Sample Solution to COMP 523 Assignment 2

Joshua Dunfield
McGill University

February 10, 2008

1 Lazy evaluation

1.1 Force function (10pts)

$$\mathsf{force} \;=\; \lambda x.\,(\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)$$

Typing derivation (not required):

$$\dfrac{\dfrac{\dfrac{x{:}\mathsf{susp}\ \alpha \in x{:}\mathsf{susp}\ \alpha}{x{:}\mathsf{susp}\ \alpha \vdash x : \mathsf{susp}\ \alpha} \qquad \dfrac{y{:}\alpha \in (x{:}\mathsf{susp}\ \alpha, y{:}\alpha)}{x{:}\mathsf{susp}\ \alpha, y{:}\alpha \vdash y : \alpha}}{x{:}\mathsf{susp}\ \alpha \vdash \mathsf{let\ delay}\ y = x\ \mathsf{in}\ y : \alpha}}{\cdot \vdash \underbrace{\lambda x.\,(\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)}_{\mathsf{force}} : \mathsf{susp}\ \alpha \to \alpha}$$

Remark 1. Several people wrote something like

$$\mathsf{force}\ (x) = \;\lambda x.\,(\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)$$

which says that force is parameterized by some x; but a lambda-expression is already parameterized, indeed, that's the point of being a function.

Several people also took an accidental (and incorrect) shortcut by saying something like

$$\mathsf{force}\ (\mathsf{delay}\ e) \;=\; \mathsf{let\ delay}\ y = \mathsf{delay}\ e\ \mathsf{in}\ y$$

But the term on the left, $\mathsf{force}\ (\mathsf{delay}\ e)$, simply uses an *abbreviation*, "force"; $\mathsf{force}\ (\mathsf{delay}\ e)$ is equal only to

$$(\lambda x.\,\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)\ (\mathsf{delay}\ e)$$

1.2 Inverse (10pts)

Show that $\mathsf{force}\ (\mathsf{delay}\ e) \Downarrow v$ *iff* $e \Downarrow v$ *(that is,* force *is the inverse of* delay*).*

1.2.1 Right-to-left: If $e \Downarrow v$ then $\mathsf{force}\ (\mathsf{delay}\ e) \Downarrow v$

Proof. By the first big-step rule in the question, $\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e$.

$e \Downarrow v$ is given. By definition of substitution, $e = [e/y]y$. Therefore $[e/y]y \Downarrow v$.

Using the second big-step rule in the question gives

$$\dfrac{\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e \qquad [e/y]y \Downarrow v}{\mathsf{let\ delay}\ y = \mathsf{delay}\ e\ \mathsf{in}\ y \Downarrow v}$$

Again by definition of substitution, $(\mathsf{let\ delay}\ y = \mathsf{delay}\ e\ \mathsf{in}\ y) = [(\mathsf{delay}\ e)/x]\,(\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)$; substituting, we get

$$[(\mathsf{delay}\ e)/x]\,(\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y) \Downarrow v$$

We obtained $\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e$ above. By the big-step rule for functions,

$$(\underbrace{\lambda x.\,\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y}_{\mathsf{force}})\ (\mathsf{delay}\ e) \Downarrow v$$ □

1.2.2 Left-to-right: If force (delay e) ⇓ v then e ⇓ v

Proof. It is given that $\mathsf{force}\ (\mathsf{delay}\ e) \Downarrow v$. We first expand our definition of force :

$$(\lambda x.\,\mathsf{let\ delay}\ y = x\ \mathsf{in}\ y)\ (\mathsf{delay}\ e) \Downarrow v$$

By inversion on the derivation of the above judgment, we have $\mathsf{delay}\ e \Downarrow v_1$ for some v_1 and $\mathsf{let\ delay}\ y = v_1\ \mathsf{in}\ y \Downarrow v$. By inversion, $v_1 = \mathsf{delay}\ e$. Substituting, we get

$$\mathsf{let\ delay}\ y = \mathsf{delay}\ e\ \mathsf{in}\ y \Downarrow v$$

which can only be derived with the second big-step rule given in the question; inverting that rule gives

$$\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e_1 \quad \text{and} \quad [e_1/y]y \Downarrow v$$

By inversion on the first judgment, $e_1 = e$, so the second judgment is $[e/y]y \Downarrow v$. By definition of substitution, $[e/y]y = e$, so $e \Downarrow v$, which was to be shown. □

1.3 Type preservation (10pts)

Theorem 1. *If* $e \Downarrow v$ *and* $\cdot \vdash e : \tau$ *then* $\cdot \vdash v : \tau$.

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

- **Case** First rule: $\dfrac{}{\underbrace{\mathsf{delay}\ e'}_{e} \Downarrow \underbrace{\mathsf{delay}\ e'}_{v}}$

  $\cdot \vdash \mathsf{delay}\ e' : \tau$ is given, and since $v = \mathsf{delay}\ e'$, we have $\cdot \vdash v : \tau$, exactly what we needed to show.

