编程语言的设计原理

Design Principles of Programming Languages

Haiyan Zhao, Di Wang

赵海燕，王迪

Peking University, Spring Term 2023

Recap: untyped lambda-calculus

Syntax

$\mathtt{t}$	::=		*terms:*
		$\mathtt{x}$	*variable*
		$\lambda\mathtt{x.t}$	*abstraction*
		$\mathtt{t\ t}$	*application*
$\mathtt{v}$	::=		*values:*
		$\lambda\mathtt{x.t}$	*abstraction value*

Evaluation $\boxed{\mathtt{t} \longrightarrow \mathtt{t}'}$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1\ \mathtt{t}_2 \longrightarrow \mathtt{t}_1'\ \mathtt{t}_2} \quad \text{(E-APP1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{v}_1\ \mathtt{t}_2 \longrightarrow \mathtt{v}_1\ \mathtt{t}_2'} \quad \text{(E-APP2)}$$

$$(\lambda\mathtt{x.t}_{12})\ \mathtt{v}_2 \longrightarrow [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12} \quad \text{(E-APPABS)}$$

Syntax

- **Definition** [Terms]:

  Let $\mathcal{V}$ be a *countable set* of variable names.

  The set of terms is *the smallest set* $\mathcal{T}$ such that

  1. $x \in \mathcal{T}$ for every $x \in \mathcal{V}$;
  2. if $t_1 \in \mathcal{T}$ and $x \in \mathcal{V}$, then $\lambda x.t_1 \in \mathcal{T}$;
  3. if $t_1 \in \mathcal{T}$ and $t_2 \in \mathcal{T}$, then $t_1\ t_2 \in \mathcal{T}$.

- **Definition:** Free Variables of term t, written as $FV(t)$:

  $FV(x) = \{x\}$

  $FV(\lambda x.t_1) = FV(t_1) \setminus \{x\}$

  $FV(t_1\ t_2) = FV(t_1) \cup FV(t_2)$

Substitution

$$
\begin{aligned}
[x \mapsto s]x &= s \\
[x \mapsto s]y &= y && \text{if } y \neq x \\
[x \mapsto s](\lambda y.t_1) &= \lambda y.\ [x \mapsto s]t_1 && \text{if } y \neq x \text{ and } y \notin FV(s) \\
[x \mapsto s](t_1\ t_2) &= [x \mapsto s]t_1\ [x \mapsto s]t_2
\end{aligned}
$$

Alpha-conversion : Terms that *differ only in the names of bound variables* are interchangeable *in all contexts*.

Example:

$$
\begin{aligned}
& [x \mapsto y\ z]\ (\lambda y.\ x\ y) \\
=\ & [x \mapsto y\ z]\ (\lambda w.\ x\ w) \\
=\ & \lambda w.\ y\ z\ w
\end{aligned}
$$

Chapter 6:
Nameless Representation of Terms

Terms and Contexts

Shifting and Substitution

Bound Variables

- Recall that bound variables can be renamed, at any moment, to enable substitution:

$$
\begin{array}{lcll}
[x \mapsto s]x & = & s & \\
[x \mapsto s]y & = & y & \text{if } y \neq x \\
[x \mapsto s](\lambda y.t_1) & = & \lambda y.\ [x \mapsto s]t_1 & \text{if } y \neq x \text{ and } y \notin FV(s) \\
[x \mapsto s](t_1\ t_2) & = & [x \mapsto s]t_1\ [x \mapsto s]t_2 &
\end{array}
$$

- Variable Representation
  - Represent variables symbolically, with variable renaming mechanism
  - Represent variables symbolically, with bound variables are all different
  - "Canonically" represent variables in a way such that renaming is unnecessary
  - No use of variables: combinatory logic

Terms and Contexts

Nameless Terms

- *De Bruijin* idea: Replacing named variables by *natural numbers*, where the number k stands for "the variable bound by the $k'th$ enclosing λ". e.g.,
  - λx.x λ.0
  - λx.λy. x (y x) λ.λ. 1 (0 1)
- *e.g.,* the corresponding nameless term for the following:

  c0 = λs. λz. z;

  c2 = λs. λz. s (s z);

  plus = λm. λn. λs. λz. m s (n z s);

  fix = λf. (λx. f (λy. (x x) y)) (λx. f (λy. (x x) y));

  foo = (λx. (λx. x)) (λx. x);

Nameless Terms

- Need to keep careful track of how many free variables each term may contain.

