OpSem Theory

COMP105 Fall 2015

Benjamin Tanen

Problem 16

(a) Awk-like semantics

$$\frac{x \notin \mathsf{dom}\ \rho \qquad x \notin \mathsf{dom}\ \xi}{\langle \mathsf{VAR}(x),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle \xi'(x),\ \xi'\{x \to 0\},\ \phi,\ \rho\rangle}\mathsf{AwkVar}$$

$$\frac{x \notin \mathsf{dom}\ \xi \quad \langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle}{\langle \mathsf{SET}(x,\ e),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi'\{x \to v\},\ \phi,\ \rho'\rangle}\mathsf{AwkSet}$$

(b) Icon-like semantics

$$\frac{x \notin \mathsf{dom}\ \rho \qquad x \notin \mathsf{dom}\ \xi}{\langle \mathsf{VAR}(x),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle \rho'(x),\ \xi',\ \phi,\ \rho'\{x \to 0\}\rangle}\mathsf{IconVar}$$

$$\frac{x \notin \mathsf{dom}\ \rho \quad \langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle}{\langle \mathsf{SET}(x,\ e),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\{x \to v\}\rangle}\mathsf{IconSet}$$

(c) Which do you prefer and why?

I personally would prefer to have the Icon-like semantics. This is because if a user is going to attempt to use a variable without declaring it, this variable is likely to be of relatively low importance to the user and the program (e.g. a counter variable). Therefore, having this variable only stay in the local environment makes sense because the user was likely not planning on using it in a larger (global) scope. Thus placing it in the global variable environment would be unnecessary, making the Icon-like semantics more appropriate.

Problem 13

$$\frac{\dfrac{x \in \mathsf{dom}\ \rho \qquad \dfrac{}{\langle \mathsf{LITERAL}(3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho\rangle}}{\langle \mathsf{SET}(x,\ \mathsf{LITERAL}(3)),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho\{x \to 3\}\rangle} \qquad \frac{x \in \mathsf{dom}\ \rho\{x \to 3\}}{\langle \mathsf{VAR}(x),\ \xi,\ \phi,\ \rho\{x \to 3\}\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho\{x \to 3\}\rangle}}{\langle \mathsf{BEGIN}(\mathsf{SET}(x,\ \mathsf{LITERAL}(3)),\ \mathsf{VAR}(x)),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 3,\ \xi,\ \phi,\ \rho\{x \to 3\}\rangle}$$

Problem 14

For the following proof, there are two cases given where $x \in \mathsf{dom}\ \rho$. There are additionally two more cases such that $x \in \mathsf{dom}\ \xi$ for $\xi(x) = 0$ and for $\xi(x) \neq 0$. However, the cases for $x \in \mathsf{dom}\ \rho$ are given without loss of generality. The differences between the formal variable evaluation and the global variable evaluation are subtle and are similar enough for this proof to be shown WLOG for just the two cases for which $x \in \mathsf{dom}\ \rho$.

Thus there are two cases to consider for this derivation. First, if $\mathsf{VAR}(x) = 0$ (*IfTrue*) and second, if $\mathsf{VAR}(x) \neq 0$ (*IfFalse*):

$$\dfrac{\dfrac{x \in \mathsf{dom}\,\rho \quad \rho(x) = v_2 \neq 0}{\langle \mathsf{VAR}(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \rho(x),\, \xi,\, \phi,\, \rho\rangle} \quad \dfrac{x \in \mathsf{dom}\,\rho \quad \rho(x) = v_2 \neq 0}{\langle \mathsf{VAR}(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \rho(x),\, \xi,\, \phi,\, \rho\rangle} \; \rho(x) \neq 0}{\langle \mathsf{IF}(\mathsf{VAR}(x),\, \mathsf{VAR}(x),\, \mathsf{LITERAL}(0)),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \rho(x),\, \xi,\, \phi,\, \rho\rangle}$$

and

$$\dfrac{\dfrac{x \in \mathsf{dom}\,\rho \quad \rho(x) = v_2 = 0}{\langle \mathsf{VAR}(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle \rho(x),\, \xi,\, \phi,\, \rho\rangle} \quad \dfrac{}{\langle \mathsf{LITERAL}(0),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle 0,\, \xi,\, \phi,\, \rho\rangle} \; \rho(x) = 0}{\langle \mathsf{IF}(\mathsf{VAR}(x),\, \mathsf{VAR}(x),\, \mathsf{LITERAL}(0)),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle 0,\, \xi,\, \phi,\, \rho\rangle}$$

As can be seen, evaluating $\mathsf{VAR}(x)$ in both cases yields $\rho(x)$. In the first case, $\rho(x) \neq 0$, resulting in the if-statement to evaluate $e_2 = \mathsf{VAR}(x)$ which again evaluates to $\rho(x)$. This result of $\rho(x)$ is thus the final evaluation of the if-statement, so it can be seen that the evaluation of $\mathsf{VAR}(x)$ is equal to the final if-statement evalutation $\rho(x)$.

