IIT CS595: Topics & Applications in Programming Languages

Homework 2: Memory and Cost Semantics

Dustin Van Tate Testa

Out: Tuesday, Oct. 19
Due: Wednesday, Nov. 3 11:59pm CDT

Logistics

The submission and collaboration instructions (as well as the preference for submissions typeset in LaTeX) are the same as for HW0 and HW1.

In particular: Submit your answers as a single .pdf or .doc file on Blackboard under the correct assignment.

1 Allocation and Memory Use

As we discussed in class, cost semantics can be useful for more than just counting the number of steps a program will take to execute. In this assignment, we'll use them to track memory usage. Consider this new variant of STLC:

$$\begin{array}{rcl} \tau & ::= & \mathsf{unit} \mid \tau \to \tau \mid \tau \times \tau \\ e & ::= & x \mid () \mid \lambda x : \tau . e \mid e\ e \mid (e, e) \mid \langle e, e \rangle \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \end{array}$$

We now have two forms for pairs: (e_1, e_2) and $\langle e_1, e_2 \rangle$. The latter form, $\langle e_1, e_2 \rangle$, is used for a pair that has fully evaluated and been allocated in memory (this models how structures like pairs are executed in real languages; when they are constructed, memory for the structure is allocated, e.g., using `malloc`). This is made clearer in the dynamic semantics. Like when we covered imperative languages, the state has a store σ. These stores are actually just lists of all of the pairs that have been produced/allocated. We'll use $\sigma_1 \uplus \sigma_2$ to combine stores and $|\sigma|$ to refer to the size of a store (i.e., the length of the list).

The key different rules are (V-3) and (S-6). Note from the (V-#) rules that (e_1, e_2) is *not* a value, even if e_1 and e_2 are values. Instead, (S-6) says that when v_1 and v_2 are values, (v_1, v_2) takes an additional step to allocate the memory for the pair by adding it to the store and steps to $\langle v_1, v_2 \rangle$, which is now a value[1]. For the same reason, $\mathsf{fst}\ e$ and $\mathsf{snd}\ e$ evaluate e until it looks like $\langle v_1, v_2 \rangle$: the pair must be allocated before we can take its components and if we have a pair $\langle v_1, v_2 \rangle$, we know it's been allocated.

Make sure you understand the language and the distinctions made above before continuing. If you have questions, feel free to ask for clarifications on Discord or at office hours.

[1] If we really wanted to faithfully model how this works in real languages, we would return a representation of the location in memory where the pair is stored (basically a pointer), and look it up in memory if we need to get the components of the pair later. The way we're doing it here is a little contrived but avoids a bunch of cumbersome notation I didn't want to burden you with.

$$\frac{}{() \text{ val}} \text{ (V-1)} \qquad \frac{}{\lambda x : \tau.e \text{ val}} \text{ (V-2)} \qquad \frac{v_1 \text{ val} \qquad v_2 \text{ val}}{\langle v_1, v_2 \rangle \text{ val}} \text{ (V-3)} \qquad \frac{\sigma; e_1 \mapsto \sigma'; e_1'}{\sigma; e_1\ e_2 \mapsto \sigma'; e_1'\ e_2} \text{ (S-1)}$$

$$\frac{\sigma; e_2 \mapsto \sigma'; e_2'}{\sigma; (\lambda x : \tau.e)\ e_2 \mapsto \sigma'; (\lambda x : \tau.e)\ e_2'} \text{ (S-2)} \qquad \frac{v \text{ val}}{\sigma; (\lambda x : \tau.e)\ v \mapsto \sigma; [v/x]e} \text{ (S-3)} \qquad \frac{\sigma; e_1 \mapsto \sigma'; e_1'}{\sigma; (e_1, e_2) \mapsto \sigma'; (e_1', e_2)} \text{ (S-4)}$$

$$\frac{v_1 \text{ val} \qquad \sigma; e_2 \mapsto \sigma'; e_2'}{\sigma; (v_1, e_2) \mapsto \sigma'; (v_1, e_2')} \text{ (S-5)} \qquad \frac{v_1 \text{ val} \qquad v_2 \text{ val}}{\sigma; (v_1, v_2) \mapsto \sigma \uplus [\langle v_1, v_2 \rangle]; \langle v_1, v_2 \rangle} \text{ (S-6)} \qquad \frac{\sigma; e \mapsto \sigma'; e'}{\sigma; \mathsf{fst}\ e \mapsto \sigma'; \mathsf{fst}\ e'} \text{ (S-7)}$$