- **Case** Second rule: $\dfrac{e_1 \Downarrow \mathsf{delay}\ e' \qquad [e'/x]e_2 \Downarrow v}{\underbrace{\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2}_{e} \Downarrow v}$

$\cdot \vdash \mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2 : \tau$	Given
$\cdot \vdash e_1 : \mathsf{susp}\ \tau'$	By inversion on second typing rule
$x{:}\tau' \vdash e_2 : \tau$	ditto
$e_1 \Downarrow \mathsf{delay}\ e'$	Subderivation
$\cdot \vdash e_1 : \mathsf{susp}\ \tau'$	Above
$\cdot \vdash \mathsf{delay}\ e' : \mathsf{susp}\ \tau'$	By i.h.
$\cdot \vdash e' : \tau'$	By inversion on first typing rule
$x{:}\tau' \vdash e_2 : \tau$	Above
$\cdot \vdash [e'/x]e_2 : \tau$	By substitution lemma
$[e'/x]e_2 \Downarrow v$	Subderivation
$\cdot \vdash v : \tau$	By i.h.

□

Inducting on the derivation of $\cdot \vdash e : \tau$, as several people did, doesn't work: in the "Second rule" case, the substitution lemma can give a *bigger* typing derivation. Think about the case when e' has an enormous typing derivation, and e_2 is some variable x; the derivation of $x{:}\tau' \vdash x : \tau'$ is very short, but the substitution lemma produces the enormous derivation of $\cdot \vdash e' : \tau'$. Inducting on $e \Downarrow v$ avoids this, because $[e'/x]e_2 \Downarrow v$ *is* a subderivation of $e \Downarrow v$.

1.4 Failure of type preservation under modified rule (10pts)

The problem is that the proposed rule substitutes e_1, which is a suspension—$e_1 = \mathsf{delay}\ e_1'$ for some e_1'—yet in the evaluation rule, we substitute e_1', the suspended expression. When we invert the rule in the proof above, instead of $x{:}\tau' \vdash e_2 : \tau$ (the old rule), we would get $\cdot \vdash [e_1/x]e_2 : \tau$. This does not match the relevant evaluation subderivation $[e'/x]e_2 : \tau$, and we cannot apply the i.h.

A simple counterexample is $\mathsf{let\ delay}\ x{=}\mathsf{delay}\ e'\ \mathsf{in}\ x$, where e' has type α. We can easily obtain $\cdot \vdash [(\mathsf{delay}\ e')/x]x : \mathsf{susp}\ \alpha$, so by the proposed rule $\cdot \vdash \mathsf{let\ delay}\ x = \mathsf{delay}\ e'\ \mathsf{in}\ x : \mathsf{susp}\ \alpha$. However, the operational semantics (supposing $e' \Downarrow v'$) gives

$$\mathsf{let\ delay}\ x = \mathsf{delay}\ e'\ \mathsf{in}\ x \Downarrow v'$$

(because $\mathsf{delay}\ e' \Downarrow \mathsf{delay}\ e'$ and $[e'/x]x \Downarrow v'$), but $\cdot \vdash v' : \alpha$, not $\cdot \vdash v' : \mathsf{susp}\ \alpha$.

1.5 Values (10pts)

From the first big-step evaluation rule given,

$$\frac{}{\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e}$$

it is apparent that $\mathsf{delay}\ e$ must be among the values: value soundness (which we're about to try to prove) says that $e_1 \Downarrow e_2$ is derivable only if e_2 is a value. So we add $\mathsf{delay}\ e$ to the values.

Theorem 2 (Value soundness)**.** *If $e \Downarrow e'$ then e' is a value.*[1]

Proof. By induction on the derivation of $e \Downarrow e'$.

We case-analyze the rule concluding that derivation, showing the cases for the new rules.

- **Case** First rule: $\dfrac{}{\underbrace{\mathsf{delay}\ e_1}_{e} \Downarrow \underbrace{\mathsf{delay}\ e_1}_{e'}}$

  By our definition of values, $\mathsf{delay}\ e_1$ is a value, which was to be shown.

- **Case** Second rule: $\dfrac{e_1 \Downarrow \mathsf{delay}\ e'' \qquad [e''/x]e_2 \Downarrow \overbrace{v}^{e'}}{\underbrace{\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2}_{e} \Downarrow \underbrace{v}_{e'}}$

  We have a subderivation of $[e''/x]e_2 \Downarrow e'$. By i.h., e' is a value, which was to be shown. □

1.6 Alternate primitives (10pts)

An appropriate evaluation rule for $\mathsf{force}\ e$ is:

$$\frac{e \Downarrow \mathsf{delay}\ e' \qquad e' \Downarrow v}{\mathsf{force}\ e \Downarrow v}$$

[1] The statement in the assignment, "if $e \Downarrow v$ then v is a value", is problematic: v is a value just by virtue of being written with the letter v and not, say, e'—but that is not what was intended.