In the case that $\rho(x) = 0$, the evaluation of $e_1 = \mathsf{VAR}(x)$ gives 0, resulting in an evaluation of $e_3 = \mathit{LITERAL}(0)$. This ultimately returns 0 for the whole if-statement, again showing that the evaluation of $\mathsf{VAR}(x)$ is equal to the evaluation of $\mathit{IF}(\mathsf{VAR}(x), \mathsf{VAR}(x), \mathsf{LITERAL}(0))$. Thus, since it was given that $\langle \mathit{IF}(\mathsf{VAR}(x), \mathsf{VAR}(x), \mathsf{LITERAL}(0)),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_1,\, \xi,\, \phi,\, \rho\rangle$, that $\langle \mathsf{VAR}(x),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_2,\, \xi,\, \phi,\, \rho\rangle$, and that the $\mathit{IF}(\mathsf{VAR}(x), \mathsf{VAR}(x), \mathsf{LITERAL}(0))$ and $\mathsf{VAR}(x)$ evaluate to the same thing in both cases, thus showing $v_1 = v_2$.

Problem 23

For this proof, we will be showing that any evaluation judgement done in `impcore` can be written and described as a series of pops and pushes of the formal variable environment ρ. More specifically, we are showing:

a. In `eval`, the implementation of every proof rule that ends in the judgement form $\langle e,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho'\rangle$ can be changed to pop ρ off the stack and push ρ' onto the stack. (It is possible that $\rho' = \rho$)
b. If $\rho' = \rho$, then the only copy of ρ is the one on top of the stack. If $\rho' = \rho$, then one ρ is popped off the stack, it is thrown away and never used again. In particular, no environment ever needs to be copied anywhere except on the stack; that is, the stack holds all the environments that will ever be used in any future evaluation.

This can be accomplished by breaking down each evaluation judgement into their appropriate parts and proving each one via induction. In a sense, we can subdivide the evaluations into base cases and build from there.

$$\dfrac{}{\langle \mathsf{LITERAL}(v),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v,\, \xi,\, \phi,\, \rho\rangle}$$

The first of these base cases is the assessment of a literal. This first assessement (seen above) obviously does not require any looking at or dealing with any environment at all and can actually be down without popping the environment

at all. Thus this evaluation can obviously be replicated in this new pop / push standard and can be seen as a base case for other evaluations.

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

Next we can consider the local evaluation of a variable x. Unlike the literal assessment, this does require popping ρ off the stack to evaluate the value of $\rho(x)$. However, as can be seen in the environments before and after the evaluation, assessing the value of the variable cannot take any effect or change on the environment itself. Therefore once the variable is evaluated, ρ can simply be pushed back onto the stack unchanged. Again this shows that this can be replicated in a pop and push environment setting and this evaluation can also be seen as another base case.

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

The simplist next step is to look at the global variable evaluation. For this, it would simply only require popping ρ off the stack, checking if variable x is in the environment domain (and concluding no), and then pushing it back on the stack. Since no changes are being made to the local environment ρ, this evaluation again stays within the lines of the stack architecture.

$$\frac{}{\langle \mathsf{BEGIN}(),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle 0,\ \xi,\ \phi,\ \rho\rangle}$$

The final base case evaluation is the empty begin case. As can be seen through the rule above, if the begin statement is empty, nothing is evaluated and the expression simply returns 0. It can also be seen that the environments cannot possibly change over this evaluation, similiarly showing that this can be replicated via popping and pushing. In fact, since the assessment doesn't require any environment evaluation at all, this can be done without even touching the stack.

$$\langle e,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v,\ \xi',\ \phi,\ \rho'\rangle$$

From here, we can analyze the evaluation of an expression e (such as above). During this evaluation, ρ would be popped off the stack, the expression e would be evaluated (this expression could be any of the other expressions which could then in turn evaluate to other expressions), and after evaluating e, ρ' would be pushed back onto the stack. Note that this environment is ρ', a new mutated verison of the original ρ that was made without making a copy of ρ. At the end of this whole evaluation, ρ' is the only environment of such with no outstanding copies. Even though it is unclear if the evaluation of e results in $\rho \neq \rho'$, this assessment still meets within the bounds of (a) and (b) because there remains only

one copy of the environment and it is at the top of the stack. Thus an evaluation of expression e stays within the grounds of this pop and push environment. We can use this to help build out and prove the rest of our rules.