$$\frac{}{\sigma; \mathsf{fst}\ \langle v_1, v_2 \rangle \mapsto \sigma; v_1} \text{ (S-8)} \qquad \frac{\sigma; e \mapsto \sigma'; e'}{\sigma; \mathsf{snd}\ e \mapsto \sigma'; \mathsf{snd}\ e'} \text{ (S-9)} \qquad \frac{}{\sigma; \mathsf{snd}\ \langle v_1, v_2 \rangle \mapsto \sigma; v_2} \text{ (S-10)}$$

The cost semantics to track memory usage then should correspond not to the number of steps a program will take to execute, but the number of pairs it allocates, that is, the length of σ when it is done. I've filed out the big-step semantics for the language for you. Note that the big-step semantics *don't* carry around a σ; that's too low-level a detail for the abstract big-step semantics.

$$\frac{v \text{ val}}{v \Downarrow^0 v} \text{ (E-1)} \qquad \frac{e_1 \Downarrow^{n_1} \lambda x : \tau.e \qquad e_2 \Downarrow^{n_2} v \qquad [v/x]e \Downarrow^{n_3} v'}{e_1\ e_2 \Downarrow^{n_1+n_2+n_3} v'} \text{ (E-2)} \qquad \frac{e_1 \Downarrow^{n_1} v_1 \qquad e_2 \Downarrow^{n_2} v_2}{(e_1, e_2) \Downarrow^{n_1+n_2+1} \langle v_1, v_2 \rangle} \text{ (E-3)}$$

$$\frac{e \Downarrow^n \langle v_1, v_2 \rangle}{\mathsf{fst}\ e \Downarrow^n v_1} \text{ (E-4)} \qquad \frac{e \Downarrow^n \langle v_1, v_2 \rangle}{\mathsf{snd}\ e \Downarrow^n v_2} \text{ (E-5)}$$

Task 1. *Fill in the blanks in the big-step semantics to turn it into a cost semantics for tracking memory usage. If you're having trouble deciding exactly what operations to "charge" for, look ahead at the theorems we want to prove about the cost semantics. (Conversely, if you're having trouble proving the results later, you might want to come back and re-evaluate whether you need to change the cost semantics.)*

We'll now prove that the cost semantics corresponds to the memory used by the small-step semantics. That is:

$$e \Downarrow^n v \Leftrightarrow (\emptyset; e \mapsto^* \sigma; v \text{ and } n = |\sigma|)$$

where $|\sigma|$ is the size of the store and $\emptyset$ is the empty store. As you'd expect, $|\emptyset| = 0$.

We prove this in several pieces. Recall that the forward direction involves proving several annoying lemmas. Here's the first annoying lemma. It says that if $e \mapsto^* e'$ and e is actually a part of a bigger expression, that bigger expression also steps appropriately. It has 6 parts for each kind of "bigger expression."

Lemma 1. *1. If $\sigma; e_1 \mapsto^* \sigma'; e_1'$ then $\sigma; e_1\ e_2 \mapsto^* \sigma'; e_1'\ e_2$.*

2. If $\sigma; e_2 \mapsto^* \sigma'; e_2'$ then $\sigma; e_1\ e_2 \mapsto^* \sigma'; e_1\ e_2'$.

3. If $\sigma; e_1 \mapsto^* \sigma'; e_1'$ then $\sigma; (e_1, e_2) \mapsto^* \sigma'; (e_1', e_2)$.

4. If $\sigma; e_2 \mapsto^* \sigma'; e_2'$ then $\sigma; (e_1, e_2) \mapsto^* \sigma'; (e_1, e_2')$.

5. If $\sigma; e \mapsto^* \sigma'; e'$ then $\sigma; \mathsf{fst}\ e \mapsto^* \sigma'; \mathsf{fst}\ e$.

6. If $\sigma; e \mapsto^* \sigma'; e'$ then $\sigma; \mathsf{snd}\ e \mapsto^* \sigma'; \mathsf{snd}\ e$.

To do the proof, recall that the $\mapsto^*$ judgment is also defined inductively:

$$\frac{}{\sigma; e \mapsto^* \sigma; e} \text{ (M-1)} \qquad \frac{\sigma; e \mapsto \sigma''; e'' \qquad \sigma''; e'' \mapsto^* \sigma'; e'}{\sigma; e \mapsto^* \sigma'; e'} \text{ (M-2)}$$

The parts are all reasonably similar. I'll prove one, you'll prove another.