Definition [Terms]: Let $\mathcal{T}$ be *the smallest family of sets* $\{\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots\}$ such that

1. $k \in \mathcal{T}_n$ whenever $0 \leq k < n$;
2. if $t_1 \in \mathcal{T}_n$ and $n>0$, then $\lambda.t_1 \in \mathcal{T}_{n-1}$;
3. if $t_1 \in \mathcal{T}_n$ and $t_2 \in \mathcal{T}_n$, then $(t_1\ t_2) \in \mathcal{T}_n$.

- **Note:**
  - terms with no free variables are called the 0-terms.
  - $\mathcal{T}_n$ are set of terms with at most n free variables, n-terms, numbered between 0 and n-1: a given element of $\mathcal{T}_n$ need not have free variables with all these numbers, or indeed any free variables at all. When t is closed, for example, it will be an element of $\mathcal{T}_n$ for every n.
  - two ordinary terms are *equivalent* modulo renaming of bound variables iff they have the same de Bruijn representation.

Name Context

- To deal with terms containing free variables,

  Definition: Suppose x_0 through x_n are variable names from $\mathcal{V}$. The naming context $\Gamma = x_n, x_{n-1}, \ldots x_1, x_0$ assigns to each x_i the *de Bruijn index* i. Note that the *rightmost variable* in the sequence is given the index *0*; this matches the way we count *λ binders* — from right to left — when converting a named term to nameless form.

  We write dom(Γ) for the set $\{x_n, \ldots x_1, x_0\}$ of variable names mentioned in Γ .

- e.g., $\Gamma = x \mapsto 4; y \mapsto 3; z \mapsto 2; a \mapsto 1; b \mapsto 0$, under this Γ, we have
  - x (y z) ? 4 (3 2)
  - λw. y w λ. 4 0
  - λw. λa. x λ. λ. 6

Shifting and Substitution

How to define substitution $[k \mapsto s]\ t$?

Shifting

- Under the naming context Γ : $x \mapsto 1, z \mapsto 2$

  $[1 \mapsto 2\ (\lambda.\ 0)]\ \lambda.\ 2 \longrightarrow\ \ ?$

  i.e., $[\ x \mapsto z\ (\lambda w.\ w)\ ]\ \lambda y.\ x \longrightarrow\ ?$

- When a substitution goes under a λ-abstraction, as in $[1 \mapsto s](\lambda.2)$ (i.e.,$[x \mapsto s]\ (\lambda y.x)$, assuming that 1 is the index of x in the outer context), *the context* in which the substitution is taking place becomes *one variable longer than the original*;
- We need to *increment the indices* of the *free variables* in s so that they keep referring to *the same names in the new context* as they did before.
- e.g., s = 2 (λ. 0), , i.e., s = z (λw.w), assuming 2 is the index of z in the outer context, we need to shift the 2 but not the 0

- An auxiliary operation: renumber the indices of the free variables in a term.

Shifting

DEFINITION [SHIFTING]: The d-place shift of a term $\mathtt{t}$ above cutoff c, written $\uparrow^d_c(\mathtt{t})$, is defined as follows:

$$
\begin{aligned}
\uparrow^d_c(\mathtt{k}) &= \begin{cases} \mathtt{k} & \text{if } k < c \\ \mathtt{k} + d & \text{if } k \geq c \end{cases} \\
\uparrow^d_c(\lambda.\mathtt{t}_1) &= \lambda.\ \uparrow^d_{c+1}(\mathtt{t}_1) \\
\uparrow^d_c(\mathtt{t}_1\ \mathtt{t}_2) &= \uparrow^d_c(\mathtt{t}_1)\ \uparrow^d_c(\mathtt{t}_2)
\end{aligned}
$$

