Solution by Xiao Jia

Homework 4

Problem 1. (30 points) Extend tuples to records, and write the (a) syntax and (b) semantic rules for records. Example usage:

- Elements are indexed by labels: (Labels are not integers!)
  - $\{y = 10\}$
  - $\{id = 1, salary = 50000, active = \mathbf{true}\}$
- The order of the record fields is insignificant:
  - $\{y = 10, x = 5\}$ is the same as $\{x = 5, y = 10\}$
- To access fields of a record:
  - $a.id$
  - $b.salary$

Solution.

(a) Syntax: (x and x_i are names)

$$\begin{aligned} e &::= \ldots \mid \{x_1 = e_1, \ldots, x_n = e_n\} \mid e.x \\ v &::= \ldots \mid \{(x_1, v_1), \ldots, (x_n, v_n)\} \text{ (a set)} \\ t &::= \ldots \mid \{(x_1, t_1), \ldots, (x_n, t_n)\} \text{ (a set)} \end{aligned}$$

(b) Semantics:

$$\frac{e_i \to e_i'}{\{\ldots, x_i = e_i, \ldots\} \to \{\ldots, x_i = e_i', \ldots\}} \quad \text{(E-record1)}$$

$$\frac{}{\{x_1 = v_1, \ldots, x_n = v_n\} \to \{(x_1, v_1), \ldots, (x_n, v_n)\}} \quad \text{(E-record2)}$$

$$\frac{e \to e'}{e.x \to e'.x} \quad \text{(E-label1)}$$

$$\frac{}{\{(x_1, v_1), \ldots, (x_n, v_n)\}.x_i \to v_i} \quad \text{(E-label2)}$$

$$\frac{\Gamma \vdash e_i : t_i}{\Gamma \vdash \{x_1 = e_1, \ldots, x_n = e_n\} : \{(x_1, t_1), \ldots, (x_n, t_n)\}} \quad \text{(T-record)}$$

$$\frac{\Gamma \vdash e : \{(x_1, t_1), \ldots, (x_n, t_n)\}}{\Gamma \vdash e.x_i : t_i} \quad \text{(T-label)}$$

Names are very important in the type of records. $\square$

Problem 2. (30 points) Binary sums generalizes to variants just like pairs generalized to labeled records.

$$e ::= \ldots \mid \texttt{in}_i\ e_i$$
$$t ::= \ldots \mid t_1 + \ldots + t_n$$

Write the (a) syntax and (b) semantic rules for variants.

Solution.

(a) Syntax:

$$\begin{aligned}
\tau &::= t_1 + \cdots + t_n \\
e &::= \ldots \mid \texttt{in}_i[\tau]\ e \mid \texttt{case}\ e\ \texttt{of}\ \texttt{in}_1\ x \Rightarrow e_1 | \ldots | \texttt{in}_n\ x \Rightarrow e_n \\
v &::= \ldots \mid \mathbf{in}_i[\tau]\ v \\
t &::= \ldots \mid \tau
\end{aligned}$$

(b) Semantics:

$$\frac{e \to e'}{\texttt{in}_i[\tau]\ e \to \texttt{in}_i[\tau]\ e'}\ \text{(search rule)} \qquad \text{(E-inject1)}$$

$$\frac{}{\texttt{in}_i[\tau]\ v \to \mathbf{in}_i[\tau]\ v}\ \text{(inject into a value)} \qquad \text{(E-inject2)}$$

$$\frac{e \to e'}{\texttt{case}\ e\ \texttt{of}\ \cdots \to \texttt{case}\ e'\ \texttt{of}\ \ldots} \qquad \text{(E-case1)}$$

$$\frac{}{\texttt{case}\ \mathbf{in}_i[\tau]\ v\ \texttt{of}\ \texttt{in}_1\ x \Rightarrow e_1 | \ldots | \texttt{in}_n\ x \Rightarrow e_n \to e_i[v/x]} \qquad \text{(E-case2)}$$

$$\frac{\Gamma \vdash e : t_i}{\Gamma \vdash in_i[\tau]\ e : \tau} \qquad \text{(T-inject)}$$

$$\frac{\Gamma \vdash e : t_1 + \cdots + t_n \qquad \Gamma, x : t_i \vdash e_i : t}{\Gamma \vdash \texttt{case}\ e\ \texttt{of}\texttt{in}_1\ x \Rightarrow e_1 | \ldots | \texttt{in}_n\ x \Rightarrow e_n : t} \qquad \text{(T-case)}$$

□

Problem 3. (20 points) Use environment model to write the evaluation steps for

(a) factorial 3

(b) reverse (4 :: 3 :: 2 :: 1 :: nil)

Solution.

