1. Simply typed λ-calculus (ala Church)

Definition: A **simple type** is a propositional formula built from atoms and the connective $\rightarrow$. The set of all simple types is denoted $\Phi_{\rightarrow}$.

Definition: An **environment** Γ is a finite set of pairs of the form $\{x_1 : \tau_1, ..., x_n : \tau_n\}$, where the x_i are variables and the τ_i are types.

$x_1 : \tau_1$ means x_1 has type τ_1. Often this is written $x_1^{\tau_1}$ to improve readability.

Definition: We say M is a **term of type τ** in Γ, written $\Gamma \vdash M : \tau$, when M can be derived using the following rules.

$$\text{(Var)} \qquad \Gamma, x : \tau \vdash x : \tau$$

$$\text{(Abs)} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma\ M) : \sigma \rightarrow \tau}$$

$$\text{(App)} \qquad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\,N) : \tau}$$

Note in particular that it is only sensible to talk about applications when the types are compatible. There exist λ-terms which cannot be assigned types according to the above rules. One example is $(\lambda x.xx)$.

Examples:

$\vdash \lambda x^{\sigma}.x : \sigma \rightarrow \sigma$

$\vdash \lambda x^{\sigma} y^{\tau}.x : \sigma \rightarrow \tau \rightarrow \sigma$ (where $\rightarrow$ is right-associative; that is $\sigma \rightarrow \tau \rightarrow \sigma = \sigma \rightarrow (\tau \rightarrow \sigma)$)

$\vdash \lambda x^{\sigma \rightarrow \tau \rightarrow \rho} y^{\sigma \rightarrow \tau} z^{\sigma}.xz(yz) : (\sigma \rightarrow \tau \rightarrow \rho) \rightarrow (\sigma \rightarrow \tau) \rightarrow \sigma \rightarrow \rho$

Some useful properties of the simply typed λ-calculus include:

Theorem: (subject reduction) If $\Gamma \vdash M : \sigma$ and $M \rightarrow_{\beta} N$ then $\Gamma \vdash N : \sigma$.

This is important as it tells us that well-typed expressions remain well-typed after reductions.

Theorem: If $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$ then $\sigma = \tau$.

(Aside: there exists a variant of the simply typed λ-calculus in which bound variables are not given types, in which case the previous theorem does not hold; for instance, $\lambda x.x$ has both the type $\sigma \rightarrow \sigma$ and the type $\tau \rightarrow \tau$.)

2. Normalization properties

Definition: A λ-term M is said to be in **normal form** if M contains no β-redex; that is, M contains no subterm of the form $(\lambda x.P)Q$.

Definition: Let M be a λ-term. We say that M is **weakly normalizing** if there exists some M' in normal form such that $M \twoheadrightarrow_{\beta} M'$. We say M is **strongly normalizing** if there does not exist some infinite β-reduction path starting from M.

There exist terms in the untyped λ-calculus which are neither strongly nor weakly normalizing; for instance $(\lambda x.xx)(\lambda x.xx)$, as well as terms which are weakly but not strongly normalizing such as $(\lambda xy.y)((\lambda x.xx)(\lambda x.xx))$. However, we will see that in the simply typed λ-calculus, every term is strongly normalizing.

Definition: Suppose that M is strongly normalizing. Then $\nu(M)$ denotes the maximum possible length of a normalization sequence beginning with N.

The fact that ν is well defined (that is, always finite) deserves justification; perhaps M could contain infinitely many reduction paths each of finite length, but unbounded overall. To see that this cannot happen, we note that any given λ-term is finite and hence contains only finitely many β-redexes. Hence at any given point in a β-reduction path, we only have finitely many choices as to which β-redex to reduce next. Hence we can apply König's Lemma; a finitely branching tree with no infinite branch is

finite. Specifically, our nodes correspond to λ-terms, and our edges correspond to the choices we can make at a given node.

Definition: Let τ be a type. The **degree** of a type τ is given by:

$$deg(\tau) = \begin{cases} 0 & \text{if } \tau \text{ is atomic} \\ 1 + \max\{deg(\sigma), deg(\rho)\} & \text{if } \tau = \sigma \to \rho \end{cases}$$

Essentially, one should think of the degree of a type as its "complexity", a measure of the maximum amount of recursion it contains.

