The untyped lambda calculus

CIS 6700, Spring 2023

August 15, 2023

1 Untyped lambda calculus

This document describes the untyped lambda-calculus, with the following grammar. References are to definitions in Barendregt, "The Lambda Calculus: Its Syntax and Semantics".

$tm,\ t,\ u,\ v,\ w$	$::=$		terms
	$\mid$	x	variables
	$\mid$	$\lambda x.t$	abstractions
	$\mid$	$t\ u$	function application, list indexing
	$\mid$	k	
	$\mid$	$\mathbf{add}$	
	$\mid$	$\mathbf{nil}$	
	$\mid$	t, u	

In β-reduction, the argument of an application is substituted for the bound variable in an abstraction. We use the $t[x \rightsquigarrow u]$ notation for substituting the term u for the variable x in the term t.

$\boxed{t\ \beta\text{-reduces to}\ u}$ *(Definition 3.1.3)*

$$\textsc{beta-reduct}\ \frac{\mathbf{value}\, v}{(\lambda x.t)\ v\ \beta\text{-reduces to}\ t[x \rightsquigarrow v]} \qquad \textsc{beta-app-0}\ \frac{\mathbf{value}\,(v, w)}{(v, w)\ 0\ \beta\text{-reduces to}\ t}$$

$$\textsc{beta-app-k}\ \frac{\mathbf{value}\,(v, w)}{(v, w)\ (1 + k)\ \beta\text{-reduces to}\ (w\ k)} \qquad \textsc{beta-plus}\ \frac{}{\mathbf{add}\ (j, (k, \mathbf{nil}))\ \beta\text{-reduces to}\ (j + k)}$$

We can define a deterministic, small-step evaluation relation by reducing in the heads of applications. This is a *call-by-name* semantics and performs head-reduction.

$\boxed{t \rightsquigarrow t'}$ *(Small-step evaluation)*

$$\frac{t \rightsquigarrow t'}{t\ u \rightsquigarrow t'\ t}\ \textsc{S-app1} \qquad \frac{}{(\lambda x.t)\ u \rightsquigarrow t[x \rightsquigarrow u]}\ \textsc{S-beta} \qquad \frac{t \rightsquigarrow t'}{t, u \rightsquigarrow t', t}\ \textsc{S-cons1} \qquad \frac{t \rightsquigarrow t' \qquad \textbf{value}\, v}{v, t \rightsquigarrow v, t'}\ \textsc{S-cons2}$$

$$\frac{\textbf{value}\,(v, w)}{(v, w)\ 0 \rightsquigarrow v}\ \textsc{S-prj-zero} \qquad \frac{\textbf{value}\,(v, w)}{(v, w)\ (1 + k) \rightsquigarrow (v, w)\ k}\ \textsc{S-prj-suc} \qquad \frac{}{\textbf{add}\ (k_1, k_2) \rightsquigarrow k_1 + k_2}\ \textsc{S-add}$$

We can also define a nondeterministic single-step full-reduction by performing β-reduction in any subterm. Iterating this reduction will convert a term into its β-normal form.

$\boxed{t \longrightarrow_\beta u}$ *(Full CBV β-reduction)*

$$\frac{\textbf{value}\, v}{(\lambda x.t)\ v \longrightarrow_\beta t[x \rightsquigarrow v]}\ \textsc{F-beta} \qquad \frac{t \longrightarrow_\beta t'}{\lambda x.t \longrightarrow_\beta \lambda x.t'}\ \textsc{F-abs} \qquad \frac{t \longrightarrow_\beta t'}{t\ u \longrightarrow_\beta t'\ u}\ \textsc{F-app1}$$

