Mid-term Exam

Foundations of Software

November 11, 2015

Last Name : ______________________________

First Name : ______________________________

Section : ______________________________

Exercise	Points	Achieved Points
1	10	
2	10	
3	10	
Total	30	

Exercise 1 : Hacking with the untyped call-by-value lambda calculus (10 points)

In this exercise, you have to implement some operations for Church encoding of lists. There are several ways to Church encode a list, among which Church encoding based on its right fold function is more popular. As an example, an empty list (`nil`) and the `cons` construct are represented as follows in this encoding:

```
nil = λc.  λn.  n
cons = λh.  λt.  λc.  λn.  c x (t c n)
```

As another example, a list of 3 elements x, y, z is encoded as:

```
λc.  λn.  c x (c y (c z n))
```

The complete list of predefined operations can be found in the appendix, and only these operations can be used in the exercise. Define the following operations on a list:

1. (2 points) The *map* function which applies the given function to each element of the given list.

   ```
   map = λl. λf. λc. λn. l (λh. λt. c (f h) t) n
   ```

2. (2 points) The *length* function which returns the size of the given list. The result should be in Church encoding.

   length = λl. l (λa. λb. scc b) c_0

3. (2 points) The *sum* function which returns the sum of all elements of the given list. Assume all elements and the result are Church encoded numbers.

   sum = λl. l plus c_0

4. (2 points) The *concat* function which concatenates two input lists.

   ```
   concat = λl1. λl2. λc. λn. l1 c (l2 c n)
   ```

5. (2 points) The *exists* function which checks if there is any element satisfying the given predicate. The given predicate and the result should be both in Church encoding.

   ```
   exists = λl. λp. l (λa. λb. p a tru b) fls
   ```

Exercise 2 : Curry-Howard Isomorphism (10 points)

Give proofs of the following propositional formulae using the Curry-Howard isomorphism between constructive logic and typed λ-calculus with products and sums (see appendix for details).

1. (2 points) $(A \Rightarrow B) \Rightarrow A \Rightarrow (A \vee B)$

   Solution:

   $\lambda f : A \to B.$
   $\lambda a : A.$
   $\texttt{inr}\ (f\ a)\ \texttt{as}\ A + B$

2. (2 points) $((A \Rightarrow B) \Rightarrow (B \Rightarrow C)) \Rightarrow B \Rightarrow (A \Rightarrow C)$

   Solution:

   $\lambda f : (A \to B) \to (B \to C).$
   $\lambda b : B.$
   $\lambda a : A.\ (f\ (\lambda dummy : A.\ b))\ b$

3. (3 points) $(A \wedge B \wedge C) \Rightarrow ((A \wedge B) \Rightarrow C) \Rightarrow (((C \vee D) \Rightarrow E) \Rightarrow E)$

   Solution:

   $\lambda p : ((A,\ B),\ C).$
   $\lambda f : (A,\ B) \to C.$
   $\lambda g : C + D \to E.$
   $g\ (\texttt{inl}\ (f\ p.\_1)\ \texttt{as}\ C + D)$

4. (3 points) $A \Rightarrow (A \Rightarrow B \vee C) \Rightarrow ((B \Rightarrow D) \Rightarrow (C \Rightarrow E) \Rightarrow (D \vee E))$

   Solution:

   $\lambda a : A.$
   $\lambda f : A \to B + C.$
   $\lambda g : B \to D.$
   $\lambda h : C \to E.$
   $\texttt{case}\ (f\ a)\ \texttt{of}$
   $\texttt{inl}\ b\ \Rightarrow \texttt{inl}\ (g\ b)\ \texttt{as}\ D + E\ |$
   $\texttt{inr}\ c\ \Rightarrow \texttt{inr}\ (h\ c)\ \texttt{as}\ D + E$

Exercise 3 : Adding the update operation to labeled records (10 points)

In this exercise, we build an extension to a call-by-value simply typed lambda calculus with labeled records. We would like to enhance the original calculus introduced in TAPL (see the appendix) with a construct that would make it easy to create slightly different records from existing ones. The syntax of the new operation is shown in the EBNF form:

$$\begin{array}{rcll} t & ::= & \dots & \textbf{terms} \\ & | & t_1 \texttt{ update } l \texttt{ with } t_2 & \text{updates field } l \text{ of a record } t_1 \text{ with } t_2 \end{array}$$