Proof. 1. By induction on the derivation of $\sigma; e_1 \mapsto^* \sigma'; e_1'$.

- (M-1). Then $e_1 = e_1'$ and $\sigma = \sigma'$. By (M-1), $\sigma; e_1\ e_2 \mapsto^* \sigma; e_1\ e_2$.
- (M-2) Then $\sigma; e_1 \mapsto \sigma''; e_1''$ and $\sigma''; e_1'' \mapsto^* \sigma'; e_1'$. By (S-1), $\sigma; e_1\ e_2 \mapsto \sigma''; e_1''\ e_2$. By induction, $\sigma''; e_1''\ e_2 \mapsto^* \sigma'; e_1'\ e_2$ so by (M-2), $\sigma; e_1\ e_2 \mapsto^* \sigma'; e_1'\ e_2$.

□

Task 2. *Prove part 5 of Lemma 1.*

Proof. By induction on the derivation of $\sigma; e \mapsto^* \sigma'; e'$.

- (M-1). Then $e = e'$ and $\sigma = \sigma'$. By (M-1), $\sigma; \mathsf{fst}\ e \mapsto^* \sigma; \mathsf{fst}\ e$.
- (M-2) Then $\sigma; e \mapsto \sigma''; e''$ and $\sigma''; e'' \mapsto^* \sigma; e'$.
  By (S-7), $\sigma; \mathsf{fst}\ e \mapsto \sigma''; \mathsf{fst}\ e''$. By induction, $\sigma''; \mathsf{fst}\ e'' \mapsto^* \sigma'; \mathsf{fst}\ e'$ so by (M-2), $\sigma; \mathsf{fst}\ e \mapsto^* \sigma'; \mathsf{fst}\ e'$

□

We also need to know that if we have two sequences of steps, we can "glue" them together.

Lemma 2. *If $\sigma; e \mapsto^* \sigma'; e'$ and $\sigma'; e' \mapsto^* \sigma''; e''$ then $\sigma; e \mapsto^* \sigma''; e''$.*

Task 3. *Prove Lemma 2. You have two choices of what to induct on. Take an educated guess based on the rules you'll have to apply to conclude that $\sigma; e \mapsto^* \sigma''; e''$. If the proof isn't working out, go back and try inducting on the other assumption.*

Proof. Your proof here.

- by induction on derivation of the step judgement, either
  - $e = e' = e''$ and $\sigma = \sigma' = \sigma''$:
    in which case for both step judgements (M-1) would be applied
  - $e' = e''$ and $\sigma' = \sigma''$:
    in which case (M-2) is applied to the first step and (M-1) applied after
  - $e = e'$ and $\sigma = \sigma'$:
    in which case (M-1) is applied to make the first step and then (M-2) after
  - all e and all σ values are unique:
    in which case (M-2) is applied twice

□

Now we can prove the forward direction of the theorem. Note that we don't actually prove that $\emptyset; e \mapsto^* \sigma; v$. We actually need to be able to start with a non-empty store in order to have a strong enough induction hypothesis. Think about why. We also prove that when e steps to v starting with store σ, the resulting store is strictly an extension of σ: that is, it's $\sigma \uplus \sigma'$, where σ' contains what we've added to the store while stepping. This means you don't have to worry about tracking things getting de-allocated or changed (we don't have garbage collection.)

Theorem 1. *If $e \Downarrow^n v$ then for all σ, it is the case that $\sigma; e \mapsto^* \sigma \uplus \sigma'; v$ and $|\sigma'| = n$.*

I'll prove one case as an example.