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

For formal assignment, this process can be replicated by popping ρ and checking if $x \in \mathsf{dom}\ \rho$ and pushing ρ back onto the stack. From here, we can evaluate the expression e as we did above resulting in ρ' on the stack while we throw away ρ since it will not be used again. Finally, we can pop ρ' off the stack and remap the variable x to the value v (the result of the evaluation of e) and pushing this altered ρ' back onto the stack. This operation maintains only one copy of the formal variable environment at any one time so it again meets our standards.

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

The global assignment is similar to the formal assignment with fewer steps. Like the formal assign, ρ is popped off the stack, we check if $x \notin \mathsf{dom}\ \rho$ (this time we conclude yes), and we push ρ back on the stack. Again we evaluate expression e as we did before, resulting in ρ' at the top of the stack with ρ thrown away. This time, however, ρ' is not mutated / remapped for a new value of x so it can again easily be seen that there exists only one copy of the environment (and it is at the top of the stack).

$$\frac{\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle\ v_1 \neq 0\ \langle e_2,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_2,\ \xi'',\ \phi,\ \rho''\rangle}{\langle \mathsf{IF}(e_1,\ e_2,\ e_3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_2,\ \xi'',\ \phi,\ \rho''\rangle}$$

Based on the our previous knowledge of the evaluation of any expression e, we know that the evaluation of expressions e_1 and e_2 are valid through induction. Through these two evaluations, we pop ρ, push ρ', pop ρ', and finally push back ρ''. In the case that $v_1 \neq 0$, we can see that our compuation can be done through popping and pushing only one copy of the formal environment.

$$\frac{\langle e_1,\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_1,\ \xi',\ \phi,\ \rho'\rangle\ v_1 = 0\ \langle e_3,\ \xi',\ \phi,\ \rho'\rangle \Downarrow \langle v_3,\ \xi'',\ \phi,\ \rho''\rangle}{\langle \mathsf{IF}(e_1,\ e_2,\ e_3),\ \xi,\ \phi,\ \rho\rangle \Downarrow \langle v_3,\ \xi'',\ \phi,\ \rho''\rangle}$$

The same truths that held for IfTrue above hold true for IfFalse. Thus we can see, through induction, that an IfFalse can be evaluated within the defined bounds (one copy of the environment via popping and pushing from the stack).

$$\frac{\langle e_1,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_1,\, \xi',\, \phi,\, \rho'\rangle \qquad v_1 \neq 0 \qquad \langle e_2,\, \xi',\, \phi,\, \rho'\rangle \Downarrow \langle v_2,\, \xi'',\, \phi,\, \rho''\rangle \;\; \langle \mathsf{WHILE}(e_1,\, e_2),\, \xi'',\, \phi,\, \rho''\rangle \Downarrow \langle v_3,\, \xi''',\, \phi,\, \rho'''\rangle}{\langle \mathsf{WHILE}(e_1,\, e_2),\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_3,\, \xi''',\, \phi,\, \rho'''\rangle}$$

For WhileIterate, it has already been shown that we can safely evalute $\langle e_1,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_1,\, \xi',\, \phi,\, \rho'\rangle$ and $\langle e_2,\, \xi',\, \phi,\, \rho'\rangle \Downarrow \langle v_2,\, \xi'',\, \phi,\, \rho''\rangle$ by popping ρ, throwing it away, replacing it with ρ', pushing it to the stack and repeating again with ρ' to ρ''. Since these have been shown, we can also show that $\langle \mathsf{WHILE}(e_1,\, e_2),\, \xi'',\, \phi,\, \rho''\rangle \Downarrow \langle v_3,\, \xi''',\, \phi,\, \rho'''\rangle$ can be done safely through popping and throwing away ρ'' and pushing ρ''' to the stack. Thus the operation can be done correctly / safely without copying ρ.

