THE NORMALIZATION THEOREM

DAVID WHITE

1. Preliminaries

Recall the lambda calculus $L ::= x|LL|\lambda x.L$. Recall how redexes $(\lambda x.M)N$ get normalized to $M[N/x]$. A normal form is one without a redex subterm, i.e. one which cannot be beta reduced. The following examples are for the untyped lambda calculus.

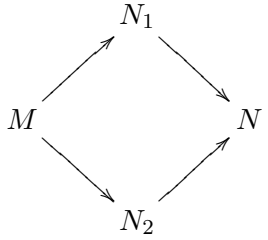
An example of a non-normalizing term is $\Omega = (\lambda x.xx)(\lambda x.xx)$. It is not in normal form but beta-reduces to itself.

A strongly normalizing term is one such that any reduction sequence terminates (in a normal form). An example is a term which is already in normal form, e.g. $\lambda x.x$. Another example is a simple redex $(\lambda x.xy)z$.

A weakly normalizing term is one such that there is a reduction sequence which terminates in a normal form. An example (from the slides) which is weakly but not strongly normalizing is $(\lambda xy.y)\Omega$. This is because the lazy reduction realizes there is no x into which I should substitute the Ω in y so this reduces to $\lambda y.y$. But trying to evaluate the Ω will never yield a result. The lazy way to beta-reduce is to reduce the left-most, outermost redex. This always works if the term is weakly normalizing. The eager way is to do the outermost redex only when the right hand side is a value (call-by-value). This does not always find the reduction of a weakly normalizing term.

Rewriting can be non-deterministic since you can choose different ways to reduce. Here is an example. Consider $(\lambda xyz.yz)a\Omega$. As in the above example, lazy evaluation realizes the Ω doesn't matter and returns $(\lambda yz.yz)a$, but eager evaluation gives $(\lambda xyz.yz)a\Omega$ because it tries to reduce Ω and gets Ω again. This is because it looks at the outermost redex (so can't touch a) and can only reduce when the term to which an abstraction is being applied (in this case Ω) is a VALUE, which Ω is not. So it tries to reduce Ω. Church-Rosser says there is a term N which both of these terms beta-reduce to. Indeed, $(\lambda yz.yz)a \to_1^\beta \lambda z.az$ and we get a normal form. On the other hand, $(\lambda xyz.yz)a\Omega \to_*^\beta \lambda z.az$ using the two lazy steps as above. So both paths of reduction result in the same normal term but the one which started with eager evaluation took longer.

Here is the picture for Church-Rosser, which says given M, N_1, and N_2 then N exists:



Date: April 27, 2010.

2. Typed Lambda Calculus

To get rid of the self-reference we introduced the typed lambda calculus. This does not allow self-reference. To have tu you must know that if u has type α (denoted u_α) then t has type $\alpha \to \beta$.

types::= $c|\alpha \to \alpha|\alpha * \alpha$

$v ::= x_\alpha|t_{\alpha\to\beta}u_\alpha|\lambda x_\alpha.t_\beta|(t_\alpha, t_\beta)$

The types above are α, β, $\alpha \to \beta$, and $\alpha * \beta$.

We have rules to deduce the type of a term based on the types of its subterms:

$$\begin{array}{c}[x:\alpha]\\ \vdots\\ t{:}\beta\\ \hline (\lambda x.t){:}\alpha\to\beta\end{array} \qquad \frac{f{:}\alpha\to\beta \quad u{:}\alpha}{fu{:}\beta} \qquad \frac{a{:}\alpha \quad b{:}\beta}{(a,b){:}\alpha*\beta} \qquad \frac{u{:}\alpha*\beta}{\pi_1 u{:}\alpha} \qquad \frac{u{:}\alpha*\beta}{\pi_2 u{:}\beta}$$

You can also use the Curry Howard Isomorphism to take propositions to terms and proofs to types. Here $a \wedge b$ goes to $a * b$, $a \to b$ goes to $a \to b$, and a goes to a.