Proof. By induction on the derivation of $e \Downarrow^n v$.

- (E-2). Then $e = e_1\ e_2$ and $e_1 \Downarrow^{n_1} \lambda x : \tau.e$ and $e_2 \Downarrow^{n_2} v$ and $[v/x]e \Downarrow^{n_3} v'$ and $n = n_1 + n_2 + n_3$. By induction,

  $$\sigma; e_1 \mapsto^* \sigma \uplus \sigma_1; \lambda x : \tau.e \text{ and } \sigma \uplus \sigma_1; e_2 \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v \text{ and } \sigma \uplus \sigma_1 \uplus \sigma_2; [v/x]e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2 \uplus \sigma_3; v'$$

  and

  $$|\sigma_1| = n_1 \text{ and } |\sigma_2| = n_2 \text{ and } |\sigma_3| = n_3$$

  By Lemma 1, $\sigma; e \mapsto^* \sigma \uplus \sigma_1; (\lambda x : \tau.e)\ e_2$ and $\sigma \uplus \sigma_1; (\lambda x : \tau.e)\ e_2 \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; (\lambda x : \tau.e)\ v$.
  By Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; (\lambda x : \tau.e)\ v$.
  By (S-3) and Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2 \uplus \sigma_3; v'$. We have $|\sigma_1 \uplus \sigma_2 \uplus \sigma_3| = |\sigma_1| + |\sigma_2| + |\sigma_3| = n$.

- (E-1). Then e val and $e \Downarrow^0 v$ and $e = v$ and $n = 0$ which by lemma 1 translates to $\emptyset; e \mapsto^* \sigma; e$. Because values cannot step according to the defined small-step semantics rules (exhaustion of rules (V- 1..3)), the $\mapsto^*$ judgement must follow rule (M-1), thus $\emptyset = \sigma$ and $|\sigma| = n = 0$.

- (E-3). Then $e = (e_1, e_2)$ and $e_1 \Downarrow^{n_1} v_1$ and $e_2 \Downarrow^{n_2} v_2$ and $(v_1, v_2) \Downarrow^1 \langle v_1, v_2 \rangle$ and $n = n_1 + n_2 + 1$. By induction,

  $$\sigma; e_1 \mapsto^* \sigma \uplus \sigma_1; v_1 \text{ and } \sigma; e_2 \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_2 \text{ and } \sigma; e_2 \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2 \uplus \sigma_3; v_2$$

  and

  $$|\sigma_1| = n_1 \text{ and } |\sigma_2| = n_2 \text{ and } |\sigma_3| = 1$$

  By Lemma 1, $\sigma; e \mapsto^* \sigma \uplus \sigma_1; (v_1, e_1)$and $\sigma \uplus \sigma_1; (v_1, e_1) \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; (v_1, v_2)$.
  By Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; (v_1, v_2)$.
  By (S-6) and Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2 \uplus [\langle v_1, v_2 \rangle]; \langle v_1, v_2 \rangle$. We have $|\sigma_1 \uplus \sigma_2 \uplus [\langle v_1, v_2 \rangle]| = |\sigma_1| + |\sigma_2| + |[\langle v_1, v_2 \rangle]| = n$.

- (E-4). Then $e = \mathsf{fst}\ e_1$ and $e_1 \Downarrow^{n_1} \langle v_1, v_2 \rangle$ and $e \Downarrow^0 v_1$ and $n = n_1 + 0$. By induction,

  $$\sigma; e_1 \mapsto^* \sigma \uplus \sigma_1; \langle v_1, v_2 \rangle \text{ and } \sigma \uplus \sigma_1; \mathsf{fst}\ \langle v_1, v_2 \rangle \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_1$$

  and

  $$|\sigma_1| = |\sigma_2| = n_1 \text{ and } n = n_1$$

  By Lemma 1, $\sigma; e \mapsto^* \sigma \uplus \sigma_1; \mathsf{fst}\ \langle v_1, v_2 \rangle$ and $\sigma \uplus \sigma_1; \mathsf{fst}\ \langle v_1, v_2 \rangle \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_1$.
  By Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_1$.
  By (S-8) and Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_1$. We have $|\sigma_1 \uplus \sigma_2| = |\sigma_1| = n$.

