15a

$x \notin \text{dom } \rho \quad x \notin \text{dom } \xi$

20. For the purposes of simplicity — I will leave out the global environment ξ in my informal inductive proof:

Base Cases:

1. Consider the rule for LITERAL:

$$\frac{}{\langle \text{LITERAL}(v), \phi, \rho \rangle \Downarrow \langle v, \phi, \rho \rangle}$$

Examining this rule reveals that it can be implemented as follows: pop ρ, then push ρ back onto the environment stack — after which the only copy is at the top of the stack.

2. Consider the rule for FORMAL VAR:

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

We see that this rule can be implemented as pop ρ, test $x \in \text{dom } \rho$, and compute $\rho(x)$ — then push ρ back onto the environment stack — after which the only copy is at the top of the stack.

Thus, our induction hypothesis is that every proof rule ending in the judgment form $\langle e, \phi, \rho \rangle \Downarrow \langle v, \phi, \rho' \rangle$ can be changed to pop ρ off the stack and push ρ' onto the stack.

→

Inductive Steps:

Consider the rule for formal assign:

$$\frac{x \in \text{dom } P \quad \langle e, \Phi, P\rangle \Downarrow \langle v, \Phi, P'\rangle}{\langle SET(X, e), \Phi, P\rangle \Downarrow \langle v, \Phi, P'\{x_1 \mapsto v_3\}\rangle}$$

By the induction hypothesis, we can evaluate $\langle e, \Phi, P\rangle \Downarrow \langle v, \Phi, P\rangle$ using a stack and the evaluation will pop P at push in P' without making a copy of P. Because P does not appear anywhere else in the rule, it is safe to pop it and discard it. P' is only used as a part of the result of the rule. Thus, we can safely conclude that when x is in the domain P, we can safely evaluate $SET\langle X, e\rangle$ and the evaluation effectively pops P (which is never used again, and pushes P'.

Consider the rule for IFTRUE:

$$\frac{\langle e_1, \Phi, P\rangle \Downarrow \langle v_1, \Phi, P'\rangle \quad v_1 \neq 0 \quad \langle e_2, \Phi, P'\rangle \Downarrow \langle v_2, \Phi, P''\rangle}{\langle IF\langle e_1, e_2, e_3\rangle, \Phi, P\rangle \Downarrow \langle v_3, \Phi, P''\rangle}$$

By the induction hypothesis, we can evaluate $\langle e_1, \Phi, P\rangle \Downarrow \langle v_1, \Phi, P'\rangle$ using a stack and the evaluation will pop P' and push P' without making a copy of P. Because P does not appear anywhere else in the rule, it is safe to pop it and discard it. We recall the induction hypothesis again to show that the evaluation of e_2 can pop P' and push P'' and P' is not copied. Also, P' is not used again in the rule after the evaluation of P'', so P' may be discarded. Lastly, we see that P'' is used only as a part of the result of the rule. Thus, we can conclude that when e_1 evaluates to a nonzero value, we can safely evaluate $IF\langle e_1, e_2, e_3\rangle$ on a stack, and the evaluation effectively pops P and pushes P'', never using P again.

To prove for IFFALSE,

$$\frac{\langle e_1, \phi, P\rangle \Downarrow \langle V_1, \phi, P'\rangle \quad V_1 = 0 \quad \langle e_3, \phi, P'\rangle \Downarrow \langle V_3, \phi, P''\rangle}{\langle IF(e_1, e_2, e_3), \phi, P\rangle \Downarrow \langle V_3, \phi, P''\rangle}$$

We remember that by IH, we can evaluate $\langle e_1, \phi, P\rangle \Downarrow \langle V_1, \phi, P'\rangle$ using a stack and the evaluation will pop P and push P' without copying P. Once again, because P does not show up anywhere else in the rule, it is safe to pop it and throw it away. The IH can be used again to show that the evaluation of e_3 can pop P' and push P'' and P' is not copied — P' is also not used again in the rule after the evaluation of e_3.

Also, P'' is only used as part of the result of the rule. Thus, when e_1 evaluates to 0 we can safely evaluate $IF(e_1, e_2, e_3)$ on a stack — the evaluation pops P and pushes P''.

