

Fundamentals of Programming Languages

Assignment 2

Cost Semantics and Higher-Order Recursion

Guilherme João Correia Lopes
fc52761

2022/2023

1 Big-step semantics with cost

Assuming all the problem context given for the questions in group 1, we have the following as the answer for each question:

A. The big step evaluation rules

$$\frac{\text{B-SUCC} \quad t1 \Downarrow^n \text{nv1}}{\text{succ } t1 \Downarrow^{n+1} \text{succ nv1}}$$

$$\frac{\text{B-PREDZERO} \quad t1 \Downarrow^n 0}{\text{pred } t1 \Downarrow^{n+1} 0}$$

$$\frac{\text{B-PREDSUCC} \quad t1 \Downarrow^n \text{succ nv1}}{\text{pred } t1 \Downarrow^{n+1} \text{nv1}}$$

B. Example of reduction

$$\begin{array}{c}
\text{B-VALUE} \\
\frac{}{0 \Downarrow^0 0} \\
\text{B-SUCC} \\
\frac{\text{succ } 0 \Downarrow^{0+1} \text{ succ } 0}{\text{succ}(\text{succ } 0) \Downarrow^{1+1} \text{ succ}(\text{succ } 0)} \\
\text{B-SUCC} \\
\frac{\text{succ}(\text{succ } 0) \Downarrow^{1+1} \text{ succ}(\text{succ } 0)}{\text{pred } 2 \Downarrow^{1+2} \text{ succ } 0} \\
\text{B-PREDSUCC} \\
\frac{\text{pred } 2 \Downarrow^{1+2} \text{ succ } 0}{\text{pred}(\text{pred } 2) \Downarrow^{1+3} 0} \\
\text{B-PREDSUCC} \\
\text{*(lacking space for } 2^{nd} \text{ B-APP, see below)} \quad \frac{}{2 \Downarrow^0 2} \quad \frac{\text{pred}(\text{pred } 2) \Downarrow^{1+3} 0}{\text{pred } 2 \Downarrow^{1+0+4} 0} \\
\text{B-APP} \\
\hline
\lambda f. \lambda x. f(fx) \text{ pred } 2 \Downarrow^{1+0+4} 0
\end{array}$$

$$\frac{\begin{array}{c} \text{B-VALUE} \\ \hline \lambda f.\lambda x.f(fx) \Downarrow^0 \lambda f.\lambda x.f(fx) \end{array} \quad \begin{array}{c} \text{B-VALUE} \\ \hline \text{pred} \Downarrow^0 \text{pred} \end{array} \quad \begin{array}{c} \text{B-VALUE} \\ \hline \lambda x.\text{pred}(\text{pred } x) \Downarrow^0 \lambda x.\text{pred}(\text{pred } x) \end{array}}{\lambda f.\lambda x.f(fx) \text{ pred} \Downarrow^{0+0+0+1} \lambda x.\text{pred}(\text{pred } x)} \text{B-APP}$$

We can conclude that we have $applyTwice \text{ pred } 2 \Downarrow^k v$ for: cost $k = 5$; value $v = 0$;

C. Evaluation returns a value

Induction on the rules for $t \Downarrow^n v$

1. Induction on B-VALUE: $v \Downarrow^0 v$
This is straightforward, because the rule tells that every value reduces to a value in 0 steps.

2. Induction on B-APP:

Similarly to the B-SUCC, PREDZERO and B-PREDSUCC cases below, the rule B-APP tells us that $t \Downarrow^n v$ is true if it is confirmed to be the case that $t_1 \Downarrow^n \lambda x.t'_1$ and $t_2 \Downarrow^m v_2$ and $[x \rightarrow v_2]t'_1 \Downarrow^o v_1$. In this case, we have to explore 3 subcases for each premise.

(a) Subcase $t_1 \Downarrow^n \lambda x.t'_1$

This is somewhat straightforward, because t_1 reduces to $\lambda x.t'_1$, and $\lambda x.t'_1$ is a value.

(b) Subcase $t_2 \Downarrow^m v_2$

This is straightforward, because t_2 reduces to a value.

(c) Subcase $[x \rightarrow v_2]t'_1 \Downarrow^o v_1$

This is also straightforward, because the substitution $[x \rightarrow v_2]t'_1$ reduces to a value.

We can then see that, when applying B-APP, every subcase meets the requirements, so we can then conclude that v_1 is a value and have $t \Downarrow^n v$ for B-APP.