- (E-5). Then $e = \mathsf{snd}\ e_1$ and $e_1 \Downarrow^{n_1} \langle v_1, v_2 \rangle$ and $e \Downarrow^0 v_1$ and $n = n_1 + 0$. By induction,

  $$\sigma; e_1 \mapsto^* \sigma \uplus \sigma_1; \langle v_1, v_2 \rangle \text{ and } \sigma \uplus \sigma_1; \mathsf{snd}\ \langle v_1, v_2 \rangle \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_2$$

  and

  $$|\sigma_1| = |\sigma_2| = n_1 \text{ and } n = n_1$$

  By Lemma 1, $\sigma; e \mapsto^* \sigma \uplus \sigma_1; \mathsf{snd}\ \langle v_1, v_2 \rangle$ and $\sigma \uplus \sigma_1; \mathsf{snd}\ \langle v_1, v_2 \rangle \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_2$.
  By Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_2$.
  By (S-9) and Lemma 2, $\sigma; e \mapsto^* \sigma \uplus \sigma_1 \uplus \sigma_2; v_2$. We have $|\sigma_1 \uplus \sigma_2| = |\sigma_1| = n$.

□

Task 4. *Prove the remaining cases of Theorem 1. There are many similarities between the cases, but some small, important differences. Copying and pasting between cases is allowed and encouraged, but make sure you know what to change!*

Note: *I actually proved the* second *case; you still have to go back and prove (E-1) as well as (E-3) through (E-5).*

The reverse direction is pretty straightforward given a key lemma introduced in class. The lemma says that if $e' \Downarrow^n v$ and e steps to e', then $e \Downarrow^{n+k} v$, where k counts any pairs allocated by that step.

Lemma 3. *If $e' \Downarrow^n v$ and $\sigma; e \mapsto \sigma \uplus \sigma'; e'$ then $e \Downarrow^{n+k} v$ and $|\sigma'| = k$.*

Proof. By induction on the derivation of $\sigma; e \mapsto \sigma \uplus \sigma'; e'$.

