Assignment 3: Adding streams to MinML
Model Solution

15-312 Foundations of Programming Languages
Kevin Watkins ⟨kw@cmu.edu⟩

April 28, 2005

Please refer to the assignment itself for the full description and statement of each problem.

§ 1. Static semantics **A 1.1.** [5 pts] The question didn't specify whether to write the grammar in natural style or in higher-order abstract syntax style. In natural style:

$$
\begin{array}{ll}
\text{Types} & \tau ::= \mathbf{int} \mid \mathbf{bool} \mid \tau_1 \to \tau_2 \mid \mathbf{stream} \\
\text{Exprs} & e ::= x \mid k \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{if}\ e\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2 \mid o(e_1, \ldots, e_n) \\
& \quad \mid \mathbf{fun}\ f(x : \tau_1) : \tau_2\ \mathbf{is}\ e\ \mathbf{end} \mid e_1\ e_2 \mid \mathbf{let}\ x = e_1\ \mathbf{in}\ e_2\ \mathbf{end} \\
& \quad \mid \mathbf{nil} \mid \mathbf{cons\ rec}\ x = e_1 \# e_2 \\
& \quad \mid (\mathbf{case}\ e\ \mathbf{of\ nil} \Rightarrow e_1 \mid x \# y \Rightarrow e_2)
\end{array}
$$

In h.o.a.s. style:

$$
\begin{array}{ll}
\text{Types} & \tau ::= \mathbf{int}() \mid \mathbf{bool}() \mid \mathbf{arrow}(\tau_1, \tau_2) \mid \mathbf{stream}() \\
\text{Exprs} & e ::= x \mid k() \mid \mathbf{true}() \mid \mathbf{false}() \mid \mathbf{if}(e, e_1, e_2) \mid o(e_1, \ldots, e_n) \\
& \quad \mid \mathbf{fun}(\tau_1, \tau_2, f.\, x.\, e) \mid \mathbf{app}(e_1, e_2) \mid \mathbf{let}(e_1, x.\, e_2) \\
& \quad \mid \mathbf{nil}() \mid \mathbf{consrec}(x.\, e_1, x.\, e_2) \mid \mathbf{case}(e, e_1, x.\, y.\, e_2)
\end{array}
$$

A 1.2. [5 pts]

$$\frac{}{\Gamma \vdash \mathbf{nil} : \mathbf{stream}}$$

$$\frac{\Gamma, (x : \mathbf{stream}) \vdash e_1 : \mathbf{int} \quad \Gamma, (x : \mathbf{stream}) \vdash e_2 : \mathbf{stream}}{\Gamma \vdash (\mathbf{cons\ rec}\ x = e_1 \# e_2) : \mathbf{stream}}$$

$$\frac{\Gamma \vdash e : \mathbf{stream} \quad \Gamma \vdash e_1 : \tau \quad \Gamma, (x : \mathbf{int}), (y : \mathbf{stream}) \vdash e_2 : \tau}{\Gamma \vdash (\mathbf{case}\ e\ \mathbf{of\ nil} \Rightarrow e_1 \mid x \# y \Rightarrow e_2) : \tau}$$

§ 2. Dynamic semantics **A 2.1.** [5 pts] In natural style:

$$\text{Values} \qquad v ::= \mathbf{nil} \mid \mathbf{cons\ rec}\ x = e_1 \# e_2$$

In h.o.a.s. style:

$$\text{Values} \qquad v ::= \mathbf{nil}() \mid \mathbf{consrec}(x.\, e_1, x.\, e_2)$$

A 2.2. [5 pts]

$$\frac{e \mapsto e'}{(\textbf{case } e \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto (\textbf{case } e' \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)}$$

$$\frac{}{(\textbf{case nil of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto e_1}$$

$$\frac{v = (\textbf{cons rec } z = hd\#tl)}{(\textbf{case } v \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto (\textbf{let } x = \{v/z\}hd \textbf{ in let } y = \{v/z\}tl \textbf{ in } e_2)}$$

§ 3. Type safety **A 3.1.** [30 pts]

First, the preservation theorem:

If $\cdot \vdash e : \tau$ *and* $e \mapsto e'$, *then* $\cdot \vdash e' : \tau$.

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

Case [$\dfrac{e_0 \mapsto e_0'}{(\textbf{case } e_0 \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto (\textbf{case } e_0' \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)}$]:
where $e = (\textbf{case } e_0 \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$ and $e' = (\textbf{case } e_0' \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$
wts $\cdot \vdash (\textbf{case } e_0' \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau$

1. $\cdot \vdash (\textbf{case } e_0 \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau$ (given)

2a. $\cdot \vdash e_0 : \textbf{stream}$ (inversion on 1)

2b. $\cdot \vdash e_1 : \tau$ (inversion on 1)

2c. $(x : \textbf{int}), (y : \textbf{stream}) \vdash e_2 : \tau$ (inversion on 1)

3. $e_0 \mapsto e_0'$ (given)

4. $\cdot \vdash e_0' : \textbf{stream}$ (i.h. on 2a and 3)

5. $\cdot \vdash (\textbf{case } e_0' \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau$ (inference rule on 4, 2b, and 2c)

Case [$\dfrac{}{(\textbf{case nil of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto e_1}$]:
where $e = (\textbf{case nil of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$ and $e' = e_1$
wts $\cdot \vdash e_1 : \tau$

1. $\cdot \vdash (\textbf{case nil of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau$ (given)

2. $\cdot \vdash e_1 : \tau$ (inversion on 1)

