CS350: Principles of Programming Languages

Aditya Tanwar
200057

Aryan Sharma
200203

September 2023

Diamond Property: Let R be a binary relation. We say that R has diamond property if, whenever aRb and aRc, then $\exists\, d$ such that bRd and cRd.

β-reduction as a binary relation: We can define $\xrightarrow{\beta}$ as a binary relation between two λ-terms t_1 and t_2:

$$t_1 \xrightarrow{\beta} t_2 \text{ iff } t_2 \text{ can be obtained from } t_1 \text{ using 1 } \beta\text{-reduction step.}$$

1. Show that $\xrightarrow{\beta}$ does not have the Diamond property. That is, give an example λ-term t_1 such that

    1. $t_1 \xrightarrow{\beta} t_2$
    2. $t_1 \xrightarrow{\beta} t_3$
    3. $t_2 \neq t_3$
    4. $\nexists\, t_4$ s.t. $t_2 \xrightarrow{\beta} t_4$ and $t_3 \xrightarrow{\beta} t_4$

Sol. We provide the following example to show that $\xrightarrow{\beta}$ does not have the Diamond property:

- $t_1 \equiv (\lambda xy.y)\ ((\lambda u.uu)\ (\lambda v.v))\ (\lambda z.z)$
- $t_2 \equiv (\lambda y.y)\ (\lambda z.z)$
  To verify, observe that:

$$\begin{aligned} t_1 &\equiv (\lambda \underline{x}y.y)\ \underline{((\lambda u.uu)\ (\lambda v.v))}\ (\lambda z.z) \\ &\xrightarrow{\beta} (\lambda y.y)\ (\lambda z.z) \equiv t_2 \end{aligned}$$

  Therefore, $t_1 \xrightarrow{\beta} t_2$.
- $t_3 \equiv (\lambda xy.y)\ ((\lambda v_1.v_1)\ (\lambda v_2.v_2))\ (\lambda z.z)$
  To verify, observe that:

$$\begin{aligned} t_1 &\equiv (\lambda xy.y)\ ((\lambda \underline{u}.uu)\ \underline{(\lambda v.v)})\ (\lambda z.z) \\ &\xrightarrow{\beta} (\lambda xy.y)\ (\underline{(\lambda v.v)}\ (\lambda v.v))\ (\lambda z.z) \\ &\xrightarrow{\alpha} (\lambda xy.y)\ ((\lambda v_1.v_1)\ \underline{(\lambda v.v)})\ (\lambda z.z) \\ &\xrightarrow{\alpha} (\lambda xy.y)\ ((\lambda v_1.v_1)\ (\lambda v_2.v_2))\ (\lambda z.z) \equiv t_3 \end{aligned}$$

  Therefore, $t_1 \xrightarrow{\beta} t_3$.
- Now, the only β-reduction possible in t_2 is:

$$t_2 \equiv (\lambda \underline{y}.y)\ \underline{(\lambda z.z)} \xrightarrow{\beta} \lambda z.z$$

  However, the only β-reductions possible in t_3 are:

$$\begin{aligned} t_3 &\equiv (\lambda \underline{x}y.y)\ \underline{((\lambda v_1.v_1)\ (\lambda v_2.v_2))}\ (\lambda z.z) \\ &\xrightarrow{\beta} (\lambda y.y)\ (\lambda z.z) \end{aligned}$$

  and

$$\begin{aligned} t_3 &\equiv (\lambda xy.y)\ ((\lambda \underline{v_1}.v_1)\ \underline{(\lambda v_2.v_2)})\ (\lambda z.z) \\ &\xrightarrow{\beta} (\lambda xy.y)\ (\lambda v_2.v_2)\ (\lambda z.z) \end{aligned}$$

2. **Uniqueness of One-step Evaluation** ($\rightarrow$): Consider the language of arithmetic expressions. Prove that if $t \rightarrow t'$ and $t \rightarrow t''$, then $t' = t''$.

Sol. We prove the same by applying induction on the size of the term. We assume that the term t under construction is well-formed, and **not** stuck, since otherwise, it cannot be evaluated further.

Base Case: Firstly, all the values are contained in the base case; as there are no derivation rules applicable on them, therefore, there is a unique representation of each value. Thus, the base case covers all terms of size 1 by default (*0, true, false*). It also covers some terms of size > 1, of the form *succ v*.