$\mathrm{CH(A)} = a : A$

$\mathrm{CH}(\frac{B}{A\to B}) = \lambda x : A.\mathrm{CH}(B)$

$\mathrm{CH}(\frac{A\to B \quad A}{B}) = \mathrm{CH}(A \to B)\mathrm{CH(A)}$

$\mathrm{CH}(\frac{A \quad B}{A\wedge B}) = (\mathrm{CH(A)},\mathrm{CH(B)})$

$\mathrm{CH}(\frac{A\wedge B}{A}) = \pi_1\mathrm{CH}(A \wedge B)$

$\mathrm{CH}(\frac{A\wedge B}{B}) = \pi_2\mathrm{CH}(A \wedge B)$

3. Weak Normalization for Simply Typed Lambda Calc

Note: if every term has a reduction to a normal form then we have proven that the question of whether two terms are equal is decidable.

The naive proof idea is to induct on terms, but you can have simple terms with arbitrarily complicated type. The idea to induct on types similarly fails. So we need the notion of degree:

For a type, $\partial(T) = 1$ if T is atomic, $\partial(U * V) = \partial(U \to V) = \max(\partial(U), \partial(V)) + 1$

For a redex, $\partial(\pi_1(u, v)) = \partial(\pi_2(u, v)) = \partial(U * V)$ while $\partial((\lambda x.v)u) = \partial(U \to V)$

For a term t we say the degree is $d(t) = \sup\{\partial(r) \mid t \text{ contains redex } r\}$.

Easy facts:

(1) $\partial(r) \leq d(r)$ because r contains itself

(2) A redex of type T has $\partial(r) > \partial(T)$ because $\partial(U * V) > \partial(U), \partial(V)$ and $\partial(U \to V) > \partial(U), \partial(V)$

(3) $d(t[u/x]) \leq \max(d(t), d(u), \partial(X))$ because $t[u/x]$ contains redexes of t and u (wherever x occurred) as well as possibly new redexes (e.g. $\pi_1(x) \to \pi_1(u) = \pi_1(u', u'') = u'$ or

$xv \to uv = (\lambda y.u')v)$ which are of the form above for redexes and hence are bounded by $\partial(X)$.

(4) $t \to u$ by replacing r by c implies $d(u) \leq d(t)$ because non-r redexes survive, redexes of c are simpler than those of r by the above ($d(c) \leq \partial(X) < d(r)$), and new redexes with same inequality.

Lemma 1. *Suppose $\partial(r) = n$ is maximal in t. Then converting r to c gives strictly fewer redexes of degree n since the only redexes in r which are duplicated have degree $< n$ and since r is replaced by smaller degree redexes.*

Theorem 1. *All terms are weakly normalzing (WN)*

Proof. Let $\mu(t) = (n, m)$ where $n = d(t)$ and m is the number of redexes of degree m. We proceed by a double induction (on n and m). In the base case, $n = 0$ and the term is normal. We now begin the second induction. Assuming $P(k)$ holds for all $k < n$ we must show $P(n)$, i.e. for all m, $\mu(t) = (n, m) \Rightarrow t$ is weakly normalizing (WN). $P(n)$ is equivalent to $\forall m\ Q(n, m)$ where $Q(n, m)$ is that $\mu(t) = (n, m) \Rightarrow t$ is WN. We proceed by induction on m. If $m = 0$ then there are no redexes of degree n, so $d(t) < n$. Thus, $Q(n, 0)$ is implied by $\forall m\ Q(d(t), m)$ which is implied by $P(d(t))$ which we assumed.

Now let $\mu(t) = (n, m)$ and assume $Q(k, \ell)$ holds for all $k < n$ and for all ℓ or for $k = n$ and all $\ell < m$. So all terms with $\mu(t') = (k, \ell)$ for such k, ℓ are WN. We must show t is WN. Choose a redex of t of maximal degree and apply the reduction ($t \to u$) from the lemma above. The lemma tells us that we now have strictly fewer redexes of degree n (i.e. we have $\ell < m$). Thus, we've reduced $Q(n, m)$ for t to $Q(n, \ell)$ for u, which we assumed. So the one-step reduction $t \to u$ composed with this chain of reductions shows t is WN.