$$\frac{u \longrightarrow_\beta u'}{t\ u \longrightarrow_\beta t\ u'}\ \textsc{F-app2} \qquad \frac{t \longrightarrow_\beta t'}{t, u \longrightarrow_\beta t', u}\ \textsc{F-cons1} \qquad \frac{u \longrightarrow_\beta u'}{t, u \longrightarrow_\beta t, u'}\ \textsc{F-cons2} \qquad \frac{\textbf{value}\, v \quad \textbf{listvalue}\, w}{(v, w)\ 0 \longrightarrow_\beta v}\ \textsc{F-prj-zero}$$

$$\frac{\textbf{value}\, v \quad \textbf{listvalue}\, w}{(v, w)\ (1 + k) \longrightarrow_\beta w\ k}\ \textsc{F-prj-suc} \qquad \frac{}{\textbf{add}\ (k_1, (k_2, \textbf{nil})) \longrightarrow_\beta k_1 + k_2}\ \textsc{F-add}$$

We can define when two terms are equivalent up to β.

$\boxed{t \equiv_\beta u}$ *(β-conversion or β-equality (Definition 2.1.4))*

$$\frac{}{(\lambda x.t)\ u \equiv_\beta t[x \rightsquigarrow u]}\ \textsc{eq-beta} \qquad \frac{}{t \equiv_\beta t}\ \textsc{eq-refl} \qquad \frac{u \equiv_\beta t}{t \equiv_\beta u}\ \textsc{eq-sym} \qquad \frac{t_1 \equiv_\beta t_2 \qquad t_2 \equiv_\beta t_3}{t_1 \equiv_\beta t_3}\ \textsc{eq-trans}$$

$$\frac{t \equiv_\beta t'}{t\ u \equiv_\beta t'\ u}\ \textsc{eq-app1} \qquad \frac{u \equiv_\beta u'}{t\ u \equiv_\beta t\ u'}\ \textsc{eq-app2} \qquad \frac{t \equiv_\beta t'}{\lambda x.t \equiv_\beta \lambda x.t'}\ \textsc{eq-abs} \qquad \frac{t \equiv_\beta t'}{t, u \equiv_\beta t', u}\ \textsc{eq-cons1}$$

$$\frac{u \equiv_\beta u'}{t, u \equiv_\beta t, u'}\ \textsc{eq-cons2}$$

Finally, the Church-Rosser Theorem relies on the definition of parallel reduction. This is a version of reduction that is confluent.

$\boxed{t \Longrightarrow_\beta u}$ *(Parallel reduction (3.2.3))*

$$\textsc{p-beta}\ \frac{t \Longrightarrow_\beta \lambda x.t' \qquad u \Longrightarrow_\beta v \qquad \mathbf{value}\ v}{t\ u \Longrightarrow_\beta t'[x \rightsquigarrow v]}$$

$$\textsc{p-app-k}\ \frac{t \Longrightarrow_\beta v \qquad u \Longrightarrow_\beta k \qquad \mathbf{value}\ v \qquad \mathbf{nth}\ v\ k = w}{t\ u \Longrightarrow_\beta w}$$

$$\textsc{p-add-beta}\ \frac{t \Longrightarrow_\beta j, (k, \mathbf{nil})}{\mathbf{add}\ t \Longrightarrow_\beta (j + k)}$$

$$\textsc{p-var}\ \frac{}{x \Longrightarrow_\beta x} \qquad \textsc{p-abs}\ \frac{t \Longrightarrow_\beta t'}{\lambda x.t \Longrightarrow_\beta \lambda x.t'} \qquad \textsc{p-app}\ \frac{t \Longrightarrow_\beta t' \qquad u \Longrightarrow_\beta u'}{t\ u \Longrightarrow_\beta t'\ u'} \qquad \textsc{p-cons}\ \frac{t \Longrightarrow_\beta t' \qquad u \Longrightarrow_\beta u'}{t, u \Longrightarrow_\beta t', u'}$$

$$\textsc{p-nil}\ \frac{}{\mathbf{nil} \Longrightarrow_\beta \mathbf{nil}} \qquad \textsc{p-add}\ \frac{}{\mathbf{add} \Longrightarrow_\beta \mathbf{add}} \qquad \textsc{p-nat}\ \frac{}{k \Longrightarrow_\beta k}$$

2 Relation operations

Many of the operations above can be generated by applying the following closure operations to the β reduction relation. These operations are parameterized by an arbitrary relation R.