Despite having a unified syntax, `update` operates in two different modes. In the first mode, it takes a record, a field that's already in the record, a new value of the type that matches the existing field and then updates the record with the new value of that field. For example:

$$\{a = 1, b = false\} \texttt{ update } b \texttt{ with } true \rightarrow \{a = 1, b = true\}$$

In the second mode, `update` takes a record, a new field, a new value and then creates a record with the new field added to the very end of the field list and set to the new value. For example:

$$\{a = 1, b = true\} \texttt{ update } c \texttt{ with } 2 \rightarrow \{a = 1, b = true, c = 2\}$$

Your task for this assignment is as follows:

1. (5 points) Introduce new evaluation and typing rules that follow from the given syntactic extension and the behavior described above.
2. (5 points) Prove soundness of your system by detailing new cases of inductive proofs of progress and preservation. You need only discuss the cases that result from the evaluation and typing rules that you introduced, and, if needed, you may use (without proving them) the standard lemmas of inversion of typing, canonical forms, and uniqueness of types for labeled records.

Solution:

Reduction:

$$\frac{t_1 \to t_1'}{t_1 \textit{ update } l \textit{ with } t_2 \;\to\; t_1' \textit{ update } l \textit{ with } t_2} \qquad \text{(E-Upd1)}$$

$$\frac{t_2 \to t_2'}{v_1 \textit{ update } l \textit{ with } t_2 \;\to\; v_1 \textit{ update } l \textit{ with } t_2'} \qquad \text{(E-Upd2)}$$

$$\frac{l \in \{l_i^{i\in 1..n}\}, \quad l = l_k}{\left\{l_i = v_i^{i\in 1..n}\right\} \textit{ update } l \textit{ with } v \;\to\; \left\{l_i = v_i^{i\in 1..k-1}, l_k = v, l_i = v_i^{i\in k+1..n}\right\}} \qquad \text{(E-UpdOldFld)}$$

$$\frac{l \notin \{l_i^{i\in 1..n}\}}{\left\{l_i = v_i^{i\in 1..n}\right\} \textit{ update } l \textit{ with } v \;\to\; \left\{l_i = v_i^{i\in 1..n}, l = v\right\}} \qquad \text{(E-UpdNewFld)}$$

Notes:

- E-Upd1 and E-Upd2 might come in reverse order, but then valueness of the first component (v_1 vs t_1) of *update* should be respected.
- E-Upd1 is mandatory, because then E-Upd*Fld won't work with terms that are typed as records, but aren't record values themselves.
- E-Upd2 is optional, because as long as we have E-Upd*Fld rules, we can rely on E-Rcd to evaluate label values for us.
- E-UpdOldFld and E-UpdNewFld shouldn't be merged. The previous solution said something about $i \in 1..n$ and then kinda assumed that l in UpdNewFld is l_{n+1} or something. If students go this route, it might work, but the notation must be impeccable.
- E-Upd*Fld can say both $\{l_i = v_i\}$ *update l with ...* and $\{l_i = t_i\}$ *update l with ....* The latter is a bit fishy, because it will make both E-Upd1 and E-Upd*Fld applicable for some terms, but I don't think that contradicts anything.
- E-Upd*Fld must be consistent with E-UpdN with respect to valueness. E.g. if we drop E-Upd2, then E-Upd*Fld must be ... *update l with t* and not ... *update l with v*.

Typing:

$$\frac{l \in \{l_i^{i\in 1..n}\}, \quad l = l_k, \quad \Gamma \vdash t_1 : \{l_i : T_i^{i\in 1..n}\}, \quad \Gamma \vdash t_2 : T_k}{\Gamma \vdash t_1 \; \textit{update } l \textit{ with } t_2 \; : \; \{l_i : T_i^{i\in 1..n}\}} \quad \text{(T-UpdOldFld)}$$

$$\frac{l \notin \{l_i^{i\in 1..n}\}, \quad \Gamma \vdash t_1 : \{l_i : T_i^{i\in 1..n}\}, \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 \; \textit{update } l \textit{ with } t_2 \; : \; \{l_i : T_i^{i\in 1..n}, l : T\}} \quad \text{(T-UpdNewFld)}$$

Notes:

- T-UpdOldFld and T-UpdNewFld shouldn't be merged in the same fashion as E-Upd*Fld.
- T-UpdOldFld needs to say that $t_2 : T_k$, where T_k is one of the field types in $l_i : T_i$, because the problem statement explicitly requests that.