3. Induction on B-SUCC:

The rule B-SUCC tells us that $t \Downarrow^n v$ is true if it is confirmed to exist a $t_1 \Downarrow^n nv_1$. If so, by applying the induction hypothesis, for $t_1 \Downarrow^n nv_1$, we conclude that nv_1 is a value, and so we have $t \Downarrow^n v$ for B-SUCC.

4. Induction on B-PREDZERO:

The rule B-PREDZERO tells us that $t \Downarrow^n v$ is true if it is confirmed to exist a $t_1 \Downarrow^n 0$. If so, by applying the induction hypothesis, for $t_1 \Downarrow^n 0$, we conclude that 0 is a value, and so we have $t \Downarrow^n v$ for B-PREDZERO.

5. Induction on B-PREDSUCC:

Similarly to the B-SUCC and PREDZERO cases, the rule B-PREDSUCC tells us that $t \Downarrow^n v$ is true if it is confirmed to exist a $t_1 \Downarrow^n succ\ nv_1$. If so, by applying the induction hypothesis, for $t_1 \Downarrow^n succ\ nv_1$, we conclude that $succ\ nv_1$ is a value (proved in the B-SUCC case), and so we have $t \Downarrow^n v$ for B-PREDSUCC.

D. The cost of big step and small step semantics coincide

1. Proof by rule induction:

(a) Case B-VALUE:

The rule says that a value reduces to a value in 0 steps, in other words, $v \Downarrow^0 v$, so it is equal to $v \rightarrow^0 v$ according to the reflexivity of the multi-step evaluation.

(b) Case B-APP:

$t = t_1 t_2$, $v = v_1$ and $k = n + m + o + 1$

The rule says that $t \Downarrow^k v$ only if we have the following:

i. $t_1 \Downarrow^n \lambda x.t'_1$, so by the induction hypothesis, we have $t_1 \rightarrow^n \lambda x.t'_1$.

ii. $t_2 \Downarrow^m v_2$, so by the induction hypothesis, we have $t_2 \rightarrow^m v_2$.

iii. $[x \rightarrow v_2]t'_1 \Downarrow^o v_1$, so by the induction hypothesis, we have $[x \rightarrow v_2]t'_1 \rightarrow^o v_1$.

So we can conclude that if $t \Downarrow^k v$, then $t \rightarrow^k v$.

(c) Case B-SUCC:

$t = succ\ t_1$, $v = succ\ nv_1$ and $k = n + 1$

The rule says that $t \Downarrow^k v$ only if we have $t_1 \Downarrow^n nv_1$. By the induction hypothesis, having $t_1 \Downarrow^n nv_1$ then we have $t_1 \rightarrow^n nv_1$, so we can conclude that if $t \Downarrow^k v$, then $t \rightarrow^k v$.

(d) Case B-PREDZERO:

$t = pred\ t_1$, $v = 0$ and $k = n + 1$

The rule says that $t \Downarrow^k v$ only if we have $t_1 \Downarrow^n 0$. By the induction hypothesis, having $t_1 \Downarrow^n 0$ then we have $t_1 \rightarrow^n 0$, so we can conclude that if $t \Downarrow^k v$, then $t \rightarrow^k v$.

(e) Case B-PREDSUCC:

$t = \text{pred } t_1$, $v = nv_1$ and $k = n + 1$

The rule says that $t \Downarrow^k v$ only if we have $t_1 \Downarrow^n \text{succ } nv_1$. By the induction hypothesis, having $t_1 \Downarrow^n \text{succ } nv_1$ then we have $t_1 \rightarrow^n \text{succ } nv_1$, so we can conclude that if $t \Downarrow^k v$, then $t \rightarrow^k v$.

2. Proof by induction on k :

(a) Case $k = 0$:

The only case in which $k = 0$ is when we have the rule B-VALUE $v \rightarrow^0 v$, which says that a value reduces to a value in 0 steps, so by induction hypothesis we conclude $v \Downarrow^0 v$.

(b) Case $k = n$

i. Case B-APP:

$t = t_1 t_2$ and $v = v_1$

The rule says that $t \rightarrow^k v$ only if we have the following:

A. $t_1 \rightarrow^p \lambda x. t'_1$, so by the induction hypothesis, we have $t_1 \Downarrow^p \lambda x. t'_1$.

B. $t_2 \rightarrow^m v_2$, so by the induction hypothesis, we have $t_2 \Downarrow^m v_2$.

C. $[x \rightarrow v_2] t'_1 \rightarrow^o v_1$, so by the induction hypothesis, we have $[x \rightarrow v_2] t'_1 \Downarrow^o v_1$.

