CIS 500 — Software Foundations

Midterm I

Answer key

October 12, 2005

Name: ____

Student ID: ____

Email ____

Status
- ____ registered for the course
- ____ not registered: trying to improve a previous grade
- ____ not registered: just taking the exam for practice

Program
- ____ undergrad
- ____ undergrad (MSE submatriculant)
- ____ CIS MSE
- ____ CIS MCIT
- ____ CIS PhD
- ____ other

Instructions

- This is a closed-book exam: you may not make use of any books or notes.
- You have 80 minutes to answer all of the questions. The entire exam is worth 80 points.
- Questions vary significantly in difficulty, and the point value of a given question is not always exactly proportional to its difficulty. Do not spend too much time on any one question.
- Partial credit will be given. All correct answers are short. The back side of each page may be used as a scratch pad.
- Good luck!

Operational semantics

The first four questions concern the following simple programming language:

`t` ::=		*terms*
	`true`	*constant true*
	`false`	*constant false*
	`if t then t else t`	*conditional*
	`pair t t`	*pairing*
	`fst t`	*first component*
	`snd t`	*second component*
`v` ::=		*values*
	`true`	*true value*
	`false`	*false value*
	`pair v v`	*pair value*

and its *large-step* operational semantics.

$$\texttt{true} \Downarrow \texttt{true} \tag{B-True}$$

$$\texttt{false} \Downarrow \texttt{false} \tag{B-False}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{true} \qquad \texttt{t}_2 \Downarrow \texttt{v}}{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Downarrow \texttt{v}} \tag{B-IfTrue}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{false} \qquad \texttt{t}_3 \Downarrow \texttt{v}}{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Downarrow \texttt{v}} \tag{B-IfFalse}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{v}_1 \qquad \texttt{t}_2 \Downarrow \texttt{v}_2}{\texttt{pair } \texttt{t}_1 \ \texttt{t}_2 \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2} \tag{B-Pair}$$

$$\frac{\texttt{t} \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2}{\texttt{fst t} \Downarrow \texttt{v}_1} \tag{B-Fst}$$

$$\frac{\texttt{t} \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2}{\texttt{snd t} \Downarrow \texttt{v}_2} \tag{B-Snd}$$

1. (5 points) Show the derivation of the *large-step* evaluation of the following term.

`fst (if true then pair true false else pair false true)`

Answer:

$$\dfrac{\dfrac{}{\mathit{true} \Downarrow \mathit{true}}\ \textsc{B-True} \qquad \dfrac{\dfrac{}{\mathit{true} \Downarrow \mathit{true}}\ \textsc{B-True} \quad \dfrac{}{\mathit{false} \Downarrow \mathit{false}}\ \textsc{B-False}}{\mathit{pair\ true\ false} \Downarrow \mathit{pair\ true\ false}}\ \textsc{B-Pair}}{\dfrac{\mathit{if\ true\ then\ pair\ true\ false\ else\ pair\ false\ true} \Downarrow \mathit{pair\ true\ false}}{\mathit{fst\ (if\ true\ then\ pair\ true\ false\ else\ pair\ false\ true)} \Downarrow \mathit{true}}\ \textsc{B-Fst}}\ \textsc{B-IfTrue}$$

2. (10 points) We might also want to define a *small-step* semantics for this language, such that

$$\mathtt{t} \Downarrow \mathtt{v} \text{ if and only if } \mathtt{t} \longrightarrow^{*} \mathtt{v}$$

Recall that a small-step semantics is composed of both computation and congruence rules. The congruence rules for this language are as follows:

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3} \qquad \text{(E-If)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{pair}\ \mathtt{t}_1\ \mathtt{t}_2 \longrightarrow \mathtt{pair}\ \mathtt{t}_1'\ \mathtt{t}_2} \qquad \text{(E-Pair1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{pair}\ \mathtt{v}\ \mathtt{t}_2 \longrightarrow \mathtt{pair}\ \mathtt{v}\ \mathtt{t}_2'} \qquad \text{(E-Pair2)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{fst}\ \mathtt{t}_1 \longrightarrow \mathtt{fst}\ \mathtt{t}_1'} \qquad \text{(E-Fst)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{snd}\ \mathtt{t}_1 \longrightarrow \mathtt{snd}\ \mathtt{t}_1'} \qquad \text{(E-Snd)}$$

Some of the computation rules are:

$$\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{t}_2 \qquad \text{(E-IfTrue)}$$

$$\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{t}_3 \qquad \text{(E-IfFalse)}$$

However, this list is not complete. What are the remaining computation rules for the small-step semantics?

Answer:

$$\mathit{fst}\ (\mathit{pair}\ v_1\ \ v_2) \longrightarrow v_1 \qquad \text{(E-Fst1)}$$

$$\mathit{snd}\ (\mathit{pair}\ v_1\ \ v_2) \longrightarrow v_2 \qquad \text{(E-Snd1)}$$

3. (5 points) With a small step semantics, there is always the possibility that a term could fail to produce a value. Are there any "stuck" terms in this language? If so, give an example. If not, explain why not.