Case [$\dfrac{v = (\textbf{cons rec } z = hd\#tl)}{(\textbf{case } v \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto (\textbf{let } x = \{v/z\}hd \textbf{ in let } y = \{v/z\}tl \textbf{ in } e_2)}$]:
where $e = (\textbf{case } v \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$ and $e' = (\textbf{let } x = \{v/z\}hd \textbf{ in let } y = \{v/z\}tl \textbf{ in } e_2)$
wts $\cdot \vdash (\textbf{let } x = \{v/z\}hd \textbf{ in let } y = \{v/z\}tl \textbf{ in } e_2) : \tau$

1. $\cdot \vdash (\textbf{case } v \textbf{ of nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau$ (given)

2a. $\cdot \vdash v : \textbf{stream}$ (inversion on 1)

2b. $(x : \textbf{int}), (y : \textbf{stream}) \vdash e_2 : \tau$ (inversion on 1)

3a. $(z : \textbf{stream}) \vdash hd : \textbf{int}$ (inversion on 2a)

3b. $(z : \textbf{stream}) \vdash tl : \textbf{stream}$ (inversion on 2a)

4. $\cdot \vdash \{v/z\}hd : \textbf{int}$ (Value Substitution on 2a and 3a)

5. $\cdot \vdash \{v/z\}tl : \mathbf{stream}$ (Value Substitution on 2a and 3b)
6. $(x : \mathbf{int}) \vdash (\mathbf{let}\ y = \{v/z\}tl\ \mathbf{in}\ e_2) : \tau$ (inference rule on 5 and 2b)
7. $\cdot \vdash (\mathbf{let}\ x = \{v/z\}hd\ \mathbf{in}\ \mathbf{let}\ y = \{v/z\}tl\ \mathbf{in}\ e_2) : \tau$ (inference rule on 4 and 6)

And all the rest of the cases are the same as for the original MinML.

Next, the progress theorem:

If $\cdot \vdash e : \tau$ *then* e *is a value or* $e \mapsto e'$ *for some* e'.

Proof. By rule induction on the derivation of $\cdot \vdash e : \tau$.

Case [$\overline{\Gamma \vdash \mathbf{nil} : \mathbf{stream}}$]:
where $\Gamma = \cdot$, $e = \mathbf{nil}$, and $\tau = \mathbf{stream}$
wts **nil** is a value or **nil** $\mapsto e'$ for some e'

1. **nil** is a value (syntax of values)

Case [$\dfrac{\Gamma, (x : \mathbf{stream}) \vdash e_1 : \mathbf{int} \quad \Gamma, (x : \mathbf{stream}) \vdash e_2 : \mathbf{stream}}{\Gamma \vdash (\mathbf{cons\ rec}\ x = e_1\#e_2) : \mathbf{stream}}$]:
where $\Gamma = \cdot$, $e = (\mathbf{cons\ rec}\ x = e_1\#e_2)$, and $\tau = \mathbf{stream}$
wts $\mathbf{cons\ rec}\ x = e_1\#e_2$ is a value or $(\mathbf{cons\ rec}\ x = e_1\#e_2) \mapsto e'$ for some e'

1. $\mathbf{cons\ rec}\ x = e_1\#e_2$ is a value (syntax of values)

Case [$\dfrac{\Gamma \vdash e_0 : \mathbf{stream} \quad \Gamma \vdash e_1 : \tau \quad \Gamma, (x : \mathbf{int}), (y : \mathbf{stream}) \vdash e_2 : \tau}{\Gamma \vdash (\mathbf{case}\ e_0\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) : \tau}$]:
where $\Gamma = \cdot$ and $e = (\mathbf{case}\ e_0\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$
wts $\mathbf{case}\ e_0\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2$ is a value or $(\mathbf{case}\ e_0\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto e'$ for some e'

1. $\cdot \vdash e_0 : \mathbf{stream}$ (premise)

Subcase [e_0 is not a value]:

1.2. $e_0 \mapsto e_0'$ for some e_0' (i.h. on 1)
1.3. $(\mathbf{case}\ e_0\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto (\mathbf{case}\ e_0'\ \mathbf{of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2)$ (inference rule on 1.2)

Subcase [e_0 is a value]:

Now split cases based on the last rule in the derivation of $\cdot \vdash e_0 : \mathbf{stream}$. (Officially this is the *canonical forms lemma*, but it's so simple in this case I'm not bothering to state it as a separate lemma.)

Subsubcase [$\overline{\Gamma_0 \vdash \mathbf{nil} : \mathbf{stream}}$]:
where $\Gamma_0 = \cdot$ and $e_0 = \mathbf{nil}$

2.2. $(\mathbf{case\ nil\ of\ nil} \Rightarrow e_1 \mid x\#y \Rightarrow e_2) \mapsto e_1$ (inference rule)

Subsubcase [$\dfrac{\Gamma_0, (z : \mathbf{stream}) \vdash hd : \mathbf{int} \quad \Gamma_0, (z : \mathbf{stream}) \vdash tl : \mathbf{stream}}{\Gamma_0 \vdash (\mathbf{cons\ rec}\ z = hd\#tl) : \mathbf{stream}}$]:
where $\Gamma_0 = \cdot$ and $e_0 = (\mathbf{cons\ rec}\ z = hd\#tl)$

2.2. $(\mathbf{case}\ (\mathbf{cons\ rec}\ z = hd\#tl)\ \mathbf{of\ nil} \Rightarrow e_0 \mid x\#y \Rightarrow e_2) \mapsto (\mathbf{let}\ x = \{e_0/z\}hd\ \mathbf{in\ let}\ y = \{e_0/z\}tl\ \mathbf{in}\ e_2)$

And the rest of the cases are the same as for the original MinML.