2 steps: 1. Identify when to return a particular right-hand side.
2. Return it: • Gather missing information, if any.
• Use the information you have.

4.3 Substitution

Subst(arguments) = result

Substitution for Fun++ abstract syntax

$$
\begin{aligned}
\mathit{subst}((\text{num } n), x, e2) &= (\text{num } n)\\
\mathit{subst}((\text{id } x), x, e2) &= e2\\
\mathit{subst}((\text{id } y), x, e2) &= (\text{id } y) \quad \text{if } x \neq y\\
\mathit{subst}((\text{lam } x\ eB), x, e2) &= (\text{lam } x\ eB)\\
\mathit{subst}((\text{lam } y\ eB), x, e2) &= (\text{lam } y\ \mathit{subst}(eB, x, e2))\\
&\quad \text{if } x \neq y\\
\mathit{subst}((\text{app } eFun\ eArg), x, e2) &= (\text{app } \mathit{subst}(eFun, x, e2)\ \mathit{subst}(eArg, x, e2))\\
\mathit{subst}((\text{binop op } eL\ eR), x, e2) &= (\text{binop op } \mathit{subst}(eL, x, e2)\ \mathit{subst}(eR, x, e2))\\
\mathit{subst}((\text{pair } eL\ eR), x, e2) &= (\text{pair } \mathit{subst}(eL, x, e2)\ \mathit{subst}(eR, x, e2)) \quad \text{show alternate}\\
&\quad \text{see next page}\\
\mathit{subst}((\text{bfalse}), x, e2) &= (\text{bfalse})\\
\mathit{subst}((\text{btrue}), x, e2) &= (\text{btrue})\\
\mathit{subst}((\text{ite } e\ eThen\ eElse), x, e2) &= (\text{ite } \mathit{subst}(e, x, e2)\ \mathit{subst}(eThen, x, e2)\ \mathit{subst}(eElse, x, e2))\\
\mathit{subst}((\text{with } x\ e\ eB), x, e2) &= (\text{with } x\ \mathit{subst}(e, x, e2)\ eB)\\
\mathit{subst}((\text{with } y\ e\ eB), x, e2) &= (\text{with } y\ \mathit{subst}(e, x, e2)\ \mathit{subst}(eB, x, e2))\\
&\quad \text{if } x \neq y\\
\mathit{subst}((\text{with* } ()\ eB), x, e2) &= (\text{with* } ()\ \mathit{subst}(eB, x, e2))\\
\mathit{subst}((\text{with* } ((x\ e)\ bindings)\ eB), x, e2) &= (\text{with* } ((x\ \mathit{subst}(e, x, e2))\ bindings)\ eB)\\
\mathit{subst}((\text{with* } ((y\ e)\ bindings)\ eB), x, e2) &= (\text{with* } ((y\ \mathit{subst}(e, x, e2))\ bindings')\ eB')\\
&\quad \text{if } x \neq y\\
&\quad \text{where } \mathit{subst}((\text{with* } bindings\ eB), x, e2) = (\text{with* } bindings'\ eB')\\
\mathit{subst}((\text{pair-case } e\ x1\ x2\ eB), x, e2) &= (\text{pair-case } \mathit{subst}(e, x, e2)\ x1\ x2\ eB)\\
&\quad \text{if } x = x1 \text{ or } x = x2\\
\mathit{subst}((\text{pair-case } e\ x1\ x2\ eB), x, e2) &= (\text{pair-case } \mathit{subst}(e, x, e2)\ x1\ x2\ \mathit{subst}(eB, x, e2))\\
&\quad \text{if } x \neq x1 \text{ and } x \neq x2\\
\mathit{subst}((\text{rec } x\ eB), x, e2) &= (\text{rec } x\ eB)\\
\mathit{subst}((\text{rec } y\ eB), x, e2) &= (\text{rec } y\ \mathit{subst}(eB, x, e2))\\
&\quad \text{if } x \neq y
\end{aligned}
$$

Alternate:
Suppose subst((··),··) = (with* bindings' eB').*
Return ((with* ((y subst(e,x,e2)) bindings') eB')

* What if subst returns something else?

subst((pair eL eR), x, e2) = (pair eL' eR')

where subst(eL, x, e2) = eL'

and subst(eR, x, e2) = eR'

(alternate:

Suppose subst(eL, x, e2) = eL'

and subst(eR, x, e2) = eR'.

Return (pair eL' eR').)

```
(let* ([eL-subst (subst eL x e2)]
       [eR-subst (subst eR x e2)])
  (pair eL-subst
        eR-subst))
```