(a)

$$
\begin{aligned}
\text{factorial} &\equiv \text{ fix } \ g \\
g &\equiv \lambda f.\lambda n.\texttt{if}\ n = 0\ \texttt{then}\ 1\ \texttt{else}\ n * (f\ \ (n-1)) \\
g' &\equiv \texttt{if}\ n = 0\ \texttt{then}\ 1\ \texttt{else}\ n * ((\text{fix}\ \ g)\ \ (n-1))
\end{aligned}
$$

$$
\begin{aligned}
&(\cdot, \text{factorial}\ \ 3) \\
\equiv\ &(\cdot, (\text{fix}\ \ g)\ 3) \\
\to\ &(\cdot, (\lambda n.g')\ 3) \\
\to\ &(\cdot, \{\lambda n.g', \cdot\}\ 3) \\
&\quad \to (n \mapsto 3, g') \\
&\quad \to (n \mapsto 3, n * ((\text{fix}\ \ g)\ \ (n-1))) \\
&\quad \to (n \mapsto 3, 3 * ((\text{fix}\ \ g)\ \ (n-1))) \\
&\quad \to^2 (n \mapsto 3, 3 * (\{\lambda n.g', n \mapsto 3\}\ \ (n-1))) \\
&\quad \to (n \mapsto 3, 3 * (\{\lambda n.g', n \mapsto 3\}\ \ 2)) \\
&\qquad \to (n \mapsto 2, g') \\
&\qquad \to^2 (n \mapsto 2, 2 * ((\text{fix}\ \ g)\ \ (n-1))) \\
&\qquad \to^3 (n \mapsto 2, 2 * (\{\lambda n.g', n \mapsto 2\}\ \ 1)) \\
&\qquad\quad \to (n \mapsto 1, g') \\
&\qquad\quad \to^5 (n \mapsto 1, 1 * (\{\lambda n.g', n \mapsto 1\}\ \ 0)) \\
&\qquad\qquad \to (n \mapsto 0, g') \\
&\qquad\qquad \to 1 \\
&\qquad\quad \to 1 \\
&\qquad \to 2 \\
&\quad \to 6 \\
\to\ &6
\end{aligned}
$$

(b)

$$
\begin{aligned}
\text{append} &\equiv \text{ fix } \ g \\
g &\equiv \lambda a.\lambda l.\lambda n.\texttt{case}\ l\ \texttt{of}\ \ \text{nil} \Rightarrow n :: \text{nil}\,|x :: l \Rightarrow x :: ((a\ \ l)\ \ n) \\
g' &\equiv \lambda n.g'' \\
g'' &\equiv \texttt{case}\ l\ \texttt{of}\ \ \text{nil} \Rightarrow n :: \text{nil}\,|x :: l \Rightarrow x :: (((\text{fix}\ \ g)\ \ l)\ \ n) \\
\text{reverse} &\equiv \text{ fix } \ f \\
f &\equiv \lambda r.\lambda l.\texttt{case}\ l\ \texttt{of}\ \ \text{nil} \Rightarrow \text{nil}\,|x :: l \Rightarrow (\text{append}\ \ (r\ \ l))\ \ x \\
f' &\equiv \texttt{case}\ l\ \texttt{of}\ \ \text{nil} \Rightarrow \text{nil}\,|x :: nil \Rightarrow (\text{append}\ \ ((\text{fix}\ \ f)\ \ l))\ \ x
\end{aligned}
$$