□

4. Strong Normalization for Simply Typed Lambda Calc

Lemma 2. *t is strongly normalizing (SN) iff there is a number $\nu(t)$ bounding the length of every normalization sequence beginning with t*

Proof. If $\nu(t)$ exists then clearly all chains halt so t is SN. Conversely, consider the collection of all chains starting at t. These can be arranged into a tree which is finitely branching because each term has only finitely many subterms. Since t is SN each branch terminates, so Konig's Lemma tells us the tree is finite. Thus, letting $\nu(t)$ be the number of vertices of the tree plus 1 completes the proof. □

Theorem 2. *All terms are strongly normalizing (SN)*

Note that this theorem and Church-Rosser imply the simply typed lambda calculus (STLC) is consistent, i.e. you can't prove $x = y$ if x and y are distinct. This is because if $x \to_* y$ then $x = y$ can be derived, so $x \equiv_\beta y$ implies $x = y$ =common ancestor. Conversely, if $x = y$ then reducing both to normal form and looking at Church-Rosser (normal forms are unique) proves $x \equiv_\beta y$.

Our proof will generalize to the case when the system is F (chapter 14) rather than the simply typed lambda calculus. It also holds for Godel's system T which includes the integers. Thus, this theorem implies the axioms of $\mathbb{Z}$ are consistent, violating Godel's Second Incompleteness Theorem unless it's proven using something not included in Peano's Axioms. For this reason we introduce a stronger form of induction based on the property of reduciblity (R).

(*) If t is atomic then $t \in R$ if t is SN. If $t : U * V$ then $t \in R$ if $\pi_1(t), \pi_2(t) \in R$. If $t : U \to V$ then $t \in R$ if for all $u \in R$ of type U, $tu \in R$ of type V.

Note that this last case contains information not obtainable in Peano Arithmetic because it has a universal quantifier which can't be done by induction (so it's not an arithmetic formula in t and $U \to V$).

Define a neutral term to be one which is not of the form (a, b) or $\lambda x.v$. We will prove three conditions hold on all terms by induction on the type:

(1) $t \in R$ implies t is SN

(2) $t \in R$ and $t \to_* t'$ implies $t' \in R$

(3) If t neutral and every time we convert a redex we get $t' \in R$ then $t \in R$.

For atomic t, (1) is trivial. (2) holds because t is SN so all t' are SN. For (3) we must show t is SN, so take any reduction step. You end up at an SN t' so set $\nu(t) = \nu(t') + 1$ and you've got t SN by the lemma.

For $t : U * V$, (1) holds because $\pi_1(t), \pi_2(t) \in R$ and so by induction (U and V are subtypes of $U * V$), both are SN and clearly $\nu(t) \leq \nu(\pi_i(t))$ because we can apply π_i to a reduction chain for t. So $\nu(t)$ is finite and t is SN. (2) holds because $t \in R$ implies $\pi_i(t) \in R$. So $t \to_* t'$ implies $\pi_i(t) \to_* \pi_i(t')$ and by induction $\pi_i(t') \in R$. So $t' \in R$. For (3) note that it IS possible to have $t : U * V$ and also have t neutral because t need not be reduced (so it could be ab where $b : B$ and $a : B \to U * V$). Take any step $\pi_i(t) \to \pi_i(t')$ (all steps from $\pi_i(t)$ are of this form because $\pi_i(t)$ is not a redex because t is not a pair). Then there is a $t \to t'$ and so by hypothesis t' is reducible. Thus, $\pi_i(t')$ is reducible and $\pi_i(t)$ is neutral. By induction, $\pi_i(t)$ is reducible so by (*) t is reducible.

For $t : U \to V$, (1) holds because if $x : U$ is a variable then x is neutral so induction and (3) tells us x is reducible. Thus, tx is reducible and $\nu(t) \leq \nu(tx)$ because we could apply a normalization chain for t to x and get one for tx. So $\nu(t)$ is finite and t is SN. (2) holds because if $t \to_* t'$, $t \in R$, and $u : U \in R$ then $tu \in R$ and $tu \to_* t'u$. So by induction (on type V) $t'u \in R$ for all $u : U \in R$. So by (*), $t' \in R$. For (3) let $u : U \in R$ and show $tu \in R$ so that by (*) we're done. There are two cases because tu cannot be a redex (t is not $\lambda x.v$):

$tu \to t'u$ via $t \to t'$ implies $t' \in R$ so $t'u \in R$.