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

We have already shown that $\langle e_1,\, \xi,\, \phi,\, \rho\rangle \Downarrow \langle v_1,\, \xi',\, \phi,\, \rho'\rangle$ can be safely evaluated via pops and pushes. Thus it can be seen that this assessment is safe. In fact, there are no new operations to consider this so other than evaluating e_1, we don't even need to touch the stack.

$$\frac{\begin{array}{c}\langle e_1,\, \xi_0,\, \phi,\, \rho_0\rangle \Downarrow \langle v_1,\, \xi_1,\, \phi,\, \rho_1\rangle \\ \langle e_2,\, \xi_1,\, \phi,\, \rho_1\rangle \Downarrow \langle v_2,\, \xi_2,\, \phi,\, \rho_2\rangle \\ \ldots \\ \langle e_n,\, \xi_{n-1},\, \phi,\, \rho_{n-1}\rangle \Downarrow \langle v_n,\, \xi_n,\, \phi,\, \rho_n\rangle\end{array}}{\langle \mathsf{BEGIN}(e_1,\, e_2,\, \ldots,\, e_n),\, \xi_0,\, \phi,\, \rho_0\rangle \Downarrow \langle v_n,\, \xi_n,\, \phi,\, \rho_n\rangle}$$

When evaluating e_1 with ρ_0, we can throw out ρ_0 and replace it with ρ_1 on the stack because we will not be using ρ_0 any more. This loop repeats for every expression $e_1,\, e_2,\, \ldots,\, e_n$ so it can be seen that these expresssions can be evaluated in order by popping the environment, modifying it for side-effects of evaluating the expression, and pushing it back onto the stack. Thus, the entire BEGIN statement can be evaluated using the environment stack, ultimately concluding with environment ρ_n on the top of the stack with no other copies existing.

$$\frac{\begin{array}{c}\phi(f) \;=\; \mathsf{USER}(\langle x_1,\, \ldots,\, x_n\rangle,\, e) \\ x_1,\, \ldots,\, x_n \textit{ all distinct} \\ \langle e_1,\, \xi_0,\, \phi,\, \rho_0\rangle \Downarrow \langle v_1,\, \xi_1,\, \phi,\, \rho_1\rangle \\ \ldots \\ \langle e_n,\, \xi_{n-1},\, \phi,\, \rho_{n-1}\rangle \Downarrow \langle v_n,\, \xi_n,\, \phi,\, \rho_n\rangle \\ \langle e,\, \xi_n,\, \phi,\, \{x_1 \to v_1,\, \ldots,\, x_n \to v_n\}\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho'\rangle\end{array}}{\langle \mathsf{APPLY}(f,\, e_1,\, \ldots,\, e_n),\, \xi_0,\, \phi,\, \rho_0\rangle \Downarrow \langle v,\, \xi',\, \phi,\, \rho_n\rangle}$$

Finally we can consider the application of any user defined function f, which is given. Similar to how we evaluated the series of expressions $e_1,\, e_2,\, \ldots,\, e_n$

above in the BEGIN statement, we know that the evaluation of this series of expressions is valid and can be replicated using the pop and push approach of the environment stack. We begin with ρ_0 and we end with ρ_n.

After we have correctly evaluated all of the expressions e_1, e_2, ..., e_n and gotten values v_1, v_2, ..., v_n, we can then make a new formal variable environment with parameters x_1, x_2, ..., x_n being mapped to the values obtained. This results in an environment $\{x_1 \rightarrow v_1, x_2 \rightarrow v_2, ..., x_n \rightarrow v_n\}$. As we go to apply the function *f*, we can push this environment onto the stack and use this for the duration of our evaluation of our function *f*. This makes sense because we have defined the formal variables that we will be using through the body of function *f*.

On the surface, this might seem like it breaks the rules of the popping and pushing, no environment copy approach. However, since the new environment that we are pushing onto the stack is a completely new and distinct environment, it is not a duplicate copy or mutation of any other environment on the stack. Therefore it is valid within the bounds of (a) and (b).

And thus with that final assessment of the function itself with this new environment, it can be seen that the application of user defined function *f* is possible to recreate / conduct with by simply popping and pushing formal variable environments without making copies.

Thus, through these 12 operational derivations it can be seen that each of these `impcore` operations can be replaced with / replicated through the use of operations "pop ρ^{n-1}, push ρ^n", proving that properties (a) and (b) of the lemma above hold true.