Types and Lambda Calculus Notes

by Tyler Wright

github.com/Fluxanoia fluxanoia.co.uk

These notes are not necessarily correct, consistent, representative of the course as it stands today or, rigorous. Any result of the above is not the author's fault.

Contents

1 The Axioms of Lambda Calculus **3**
1.1 Axiom of Variables 3
1.2 Axiom of Application 3
1.3 Axiom of Abstraction 3
1.4 Subterms 3
1.5 Syntactical Conventions 4

2 Alpha Equivalence **5**
2.1 Free Variables 5
2.2 Substitution 5
2.3 Alpha Equivalence 5
2.4 The Variable Convention 5

1 The Axioms of Lambda Calculus

We suppose that we have a countably infinite set of variables $\mathbb{V}$ (we usually refer to elements of this set as x, y, z, etc.), from this we define the alphabet of lambda calculus $\mathbb{V} + \{\lambda, ., (,)\}$.

The set of terms of lambda calculus Λ is defined inductively by the axioms. A string is a term if and only if there is a proof tree using the following axioms that concludes it is a term.

1.1 Axiom of Variables

For some x in $\mathbb{V}$, we have the axiom:

$$\frac{}{x \in \Lambda},$$

which says that each variable is a term.

1.2 Axiom of Application

We have the axiom:

$$\frac{M \in \Lambda \qquad N \in \Lambda}{(MN) \in \Lambda},$$

which says that for some terms M and N, (MN) is also a term.

1.3 Axiom of Abstraction

For some x in $\mathbb{V}$, we have the axiom:

$$\frac{M \in \Lambda}{(\lambda x.M) \in \Lambda},$$

which says that for some term M, $(\lambda x.M)$ is also a term.

1.4 Subterms

Subterms of a term M are substrings of M that are themselves terms and not captured by a λ (directly preceeded by).

1.5 Syntactical Conventions

Parentheses allow our lambda calculus to be unambigious, but for the sake of simplicity, we will construct conventions that will allow us to retain unique meaning with less parentheses:

- Omit outermost parentheses,
- Terms associate to the left, (MNP) parses as $((MN)P)$,
- Bodies of abstractions end at parentheses, $(\lambda x.MN)$ parses as $(\lambda x.(MN))$,
- Group repeated abstractions, $(\lambda xy.M)$ parses as $(\lambda x.(\lambda y.M))$.

2 Alpha Equivalence

2.1 Free Variables

We define the function $FV : \Lambda \to \mathcal{P}(\mathbb{V})$, which returns the set of variables contained within a term M that are not bound. We define it recursively on the structure of terms:

$$\begin{aligned} FV(x) &= \{x\}, \\ FV(MN) &= FV(M) \cup FV(N), \\ FV(\lambda x.M) &= FV(M) \backslash \{x\}. \end{aligned}$$

If a term has no free variables we say it is closed, and if a term has at least one free variable then we say it is open. The set of all closed terms is denoted by Λ^0.

2.2 Substitution

We define 'capture-avoiding' substitution of a term M for a variable x recursively on the structure of terms:

$$\begin{array}{rcll} y[M/x] & = & y & \text{if } y \neq x, \\ y[M/x] & = & M & \text{if } y = x, \\ (PQ)[M/x] & = & P[M/x] \cup Q[M/x], & \\ (\lambda y.P)[M/x] & = & \lambda y.P & \text{if } y = x, \\ (\lambda y.P)[M/x] & = & \lambda y.P[M/x] & \text{if } y \neq x \text{ and } y \notin FV(M). \end{array}$$

On the final case, we stipulate that y cannot be a free variable of M because otherwise free variables in the substitution could be captured by the lambda.

2.3 Alpha Equivalence

Suppose we have a term $\lambda x.M$ and y in $\mathbb{V} \setminus FV(M)$, then substituting y for x is a change of bound variable name. If two terms can be made identical through changes of bound variable name, they are α-equivalent. The set of λ-terms is the set Λ under α-equivalence.

This equivalence is much more useful to us than string comparision, so for the remainder of the notes we will always be referring to λ-terms as terms.

2.4 The Variable Convention

For $M_1, \ldots, M_k$ terms occuring in the same scope, we assume each term has distinct bound variables. We can make this assumption as otherwise, we can use changes of bound variable names to make it so.