So we can conclude that if $t \rightarrow^k v$, then $t \Downarrow^k v$.

ii. Case B-SUCC:

$t = \text{succ } t_1$ and $v = \text{succ } nv_1$

The rule says that $t \rightarrow^k v$ only if we have $t_1 \rightarrow^{k-1} nv_1$. By the induction hypothesis, having $t_1 \rightarrow^{k-1} nv_1$ then we have $t_1 \Downarrow^{k-1} nv_1$, so we can conclude that if $t \rightarrow^k v$, then $t \Downarrow^k v$.

iii. Case B-PREDZERO:

$t = \text{pred } t_1$ and $v = 0$

The rule says that $t \rightarrow^k v$ only if we have $t_1 \rightarrow^{k-1} 0$. By the induction hypothesis, having $t_1 \rightarrow^{k-1} 0$ then we have $t_1 \Downarrow^{k-1} 0$, so we can conclude that if $t \rightarrow^k v$, then $t \Downarrow^k v$.

iv. Case B-PREDSUCC:

$t = \text{pred } t_1$ and $v = nv_1$

The rule says that $t \rightarrow^k v$ only if we have $t_1 \rightarrow^{k-1} \text{succ } nv_1$. By the induction hypothesis, having $t_1 \rightarrow^{k-1} \text{succ } nv_1$ then we have $t_1 \Downarrow^{k-1} \text{succ } nv_1$, so we can conclude that if $t \rightarrow^k v$, then $t \Downarrow^k v$.

2 Godel's system T

Assuming all the problem context given for the questions in group 2, we have the following as the answer for each question:

A. Example of reduction

$$\frac{\text{E-TRANSITIVITY} \quad \begin{array}{c} t \rightarrow t' \quad t' \rightarrow^* t'' \\ \hline t \rightarrow^* t'' \end{array}}{\quad}$$

$$\frac{\text{E-REFLEXIVITY} \quad \quad}{t \rightarrow^* t}$$

For the sake of space and readability, it will only be expressed the $t \rightarrow t'$ derivation directly, in multiple one-step transitions, correspondent to what the derivation tree using E-TRANSITIVITY and E-REFLEXIVITY rules would look like.

$$\begin{aligned} & (\lambda n : \text{Nat}. \text{rec}(0; x.y.(\text{succ } x) + y)(n))(\text{succ}(\text{succ}(\text{succ } 0))) \rightarrow \\ & \text{rec}(0; x.y.(\text{succ } x) + y)(\text{succ}(\text{succ}(\text{succ } 0))) \rightarrow \end{aligned}$$

$$\begin{aligned}
& (\text{succ}(\text{succ}(\text{succ } 0))) + (\text{rec}(0; x.y.(\text{succ } x) + y)(\text{succ}(\text{succ } 0))) \rightarrow \\
& (\text{succ}(\text{succ}(\text{succ } 0))) + (\text{succ}(\text{succ } 0)) + (\text{rec}(0; x.y.(\text{succ } x) + y)(\text{succ } 0)) \rightarrow \\
& (\text{succ}(\text{succ}(\text{succ } 0))) + (\text{succ}(\text{succ } 0)) + (\text{succ } 0) + (\text{rec}(0; x.y.(\text{succ } x) + y)(0)) \rightarrow \\
& (\text{succ}(\text{succ}(\text{succ } 0))) + (\text{succ}(\text{succ } 0)) + (\text{succ } 0) + 0 \rightarrow \\
& (\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ } 0))))))
\end{aligned}$$

B. A definition for +

$$\text{sum} = \lambda n. \lambda m. \text{rec}(m; x.y. \text{succ } y)(n)$$

C. Typing the recursor

$$\frac{\Gamma \vdash t_0 : T \quad \Gamma, x : \text{Nat}, y : T \vdash t_1 : T \quad \Gamma \vdash t : \text{Nat}}{\Gamma \vdash \text{rec}(t_0; x.y.t_1)(t) : T} \text{ T-REC}$$

D. A typing derivation

Assuming the existence of the arithmetic expressions typing rules for numbers (e.g. T-ZERO) and of the simply typed lambda-calculus typing rules (e.g. T-VAR and T-ABS).

$$\frac{\frac{\frac{}{n : \text{Nat} \vdash 0 : \text{Nat}} \text{ T-ZERO} \quad x : \text{Nat}, y : \text{Nat}, n : \text{Nat} \vdash (\text{succ } x) + y : \text{Nat}}{n : \text{Nat} \vdash \text{rec}(0; x.y.(\text{succ } x) + y)(n) : \text{Nat}} \text{ T-REC} \quad \frac{n : \text{Nat} \in n : \text{Nat}}{n : \text{Nat} \vdash n : \text{Nat}} \text{ T-VAR}}{\vdash \lambda n : \text{Nat}. \text{rec}(0; x.y.(\text{succ } x) + y)(n) : \text{Nat} \rightarrow \text{Nat}} \text{ T-ABS}$$

A term t is typable (or well typed) if there is some T such that $t : T$. With this, it is observable, that for the term S there is a T such that $S : T$, because, as the derivation shows, $S : \text{Nat} \rightarrow \text{Nat}$, in other words, $T = \text{Nat} \rightarrow \text{Nat}$.