This notion of degree of type allows us to quite readily prove the **weak normalisation theorem** for simply typed lambda calculus; that is, the statement that every term has a normal form. As a proof sketch, the idea is to come up with a β-reduction strategy that is guaranteed to terminate in a normal form. Given a term M containing one or more β-redexes, the strategy will be to reduce the rightmost redex within M with type of maximum degree n. This may produce new redexes, or copy existing ones, but any such redexes are guaranteed to have type of degree less than n. Hence each reduction decreases the number of redexes of maximal degree; repeating this process is therefore guaranteed to terminate in a normal form.

While the simply typed λ-calculus is strongly normalizing, this important property does not come for free. In particular, untyped λ-calculus is well known to be Turing complete, which is not true of simply typed systems. One hint that this is the case is that the strong normalization theorem essentially tells us that any function which is able to be encoded by the simply typed system must be calculable by a program which halts.

In fact the expressive power of the simply typed system is even weaker than we would hope. The exact class of functions it can define are the **extended polynomials**; the smallest class of functions closed under compositions, and containing the constants 0 and 1, projections, addition, multiplication, and the conditional function ($cond(a, b, c) = b$ if $a = 0$, and $= c$ otherwise). In particular, we do not have primitive recursion or minimisation in general.

One possible modification of the simply typed calculus which increases the expressive power while still maintaining strong normalization is System F, which introduces polymorphism.

3. Proof of the strong normalisation theorem

Definition: We first define $\boldsymbol{RED_\tau}$**: the set of reducible terms of type** $\boldsymbol{\tau}$, by induction on types τ.

1. If τ is atomic, then a term of type τ is reducible iff it is strongly normalizable.

2. Otherwise $\tau = \sigma \to \rho$. A term M of type τ is reducible iff, for every reducible term N of type σ, $M\,N$ is reducible of type ρ.

Lemma 1: If $M \in RED_\tau$, then M is strongly normalizable.

Proof: Prove by induction on (degree of) type. This is true by definition if τ atomic. Otherwise, let $\tau = \sigma \to \rho$, and assume result is true for each type of degree less than $deg(\tau)$. Let N be any reducible term of type σ (For instance, setting $N = x^\sigma$ works). Since M is reducible, we know that $M\,N$ is reducible and of type ρ. Since $deg(\rho) < deg(\sigma \to \rho)$, by the induction hypothesis $M\,N$ is strongly normalizing. Hence M must also be strongly normalizing; for if there existed an infinite reduction path in M, there would also exist one in $M\,N$, as we could simply expand within M in exactly the same way, ignoring the presence of N entirely.

Lemma 2: If $M \in RED_\tau$ and $M \twoheadrightarrow_\beta M'$, then $M' \in RED_\tau$.

Proof: Prove by induction on degree of type. If τ is atomic, then M is strongly normalizable by definition; hence every term M' to which M reduces is also strongly normalizable.

Otherwise, let $\tau = \sigma \to \rho$, and assume the result is true for each type of degree less than $deg(\tau)$. Let $M \in RED_\tau$ such that $M \twoheadrightarrow_\beta M'$. We wish to show that for each reducible N of type σ, that $M'\,N$ is reducible.

Let $N \in RED_\sigma$. We know that:

1. $M\,N \in RED_\rho$, since $M \in RED_{\sigma\to\rho}$ and $N \in RED_\sigma$.
2. $M\,N \twoheadrightarrow_\beta M'\,N$, since $M \twoheadrightarrow_\beta M'$ (the same sequence of β-reductions within M works)

Since $deg(\rho) < deg(\sigma \to \rho)$, it follows from the induction hypothesis that $M'\,N \in RED_\rho$. Since this holds for any $N \in RED_\sigma$, we conclude that $M' \in RED_\tau$.

