OpSem Theory

COMP105 Fall 2015

Andrew Burgos

Problem 16

(a) Awk-like semantics

Unbound
$x \notin \mathsf{dom}\, \rho$
$x \notin \mathsf{dom}\, \xi$

$\langle VAR(x),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 0,\ \xi(x- > \ 0),\ \phi,\ \rho\rangle$
Global
$x \notin \mathsf{dom}\, \rho$
$\langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle$

$\langle SET(x, e)\, \xi,\ \phi,\ \rho\rangle \Downarrow \langle \mathsf{v},\ \xi'\ (x- > \ v),\ \phi,\ \rho'\rangle$

(b) Icon-like semantics

Unbound
$x \notin \mathsf{dom}\, \rho$
$x \notin \mathsf{dom}\, \xi$

$\langle VAR(x),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 0,\ \xi,\ \phi,\ \rho(x- > \ 0)\rangle$
Formal
$x \notin \mathsf{dom}\, \xi \langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle$

$\langle SET(x, e)\, \xi,\ \phi,\ \rho\rangle \Downarrow \langle \mathsf{v},\ \xi',\ \phi,\ \rho'\ (x- > \ v)\rangle$

(c) Which do you prefer and why?

Icons method of implementation is my preferred method. Awk's method seems like a risk when dealing with extensive amounts of code because of the likely scenario that there will be conflicting unbound names.

Problem 13

$(begin(set\, x\, 3)\ x) \qquad \rho(x)\ =\ 99$
$\langle LIT\, 3,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho\rangle$
$x \in \mathsf{dom}\, \rho(x- > \ 3)$ - Formal Var
$x \in \mathsf{dom}\, \rho$ - Formal Assign

$$\frac{Formal\ Assign\ -\ \langle SET\,(x,\ LIT\, 3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho(x- > 3)\rangle\ Formal\ Var\ -\ \langle VAR(x),\ \xi,\ \phi,\ \rho}{\langle BEGIN\,(SET\,(x,\ LIT\, 3),\ VAR(x)),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi',\ \phi,\ \rho'\ (x- > 3}$$

Problem 14

$\langle IF\,(\,VAR(x),\ VAR(x),\ LIT\, 0),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle\, V,\ \xi',\ \phi,\ \rho'\rangle$

$\langle VAR(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle V_2,\, \xi'',\, \phi,\, \rho''\rangle$

If false both V1 and the resulting V2 are 0. If true Var x is returned which results in Var x = Var x

If False:

$\langle VAR(x),\, \xi,\, \phi,\, \rho\rangle \langle V_1,\, \xi',\, \phi,\, \rho'\rangle$, $V_1 \,=\, 0$, $\langle LIT\, 0,\, \xi',\, \phi,\, \rho'\rangle \Downarrow \langle 0,\, \xi'',\, \phi,\, \rho''\rangle$

If True:

$\langle VAR(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle V_1,\, \xi',\, \phi,\, \rho'\rangle$, $V_1 \;=\, / \,=\; 0$, $\langle VAR\, x,\, \xi',\, \phi,\, \rho'\rangle \Downarrow \langle V_2,\, \xi',\, \phi,\, \rho''\rangle$

$\langle IF\,(\, VAR(x),\; VAR(x),\; LIT\, 0),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle V_2,\, \xi',\, \phi,\, \rho'\rangle$

Problem 23

LITERAL

$\langle LIT(v),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi,\, \phi,\, \rho\rangle$

We can evaluate the literal without touching the stack

FORMAL VAR

$$\frac{x \in \mathsf{dom}\, \rho}{\langle VAR(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \rho(x),\, \xi,\, \phi,\, \rho\rangle}$$

We can pop ρ off the stack and see if x exists within domain ρ. We then push ρ back onto the stack

FORMAL ASSIGN

$$\frac{x \in \mathsf{dom}\, \rho, \quad \langle e,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho'\rangle}{\langle SET(x, e),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho' x -> \; v\rangle}$$

Pop ρ off the stack and check to see if x exists within the domain. Then use the inductive hypothesis to evaluate $\langle e,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho'\rangle$ which will pop and push ρ and ρ' Now we can pop ρ', and push the resulting environment $\rho'\,(x -> \; v)$

GLOBAL VAR

$$\frac{x \notin \mathsf{dom}\, \rho,\;\; x \in \mathsf{dom}\, \xi}{\langle VAR(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \xi(x),\, \xi,\, \phi,\, \rho\rangle}$$