E. Progress

[Proof] By induction on a derivation of $t : T$. The T-ZERO and T-ABS cases are immediate, since t in these cases is a value. For the other cases, we argue as follows.

1. Case T-SUCC:
 $t = \text{succ } t_1$ and $t_1 : \text{Nat}$
 By induction hypothesis, either t_1 is a value or else there is some t'_1 such that $t_1 \rightarrow t'_1$. If t_1 is a value, then the canonical forms lemma assures us that it must be a numeric value, in which case so is t . On the other hand, if $t_1 \rightarrow t'_1$, then, by E-SUCC, $\text{succ } t_1 \rightarrow \text{succ } t'_1$.
2. Case T-VAR:
 This case cannot occur, because t is closed.
3. Case T-APP:
 Using the Inversion of Typing Relation, we know that $t = t_1 t_2$, $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$ and $\Gamma \vdash t_2 : T_{11}$. By induction hypothesis, either t_1 is a value or else there is some t'_1 such that $t_1 \rightarrow t'_1$. By induction hypothesis, either t_2 is a value or else there is some t'_2 such that $t_2 \rightarrow t'_2$. With that said, we have the following:
 - (a) If $t_1 \rightarrow t'_1$, then apply E-APP1.
 - (b) If t_1 is a value and $t_2 \rightarrow t'_2$, then apply E-APP2.
 - (c) If t_1 is a value and t_2 is also a value, we know that $\vdash t_1 : T_{11} \rightarrow T_{12}$ and the canonical form tells that $t_1 = \lambda x. t_{11}$, then we apply E-APPABS.

4. Case T-REC:

$t = \text{rec}(t_0; x.y.t_1)(t_2)$, $\Gamma \vdash t_0 : T$, $\Gamma, x : \text{Nat}, y : T \vdash t_1 : T$ and $\Gamma \vdash t_2 : \text{Nat}$

By induction hypothesis, either t_2 is a value or else there is some t'_2 such that $t_2 \rightarrow t'_2$. If t_2 is a value, then the canonical forms lemma assures us that it must be a numeric value, in other words, either 0 or $\text{succ } nv$, and one of the rules E-REC-Z or E-REC-S applies to t . On the other hand, if $t_2 \rightarrow t'_2$, then, by E-REC-A, $\text{rec}(t_0; x.y.t_1)(t_2) \rightarrow \text{rec}(t_0; x.y.t_1)(t'_2)$.

F. Preservation

[Proof] By induction on a derivation of $t : T$. At each step of the induction, we assume that the desired property holds for all subderivations (i.e., that if $s : S$ and $s \rightarrow s'$, then $s' : S$ whenever $s : S$ is proved by a subderivation of the present one) then and proceed by case analysis on the final rule in the derivation.

1. Case T-ZERO:

$t = 0$ and $T = \text{Nat}$

If the last rule in the derivation is T-ZERO, then we know from the form of this rule that t must be a value and T must be $: \text{Nat}$. But if t is a value, then it cannot be the case that $t \rightarrow t'$ for any t' , and the requirements of the theorem are vacuously satisfied.

2. Case T-SUCC:

$t = \text{succ } t_1$, $T = \text{Nat}$ and $t_1 : \text{Nat}$

By inspecting the evaluation rules, we see that there is just one rule, E-SUCC, that can be used to derive $t \rightarrow t'$. The form of this rule tells us that $t_1 \rightarrow t'_1$. Since we also know $t_1 : \text{Nat}$, we can apply the induction hypothesis to obtain $t'_1 : \text{Nat}$, from which we obtain $\text{succ}(t'_1) : \text{Nat}$, i.e., $t' : T$, by applying rule T-SUCC.

3. Case T-VAR:

$t = x$ and $x : T$

If the last rule in the derivation is T-VAR, then we know from the form of this rule that t must be a value and T must be $: T$. But if t is a value, then it cannot be the case that $t \rightarrow t'$ for any t' , and the requirements of the theorem are vacuously satisfied.

