How to prove normalisation of STLC

Jules Jacobs

March 7, 2021

Abstract

This proof is a simplification of the one in a note by Harper.[1] The proof given here does not need a Kripke logical relation, yet works for reduction under lambda. The trick is to not insist that terms are typed, which removes the need for keeping track of the types of free variables. We also no longer rely on type preservation as a lemma.

We prove weak normalisation of simply typed lambda calculus with $\mathbb{N}$ as the base type:

$$e ::= x \mid n \mid \lambda x.e \mid \mathsf{ap}(e,e) \qquad\qquad T ::= \mathbb{N} \mid T \to T$$

The typing rules are standard:

$$\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \frac{n \in \mathbb{N}}{\Gamma \vdash n : \mathbb{N}} \qquad \frac{\Gamma, x : A \vdash e : B \qquad x \notin \Gamma}{\Gamma \vdash \lambda x.e : A \to B} \qquad \frac{\Gamma \vdash e_1 : A \to B \qquad \Gamma \vdash e_2 : A}{\Gamma \vdash \mathsf{ap}(e_1, e_2) : B}$$

The operational semantics allows reduction under lambda:

$$\frac{.}{\mathsf{ap}(\lambda x.e_1, e_2) \to e_1[x \mapsto e_2]} \qquad \frac{e_1 \to e_1'}{\mathsf{ap}(e_1,e_2) \to \mathsf{ap}(e_1', e_2)} \qquad \frac{e_2 \to e_2'}{\mathsf{ap}(e_1,e_2) \to \mathsf{ap}(e_1, e_2')} \qquad \frac{e \to e'}{\lambda x.e \to \lambda x.e'}$$

Definition 1. Let the set W of weakly normalising terms be such that $e \in W$ iff there exists a reduction sequence $e \to^* e'$ such that e' is irreducible, i.e. e' cannot take another step.

Our goal is to prove:

Theorem 1. $\Gamma \vdash e : T \implies e \in W$

Doing this by induction on typing doesn't work: the induction hypothesis $e_1, e_2 \in W$ is not strong enough to show that $\mathsf{ap}(e_1, e_2) \in W$. The solution is to strengthen the theorem to $e \in W(T)$ where $W(T) \subseteq W$ is defined by recursion on T to make the induction go through:

$$W(\mathbb{N}) = W \qquad\qquad W(A \to B) = \{e \mid \mathsf{ap}(e, e') \in W(B) \text{ for all } e' \in W(A)\}$$

It is not trivial that $W(T) \subseteq W$; induction on T fails to go through because we need to know that variables $x \in W(T)$, which depends on $\mathsf{ap}(x, e') \in W(T)$, which depends on $\mathsf{ap}(\mathsf{ap}(x,e'), e'') \in W(T)$, and so on. In order to strengthen the induction hypothesis we need the set of neutral terms N, defined inductively by:

$$x \in N \qquad\qquad \mathsf{ap}(e_1, e_2) \in N \text{ if } e_1 \in N \text{ and } e_2 \in W$$

With a strengthened induction hypothesis, the proof goes through.

[1]Kripke-Style Logical Relations for Normalization, Harper 2021
http://www.cs.cmu.edu/~rwh/courses/chtt/pdfs/kripke.pdf

Lemma 2. (Pas de deux). $N \subseteq W(T) \subseteq W$

Proof. By induction on T.

1. $T = \mathbb{N}$:
   (a) $N \subseteq W(\mathbb{N})$: strengthen the statement $N \subseteq W$ to $N \subseteq W'$, where $W' \subseteq W$ are terms that normalise to a neutral term, then perform induction on N.
   (b) $W(\mathbb{N}) \subseteq W$: true by definition.
2. $T = A \to B$:
   (a) $N \subseteq W(A \to B)$: Let $e \in N$. If $v \in W(A) \subseteq W$, then $\mathsf{ap}(e, v) \in N \subseteq W(B)$, so $e \in W(A \to B)$.
   (b) $W(A \to B) \subseteq W$: Let $e \in W(A \to B)$. A variable $x \in N \subseteq W(A)$, so $\mathsf{ap}(e, x) \in W(B) \subseteq W$. A careful analysis of the reductions shows that $e \in W$.

□

From this point on, we may forget about neutral terms, and only remember $W(T) \subseteq W$ and $x \in W(T)$.

Another esential property that $W(T)$ shares with W is that of head expansion:

Lemma 3. (Head expansion). If $e \to e'$ then $e' \in W(T) \implies e \in W(T)$.

Proof. By induction on T. For $T = \mathbb{N}$, it follows by definition of W. For $T = A \to B$ it follows from the induction hypothesis for B and the rule for stepping in the function position of ap. □

With $W(T)$ instead of W in the theorem statement, the difficulty has been moved to the lambda case, for which we need a further strengthening of the induction hypothesis to obtain information about variables.

A substitution γ is a map from variables to terms. We extend $\gamma(e)$ to all terms e to perform the substitution. By abuse of notation, we say that $\gamma \in W(\Gamma)$ if $\gamma(x) \in W(\Gamma(x))$ for all x.