Induction proofs are fairly trivial, so we're not including them here.

For reference: **Untyped lambda-calculus**

The complete reference of the untyped lambda-calculus with call-by-value semantics is:

$$\begin{array}{rclr}
t & ::= & & \textbf{terms}: \\
 & | & x & \textit{variable} \\
 & | & \lambda\, x\ .\ t & \textit{abstraction} \\
 & | & t\ t & \textit{application} \\
 \\
v & ::= & & \textbf{values}: \\
 & | & \lambda\, x .\ t & \textit{abstraction} - \textit{value}
\end{array}$$

Evaluation rules:

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \qquad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \qquad \text{(E-App2)}$$

$$(\lambda\, x .\ t_1)\ v_2 \longrightarrow [x \to v_2]\ t_1 \qquad \text{(E-AppAbs)}$$

For reference: **Predefined Lambda Terms**

Predefined lambda terms that can be used as-is in the exam

`tru` =	$\lambda t.\ \lambda f.\ t$
`fls` =	$\lambda t.\ \lambda f.\ f$
`iszro` =	$\lambda m.\ m\ (\lambda x.\ fls)\ tru$
`test` =	$\lambda b.\ \lambda t.\ \lambda f.\ b\ t\ f\ unit$
`pair` =	$\lambda f.\ \lambda s.\ \lambda b.\ b\ f\ s$
`fst` =	$\lambda p.\ p\ tru$
`snd` =	$\lambda p.\ p\ fls$
$\mathtt{c}_0$ =	$\lambda s.\ \lambda z.\ z$
$\mathtt{c}_1$ =	$\lambda s.\ \lambda z.\ s\ z$
`scc` =	$\lambda n.\ \lambda s.\ \lambda z.\ s\ (n\ s\ z)$
`plus` =	$\lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ s\ (n\ s\ z)$
`times` =	$\lambda m.\ \lambda n.\ m\ (plus\ n)\ c_0$
`zz` =	$pair\ c_0\ c_0$
`ss` =	$\lambda p.\ pair\ (snd\ p)\ (scc\ (snd\ p))$
`prd` =	$\lambda m.\ fst\ (m\ ss\ zz)$
`fix` =	$\lambda f.\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))$

For reference: **Curry-Howard Isomorphism**

Curry-Howard isomorphism or *Curry-Howard correspondence* establishes a connection between type systems and logical calculi based on an observation that the ways we build types are structurally similar to the ways we build formulae.

According to Curry-Howard isomophism proofs can be represented as programs and formulae they prove can be represented as types of those programs. Here is a (non-comprehensive) list of some examples of how concepts from constructive logic are correlated with concepts from simply typed lambda calculus.

Constructive logic	Simply typed lambda calculus
Formula	Type
$A \Rightarrow B$	$A \longrightarrow B$
$A \wedge B$	$A \times B$
$A \vee B$	$A + B$
Proof of a formula	Term that inhabits a type

For reference: **Simply Typed Lambda Calculus**

The complete reference of the simply typed lambda calculus is:

$$\begin{array}{rcll}
t & ::= & & \textbf{terms}: \\
& | & x & \textit{variable} \\
& | & \lambda\, x : \texttt{T}.\ t & \textit{abstraction} \\
& | & t\ t & \textit{application} \\
\\
v & ::= & & \textbf{values}: \\
& | & \lambda\, x : \texttt{T}.\ t & \textit{abstraction} - \textit{value} \\
\\
\texttt{T} & ::= & & \textbf{types}: \\
& | & \texttt{T} \to \texttt{T} & \textit{type of functions}
\end{array}$$

Evaluation rules:

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \qquad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \qquad \text{(E-App2)}$$

$$(\lambda\, x\colon \texttt{T}_1.\ t_1)\ v_2 \longrightarrow [x \to v_2]\ t_1 \qquad \text{(E-AppAbs)}$$

