Jason Zhang jzhan127

Collaborators: Dan Qian

1. a. (Fun x → x+1)[3/x] ⇒ Fun x → x+1

b. (Fun x → Fun y → x and y)[True/x] ⇒ ~~F~~ Fun x → Fun y → x And y

c. (Fun x → Fun y → x And y)[True/y] ⇒ Fun x → Fun y → x And y

d. (Fun x → Fun y → x And y)[True/z] ⇒ Fun x → Fun y → x And Y

e. ((Fun x → x)(Fun y → x))[2/x] ⇒ (Fun x → x)(Fun y → 2) ⇒ Fun y → 2

f. (Fun x → Let y = x+1 In y+2)[5/y] ⇒ Fun x → Let y = x+1 In y+2

g. (Fun x → Let y = x+1 In z+2)[5/z] ⇒ Fun x → Let y = x+1 In (5+2)

2. (Fun f → (Fun z → (f z) + 1))(Fun y → y+1)

We Let $v \Rightarrow \star$ as an abbreviation for $v \Rightarrow v$ (when lengthy)

a. part 1.

$$\frac{\dfrac{}{\text{Fun } f \to \text{Fun } z \to (f\ z)+1 \Rightarrow \star} \quad \dfrac{}{\text{Fun } y \to y+1 \Rightarrow \star} \quad \dfrac{}{\text{Fun } z \to (\text{Fun } y \to y+1)\ z + 1 \Rightarrow \star}}{(\text{Fun } f \to (\text{Fun } z \to (f\ z)+1))(\text{Fun } y \to y+1) \Rightarrow \text{Fun } z \to (\text{Fun } y \to y+1)\ z+1}$$

b.

$$\frac{\text{using part a} \quad \dfrac{}{4 \Rightarrow 4} \quad \dfrac{\dfrac{\dfrac{}{\text{Fun } y \to y+1 \Rightarrow \star} \quad \dfrac{}{4 \Rightarrow 4} \quad \dfrac{\dfrac{}{4 \Rightarrow 4} \quad \dfrac{}{1 \Rightarrow 1}}{4+1 \Rightarrow 5}}{(\text{Fun } y \to y+1)\ 4 \Rightarrow 5} \quad \dfrac{}{1 \Rightarrow 1}}{(\text{Fun } y \to y+1)\ 4 + 1 \Rightarrow 6}}{(\text{Fun } f \to (\text{Fun } z \to (f\ z) + 1))(\text{Fun } y \to y+1)\ 4 \Rightarrow 6}$$

2. c. (Fun → x (Fun a → a+1)(If x=2 Then 0 Else 1)) 2

~~Fun x → (Fun a → a+1~~

$$\dfrac{\dfrac{}{\text{(Fun x → (Fun a → a+1)(If x=2 then 0 Else 1))} \Rightarrow ☆} \quad \dfrac{}{2 \Rightarrow 2} \quad \dfrac{\dfrac{}{\text{Fun a → a+1} \Rightarrow ☆} \quad \dfrac{\dfrac{\dfrac{}{2\Rightarrow 2}\quad\dfrac{}{2\Rightarrow 2}}{2=2 \Rightarrow \text{True}} \quad \dfrac{}{0 \Rightarrow 0}}{\text{If 2=2 Then 0 Else 1} \Rightarrow 0} \quad \dfrac{\dfrac{}{0\Rightarrow 0}\quad\dfrac{}{1\Rightarrow 1}}{0+1 \Rightarrow 1}}{\text{(Fun a → a+1)(If 2=2 Then 0 Else 1)} \Rightarrow 1}}{\text{(Fun X → (Fun a → a+1) (If x=2 Then 0 Else 1)) 2} \Rightarrow 1}$$

3. (Fun q → q = False) 0 ⇒ False

$$\dfrac{\dfrac{}{\text{Fun q → q = False} \Rightarrow ☆} \quad \dfrac{}{0 \Rightarrow 0} \quad \dfrac{\dfrac{}{0 \Rightarrow 0}\quad\dfrac{}{\text{False} \Rightarrow \text{False}}}{0 = \text{False} \Rightarrow \text{False}}}{\text{(Fun q → q = False) 0} \Rightarrow \text{False}}$$

Cannot hold due to the fact that we have no rule for Comparing Bool and Int

4. a. (Negate) (plus) (minus) (Times)

$e ::= v \mid -e \mid e+e \mid e-e \mid e*e \mid$

$v ::= \ldots -1 \mid 0 \mid 1 \mid 2 \mid 3 \ldots$ Integer values

b. Int value rule $\dfrac{}{v \Rightarrow v}$

Negation rule $\dfrac{e \Rightarrow v}{-e \Rightarrow -v}$ (Integer Negation)

plus rule $\dfrac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2}{e_1 + e_2 \Rightarrow v_1 + v_2}$ (Integer Sum)

Minus rule $\dfrac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2}{e_1 - e_2 \Rightarrow v_1 - v_2}$ (Integer difference)

Times rule $\dfrac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2}{e_1 * e_2 \Rightarrow v_1 * v_2}$ (Integer multiplication)

C. To prove Arith is normalizing, we will perform Induction on the size of expression e.

To Start, we define size(e) to be the number of operations in the expression e. Operations are: negation, plus, minus, times.

For our Base case where size(e) = ϕ, we perform only 1 operation. Based on our rules, we know that every operation outputs a value for any Integer input(s). ~~inputs~~ Thus for any e with size(e) = 1 : $e \Rightarrow v$

We now will use Strong induction. Assume for e where $1 \leq \text{Size}(e) \leq K$, $e \Rightarrow v$, prove that $e \Rightarrow v$ holds for Size(e) = K+1.

Lets assume we have some expression f s.t. Size(f) = K+1. f is constructed from the same grammar above. f can be deconstructed as $f = Op(e_0)$ ~~where op~~ is ~~some operation~~ or $f = Op(e_1, e_2)$ where op is an operation (in the grammar) and e_0, e_1, e_2 are expressions of size $\leq K$. Based on our Induction $e_0 \Rightarrow v_0$, $e_1 \Rightarrow v_1$, $e_2 \Rightarrow v_2$ and thus f can be deconstructed as $f = Op(v_0)$ or $f = Op(v_1, v_2)$. ~~From our base case also This is essentially f~~ f essentially applies one operation on integer values and from our base case we know that based on our rules, any of the operations in Arith that is applied on integer values has a well defined integer output value. Thus $f = Op(v_1, v_2) \Rightarrow v$ or $f = Op(v_0) \Rightarrow v$, thus we have that

$f \Rightarrow v$ where $size(f) = K+1$

Since our induction holds, the the Arith language is normalizing for all Arith expressions e.