$tu \to tu'$ via $u \to u'$ implies $u' \in R$ by the induction hypothesis (2) on U so $\nu(u') < \nu(u)$. Thus, the induction hypothesis for u' tells us $tu' \in R$.

Either way tu converts into reducible terms only, so the induction hypothesis (3) tells us $tu \in R$ for all u.

Lemma 3. $u, v \in R \Rightarrow (u, v) \in R$

Proof. We'll induct on $\nu(u) + \nu(v)$ and show $\pi_1(u, v) \in R$. It could reduce to $u \in R$. Or it could reduce to $\pi_1(u', v)$ with $u \to u'$. Then (2) tells us $u' \in R$ so $\nu(u') < \nu(u)$ and hence $\nu(\pi_1(u', v)) < \nu(u) + \nu(v)$ is in R by induction. Finally, it could reduce to $\pi_1(u, v')$ but this is similar. So $\pi_1(u, v)$ converts to reducible terms only and by (3) must be reducible. Similarly, $\pi_2(u, v) \in R$. So $(u, v) \in R$. □

Lemma 4. *If $u : U \in R$ and $v[u/x] \in R$ then $\lambda x.v \in R$*

Proof. We'll induct on $\nu(u) + \nu(v)$ and show $(\lambda x.v)u \in R$. It could reduce to $v[u/x] \in R$. Or it could reduce to $(\lambda x.v')u$ with $v \to v'$. Then (2) tells us $v' \in R$ so $\nu(v') < \nu(v)$ and hence

$\nu((\lambda x.v')u) < \nu(u) + \nu(v)$ is in R by induction. Finally, it could reduce to $(\lambda x.v)u'$ but this is similar. So $(\lambda x.v)u$ converts to reducible terms only and by (3) must be reducible for all u. So $\lambda x.v \in R$. □

Proposition 1. *For a term t with free variables contained in $x_1 : U_1, \ldots, x_n : U_n$, if $u_1 : U_1, \ldots, u_n : U_n$ are reducible then $t[u_1/x_1, \ldots, u_n/x_n] \in R$.*

Proof. Induct on t. If $t = x_i$ we have a tautology. If $t = \pi_i(w)$ then by induction $w[\underline{u}/\underline{x}] \in R$. So $\pi_i(w[\underline{u}/\underline{x}]) = t[\underline{u}/\underline{x}] \in R$. If $t = (v, w)$ then $v[\underline{u}/\underline{x}], w[\underline{u}/\underline{x}] \in R$. The first lemma above says $t[\underline{u}/\underline{x}] = (v[\underline{u}/\underline{x}], w[\underline{u}/\underline{x}]) \in R$.

If $t = wv$ then $w[\underline{u}/\underline{x}], v[\underline{u}/\underline{x}] \in R$ so $t[\underline{u}/\underline{x}] = w[\underline{u}/\underline{x}](v[\underline{u}/\underline{x}]) \in R$. If $t = \lambda y.w$ has type $V \to W$ then $w[\underline{u}/\underline{x}, \underline{v}/\underline{y}] \in R$ for all $v : V$. The second lemma says $t[\underline{u}/\underline{x}] = \lambda y.(w[\underline{u}/\underline{x}]) \in R$. □

To prove the theorem, let $u_i = x_i$. Then for any term t, $t = t[\underline{x}/\underline{x}] \in R$.

5. Proof of Konig's Lemma

Given a finitely branching tree T with no infinite branch, we prove T is finite. Suppose T is infinite and let v be a vertex. Then v has paths to all vertices in T, but v has only finitely many neighbors. By the pigeonhole principle, one of the neighbors of v (call it v_2) must connect to infinitely many vertices without going through v. The same logic holds for v_2 and gives a vertex v_3 which connects to infinitely many others without going through v_1 or v_2. Continue in this way. We now have an infinite branch $v, v_2, v_3, \ldots$ and it has no repeated vertex because of how it was constructed. So this path contradicts the hypotheses on T.