Typing rules:

$$\frac{x : \texttt{T} \in \Gamma}{\Gamma \vdash x : \texttt{T}} \qquad \text{(T-Var)}$$

$$\frac{\Gamma,\ x : \texttt{T}_1 \vdash t_2 : \texttt{T}_2}{\Gamma \vdash (\lambda\, x\colon \texttt{T}_1.\ t_2) : \texttt{T}_1 \to \texttt{T}_2} \qquad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{T}_1 \to \texttt{T}_2 \quad \Gamma \vdash t_2 : \texttt{T}_1}{\Gamma \vdash t_1\ t_2 : \texttt{T}_2} \qquad \text{(T-App)}$$

For reference: **Sum extension to STLC**

$$
\begin{array}{rcll}
t & ::= & \ldots & \textbf{terms}: \\
 & | & \texttt{inl}\ t & \textit{tagging (left)} \\
 & | & \texttt{inr}\ t & \textit{tagging (right)} \\
 & | & \texttt{case}\ t\ \texttt{of}\ \texttt{inl x} \Rightarrow t \mid \texttt{inr x} \Rightarrow t & \textit{case} \\
\\
v & ::= & \ldots & \textbf{values}: \\
 & | & \texttt{inl}\ v & \textit{tagged value (left)} \\
 & | & \texttt{inr}\ v & \textit{tagged value (right)} \\
\\
\texttt{T} & ::= & \ldots & \textbf{types}: \\
 & | & \texttt{T} + \texttt{T} & \textit{sum type}
\end{array}
$$

Typing rules:

$$\frac{\Gamma \vdash t_1 : \texttt{T}_1}{\Gamma \vdash \texttt{inl}\ t_1 : \texttt{T}_1 + \texttt{T}_2} \qquad \text{(T-INL)}$$

$$\frac{\Gamma \vdash t_2 : \texttt{T}_2}{\Gamma \vdash \texttt{inl}\ t_2 : \texttt{T}_1 + \texttt{T}_2} \qquad \text{(T-INR)}$$

$$\frac{\Gamma \vdash t_0 : \texttt{T}_1 + \texttt{T}_2 \quad \Gamma, x{:}\texttt{T}_1 \vdash t_1 : \texttt{T} \quad \Gamma, x{:}\texttt{T}_2 \vdash t_2 : \texttt{T}}{\Gamma \vdash \texttt{case}\ t_0\ \texttt{of}\ \texttt{inl x} \Rightarrow t_1 \mid \texttt{inr x} \Rightarrow t_2 : \texttt{T}} \qquad \text{(T-CASE)}$$

New evaluation rules:

$$\begin{array}{c}\texttt{case}\ (\texttt{inl}\ v_0)\ \texttt{of}\ \texttt{inl}\ \texttt{x}_1 \Rightarrow t_1 \mid \texttt{inr}\ \texttt{x}_2 \Rightarrow t_2 \\ \longrightarrow [\texttt{x}_1 \to v_0]\ t_1\end{array} \qquad \text{(E-CASEINL)}$$

$$\begin{array}{c}\texttt{case}\ (\texttt{inr}\ v_0)\ \texttt{of}\ \texttt{inl}\ \texttt{x}_1 \Rightarrow t_1 \mid \texttt{inr}\ \texttt{x}_2 \Rightarrow t_2 \\ \longrightarrow [\texttt{x}_2 \to v_0]\ t_2\end{array} \qquad \text{(E-CASEINR)}$$

$$\frac{t_0 \longrightarrow t_0'}{\begin{array}{c}\texttt{case}\ t_0\ \texttt{of}\ \texttt{inl}\ \texttt{x}_1 \Rightarrow t_1 \mid \texttt{inr}\ \texttt{x}_2 \Rightarrow t_2 \\ \longrightarrow \texttt{case}\ t_0'\ \texttt{of}\ \texttt{inl}\ \texttt{x}_1 \Rightarrow t_1 \mid \texttt{inr}\ \texttt{x}_2 \Rightarrow t_2\end{array}} \qquad \text{(E-CASE)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inl}\ t_1 \longrightarrow \texttt{inl}\ t_1'} \qquad \text{(E-INL)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inr}\ t_1 \longrightarrow \texttt{inr}\ t_1'} \qquad \text{(E-INR)}$$