We are ready to state the final theorem:

Theorem 4. If $\Gamma \vdash e : T$ then $\gamma(e) \in W(T)$ for all $\gamma \in W(\Gamma)$.

Proof. By induction on typing.

VAR We have $\Gamma(x) = T$ and $e = x$. The result $\gamma(x) \in W(T)$ follows by assumption $\gamma \in W(\Gamma)$.

APP We need to show that $\gamma(\mathsf{ap}(e_1, e_2)) \in W(B)$ when $\gamma(e_1) \in W(A \to B)$ and $\gamma(e_2) \in W(A)$. This follows from $\gamma(\mathsf{ap}(e_1, e_2)) = \mathsf{ap}(\gamma(e_1), \gamma(e_2))$ and the definition of $W(A \to B)$.

LAM We have assumptions $\gamma \in W(\Gamma)$, $x \notin \Gamma$, and for all $\gamma' \in W(\Gamma; x : A)$ we have $\gamma'(e) \in W(B)$.

We need to show that $\gamma(\lambda x.e) \in W(A \to B)$, i.e. $\mathsf{ap}(\gamma(\lambda x.e), e') \in W(B)$ for all $e' \in W(A)$.

Let $e' \in W(A)$. We have:

$$\mathsf{ap}(\gamma(\lambda x.e), e') = \mathsf{ap}((\lambda x.\gamma(e)), e') \to_\beta \gamma(e)[x \mapsto e'] = ([x \mapsto e'] \cup \gamma)(e)$$

Using $\gamma' = [x \mapsto e'] \cup \gamma$ in our assumption, the right hand side $\gamma'(e) \in W(B)$. Then, by head expansion, also the left hand side $\mathsf{ap}(\gamma(\lambda x.e), e') \in W(B)$.

□

Corollary 1. If $\Gamma \vdash e : T$ then $e \in W$.

Proof. Take γ to be the identity substitution. We have $\gamma \in W(\Gamma)$ because variables $x \in W(T')$ for all T', by the corollary of the pas de deux. The theorem gives us that $e \in W(T)$. By the other corollary of the pas de deux, $e \in W$. □

On the representation of lambda terms

One might worry about variable names. If we represent lambda terms as abstract syntax trees where the lambda nodes and variable nodes carry natural numbers as identifiers (or strings of text), then we have issues. The definition of the substitution function becomes rather complicated, particularly when the terms we substitute are open. The step $\gamma(\lambda x.e) = \lambda x.\gamma(e)$ in the proof doesn't hold, strictly speaking: the substitution may have to rename the variable x to x' in order to avoid capture if some term in γ contains x as a free variable.

Another problem is that the substitution may have to rename *other* variables inside e. If you attempt to make everything fully formal with capture avoiding substitution in, say, Coq, you may find that it's like playing whack-a-mole: fixing one issue will cause another step of the proof to fail.

Note that this issue isn't solved by Kripke.

The solution is to consider terms up to α-equivalence of bound variables. De Bruijn terms are an explicit construction of this quotient. If Λ is the set of lambda terms and Λ_{DB} is the set of De Bruijn terms where bound variables are represented using De Bruijn indices, then we have a conversion $\pi : \Lambda \to \Lambda_{DB}$ and for any map $f : \Lambda \to A$ that respects α-equivalence, we have a unique map $\hat{f} : \Lambda_{DB} \to A$ such that $f = \hat{f} \circ \pi$, which is what it means for Λ_{DB} to be the quotient.

The problematic step of the proof is:

$$\mathsf{ap}(\gamma(\lambda x.e), e') \to_\beta ([x \mapsto e'] \cup \gamma)(e)$$

In De Bruijn representation, this is written:

$$\mathsf{ap}(\gamma(\lambda.e), e') \to_\beta (e' \triangleright \gamma)(e)$$

Where $\gamma' = e' \triangleright \gamma$ is the substitution such that $\gamma'(0) = e'$ and $\gamma'(n+1) = \gamma(n)$. The condition $\gamma' \in W(\Gamma; x : A)$ is written $\gamma' \in W(A \triangleright \Gamma)$ in De Bruijn, and is satisfied because $\gamma'(0) = e' \in W(A)$ and $\gamma'(n+1) = \gamma(n) \in W(\Gamma(n)) = W((A \triangleright \Gamma)(n+1))$.

It is non-trivial to prove $\mathsf{ap}(\gamma(\lambda.e), e') \to_\beta (e' \triangleright \gamma)(e)$: the definition of substitution has to shift De Bruijn indices when traversing under a lambda, and beta reduction removes the outermost binder, so it must also shift indices in the inner term. As it turns out, all this index shifting cancels out in the end, to make the step work.

Although it is non-trivial to prove, one can see that *if* substitution and β-reduction have been defined correctly on De Bruijn terms, then the lemma $\mathsf{ap}(\gamma(\lambda.e), e') \to_\beta (e' \triangleright \gamma)(e)$ *must* hold, since (1) substitution on De Bruijn terms it must agree with capture avoiding substitution under π, and (2) the problematic step of the proof on ordinary lambda terms does hold, if the variable names are such that no capture avoiding renaming is necessary.