4. Case T-ABS:

$t = \lambda x : T_1. t_2$, $T = T_1 \rightarrow T_2$ and $t_2 : T_2$

If the last rule in the derivation is T-ABS, then we know from the form of this rule that t must be a value and T must be $: T_1 \rightarrow T_2$. But if t is a value, then it cannot be the case that $t \rightarrow t'$ for any t' , and the requirements of the theorem are vacuously satisfied.

5. Case T-APP:

If the last rule in the derivation is T-APP, then, using the Inversion of Typing Relation, we know that $t = t_1 t_2$, $T = T_{12}$, $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$ and $\Gamma \vdash t_2 : T_{11}$. By inspecting the evaluation rules, we find that there are three rules, E-APP1, E-APP2 and E-APPABS, that can be used to derive $t \rightarrow t'$.

(a) E-APP1: $t_1 \rightarrow t'_1$, $t' = t'_1 t_2$

Since we know $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$, we can apply the induction hypothesis to obtain $\Gamma \vdash t'_1 : T_{11} \rightarrow T_{12}$, from which we obtain $\Gamma \vdash t'_1 t_2 : T$, i.e., $t' : T$, by applying rule T-APP.

(b) E-APP2: $t_2 \rightarrow t'_2$, $t' = t_1 t'_2$

Since we know $\Gamma \vdash t_2 : T_{11}$, we can apply the induction hypothesis to obtain $\Gamma \vdash t'_2 : T_{11}$, from which we obtain $\Gamma \vdash t_1 t'_2 : T$, i.e., $t' : T$, by applying rule T-APP.

(c) E-APPABS: $t' = \Gamma \vdash [x \rightarrow t_2] t'_1$

We know that $\vdash t_1 : T_{11} \rightarrow T_{12}$ and the canonical form tells that $t_1 = \lambda x. t'_1$

$\Gamma, x : T_{11} \vdash t'_1 : T_{12}$

t_2 is a value $\Gamma \vdash t_2 : T_{11}$

We obtain $\Gamma \vdash [x \rightarrow t_2] t'_1 : T$, i.e., $t' : T$, by applying the Preservation of Types Under Substitution, in which its result preserves the type T .

6. Case T-REC:

$t = \text{rec}(t_0; x.y.t_1)(t_2)$, $\Gamma \vdash t_0 : T$, $\Gamma, x : \text{Nat}, y : T \vdash t_1 : T$ and $\Gamma \vdash t_2 : \text{Nat}$

By inspecting the evaluation rules, we see that there are three rules, E-REC-A, E-REC-Z and E-REC-S, that can be used to derive $t \rightarrow t'$.

(a) E-REC-A: $t_2 \rightarrow t'_2$, $t' = \text{rec}(t_0; x.y.t_1)(t'_2)$

Since we know $\Gamma \vdash t_2 : \text{Nat}$, we can apply the induction hypothesis to obtain $\Gamma \vdash t'_2 : \text{Nat}$, from which we obtain $\text{rec}(t_0; x.y.t_1)(t'_2) : T$, i.e., $t' : T$, by applying rule T-REC.

(b) E-REC-Z: $t_2 = 0$, $t' = t_0$

If $t \rightarrow t'$ is derived using E-REC-Z, then from the form of this rule we see that t_2 must be 0 and the resulting term t' is t_0 . This means we are finished, since we know (by the assumptions of the T-REC case) that $t_0 : T$, which is what we need.

(c) E-REC-S: $t_2 = \text{succ } nv$, $t' = \Gamma \vdash [x \rightarrow nv][y \rightarrow s]t_1$

If $t \rightarrow t'$ is derived using E-REC-S, then from the form of this rule we have the following:

i. First Substitution:

$\Gamma, x : \text{Nat} \vdash t_1 : T$

$t_2 = \text{succ } nv$ and $\Gamma \vdash t_2 : \text{Nat}$

We obtain $\Gamma \vdash [x \rightarrow nv]t_1 : T$, i.e., $t' : T$, by applying the Preservation of Types Under Substitution, in which its result preserves the type T .

ii. Second Substitution:

Assuming t' to be the result of the first substitution.

$\Gamma, y : T \vdash t' : T$

$s = \text{rec}(t_0; x.y.t_1)(nv)$ and $s : T$

We obtain $\Gamma \vdash [y \rightarrow s]t' : T$, i.e., $t'' : T$, by applying the Preservation of Types Under Substitution, in which its result preserves the type T .