For reference: **Product extension to STLC**

$$
\begin{array}{rcll}
t & ::= & \ldots & \textbf{terms}: \\
 & \mid & \{t, t\} & \textit{pair} \\
 & \mid & t.1 & \textit{first projection} \\
 & \mid & t.2 & \textit{second projection} \\
\\
v & ::= & \ldots & \textbf{values}: \\
 & \mid & \{v, v\} & \textit{pair value} \\
\\
\mathtt{T} & ::= & \ldots & \textbf{types}: \\
 & \mid & \mathtt{T}_1 \times \mathtt{T}_2 & \textit{product type}
\end{array}
$$

Typing rules:

$$\frac{\Gamma \vdash t_1 : \mathtt{T}_1 \quad \Gamma \vdash t_2 : \mathtt{T}_2}{\Gamma \vdash \{t_1, t_2\} : \mathtt{T}_1 \times \mathtt{T}_2} \qquad \text{(T-Pair)}$$

$$\frac{\Gamma \vdash t : \mathtt{T}_1 \times \mathtt{T}_2}{\Gamma \vdash t.1 : \mathtt{T}_1} \qquad \text{(T-Proj1)}$$

$$\frac{\Gamma \vdash t : \mathtt{T}_1 \times \mathtt{T}_2}{\Gamma \vdash t.2 : \mathtt{T}_2} \qquad \text{(T-Proj2)}$$

New evaluation rules:

$$\{v_1, v_2\}\mathtt{.1} \longrightarrow v_1 \qquad \text{(E-PairBeta1)}$$

$$\{v_1, v_2\}\mathtt{.2} \longrightarrow v_2 \qquad \text{(E-PairBeta2)}$$

$$\frac{t \longrightarrow t'}{t\mathtt{.1} \longrightarrow t'\mathtt{.1}} \qquad \text{(E-Proj1)}$$

$$\frac{t \longrightarrow t'}{t\mathtt{.2} \longrightarrow t'\mathtt{.2}} \qquad \text{(E-Proj2)}$$

$$\frac{t_1 \longrightarrow t_1'}{\{t_1, t_2\} \longrightarrow \{t_1', t_2\}} \qquad \text{(E-Pair1)}$$

$$\frac{t_2 \longrightarrow t_2'}{\{v_1, t_2\} \longrightarrow \{v_1, t_2'\}} \qquad \text{(E-Pair2)}$$

For reference: **Record extension to STLC**

$$
\begin{array}{lcll}
l & ::= & & \textbf{labels}: \\
 & | & x & \textit{name} \\[1em]
t & ::= & \ldots & \textbf{terms}: \\
 & | & \{l_i = t_i{}^{i \in 1..n}\} & \textit{record} \\
 & | & t.l & \textit{projection} \\[1em]
v & ::= & \ldots & \textbf{values}: \\
 & | & \{l_i = v_i{}^{i \in 1..n}\} & \textit{record value} \\[1em]
\mathtt{T} & ::= & \ldots & \textbf{types}: \\
 & | & \{l_i : T_i{}^{i \in 1..n}\} & \textit{record type}
\end{array}
$$

Typing rules:

$$\frac{\texttt{for each } i \quad \Gamma \vdash t_i : \mathtt{T}_i}{\Gamma \vdash \{l_i = t_i{}^{i \in 1..n}\} : \{l_i : \mathtt{T}_i{}^{i \in 1..n}\}} \qquad \text{(T-Rcd)}$$

$$\frac{\Gamma \vdash t_1 : \{l_i : \mathtt{T}_i{}^{i \in 1..n}\}}{\Gamma \vdash t_1.l_j : \mathtt{T}_j} \qquad \text{(T-Proj)}$$

New evaluation rules:

$$\{l_i = v_i{}^{i \in 1..n}\}.l_j \longrightarrow v_j \qquad \text{(E-ProjRcd)}$$

$$\frac{t \longrightarrow t'}{t.\ l \longrightarrow t'.\ l} \qquad \text{(E-Proj)}$$

$$\frac{t_j \longrightarrow t'_j}{\{l_i = v_i{}^{i \in 1..j-1},\ l_j = t_j,\ l_k = t_k{}^{k \in j+1..n}\} \longrightarrow \{l_i = v_i{}^{i \in 1..j-1},\ l_j = t'_j,\ l_k = t_k{}^{k \in j+1..n}\}}$$

(E-Rcd)