Inductive Step:

Case 1: $t = if\ t1\ then\ t2\ else\ t3$:
Note that if t is a well defined term then $t1$ can take only be either $True$ or $False$ or a non value. It cannot be 0 or some other value, since this would lead to t being stuck, and the non-existence of t', and t''.
In each of these 3 cases exactly one rule is applicable:
i.e. if $t1\ \equiv True$, then $t\ \leftarrow\ t2$.
else if $t1\ \equiv False$, then $t\ \leftarrow\ t3$
else $t\ \leftarrow\ if\ t1'\ then\ t2\ else\ t3$, where $t1'$ is unique by induction hypothesis since $t1$ is a term of a smaller size. In the case when $t1$ is *true* (or *false*), it is trivial to show that $t' = t2 = t''$ (or $t' = t3 = t''$) as only one rule is applicable.

Case 2: $t = succ\ t1$:
Note that if t is a well defined term then $t1$ can take only be either a value or a non value.
In each of these cases exactly one rule is applicable:
i.e. if $t1\ \equiv value$, then t is also a value by definition and hence unique.
else $t\ \leftarrow\ succ\ t1'$ where $t1'$ is unique by inductive hypothesis since $t1$ is a term of smaller size, thus $succ\ t1'$ is also unique, and hence $t' = succ\ t1' = t''$.

Case 3: $t = pred\ t1$:
Note that if t is a well defined term then $t1$ can take only be either a 0, or $succ\ v$ or a non value.
In each of these cases exactly one rule is applicable:
i.e. if $t1\ \equiv 0$, then $t \leftarrow 0$ which is a value by and hence unique by definition,
else if $t\ \equiv succ\ v$ then $t\ \leftarrow\ v$ which is a value and hence unique by definition,
else $t\ \leftarrow\ pred\ t1'$ where $t1'$ is unique by inductive hypothesis since $t1$ is a term of smaller size.

Case 4: $t = iszerot1$:
Note that if $t1$ is a value, then either $t1 = 0 \Rightarrow$ the only evaluation possible is $t \rightarrow true$, in which case $t' = true = t''$, or
$t1 = succ\ v \Rightarrow$ the only evaluation possible is $t \rightarrow false$, in which case $t' = false = t''$.
Otherwise, $t1$ is a non-value, then the only evaluation possible is $t \rightarrow iszero\ t1'$, where $t1 \rightarrow t1'$. Then, by IH, as size of $t1$ is lesser than t, $t1'$ will be unique, and thus $iszero\ t1'$ is also unique, resulting in $t' = t''$.

Thus, we have shown using structural induction that in each one-step evaluation rule, if $t \rightarrow t'$ and $t \rightarrow t''$, then we have that $t' = t''$ necessarily.

3. **Non-associativity of Substitutions**: Let M, N, and P be λ-terms. Assume $x \neq y$. Show that the order of substitution matters, i.e., in general

$$M[x := N][y := P] \not\equiv M[y := P][x := N]$$

Sol. We show the above by providing an example as follows:

- $M \equiv xy$

- $N \equiv y$
- $P \equiv x$

Now,

$$\begin{aligned} M[x := N][y := P] &\equiv (xy)[x := N][y := P] \\ &\equiv ((xy)\ [x := N])\ [y := P] \\ &\equiv ((\underline{x}y)\ [x := y])\ [y := P] \\ &\equiv (yy)\ [y := P] \\ &\equiv (\underline{yy})\ [y := x] \\ &\equiv xx \end{aligned}$$

and

$$\begin{aligned} M[y := P][x := N] &\equiv (xy)[y := P][x := N] \\ &\equiv ((xy)\ [y := P])\ [x := N] \\ &\equiv ((x\underline{y})\ [y := x])\ [x := N] \\ &\equiv (xx)\ [x := N] \\ &\equiv (\underline{xx})\ [x := y] \\ &\equiv yy \end{aligned}$$

As $xx \neq yy$ in general, we arrive at $M[x := N][y := P] \not\equiv M[y := P][x := N]$.

4. **Constrained-associativity of Substitutions**: Let M, N, and P be λ-terms. Assume $x \neq y$ and $x \notin FV(P)$. Show that

$$M[x := N][y := P] \not\equiv M[y := P][x := N'] \text{ where } N' = N[y := P]$$

Sol. In order to prove the above result we perform an induction on the size of M. We define the size of a λ-term as follows (using derivation rules):

- $t \to x \Rightarrow \mathit{size}(t) = 1$
- $t \to \lambda x.\ t_1 \Rightarrow \mathit{size}(t) = 1 + \mathit{size}(t_1)$
- $t \to t_1\ t_2 \Rightarrow \mathit{size}(t) = \mathit{size}(t_1) + \mathit{size}(t_2)$

Base case: Size of M is one, i.e., M is a variable:

Case 1: $M = x$:

$LHS:\ M[x := N][y := P] = x[x := N][y := P] = N[y := P]$
$RHS:\ M[y := P][x := N] = x[y := P][x := N[y := P]] = x[x := N[y := P]] = N[y := P]$
Clearly, $LHS = RHS$.