Note that $t \to_R u$ with R equal to β is the same relation as $t \longrightarrow_\beta u$. And, the compatible, reflexive, symmetic and transitive closure of β is the same relation as $t \equiv_\beta u$.

$\boxed{t \to_R u}$ *(Compatible closure of R (aka one-step reduction (3.1.5))*

$$\textsc{cc-rel}\ \frac{R\ t\ u}{t \to_R u} \qquad \textsc{cc-abs}\ \frac{t \to_R u}{\lambda x.t \to_R \lambda x.u} \qquad \textsc{cc-app1}\ \frac{t \to_R t'}{t\ u \to_R t'\ u} \qquad \textsc{cc-app2}\ \frac{u \to_R u'}{t\ u \to_R t\ u'}$$

$$\textsc{cc-cons1}\ \frac{t \to_R t'}{t, u \to_R t', u} \qquad \textsc{cc-cons2}\ \frac{u \to_R u'}{t, u \to_R t, u'}$$

$\boxed{t \to_R^= u}$ *(reflexive closure of R (3.1.4))*

$$\textsc{r-rel}\ \frac{R\ t\ u}{t \to_R^= u} \qquad \textsc{r-refl}\ \frac{}{t \to_R^= t}$$

$\boxed{t \to_R^+ u}$ *(transitive closure of R (3.1.4))*

$$\textsc{t-rel}\quad \frac{R\,t\,u}{t \to_R^+ u} \qquad \textsc{t-trans}\quad \frac{t_1 \to_R^+ t_2 \qquad t_2 \to_R^+ t_3}{t_1 \to_R^+ t_3}$$

$\boxed{t \to_R^* u}$ *(reflexive-transitive closure of R)*

$$\textsc{rt-rel}\quad \frac{R\,t\,u}{t \to_R^* u} \qquad \textsc{rt-refl}\quad \frac{}{t \to_R^* t} \qquad \textsc{rt-trans}\quad \frac{t_1 \to_R^* t_2 \qquad t_2 \to_R^* t_3}{t_1 \to_R^* t_3}$$

$\boxed{t \leftrightarrow_R u}$ *(symmetric-transitive closure of R)*

$$\textsc{st-rel}\quad \frac{R\,t\,u}{t \leftrightarrow_R u} \qquad \textsc{st-sym}\quad \frac{u \leftrightarrow_R t}{t \leftrightarrow_R u} \qquad \textsc{st-trans}\quad \frac{t_1 \leftrightarrow_R t_2 \qquad t_2 \leftrightarrow_R t_3}{t_1 \leftrightarrow_R t_3}$$

3 Eta-reduction

By changing the definition of the underlying primitive reduction, we can also reason about η-reduction and $\beta\eta$-equivalence. Note that the rule for η-reduction has been stated in a way that generates the approporiate output for the locally-nameless representation in Coq. In this output, the fact that $x \notin \mathsf{fv}t$ is implicit and does not need to be added as a precondition to the rule.

$\boxed{t\ \eta\text{-reduces to}\ u}$ *(Definition 3.1.1 (i))*

$$\textsc{eta-reduct}\quad \frac{t' = t\ x}{\lambda x.t'\ \eta\text{-reduces to}\ t}$$

$\boxed{t\ \beta\eta\text{-reduces to}\ u}$ *(Either β or η reduction (Definition 3.1.1 (ii)))*

$$\textsc{eta-beta}\quad \frac{t\ \beta\text{-reduces to}\ u}{t\ \beta\eta\text{-reduces to}\ u} \qquad \textsc{eta-eta}\quad \frac{t\ \eta\text{-reduces to}\ u}{t\ \beta\eta\text{-reduces to}\ u}$$