We write $\uparrow^d(\mathtt{t})$ for $\uparrow^d_0(\mathtt{t})$. □

1. What is $\uparrow^2(\lambda.\lambda.\ \mathtt{1}\ (\mathtt{0}\ \mathtt{2}))$?

2. What is $\uparrow^2(\lambda.\ \mathtt{0}\ \mathtt{1}\ (\lambda.\ \mathtt{0}\ \mathtt{1}\ \mathtt{2}))$?

Substitution

DEFINITION [SUBSTITUTION]: The substitution of a term $\mathtt{s}$ for variable number $\mathtt{j}$ in a term $\mathtt{t}$, written $[\mathtt{j} \mapsto \mathtt{s}]\mathtt{t}$, is defined as follows:

$$
\begin{aligned}
[\mathtt{j} \mapsto \mathtt{s}]\mathtt{k} &= \begin{cases} \mathtt{s} & \text{if } \mathtt{k} = \mathtt{j} \\ \mathtt{k} & \text{otherwise} \end{cases} \\
[\mathtt{j} \mapsto \mathtt{s}](\lambda.\mathtt{t}_1) &= \lambda.\ [\mathtt{j}{+}1 \mapsto \uparrow^1(\mathtt{s})]\mathtt{t}_1 \\
[\mathtt{j} \mapsto \mathtt{s}](\mathtt{t}_1\ \mathtt{t}_2) &= ([\mathtt{j} \mapsto \mathtt{s}]\mathtt{t}_1\ [\mathtt{j} \mapsto \mathtt{s}]\mathtt{t}_2)
\end{aligned}
$$

□

$$
\begin{aligned}
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{x} &= \mathtt{s} & & \\
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{y} &= \mathtt{y} & & \text{if } \mathtt{y} \neq \mathtt{x} \\
[\mathtt{x} \mapsto \mathtt{s}](\lambda\mathtt{y}.\mathtt{t}_1) &= \lambda\mathtt{y}.\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 & & \text{if } \mathtt{y} \neq \mathtt{x} \text{ and } \mathtt{y} \notin FV(\mathtt{s}) \\
[\mathtt{x} \mapsto \mathtt{s}](\mathtt{t}_1\ \mathtt{t}_2) &= [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2 & &
\end{aligned}
$$

Evaluation

- To define the *evaluation relation* on nameless terms, the only thing we *need to change* (i.e., the only place where *variable names* are mentioned) is the *beta-reduction rule (computation rules),* while keep the other rules identical to what as Figure 5-3.

$$(\lambda x.\, t_{12})\ t_2 \longrightarrow [x \mapsto t_2]t_{12},$$

- How to change the above rule for nameless representation?

Evaluation

- Example:

$$(\lambda x.\ t_{12})\ t_2 \longrightarrow [x \mapsto t_2]t_{12},$$

$$(\lambda.t_{12})\ v_2 \ \longrightarrow\ \uparrow^{-1}([0 \mapsto \uparrow^{1}(v_2)]t_{12})$$

$$(\lambda.1\ 0\ 2)\ (\lambda.0) \longrightarrow 0\ (\lambda.0)\ 1$$

Homework

- Read Chapter 6.
- Do Exercise 6.2.5.

6.2.5 EXERCISE [★]: Convert the following uses of substitution to nameless form, assuming the global context is Γ = a,b, and calculate their results using the above definition. Do the answers correspond to the original definition of substitution on ordinary terms from §5.3?

1. $[\mathtt{b} \mapsto \mathtt{a}]\ (\mathtt{b}\ (\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{b}))$
2. $[\mathtt{b} \mapsto \mathtt{a}\ (\lambda\mathtt{z}.\mathtt{a})]\ (\mathtt{b}\ (\lambda\mathtt{x}.\mathtt{b}))$
3. $[\mathtt{b} \mapsto \mathtt{a}]\ (\lambda\mathtt{b}.\ \mathtt{b}\ \mathtt{a})$
4. $[\mathtt{b} \mapsto \mathtt{a}]\ (\lambda\mathtt{a}.\ \mathtt{b}\ \mathtt{a})$ □

Evaluation

- $(\lambda x.\ t_{12})\ t_2 \longrightarrow [x \mapsto t_2]t_{12},$

$$(\lambda.t_{12})\ v_2 \longrightarrow \uparrow^{-1}([0 \mapsto \uparrow^{1}(v_2)]t_{12})$$

$$(\lambda.1\ 0\ 2)\ (\lambda.0) \longrightarrow 0\ (\lambda.0)\ 1$$