mathcal{R}}(\mathcal{SN})$ used in the previous section. We leave the details as exercises.

Definition 3.16 *We define a set of terms X to be a CR-***set** *if the following three conditions hold.*

(CR1) $X \subseteq \mathcal{SN}$,

(CR2) *If $M \in X$ then $[M \text{ contr } N \implies N \in X]$,*

(CR3) *If M does not have the form $\lambda x.M_0$ and $\forall N[M \text{ contr } N \implies N \in X]$ then $M \in X$.*

Let $\mathcal{CR} = \{X \mid X \text{ is a } CR\text{-set}\}$.

Exercises 3.17

1. *$\mathcal{SN} \in \mathcal{CR}$,*
2. *$\mathcal{CR}$ is a simple variable rich STT-collection of subsets of $\mathcal{SN}$,*
3. *If $\mathcal{R} = (\mathcal{R}_{\mathcal{SN}})^{\mathcal{T}}$ then $\mathcal{CR} \subseteq \mathcal{C}_{\mathcal{R}}(\mathcal{SN})$.*

3.6 Another method for strong normalisation

We describe a method for proving strong normalisation for STT-terms that uses an STT-collection of sets of variable free terms. Let $\mathcal{T}^0$ be the set of all variable free terms and for any set S of terms let $S^0 = \mathcal{T}^0 \cap S$.

Theorem 3.18 *Let S be a set of terms such that*

$$\lambda x.M \in S \implies M \in S.$$

If there is an STT-collection of subsets of S then every STT-term is in S.

Theorem 3.19 *Let S be a set of terms such that for all terms $\lambda x.M$*

$$(*) \qquad \exists N(M[N/x] \in S) \implies \lambda x.M \in S.$$

If S has a normal relation $\mathcal{R}$ then

$$\mathcal{C}'_{\mathcal{R}}(S) = \{X \subseteq S \mid X \text{ is a non-empty } \mathcal{R}\text{-closed set}\}$$

is an STT-collection of subsets of S.

Proof: We define the implication operation for this collection by

$$m(X, Y) = S \cap (X \to Y)$$

and apply the following result.

Lemma 3.20

1. $Y \neq \emptyset \implies (X \to^w Y) \neq \emptyset$,
2. *If (*) then* $[X \neq \emptyset \text{ and } Y \subseteq S] \implies (X \to^w Y) \subseteq S$.

Exercise 3.21 *Show that $(\mathcal{R}_{\mathcal{SN}^0})^S$ is a normal relation for $\mathcal{SN}^0$ whenever $\mathcal{SN}^0 \subseteq S$.*

By this exercise $\mathcal{SN}^0$ has a normal relation so that, as $\mathcal{SN}^0$ is a set S satisfying (*), by the theorem there is an STT-collection of subsets of $\mathcal{SN}^0$. This is an STT-collection of subsets of $\mathcal{SN}$ so that, by Theorem 3.6, as $\lambda x.M \in \mathcal{SN} \implies M \in \mathcal{SN}$, every STT-term is in $\mathcal{SN}$.

References

[1] H. Barendregt, *The Lambda Calculus*, North Holland Publishing Co., Amsterdam. 2nd edition, 1984.

[2] J. Gallier, "On Girard's "Candidats de Reductibilité", in *Logic and Computer Science*, ed. P. Odifreddi, Vol. 31 in APIC studies in Data Processing, pp. 123-203, Academic Press, 1990.

[3] J.Gallier, "Constructive Logics, Part I: A tutorial on proof systems and typed λ-calculi", *Theoretical Computer Science*, 110, 249-339, 1991.

[4] J. Girard, Y. Lafont and P. Taylor, *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press, 1989.

[5] J. Lambek and P.J. Scott, *Introduction to higher order categorical logic*, Cambridge Studies in Advanced Mathematics 7, Cambridge University Press, 1986.

[6] P. Martin-Lof, *Intuitionistic Type Theory*, Bibliopolis, Napoli, 1984.

[7] B. Nordstrom, K. Petersson and J. Smith, *Programming in Martin-Lof's Type Theory, An Inroduction*, Monographs on Computer Science 7, Oxford University Press, 1990.

[8] B. Nordstrom and K. Petersson and J. Smith, "Martin-Lof's Type Theory." A chapter in *Handbook of Logic in Computer Science*, written in 1994, to appear. (Available by ftp from ftp://ftp.cs.chalmers.se/pub/cs-reports/papers/smith/hlcs.ps.gz)

[9] M. Takahashi, "Parallel Reductions in λ-Calculus", *Information and Computation*, 118, 120-127, 1995.

[10] A. S. Troelstra and H. Schwichtenberg, *Basic Proof Theory*, Cambridge Tracts in Theoretical Computer Science 43, Cambridge University Press, 1996.