Now, consider the rule WHILEITERATE

$$\frac{\langle e_1, \phi, P\rangle \Downarrow \langle V_1, \phi, P'\rangle \quad V_1 \neq 0 \quad \langle e_2, \phi, P'\rangle \Downarrow \langle V_2, \phi, P''\rangle \quad \langle WHILE(e_1, e_2), \phi, P''\rangle \Downarrow \langle V_3, \phi, P'''\rangle}{\langle WHILE(e_1, e_2), \phi, P\rangle \Downarrow \langle V_3, \phi, P'''\rangle}$$

As before, we can use the IH to see that it is possible to evaluate $\langle e_1, \phi, P\rangle \Downarrow \langle V_1, \phi, P'\rangle$ and $\langle e_2, \phi, P'\rangle \Downarrow \langle V_2, \phi, P''\rangle$ using a stack and discarding P and P', respectively. We can also apply the IH twice that $\langle WHILE(e_1, e_2), \phi, P''\rangle \Downarrow \langle V_3, \phi, P'''\rangle$ can be evaluated on a stack because P'' is never used again, and can thus be discarded when P''' is pushed onto the stack. Finally, P''' is only used as part of the result of the rule. This leads us to the conclusion that $\langle WHILE(e_1, e_2), \phi, P\rangle$ can be evaluated on a stack, and that the evaluation effectively pops P, which is never used again then pushes P'', then continues this process.

For the Rule WHILEEND

$$\frac{\langle e_1, \emptyset, P\rangle \Downarrow \langle v_1, \emptyset, P'\rangle \quad v_1 = 0}{\langle \text{WHILE}(e_1, e_2), \emptyset, P\rangle \Downarrow \langle 0, \emptyset, P'\rangle}$$

By the IH, we know that we can evaluate $\langle e_1, \emptyset, P\rangle \Downarrow \langle v_1, \emptyset, P'\rangle$ using a stack and the evaluation will push P and pop P' without copying P. Because P does not appear anywhere else in the rule, it is never used again, and can be popped and thrown away. We also note that P' is only present in the result of the rule. Thus, we can conclude that if e_1 evaluates to a non zero value, then it is safe to evaluate WHILE(e_1, e_2) on a stack and the evaluation effectively pops P and pushes P', never using P again.

Consider the rules for BEGIN:

The rule for EMPTY BEGIN:

$$\frac{}{\langle \text{BEGIN}(), \emptyset, P\rangle \Downarrow \langle 0, \emptyset, P\rangle}$$

does not require the IH — it is clear that P can be popped, then pushed and will remain on the top of the stack.

The Rule for BEGIN:

$$\frac{\langle e_1, \emptyset, P_0\rangle \Downarrow \langle v_1, \emptyset, P_1\rangle \quad \langle e_2, \emptyset, P_1\rangle \Downarrow \langle v_2, \emptyset, P_2\rangle \quad \langle e_n, \emptyset, P_{n-1}\rangle \Downarrow \langle v_n, \emptyset, P_n\rangle}{\langle \text{BEGIN}(e_1, e_2, \ldots, e_n), \emptyset, P_0\rangle \Downarrow \langle v_n, \emptyset, P_n\rangle}$$

We see, by the IH, that each expression of the form $\langle e_x, \emptyset, P_x\rangle \Downarrow \langle v_x, \emptyset, P_x'\rangle$ can be evaluated using a stack — discarding P_x and pushing P_x'. Once v_n is evaluated — P_n only appears in the result of the rule. Thus, we can conclude that when we evaluate BEGIN$(e_1 \ldots e_n)$, for each expression e_n, it is safe to pop P_n and discard it, then to push P_n'.

Consider the rule APPLYUSER:

$$\frac{\phi(F) = \text{USER}(\langle x_1, \ldots, x_n \rangle, e) \quad x_1, \ldots, x_n \text{ all distinct} \quad \langle e_1, \emptyset, P_0 \rangle \Downarrow \langle v_1, \emptyset, P_1 \rangle \quad \vdots \quad \langle e_n, \emptyset, P_{n-1} \rangle \Downarrow \langle v_n, \emptyset, P_n \rangle \quad \langle e, \emptyset, \{x_1 \mapsto v_1, \ldots x_n \mapsto v_n\} \rangle \Downarrow \langle v, \emptyset, P' \rangle}{\langle \text{APPLY}(F, e_1 \ldots e_n), \emptyset, P_0 \rangle \Downarrow \langle v, \emptyset, P_n \rangle}$$

From the IH, we can evaluate any expression of the form $\langle e_n, \emptyset, P_{n-1} \rangle \Downarrow \langle v_n, \emptyset, P_n \rangle$ using a stack - popping P_{n-1}, discarding it, and then pushing P_n. The last line of the premises $\langle e, \emptyset, \{x_1 \mapsto v_1 \ldots x_n \mapsto v_n\} \rangle \Downarrow \langle v, \emptyset, P' \rangle$, is also of this form — a new P is pushed onto the stack and the older P is popped and discarded.

Thus, we can safely conclude that we can safely evaluate APPLY(F, e1 ... en) on a stack and the evaluation effectively pops each P_n and pushes P_n' — never using P_n again.