$$
\begin{array}{l}
(\cdot, \text{reverse } (4 :: 3 :: 2 :: 1 :: \text{nil})) \\
\equiv (\cdot, (\text{fix } f)\ (4 :: 3 :: 2 :: 1 :: \text{nil})) \\
\to^2 (\cdot, \{\lambda l.f', \cdot\}\ (4 :: 3 :: 2 :: 1 :: \text{nil})) \\
\quad \to (l \mapsto 4 :: 3 :: 2 :: 1 :: \text{nil}, f') \\
\quad \to (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, (\text{append } ((\text{fix } f)\ l))\ x) \\
\quad \equiv (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, ((\text{fix } g)\ ((\text{fix } f)\ l))\ x) \\
\quad \to^2 (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 4, l \mapsto \dots\}\ ((\text{fix } f)\ l))\ x) \\
\quad \to^2 (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, (\{\lambda l.g', \dots\}\ (\{\lambda l.f', \dots\}\ l))\ x) \\
\quad \to (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, (\{\lambda l.g', \dots\}\ (\{\lambda l.f', \dots\}\ (3 :: 2 :: 1 :: \text{nil})))\ x) \\
\quad\quad \to (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, f') \\
\quad\quad \to (x \mapsto 3, l \mapsto 2 :: 1 :: \text{nil}, (\text{append } ((\text{fix } f)\ l))\ x) \\
\quad\quad \to^2 (x \mapsto 3, l \mapsto 2 :: 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 3, l \mapsto 2 :: 1 :: \text{nil}\}\ ((\text{fix } f)\ l))\ x) \\
\quad\quad\quad \to^5 (x \mapsto 2, l \mapsto 1 :: \text{nil}, (\text{append } ((\text{fix } f)\ l))\ x) \\
\quad\quad\quad \to^2 (x \mapsto 2, l \mapsto 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 2, l \mapsto 1 :: \text{nil}\}\ ((\text{fix } f)\ l))\ x) \\
\quad\quad\quad\quad \to^5 (x \mapsto 1, l \mapsto \text{nil}, (\text{append } ((\text{fix } f)\ l))\ x) \\
\quad\quad\quad\quad \to^2 (x \mapsto 1, l \mapsto \text{nil}, (\{\lambda l.g', x \mapsto 1, l \mapsto \text{nil}\}\ ((\text{fix } f)\ l))\ x) \\
\quad\quad\quad\quad\quad \to^2 (x \mapsto 1, l \mapsto \text{nil}, \texttt{case}\ l\ \texttt{of}\ \text{nil} \Rightarrow \text{nil} \,|\, \dots) \\
\quad\quad\quad\quad\quad \to \text{nil} \\
\quad\quad\quad\quad \to (x \mapsto 1, l \mapsto \text{nil}, (\{\lambda l.g', x \mapsto 1, l \mapsto \text{nil}\}\ \text{nil})\ x) \\
\quad\quad\quad\quad\quad \to (x \mapsto 1, l \mapsto \text{nil}, g') \\
\quad\quad\quad\quad\quad \to \{\lambda n.g'', x \mapsto 1, l \mapsto \text{nil}\} \\
\quad\quad\quad\quad \to (x \mapsto 1, l \mapsto \text{nil}, \{\lambda n.g'', x \mapsto 1, l \mapsto \text{nil}\}\ x) \\
\quad\quad\quad\quad \to (x \mapsto 1, l \mapsto \text{nil}, \{\lambda n.g'', x \mapsto 1, l \mapsto \text{nil}\}\ 1) \\
\quad\quad\quad\quad\quad \to (l \mapsto \text{nil}, x \mapsto 1, g'') \\
\quad\quad\quad\quad\quad \to 1 :: \text{nil} \\
\quad\quad\quad\quad \to 1 :: \text{nil} \\
\quad\quad\quad \to (x \mapsto 2, l \mapsto 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 2, l \mapsto 1 :: \text{nil}\}\ (1 :: \text{nil}))\ x) \\
\quad\quad\quad \to^* 1 :: 2 :: \text{nil} \\
\quad\quad \to (x \mapsto 3, l \mapsto 2 :: 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 3, l \mapsto 2 :: 1 :: \text{nil}\}\ (1 :: 2 :: \text{nil}))\ x) \\
\quad\quad \to^* 1 :: 2 :: 3 :: \text{nil} \\
\quad \to (x \mapsto 4, l \mapsto 3 :: 2 :: 1 :: \text{nil}, (\{\lambda l.g', x \mapsto 4, l \mapsto 3 :: \dots\}\ (1 :: 2 :: 3 :: \text{nil}))\ x) \\
\quad \to^* 1 :: 2 :: 3 :: 4 :: \text{nil} \\
\to 1 :: 2 :: 3 :: 4 :: \text{nil}
\end{array}
$$

Use stack frames and closures. □

Problem 4. (20 points) In untyped lambda calculus, we can use the Y combinator to implement recursion. However, in the extension of simply typed lambda calculus, recursion is implemented by extending the syntax with fix:

$$e ::= \ldots \mid \text{fix } e$$

Explain by concrete examples (e.g. programs) why we need this new syntax for recursion. Hint: Some untyped lambda terms are not typeable in simply typed lambda calculus.

Solution. Recall that fix is defined as

$$\text{fix} = \lambda f.(\lambda x.f\ (\lambda y.x\ x\ y))\ (\lambda x.f\ (\lambda y.x\ x\ y))$$

We show that fix is untypeable in simply typed lambda calculus (ST for short), so instead of being a syntactic sugar, it is an actual need to introduce recursion.
Abstractions in ST are in the form of $\lambda x : t.e$ and have the type defined by

$$\frac{\Gamma, x : t \vdash e : t'}{\Gamma \vdash \lambda x : t.e : t \to t'} \quad \text{(T-Abs)}$$

Suppose fix is typeable, then any sub-expression of fix should be typeable.

$$\text{fix} = \lambda f.(\lambda x : t_1 \to t_2 \to t_3.f\ (\lambda y : t_2.x\ x\ y))\ \ldots$$

The type of x has the form $t_1 \to t_2 \to t_3$ because of the application form $x\ x\ y$ where x itself is the first parameter, and y is the second parameter. Then y has type t_2, and x has type t_1. This leads to $t_1 = t_1 \to t_2 \to t_3$, which is impossible.
Exercise: Why impossible? Hint: Design a partial order on arrow types. □