By popping ρ and seeing that x does not exist within domain ρ. Then we perform the evaluation and then push ρ back onto the stack

EMPTY BEGIN

$\langle BEGIN(),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle 0,\, \xi,\, \phi,\, \rho\rangle$

Can be implemented without looking at an environment or touching the stack

BEGIN

$\langle e_1,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_1,\, \xi',\, \phi,\, \rho'\rangle$

$\langle e_2,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_2,\, \xi'',\, \phi,\, \rho''\rangle$

$\langle en,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle vn,\, \xi n,\, \phi,\, \rho n\rangle$

$\langle BEGIN(e_1,\; e_2,\; ....\; en),\, \xi 0,\, \phi,\, \rho 0\rangle \Downarrow \langle vn,\, \xi n,\, \phi,\, \rho n\rangle$

Evaluate each expression e1, e2, . . . , en using the inductive hypothes. For each expression e, the implementation pops e and then pushes the next e.

GLOBAL ASSIGN

$$\frac{x \notin \mathsf{dom}\,\rho,\ x \in \mathsf{dom}\,\xi \langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle}{\langle SET(x,\ e),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi'\ x->v,\ \phi,\ \rho'\rangle}$$

We need to check to see that x does not exist within domain ρ. We do this by popping ρ and then pushing it back onto the stack. Next, using the induction hypothesis we can evaluate $\langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle$ using a stack. This evaluation will pop ρ and push ρ'

IFTRUE

$$\frac{\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle\ \ V_1\,!=\ 0\ \langle e_2,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_2,\ \xi'',\ \phi,\ \rho''\rangle}{\langle IF(e_1,\ e_2,\ e_3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_3,\ \xi''\ x->v,\ \phi,\ \rho''\rangle}$$

Use the induction hypothesis to evaluate $\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle$. Doing this will pop ρ and push ρ' onto the stack. We can use the induction hypothesis again to show that evaluating e2 can pop ρ', push ρ'' When e1 evaluates to a nonzero value we can evaluate IF(e1, e2, e3) which pops and pushes ρ''

IFFALSE

$$\frac{\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle\ \ V_1\ =\ 0\ \langle e_3,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_3,\ \xi'',\ \phi,\ \rho''\rangle}{\langle IF(e_1,\ e_2,\ e_3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_3,\ \xi''\ x->v,\ \phi,\ \rho''\rangle}$$

Holds true for the answer above.

APPLY ADD

$\langle e_1,\ \xi 0,\ \phi,\ \rho 0\rangle \Downarrow \langle v_1,\ \xi 1,\ \phi,\ \rho 1\rangle$
$\langle e_2,\ \xi 1,\ \phi,\ \rho 1\rangle \Downarrow \langle v_2,\ \xi 2,\ \phi,\ \rho 2\rangle$

$\langle APPLY(f,\ e_1,\ e_2),\ \xi 0,\ \phi,\ \rho 0\rangle \Downarrow \langle v_1\ +\ v_2,\ \xi 2,\ \phi,\ \rho 2\rangle$

By the induction hypothesis, we can evaluate e1 and e2 using a stack. Doing this for each iteration will pop $\rho 0$ and push $\rho 2$

WHILEITERATE

$\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle\ \ v_1\,!=\ 0$
$\langle e_2,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_2,\ \xi'',\ \phi,\ \rho''\rangle\,\langle WHILE(e_1,\ e_2),\ \xi'',\ \phi,\ \rho''\rangle \Downarrow \langle v_3,\ \xi''',\ \phi,\ \rho'''\rangle$

$\langle WHILE(e_1,\ e_2),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_3,\ \xi''',\ \phi,\ \rho'''\rangle$

Using the induction hypothesis we evaluate $\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle$, and the evaluation will pop ρ and push ρ'. We can do the same when evaluating $\langle e_2,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_2,\ \xi'',\ \phi,\ \rho''\rangle$ using a stack, popping ρ' and pushing ρ''. We can do this for each subsequent version of ρ environments

WHILEEND

$$\frac{\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle\ \ v_1\ =\ 0}{\langle WHILE(e_1,\ e_2),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 0,\ \xi',\ \phi,\ \rho'\rangle}$$

Using the induction hypothesis we evaluate $\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle$ using a stack, and the evaluation will pop ρ and push ρ''. This implementation does not touch the environment or the stack.