Case 2: $M = y$:

$LHS:\ M[x := N][y := P] = y[x := N][y := P] = y[y := P]] = P$
$RHS:\ M[y := P][x := N[y := P]] = y[y := P][x := N[y := P]] = P[x := N[y := P]] = P, \because\ x \notin FV(P)$
Clearly, $LHS = RHS$.

Case 3: $M = z$, where $z \neq x,\ z \neq y$

$LHS:\ M[x := N][y := P] = z[x := N][y := P] = z[y := P]] = z$
$RHS:\ z[y := P][x := N[y := P]] = z[y := P][x := N[y := P]] = z[x := N[y := P]] = z, \because\ x \notin FV(P)$
Clearly, $LHS = RHS$.

Now let us assume that our claim holds for all terms with size less than n
Inductive Step:

Case 1: $M = \lambda z.M_1$

We can use α-renaming in M, to make sure that $z \neq x, z \neq y, z \notin FV(N), z \notin FV(P), z \notin FV(N')$. So, we now assume WLOG that all of the above holds true for z, therefore,

$$\begin{aligned} M[x := N][y := P] &\equiv (\lambda z.M_1)[x := N][y := P] \\ &\equiv \lambda z.(M_1[x := N][y := P]) \\ &\equiv \lambda z.(M_1[y := P][x := N']) && \text{(As } \mathit{size}(M_1) < \mathit{size}(M)) \\ &\equiv (\lambda z.M_1)[y := P][x := N'] \\ &\equiv M[y := P][x := N'] \end{aligned}$$

This completes the proof for this case, i.e., the abstraction rule.

Case 2: $M = M_1M_2$.

Before we jump into the main step of the proof, we first claim that $M[x := N] = M_1[x := N]M_2[x := N]$. For the same, we recall the definition of a substitution, which in this case is replacing each *free occurrence* of x in M by the term N.

Now, each free occurrence of x in M_2, has to be a free occurrence in M as well. This is because the only way x can be a free occurrence in M_2, but not in M is if x is somehow bound to a corresponding λx. x was not bound to any λx in $M_2 \Rightarrow$ there was no λx in M_2. But, x is not free in $M \Rightarrow$ there is λx in M. The only way this can happen is if there is λx in M_1.
WLOG, let $M_1 \equiv M3\lambda x.M4$ for some M_3, M_4. Then, the derivation of M will have looked something like, $M \to M_1M_2 \to (M_3\lambda x.M_4)M_2$, but this means that the scope of λx, does not go beyond M_4. We have thus arrived at a contradiction; "It is possible for a free occurrence of x in M_2 to be not a free occurrence in M."
More concisely, it can be said that each free occurrence of x in M_2 is a free occurrence in M as well.

A similar argument can be run to prove that each free occurrence of x in M_1 is a free occurrence in M as well.

Finally then, when we want to substitute x in M, it is the same as replacing each free occurrence of x in M_1 as well as replacing each free occurrence of x in M_2 at the same time. More formally, we get that:

$$M[x := N] \equiv (M_1M_2)[x := N] \equiv M_1[x := N]\ M_2[x := N]$$

We therefore have the following result:

$$\begin{aligned} M[x := N][y := P] &\equiv (M_1M_2)[x := N][y := P] \\ &\equiv (M_1[x := N]\ M_2[x := N])[y := P] \\ &\equiv M_1[x := N][y := P]\ M_2[x := N][y := P] \\ &\equiv M_1[y := P][x := N']\ M_2[y := P][x := N'] && \text{(By IH)} \\ &\equiv (M_1[y := P]\ M_2[y := P])[x := N'] \\ &\equiv (M_1M_2)[y := P][x := N'] \\ &\equiv M[y := P][x := N'] \\ M[x := N][y := P] &\equiv M[y := P][x := N'] \end{aligned}$$

This completes the proof for this case, i.e., the application rule.

Since the constrained-associativity of substitutions has been shown for each derivation rule, we can conclude that the rule holds for all λ-terms.

(a) **Church Numerals**: Explain with suitable examples, what simple arithmetic function does the following λ-term represents:

$$\lambda n.\ n\ (\lambda p\ z.\ z\ (\mathit{succ}\ (p\ \mathit{true}))(p\ \mathit{true}))(\lambda z.\ z\ \mathit{zero}\ \mathit{zero})\ \mathit{false}$$

Here, *succ*, *true*, *zero*, *false* represent the λ-terms defined in the lectures.