Lastly, consider the rule APPLYADD (which will suffice for all primitives)

$$\frac{\phi(f) = \text{PRIMITIVE}(+) \quad \langle e_1, \emptyset, P_0 \rangle \Downarrow \langle v_1, \emptyset, P_1 \rangle \quad \langle e_2, \emptyset, P_1 \rangle \Downarrow \langle v_2, \emptyset, P_2 \rangle}{\langle \text{APPLY}(F, e_1, e_2), \emptyset, P_0 \rangle \Downarrow \langle v_1 + v_2, \emptyset, P_2 \rangle}$$

By the IH, both of the expressions in the premise can be evaluated using a stack - $\langle e_1, \emptyset, P_0 \rangle \Downarrow \langle v_1, \emptyset, P_1 \rangle$ pops P_0 and pushes P_1, discarding P_0 in the process, while $\langle e_2, \emptyset, P_1 \rangle \Downarrow \langle v_2, \emptyset, P_2 \rangle$ pops P_1 and pushes P_2, discarding P_2 in the process. Finally, we see that P_2 is only used in the result of the rule. We can thus conclude that in the case of primitive APPLYs, we can safely evaluate APPLY(F, e1, e2) using a stack — and the evaluation effectively pops P and pushes P_2 — discarding P_1.

From this lemma, we can safely say that the "push P pop P" operation can be replaced by the operation "mutate P in place to become P'."

13a. AWK

$$\frac{x \notin \mathrm{dom}\, P \qquad x \notin \mathrm{dom}\, \xi}{\langle \mathrm{VAR}(x), \xi, \phi, P \rangle \Downarrow \langle 0, \xi\{x \to 0\}, \phi, P \rangle} \text{ UNBOUNDVAR}$$

$$\frac{x \notin \mathrm{dom}\, \rho \qquad \langle e_1, \xi, \phi, P \rangle \Downarrow \langle v_1, \xi', \phi, P' \rangle}{\langle \mathrm{SET}(x, e), \xi, \phi, P \rangle \Downarrow \langle v, \xi'\{x \mapsto v\}, \phi, P' \rangle} \text{ GLOBALASSIGN}$$

b. ICON

$$\frac{x \notin \mathrm{dom}\, P \qquad x \notin \mathrm{dom}\, \xi}{\langle \mathrm{VAR}(x), \xi, \phi, P \rangle \Downarrow \langle 0, \xi, \phi, P'\{x \mapsto 0\} \rangle} \text{ UNBOUNDVAR}$$

$$\frac{x \notin \mathrm{dom}\, \rho \qquad x \notin \mathrm{dom}\, \xi}{\langle \mathrm{SET}(x, e), \xi, \phi, P \rangle \Downarrow \langle v, \xi', \phi, P'\{x \mapsto v\} \rangle}$$

c. I prefer the change to ICON-like semantics because it seems more logical to want to bind any newly introduced variables as local variables within a given procedure. In the AWK-like case, any created variable can be changed by any procedure or function, which seems problematic and unsafe. The ICON-like rules make it very easy to use temporary variables within functions — which is very useful.

10.

1 $$\dfrac{x \in \mathrm{dom}\, \rho \quad \rho(x) = 99}{\langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle 99, \xi, \phi, \rho\rangle}\ \text{FORMALVAR} \qquad \dfrac{x \in \mathrm{dom}\, \rho \quad \langle LIT(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}{\langle SET(VAR(x), LITERAL(3)), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi', \phi, \rho'\{x \mapsto 3\}\rangle}\ \text{FORMALASSIGN}\ \text{LITERAL}$$

2 $$\dfrac{x \in \mathrm{dom}\, \rho \quad \rho'(x) = 3}{\langle VAR(x), \xi', \phi, \rho'\rangle \Downarrow \langle 3, \xi', \phi, \rho'\rangle}\ \text{FORMALVAR}$$

$$\dfrac{1 \qquad 2}{\langle BEGIN(SET(VAR(x), LIT(3)), VAR(x)), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi', \phi, \rho'\rangle}$$

11. $\dfrac{x \in \mathrm{dom}\, \rho}{\langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle \rho(x), \xi, \phi, \rho\rangle}$

1a. $$\dfrac{\langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle v_0, \xi, \phi, \rho\rangle \quad v \neq 0 \quad \langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi'', \phi, \rho''\rangle}{\langle IF(VAR(x), VAR(x), LIT(0)), \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi'', \phi, \rho''\rangle}\ \text{IFTRUE}$$

2. $\langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle v_2, \xi'', \phi, \rho''\rangle$

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

1b. $$\dfrac{\langle VAR(x), \xi, \phi, \rho\rangle \Downarrow \langle v_0, \xi, \phi, \rho\rangle \quad v = 0 \quad \langle LIT(0), \xi', \phi, \rho'\rangle \Downarrow \langle v_1, \xi'', \phi, \rho''\rangle}{IE(VAR(x), VAR(x), LIT(0)), \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi'', \phi, \rho''\rangle}\ \text{IFFALSE}$$

IF $v_1 \neq 0$, then $1a = 2$
$v_1 = v_2$

IF $v_1 = 0$, then $1b = 2$
$v_1 = v_2$ ≡