编程语言的设计原理
Design Principles of Programming Languages

Zhenjiang Hu, Haiyan Zhao,
胡振江 赵海燕
Peking University, Spring, 2022

Chapter 6: Nameless Representation of Terms

Terms and Contexts

Shifting and Substitution

Bound Variables

- Recall: 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.,
  - $\lambda x.x$ $\lambda.\ 0$
  - $\lambda x.\lambda y.\ x\ (y\ x)$ $\lambda.\ \lambda.\ 1\ (0\ 1)$.

- 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:** $\mathcal{T}_n$ are set of terms with at most n free variables, n-terms, numbered between 0 and n -1

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 $dom(\Gamma)$ for the set $\{x_n, \ldots, x_0\}$ 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 ↦ 1, z ↦ 2

  [1 ↦ 2 (λ. 0)] λ. 2 ⟶ ?

  i.e., [x ↦ z (λw. w)] λy. x ⟶ ?

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

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 $\uparrow^d_0(\mathtt{t})$. □

Shifting

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

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

We write $\uparrow^d(\mathtt{t})$ for $\uparrow_0^d(\mathtt{t})$. □

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 s for variable number j in a term t, written $[j \mapsto s]t$, is defined as follows:

$$
\begin{aligned}
[j \mapsto s]k &= \begin{cases} s & \text{if } k = j \\ k & \text{otherwise} \end{cases} \\
[j \mapsto s](\lambda.t_1) &= \lambda.\ [j+1 \mapsto \uparrow^1(s)]t_1 \\
[j \mapsto s](t_1\ t_2) &= ([j \mapsto s]t_1\ [j \mapsto s]t_2)
\end{aligned}
$$

□

- Example

  $[\,1 \mapsto 2\ (\lambda.\ 0)\,]\ \lambda.\ 2 \longrightarrow\ \lambda.\ 3\ (\lambda.\ 0)$

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

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

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

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

- Example:

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

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. [b ↦ a] (b (λx.λy.b))

2. [b ↦ a (λz.a)] (b (λx.b))

3. [b ↦ a] (λb. b a)

4. [b ↦ a] (λa. b a) □