Sol. For the sake of brevity, we use the following shorthand:

$$s \equiv \mathit{succ} \qquad 0 \equiv \mathit{zero} \qquad T \equiv \mathit{true} \qquad F \equiv \mathit{false}$$
$$\mathcal{G} \equiv (\lambda p\ z.\ z\ (s\ (p\ T))\ (p\ T))$$

Now let us try applying $(\lambda n.\ n\ \mathcal{G}\ (\lambda\ z.\ z\ 0\ 0)F)$ to some values of n (i.e., natural numbers) and try to build an intuition for this function.

$$
\begin{aligned}
&(\underline{\lambda n}.\ n\ \mathcal{G}\ (\lambda\ z.\ z\ 0\ 0)\ F)\ \underline{0} \\
&\xrightarrow{\beta} 0\ \mathcal{G}\ (\lambda\ z.\ z\ 0\ 0)\ F \\
&\xrightarrow{\beta} (\underline{\lambda m}\ z.\ z)\ \underline{\mathcal{G}}\ (\lambda\ z.\ z\ 0\ 0)\ F \\
&\xrightarrow{\beta} (\underline{\lambda z}.\ z)\ \underline{(\lambda\ z.\ z\ 0\ 0)}\ F \\
&\xrightarrow{\beta}\ (\underline{\lambda z}.\ z\ 0\ 0)\ \underline{F} \\
&\xrightarrow{\beta} \underline{F}\ 0\ \underline{0} \\
&\rightarrow 0
\end{aligned}
$$

Similarly on applying the function to 1 we get:

$$(\lambda\ z.\ z\ 1\ 0)\ F \xrightarrow{\beta} F\ 1\ 0 \rightarrow 0$$

Thus, we claim that $(n\ \mathcal{G}\ (\lambda z.\ z\ 0\ 0))$ reduces to $(\lambda z.\ z\ n\ \mathit{pred}(n))$ after a finite number of β-reductions.

We now prove our claim using induction on n:

- **Base case**: The claim clearly holds for $n = 0$ and $n = 1$.
- **Inductive case**: Let us assume, that the Inductive Hypothesis holds for $n = k$. Now, for $n = k + 1$,

$$
\begin{aligned}
(k+1)\ \mathcal{G}\ (\lambda z.\ z\ 0\ 0)\ &\equiv\ (\underline{\lambda m\ w}.\ m\ (m(\ldots\{k\text{-times}\}w)))\ \underline{\mathcal{G}}\ \underline{(\lambda z.\ z\ 0\ 0)} \\
&\rightarrow \mathcal{G}\ \underline{(\mathcal{G}\ (\mathcal{G}\ \ldots\{k\text{-times}\}(\lambda z.\ z\ 0\ 0)))} \\
&\rightarrow \mathcal{G}\ (\lambda z.\ z\ k\ \mathit{pred}(k)) && \text{(By IH)} \\
&\rightarrow (\underline{\lambda p}\ z.\ z\ (s\ (p\ T))\ (p\ T))\ \underline{(\lambda z.\ z\ k\ \mathit{pred}(k))} \\
&\xrightarrow{\beta} (\lambda z.\ z\ (s\ ((\underline{\lambda z}.\ z\ k\ \mathit{pred}(k))\ \underline{T}))\ ((\lambda z.\ z\ k\ \mathit{pred}(k))\ T)) \\
&\xrightarrow{\beta} \lambda z.\ z\ (s\ (T\ k\ \mathit{pred}(k))\ (((\underline{\lambda z}.\ z\ k\ \mathit{pred}(k))\ \underline{T})) \\
&\xrightarrow{\beta} \lambda z.\ z\ (s\ (\underline{T\ k}\ \mathit{pred}(k))\ (\underline{T\ k}\ \mathit{pred}(k))) \\
&\rightarrow \lambda z.\ z\ (s\ k)\ k \\
&\equiv \lambda z.\ z\ (k+1)\ \mathit{pred}(k+1)
\end{aligned}
$$

Hence, proved

Now, applying n_0 to $(\lambda n.\ n\ \mathcal{G}\ (\lambda\ z.\ z\ 0\ 0)F)$, we get:

$$
\begin{aligned}
(\underline{\lambda n}.\ n\ \mathcal{G}\ (\lambda\ z.\ z\ 0\ 0)F)\ \underline{n_0}\ &\xrightarrow{\beta} (n_0\ \mathcal{G}\ (\lambda z.\ z\ 0\ 0)\ F) \\
&\equiv (\underline{\lambda z}.\ z\ n_0\ \mathit{pred}(n_0))\ \underline{F} && \text{(Using our claim above)} \\
&\xrightarrow{\beta} (\underline{F}\ n_0\ \underline{\mathit{pred}(n_0))} \\
&\rightarrow \mathit{pred}(n_0)
\end{aligned}
$$

Therefore the given function is a **natural number predecessor** function.

References

[1] https://isabelle.in.tum.de/nominal/example.html