Where we wrote let delay $x = e$ in e_2 before, we can write $(\lambda x.\, e_2)\,(\text{force } e_1)$ and produce the same result. In that sense, the two sets of primitives are equivalent. But we can also ask whether the same terms are evaluated. The given rule for let delay is call-by-name, in the sense that it does not force evaluation of the delayed term; for example, in let delay x = delay e_1 in w, the variable x does not appear in w, and so e_1 will not be evaluated at all. If functions are call-by-name, this is equivalent to $(\lambda x.\, e_2)\,(\text{force } e_1)$. However, with call-by-value functions, e_1 will always be evaluated in $(\lambda x.\, e_2)\,(\text{force } e_1)$, unlike let delay.

2 Case expression

2.1 Predecessor and iszero (5pts)

$$\begin{aligned} \mathsf{pred} &= \lambda x.\,(\mathsf{case}\ x\ \mathsf{of}\ z \Rightarrow z \mid \mathsf{succ}\ x' \Rightarrow x') \\ \mathsf{iszero} &= \lambda x.\,(\mathsf{case}\ x\ \mathsf{of}\ z \Rightarrow \mathsf{true} \mid \mathsf{succ}\ x' \Rightarrow \mathsf{false}) \end{aligned}$$

2.2 Typing rule (5pts)

$$\frac{\Gamma \vdash t : \text{NAT} \qquad \Gamma \vdash t_1 : T \qquad \Gamma, x{:}\text{NAT} \vdash t_2 : T}{\Gamma \vdash \mathsf{case}\ t\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2\ :\ T}$$

2.3 Type preservation (10pts)

Theorem 3. *If* $t \Downarrow v$ *and* $\cdot \vdash t : T$ *then* $\cdot \vdash v : T$.

Proof. By induction on the derivation of $t \Downarrow v$.

There are four "new" cases; we show the two that are relevant to our new typing rule. In both cases, $t = \mathsf{case}\ t_0\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2$ and we obtain the following by inversion on the new typing rule:

$$\cdot \vdash t_0 : \text{NAT} \qquad \cdot \vdash t_1 : T \qquad x{:}\text{NAT} \vdash t_2 : T$$

- **Case** : $\dfrac{t_0 \Downarrow z \qquad t_1 \Downarrow v}{\mathsf{case}\ t_0\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \Downarrow v}$

$\cdot \vdash t_1 : T$	Above
$t_1 \Downarrow v$	Subderivation
$\cdot \vdash v : T$	By i.h.

- **Case** : $\dfrac{t_0 \Downarrow \mathsf{succ}\ v_2 \qquad [v_2/x]t_2 \Downarrow v}{\mathsf{case}\ t_0\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \Downarrow v}$

$t_0 \Downarrow \mathsf{succ}\ v_2$	Subderivation
$\cdot \vdash t_0 : \text{NAT}$	Above
$\cdot \vdash v_2 : \text{NAT}$	By i.h.
$x{:}\text{NAT} \vdash t_2 : T$	Above
$\vdash [v_2/x]t_2 : T$	By substitution lemma
$[v_2/x]\,t_2 \Downarrow v$	Subderivation
$\cdot \vdash v : T$	By i.h.

□

2.4 Small-step evaluation (10pts)

$$\frac{t \to t'}{\mathsf{case}\ t\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \ \to\ \mathsf{case}\ t'\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2}$$

$$\frac{}{\mathsf{case}\ z\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \ \to\ t_1} \qquad \frac{}{\mathsf{case}\ \mathsf{succ}\ v\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \ \to\ [v/x]t_2}$$

2.5 Progress (10pts)

Theorem 4. *If* $\cdot \vdash t : T$ *then either* t *is a value or there exists* t' *such that* $t \to t'$.

Proof. By structural induction on the derivation of $\cdot \vdash t : T$.
We show the "new" case for our typing rule above.

- **Case** : $\dfrac{\cdot \vdash t_0 : \text{NAT} \qquad \cdot \vdash t_1 : T \qquad x{:}\text{NAT} \vdash t_2 : T}{\cdot \vdash \mathsf{case}\ t_0\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2\ :\ T}$

  We have a subderivation of $\cdot \vdash t_0 : \text{NAT}$. By i.h., either t_0 is a value or there exists t_0' such that $t_0 \to t_0'$. The second case is easier, so we give it first.

  - If $t_0 \to t_0'$: Let $t' = \mathsf{case}\ t_0'\ \mathsf{of}\ z \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2$. By our first small-step rule, $t \to t'$, which was to be shown.
  - If t_0 is a value, then by inversion on $\cdot \vdash t_0 : \text{NAT}$, it must be either z or $\mathsf{succ}\ v$ for some v.
    - If $t_0 = z$: Let $t' = t_1$. By our second small-step rule, $t \to t_1$, which was to be shown.
    - If $t_0 = \mathsf{succ}\ v$: Let $t' = [v/x]t_2$. By our third small-step rule, $t \to [v/x]t_2$, which was to be shown. □