Lemma 3: Let M be a non-abstraction of type τ. Suppose that M satisfies:

$$M \to_\beta M' \qquad \Rightarrow \qquad M' \in RED_\tau \qquad (*)$$

Then $M \in RED_\tau$.

Proof: Suppose M of type τ is not an abstraction and satisfies $(*)$. We will do induction on the degree of type.

Suppose τ atomic. Then by $(*)$, contracting any redex in M results in a strongly normalizable term. Hence M itself must be strongly normalizable, hence $M \in RED_\tau$.

Now, suppose instead that M has type $\tau = \sigma \to \rho$, and assume that for each non-abstraction λ-term M_0 of degree less than $deg(\tau)$, we have

$$M_0 \text{ satisfies } (*) \qquad \Rightarrow \qquad M \in RED_\tau$$

Let $N \in RED_\sigma$; we wish to show $M\,N \in RED_\rho$. By the induction hypothesis, it suffices to show that $M\,N$ satisfies $(*)$.

From Lemma 1, we know that N is strongly normalizable. We will reason by a secondary induction on $\nu(N)$. Specifically, our induction hypothesis will be

$$P_M(n) = \text{“}\forall N_0 \in RED_\sigma \text{ such that } \nu(N_0) = n, \text{ we have } M\,N_0 \in RED_\rho\text{”}$$

Base case $P_M(0)$: Let $N_0 \in RED_\sigma$ satisfy $\nu(N_0) = 0$. We wish to show $M\,N_0$ satisfies $(*)$. Suppose that $M\,N_0 \to_\beta P$. Then P must be of the form $M'\,N_0$ with $M \to_\beta M'$ since N_0 contains no β-redexes and $M\,N_0$ is not itself a β-redex (as M was not an abstraction). Since $M \to_\beta M'$, we know from $(*)$ that $M' \in RED_\tau$, hence by definition of RED_τ, we know that $P = M'\,N_0 \in RED_\rho$. It follows that $M\,N_0$ satisfies $(*)$, and hence by the first induction hypothesis $M\,N_0$ is reducible.

Inductive step: Let $N_0 \in RED_\sigma$ have $\nu(N_0) = n$, and suppose that $P_M(m)$ holds for any $m < n$. Suppose $M\,N_0 \to_\beta P$. Then either $P = M'\,N_0$ with $M \to_\beta M'$, or $P = M\,N_0'$ with $N_0 \to_\beta N_0'$. In the former case, the proof is the same as in the base case. In the latter, since $N_0 \in RED_\sigma$ and $N_0 \to_\beta N_0'$, we know by Lemma 2 that $N_0' \in RED_\sigma$. Furthermore, $\nu(N_0') < \nu(N) = n$. Hence by our secondary inductive hypothesis, $P = M\,N_0'$ is reducible. Hence $M\,N_0$ satisfies $(*)$, and again by the first induction hypothesis, $M\,N_0$ is reducible.

It follows that $P_M(n)$ holds for all $n \in \mathbb{N}$; hence $M\,N$ is reducible. Since $N \in RED_\sigma$ was arbitrary, we conclude $M \in RED_\sigma$.

Lemma 4: If M is a λ-term such that for each $N \in RED_\sigma$ we have $M[x := N] \in RED_\rho$, then $\lambda x.M \in RED_{\sigma\to\rho}$.

Proof: Let $N \in RED_\sigma$; we wish to show that $(\lambda x.M)N \in RED_\rho$. Since $(\lambda x.M)N$ is not an abstraction, by Lemma 3 it suffices to show that $(\lambda x.M)N$ satisfies $(*)$.

Note that M and N are both reducible (M being reducible follows from the fact that $M[x := x]$ is reducible by hypothesis). Hence they are both strongly normalizable by Lemma 1. We will reason by induction on $\nu(M) + \nu(N)$. Our induction hypothesis is:

$$P(n) = \text{“if } M_0 \in RED_\rho, N_0 \in RED_\sigma \text{ are such that (i) } \nu(M_0) + \nu(N_0) = n, \text{ and (ii)}$$
$$M_0[x := N_0] \in RED_\rho, \text{ then } (\lambda x.M_0)N_0 \in RED_\rho\text{.”}$$

Base case: Suppose that $M_0 \in RED_\rho, N_0 \in RED_\sigma$ satisfy $\nu(M_0) + \nu(N_0) = 0$ and $M_0[x := N_0] \in RED_\rho$. We will prove that $(\lambda x.M_0)N_0$ satisfies $(*)$.