   Answer: *Yes, there are stuck terms, such as* `fst true`*.*

Functional programming

The following questions are about the untyped lambda calculus. For reference, the semantics of this language appears at the end of the exam.

Recall the Church encoding of lists and booleans in the untyped lambda calculus.

```
tru = λx. λy.x
fls = λx. λy.y
not = λb. b fls tru
and = λb1. λb2. b1 b2 fls
or  = λb1. λb2. b1 tru b2

nil = λc. λn. n
cons = λh. λt. λc. λn. c h (t c n)
head = λl. l (λh.λt. h) fls
tail = λl.
          fst (l (λx. λp. pair (snd p) (cons x (snd p)))
                 (pair nil nil))
isnil = λl. l (λh.λt. fls) tru
```

4. (5 points) Which of the following terms defines the function `all` that takes a list of boolean terms and determines of **all** of the terms are true? For example,

   `all (cons tru (cons fls nil))` should be equivalent to `fls`

   and

   `all nil` should be equivalent to `tru`. Circle the correct answer.

   (a) `all = λl. l and tru`

   (b) `all = λl. all (hd l) (tail l)`

   (c) `all = λl. l (λa.λb. a tru b) fls`

   (d) `all = λl. (λa. λb. a and b) l fls`

   Answer: a

5. (5 points) Which of the following terms defines the function `map` that take a term `l`, representing a list, and a function `f`, applies `f` to each element of `l`, and yields a list of the results (just like the `List.map` function in OCaml). For example:

   `map not (cons tru (cons fls nil))`

   should be equivalent to

   `(cons fls (cons tru nil))`. Circle the correct answer.

   (a) `map = λf. λl. l (f cons) nil`

   (b) `map = λf. λl. l (λh. λt. cons t (f h)) nil`

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

   Answer: c

6. (5 points) Which of the following OCaml terms implements the map function, using recursion? Circle the correct answer.

   (a) `let rec map f l = match l with [] → f [] | (hd :: tl) → f hd :: map f tl`

   (b) `let rec map f l = match l with [] → [] | (hd :: tl) → f hd :: map f tl`

   (c) `let rec map f l = match l with [] → f [] | (hd :: tl) → hd :: map f tl`

   (d) `let rec map f l = match l with [] → [] | (hd :: tl) → hd :: map f tl`

   (e) `let rec map f l = match l with [] → f [] | (hd :: tl) → f hd :: f tl`

   (f) None of the above

   Answer: b

Proofs by induction

7. (4 points) What is the structural induction principle for the untyped lambda calculus?

   Answer: For all $\mathtt{t}$, $P(\mathtt{t})$ *if and only if*

   - $P(\mathtt{x})$
   - $P(\mathtt{t})$ *implies* $P(\lambda\mathtt{x}.\,\mathtt{t})$
   - $P(\mathtt{t}_1)$ *and* $P(\mathtt{t}_2)$ *implies* $P(\mathtt{t}_1\ \ \mathtt{t}_2)$.

8. (15 points) Complete the following proof of a property of the untyped lambda calculus, by induction on the structure of lambda terms.

   Theorem: If $\mathtt{t}$ is closed, and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}'$ is closed.

   You may use, without proving, the following lemma about substitution.

   Lemma: If $\lambda\mathtt{x}.\mathtt{t}_1$ is closed, and $\mathtt{t}_2$ is closed, then the substitution $[\mathtt{x} \mapsto \mathtt{t}_2]\mathtt{t}_1$ is also closed.

   We prove the theorem by induction on the structure of the lambda term $\mathtt{t}$.

   - Suppose $\mathtt{t}$ is a variable $\mathtt{x}$. This case is trivial because *Answer: variables are not closed.*
   - Suppose $\mathtt{t}$ is a lambda term $\lambda\mathtt{x}.\ \mathtt{t}_1$. This case is also trivial because *Answer:* $\mathtt{t} \not\longrightarrow$.
   - Suppose $\mathtt{t}$ is an application $\mathtt{t}_1\ \ \mathtt{t}_2$. Consider the possible ways that $\mathtt{t} \longrightarrow \mathtt{t}'$.
     - Suppose the last rule used was E-App1 where $\mathtt{t}_1 \longrightarrow \mathtt{t}_1'$. *Answer: As* $\mathtt{t}$ *is closed, then* $\mathtt{t}_1$ *is also closed. So by induction* $\mathtt{t}_1'$ *is also closed. Therefore, the term* $\mathtt{t}_1'\ \ \mathtt{t}_2$ *is closed.*
     - Suppose the last rule used was E-App2, where $\mathtt{t}_1$ is a value and $\mathtt{t}_2 \longrightarrow \mathtt{t}_2'$. *Answer: As* $\mathtt{t}$ *is closed, then* $\mathtt{t}_2$ *is also closed. So by induction,* $\mathtt{t}_2'$ *is also closed. Therefore the term* $\mathtt{t}_1\ \ \mathtt{t}_2'$ *is closed.*
     - Suppose the last rule used was E-AppAbs, where $\mathtt{t}_1$ is a lambda term $\lambda\mathtt{x}.\ \mathtt{t}_{11}$, $\mathtt{t}_2$ is a value, and $\mathtt{t}'$ is $[\mathtt{x} \mapsto \mathtt{t}_2]\mathtt{t}_{11}$. *Answer: As the application* $\mathtt{t}_1\ \ \mathtt{t}_2$ *is closed, then the subterms* $\lambda\mathtt{x}.\,\mathtt{t}_{11}$ *and* $\mathtt{t}_2$ *are also closed. By the lemma, this substitution is closed.*