- (S-1) Then $e = e_1\ e_2$ and $e' = e_1'\ e_2$. By inversion on (E-2), $e_1' \Downarrow^{n_1} \lambda x : \tau.e$ and $e_2 \Downarrow^{n_2} v$ and $[v/x]e \Downarrow^{n_3} v'$ and $n = n_1 + n_2 + n_3$. By induction, $e_1 \Downarrow^{n_1+k} \lambda x : \tau.e$ and $|\sigma'| = k$. By (E-2), $e \Downarrow^{n+k} v'$.
- (S-2) Then $e = \lambda x : \tau.e_1\ e_2$ and $e' = \lambda x : \tau.e_1\ e_2'$. By inversion on (E-2), $\lambda x : \tau.e_1 \Downarrow^0 \lambda x : \tau.e_1$ and $e_2 \Downarrow^{n_2} v'$ and $[v'/x]e_1 \Downarrow^{n_3} v$ and $n = 0 + n_2 + n_3$. By induction, $e_2 \Downarrow^{n'+k} v'$ and $|\sigma'| = k$. By (E-2), $e \Downarrow^{n+k} v$.
- (S-3) Then $e = \lambda x : \tau.e_1\ v'$ and $e' = \lambda x : \tau.e_1\ e_2'$ and v' val. By inversion on (E-2), $\lambda x : \tau.e_1 \Downarrow^0 \lambda x : \tau.e_1$ and $v' \Downarrow^0 v'$ and $[v'/x]e_1 \Downarrow^{n_1} v$ and $n = 0 + 0 + n_1$. By induction, $[v'/x]e_1 \Downarrow^{n'+k} v$ $|\sigma'| = k$. By (E-2), $e \Downarrow^{n+k} v$.
- (S-4) Then $e = (e_1, e_2)$ and $e' = (e_1', e_2)$. By inversion on (E-3), $e_1 \Downarrow^{n_1} v_1$ and $e_2 \Downarrow^{n_2} v_2$ and $(v_1, v_2) \Downarrow^1 \langle v_1, v_2 \rangle$ and $n = n_1 + n_2 + 1$. By induction, $e_1 \Downarrow^{n_1+k} v_1$ $|\sigma'| = k$. By (E-3), $e \Downarrow^{n+k} v$
- (S-5) Then $e = (v_1, e_2)$ and $e' = (v_1, e_2')$. By inversion on (E-3), $v_1 \Downarrow^0 v_1$ and $e_2 \Downarrow^{n_2} v_2$ and $(v_1, v_2) \Downarrow^1 \langle v_1, v_2 \rangle$ and $n = 0 + n_2 + 1$. By induction, $e_2 \Downarrow^{n_2+k} v_2$ $|\sigma'| = k$. By (E-3), $e \Downarrow^{n+k} v$
- (S-6) Then $e = (v_1, v_2)$ and $e' = (v_1, v_2)$. By inversion on (E-3), $v_1 \Downarrow^0 v_1$ and $v_2 \Downarrow^0 v_2$ and $(v_1, v_2) \Downarrow^1 \langle v_1, v_2 \rangle$ and $n = 0 + 0 + 1$. By induction, $e \Downarrow^{n_2+k} \langle v_1, v_2 \rangle$ $|\sigma'| = k = 1$. By (E-3), $e \Downarrow^{n+k} v$
- (S-7) Then $e = \mathsf{fst}\ e_1$ and $e' = \mathsf{fst}\ e_1'$. By inversion on (E-4), $e_1 \Downarrow^{n_1} \langle v_1, v_2 \rangle$ and $\mathsf{fst}\ e_1 \Downarrow^{n_1} v_1$ and $n = n_1$. By induction, $e \Downarrow^{n_1+k} v_1$ $|\sigma'| = k$. By (E-4), $e \Downarrow^{n+k} v$
- (S-8) Then $e = \mathsf{fst}\ \langle v_1, v_2 \rangle$ and $e' = v_1$. By inversion on (E-4), $\langle v_1, v_2 \rangle \Downarrow^0 \langle v_1, v_2 \rangle$ $e' \Downarrow^0 v_1$, so $n = 0$. By (S-8), $\sigma' = \emptyset$, so $k = 0$. By (E-4), $e \Downarrow^{n+k} v$
- (S-9) Then $e = \mathsf{snd}\ e_1$ and $e' = \mathsf{snd}\ e_1'$. By inversion on (E-4), $e_1 \Downarrow^{n_1} \langle v_1, v_2 \rangle$ and $\mathsf{fst}\ e_1 \Downarrow^{n_1} v_2$ and $n = n_1$. By induction, $e \Downarrow^{n_1+k} v_1$ $|\sigma'| = k$. By (E-5), $e \Downarrow^{n+k} v$
- (S-10) Then $e = \mathsf{fst}\ \langle v_1, v_2 \rangle$ and $e' = v_1$. By inversion on (E-5), $\langle v_1, v_2 \rangle \Downarrow^0 \langle v_1, v_2 \rangle$ $e' \Downarrow^0 v_2$, so $n = 0$. By (S-10), $\sigma' = \emptyset$, so $k = 0$. By (E-5), $e \Downarrow^{n+k} v$

□

Task 5. *Prove the remaining cases of Lemma 3. There are many similarities between the cases, but some small, important differences. Copying and pasting between cases is allowed and encouraged, but make sure you know what to change!*

Given that lemma, the theorem is pretty straightforward to prove. I'll just do it so you can be done with the homework.

Theorem 2. *If $\sigma; e \mapsto^* \sigma \uplus \sigma'; v$ and v val, then $e \Downarrow^n v$ and $n = |\sigma'|$.*

Proof. By induction on the derivation of $\emptyset; e \mapsto^* \sigma; v$. If $e = v$, then $\sigma = \emptyset$ and by (E-1), $v \Downarrow^0 v$. Otherwise, $\emptyset; e \mapsto \emptyset \uplus \sigma_1; e' \mapsto^* \emptyset \uplus \sigma_1 \uplus \sigma_2; v$. By induction, $e' \Downarrow^n v$ and $n = |\sigma_2|$. By Lemma 3, $e \Downarrow^{n+k} v$ where $k = |\sigma_1|$. We have $n + k = |\sigma_1 \uplus \sigma_2|$. □