Suppose that $(\lambda x.M_0)N_0 \to_\beta P$. Then since M_0, N_0 are in normal form, we must have $P = M_0[x := N_0]$, which is reducible by hypothesis. Hence $(\lambda x.M_0)N_0$ satisfies $(*)$, so by Lemma 3 $(\lambda x.M_0)N_0$ is reducible.

Inductive step: Suppose that $M_0 \in RED_\rho, N_0 \in RED_\sigma$ satisfy $\nu(M_0) + \nu(N_0) = n$ and $M_0[x := N_0] \in RED_\rho$, and suppose that $P(m)$ holds for any $m < n$. Suppose $(\lambda x.M_0)N_0 \to_\beta P$. Then either:

(a) $P = M_0[x := N_0]$ which is reducible by hypothesis.

(b) $P = (\lambda x.M_0')N_0$ with $M_0 \to_\beta M_0'$. Note then that $\nu(M_0') < \nu(M_0)$ and $M_0'[x := N_0] \in RED_\rho$ (by Lemma 2). Hence we can apply the inductive hypothesis to conclude that P satisfies $(*)$, and is therefore reducible by Lemma 3.

(c) $P = (\lambda x.M_0)N_0$ with $N_0 \to_\beta N_0'$; similarly to (b), since $\nu(N_0') < \nu(N_0)$ and $M_0[x := N_0'] \in RED_\rho$, can apply the inductive hypothesis to conclude P satisfies $(*)$, and therefore is reducible.

In any case, P is reducible. So $(\lambda x.M_0)N_0$ satisfies $(*)$, and is therefore reducible.

We conclude that $P(n)$ holds for all $n \in \mathbb{N}$. In other words, we have shown:

$$\text{if } M_0 \in RED_\rho, N_0 \in RED_\sigma \text{ satisfy } M_0[x := N_0] \in RED_\rho, \text{ then } (\lambda x.M_0)N_0 \in RED_\rho.$$

Since $(\lambda x.M_0)N_0$ is reducible for each N_0 reducible, we conclude that $\lambda x.M_0$ is reducible.

Lemma 5: Let M be a well-typed λ-term (not necessarily reducible). Suppose that the free variables of M are $\vec{x} = (x_1, ..., x_n)$ of types $\tau_1, ..., \tau_n$. Then if $\vec{N} = (N_1, ..., N_n)$ are reducible terms of types $\tau_1, ..., \tau_n$, then $M[\vec{x} := \vec{N}]$ is reducible.

Proof: Induction on the structure of M.

If M is a variable, trivial.

If $M = P\,Q$, then by induction hypothesis $P[\vec{x} := \vec{N}]$ and $Q[\vec{x} := \vec{N}]$ are reducible. Then by definition $P[\vec{x} := \vec{N}]Q[\vec{x} := \vec{N}]$ is reducible. This is exactly $(P\,Q)[\vec{x} := \vec{N}]$.

Lastly, suppose $M = (\lambda y.P)$ is of type $\sigma \to \rho$. By induction hypothesis $P[\vec{x} := \vec{N}][y := N']$ is reducible for all reducible N' of type σ. Hence by Lemma 4, $(\lambda y.P[\vec{x} := \vec{N}])$ is reducible.

From Lemma 5, by setting $N_i = x_i$, we immediately obtain that all well-typed λ-terms are reducible. Hence by applying Lemma 1:

Theorem 6: All well-typed λ terms are strongly normalizable.

4. References

1. Sørensen, Urzycz; *Lectures on the Curry-Howard Isomorphism*, Ch.3
2. Girard; *Proofs and Types*, Ch.4,6
3. Gunther; *Strong Normalization of Propositional Types*, retrieved from http://www.math.cmu.edu/~wgunther/SN.pdf