Untyped lambda-calculus

9. (9 points) What do the following lambda calculus terms step to, using the *single-step* evaluation relation $\mathtt{t} \longrightarrow \mathtt{t}'$. Write *NONE* if the term does not step. For reference, the semantics of the untyped lambda calculus appears in the appendix of the exam.

   (a) `(λx.x)(λx. x x)(λx. x x)`
   Answer: *(λx. x x) (λx. x x)*

   (b) `(λx. (λx.x) (λx. x x))`
   Answer: *NONE*

   (c) `(λx. (λz. λx. x z) x) (λx. x x)`
   Answer: *(λz. λx. x z)(λx. x x)*

10. (9 points) Now consider the leftmost/outermost evaluation relation from homework 4.

$$\frac{\mathtt{t_1} \longrightarrow \mathtt{t_1}'}{\mathtt{t_1\ t_2} \longrightarrow \mathtt{t_1}'\ \mathtt{t_2}}\ \text{E-App1} \qquad \frac{\lambda\mathtt{x.t_1} \not\longrightarrow}{(\lambda\mathtt{x.t_1})\ \mathtt{t_2} \longrightarrow [\mathtt{x} \mapsto \mathtt{t_2}]\mathtt{t_1}}\ \text{E-App2} \qquad \frac{\mathtt{t_2} \longrightarrow \mathtt{t_2}'}{\mathtt{x\ t_2} \longrightarrow \mathtt{x\ t_2}'}\ \text{E-App3}$$

$$\frac{(\mathtt{s\ t}) \not\longrightarrow \qquad \mathtt{t_2} \longrightarrow \mathtt{t_2}'}{(\mathtt{s\ t})\ \mathtt{t_2} \longrightarrow (\mathtt{s\ t})\ \mathtt{t_2}'}\ \text{E-App4} \qquad \frac{\mathtt{t_1} \longrightarrow \mathtt{t_1}'}{\lambda\mathtt{x.t_1} \longrightarrow \lambda\mathtt{x.t_1'}}\ \text{E-Abs}$$

Using this reduction relation, what do the following terms step to? Again, write *NONE* if the term does not step.

   (a) `(λx.x)(λx. x x)(λx. x x)`
   Answer: *(λx. x x)(λx. x x)*

   (b) `(λx. (λx.x) (λx. x x))`
   Answer: *(λx.(λx. x x))*

   (c) `(λx. (λz. λx. x z) x) (λx. x x)`
   Answer: *(λx. λy. y x)(λx. x x)*

Nameless representation of terms

11. (4 points) Suppose we have defined the naming context $\Gamma =$ `a,b,c,d`. Then the "nameless representation" of the term λ`x.d x (`λ`y.x)` is λ`.1 0 (`λ`.1)`.

    Write down the nameless representation for each of the following terms, in the given naming context.

    (a) λ`x.` λ`y. x c y`

    Answer: λ*. λ. 1 3 0*

    (b) λ`x. b (`λ`y. d x x) d`

    Answer: λ*. 3 (λ. 2 1 1) 1*

12. (4 points) Write down (in de Bruijn notation) the normal form of the following de Bruijn term:

    `(`λ`.` λ`. 1 (`λ`. 1)) (`λ`. 0)`

    Answer: λ*. (λ.0) (λ. 1)*

For reference: Untyped lambda calculus

Syntax

$\texttt{t}$ ::=		*terms*
	$\texttt{x}$	*variable*
	$\lambda\texttt{x.t}$	*abstraction*
	$\texttt{t t}$	*application*
$\texttt{v}$ ::=		*values*
	$\lambda\texttt{x.t}$	*abstraction value*

Evaluation

$$\frac{\texttt{t}_1 \longrightarrow \texttt{t}'_1}{\texttt{t}_1\ \texttt{t}_2 \longrightarrow \texttt{t}'_1\ \texttt{t}_2} \qquad \text{(E-APP1)}$$

$$\frac{\texttt{t}_2 \longrightarrow \texttt{t}'_2}{\texttt{v}_1\ \texttt{t}_2 \longrightarrow \texttt{v}_1\ \texttt{t}'_2} \qquad \text{(E-APP2)}$$

$$(\lambda\texttt{x.t}_{12})\ \texttt{v}_2 \longrightarrow [\texttt{x} \mapsto \texttt{v}_2]\texttt{t}_{12} \qquad \text{(E-APPABS)}$$