6. Some Logic

Various systems of typed lambda calculus including the simply typed lambda calculus, Jean-Yves Girard's System F, and Thierry Coquand's calculus of constructions are strongly normalizing.

A lambda calculus system with the normalization property can be viewed as a programming language with the property that every program terminates. Although this is a very useful property, it has a drawback: a programming language with the normalization property cannot be Turing complete. That means that there are computable functions that cannot be defined in the simply typed lambda calculus (and similarly there are computable functions that cannot be computed in the calculus of constructions or system F). As an example, it is impossible to define the normalization algorithms of any of the calculi cited above within the same calculus.

A system Turing Complete if and only if such system can simulate a single taped Turing Machine. So it needs recursion, which we've thrown out of the simply typed lambda calculus.

Various typed lambda calculi have been studied: The types of the simply typed lambda calculus are only base types (or type variables) and function types $\sigma \to \tau$. System T extends the simply typed lambda calculus with a type of natural numbers and higher order primitive recursion; in this system all functions provably recursive in Peano arithmetic are definable. System F allows polymorphism by using universal quantification over all types; from a logical perspective it can describe all functions which are provably total in second-order logic. Lambda calculi with dependent types are the base of intuitionistic type theory

Theorem 3 (Godel's First Incompleteness Theorem)**.** *Any effectively generated theory capable of expressing elementary arithmetic cannot be both consistent and complete. In particular, for any consistent, effectively generated formal theory that proves certain basic arithmetic truths, there is an arithmetical statement that is true, but not provable in the theory*

Theorem 4 (Godel's Second Incompleteness Theorem)**.** *For any formal effectively generated theory T including basic arithmetical truths and also certain truths about formal provability, T includes a statement of its own consistency if and only if T is inconsistent.*

Peano's Axioms:

The first four axioms describe the equality relation. The next four deal with 0 and the successor function S. The last is induction.

(1) For every natural number x, x = x. That is, equality is reflexive.

(2) For all natural numbers x and y, if x = y, then y = x. That is, equality is symmetric.

(3) For all natural numbers x, y and z, if x = y and y = z, then x = z. That is, equality is transitive.

(4) For all a and b, if a is a natural number and a = b, then b is also a natural number. That is, the natural numbers are closed under equality.

(5) 0 is a natural number.

(6) For every natural number n, S(n) is a natural number.

(7) For every natural number n, S(n) = 0 is False. That is, there is no natural number whose successor is 0.

(8) For all natural numbers m and n, if S(m) = S(n), then m = n. That is, S is an injection.

(9) If K is a set such that 0 is in K and for every natural number n, if n is in K, then S(n) is in K, then K contains every natural number.

7. Generalizing the theory

All of this theory generalizes to the case of rewrite systems, which consist of a set of objects and a set of relations on how to transform those objects. These systems do not provide an algorithm for changing one term to another; they give a set of possible rule applications. You can also study abstract rewriting systems (ARS) which generalize the β-reduction we studied earlier.

Logic is an example of such a system with not-not-elim, DeMorgan ($\neg(A \vee B) \to \neg A \wedge \neg B$), and distributivity ($(A \wedge B) \vee C \to (A \vee C) \wedge (B \vee C)$.

An object $x \in A$ is called reducible if there exist some other $y \in A$ and $x \to y$; otherwise it is called irreducible or a normal form. An object y is called a normal form of x if $x \xrightarrow{*} y$, and y is irreducible. If x has a unique normal form, then this is usually denoted with $x\downarrow$.

An ARS has the Church-Rosser Property if and only if $x \overset{*}{\leftrightarrow} y$ implies $x \downarrow y$. In words, the Church-Rosser propery means that the reflexive transitive symmetric closure is contained in the joinability relation.

An abstract rewriting system is said to be terminating or noetherian if there is no infinite chain $x_0 \to x_1 \to x_2 \to \cdots$