

CS421 Summer 2013 Sample Final

The actual midterm will likely have no more than 8 questions. In addition to the kinds of questions asked below, you should expect to see questions similar to those on the MPs and HWs on the exam. Proof rules for type systems, operational semantics, and Hoare logic will be provided to you on the exam; you do not have to memorize them or write them on your index card.

Problem 1. (0 points)

In the following parts you will focus on extending two languages (our in-class functional and imperative languages) with a new programming construct, the trichotomy branch. Syntactically, the extension looks like this:

`trich e -: c1 0: c2 +: c3`

In our functional language the c_i are expressions, but, in the imperative language, they are commands. In both, e is an expression. Informally, this says to examine the value of an integer resulting from e and then executes c_1 if it is negative, c_2 if it is zero, or c_3 if it is positive.

- (a) Give rule(s) that would be appropriate to add to our polymorphic type system to allow for typing programs with the trichotomy switch.

Solution:

$$\frac{\Gamma \vdash e : \text{int} \quad \Gamma \vdash c_1 : \tau \quad \Gamma \vdash c_2 : \tau \quad \Gamma \vdash c_3 : \tau}{\Gamma \vdash \text{trich } e \text{ } -: c_1 \text{ } 0: c_2 \text{ } +: c_3 : \tau} \text{TRICH}$$

- (b) Give big-step semantic rules to describe the behavior of the trichotomy switch. Do not reduce it to an if-then-else command.

Solution:

$$\frac{(e, \rho) \Downarrow n \quad n < 0 \quad (c_1, \rho) \Downarrow \rho'}{(\text{trich } e \text{ } -: c_1 \text{ } 0: c_2 \text{ } +: c_3, \rho) \Downarrow \rho'} \text{TRICHN} \quad \frac{(e, \rho) \Downarrow n \quad n = 0 \quad (c_2, \rho) \Downarrow \rho'}{(\text{trich } e \text{ } -: c_1 \text{ } 0: c_2 \text{ } +: c_3, \rho) \Downarrow \rho'} \text{TRICHZ}$$
$$\frac{(e, \rho) \Downarrow n \quad n > 0 \quad (c_3, \rho) \Downarrow \rho'}{(\text{trich } e \text{ } -: c_1 \text{ } 0: c_2 \text{ } +: c_3, \rho) \Downarrow \rho'} \text{TRICHP}$$

- (c) Give small-step semantic rules to describe the behavior of the trichotomy switch. Do not reduce it to an if-then-else command.

Solution:

$$\begin{array}{c}
 \frac{(e, \rho) \rightarrow (e', \rho)}{(\text{trich } e \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: \rho) \rightarrow (\text{trich } e' \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: c_3, \rho)} \text{TRICHE} \\
 \frac{n < 0}{(\text{trich } n \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: c_3 \rho) \rightarrow (c_1, \rho)} \text{TRICHN} \\
 \frac{n = 0}{(\text{trich } n \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: c_3, \rho) \rightarrow (c_2, \rho)} \text{TRICHZ} \\
 \frac{n > 0}{(\text{trich } n \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: c_3, \rho) \rightarrow (c_3, \rho)} \text{TRICHP}
 \end{array}$$

- (d) Give Hoare logic rule(s) for proofs involving the trichotomy switch.

Solution:

$$\frac{\{P \wedge e < 0\}c_1\{Q\} \quad \{P \wedge e = 0\}c_2\{Q\} \quad \{P \wedge e > 0\}c_3\{Q\}}{\{P\}\text{trich } e \text{ } -: c_1 \text{ } 0 : c_2 \text{ } +: c_3\{Q\}} \text{TRICH}$$

Problem 2. (0 points)

Integers can be represented as natural numbers extended with a predecessor operation in addition to a successor.

- (a) Using this idea, declare an OCaml datatype to represent integers.

Solution: `type integer = Pred of integer | Succ of integer | Zero`

- (b) Write the Church-encoded version of your constructors.

Solution:

$$\begin{aligned}\overline{\text{Pred}} &\equiv \lambda r_1 \ x_1 \ x_2 \ x_3. \ x_1 \ (r_1 \ x_1 \ x_2 \ x_3) \\ \overline{\text{Succ}} &\equiv \lambda r_1 \ x_1 \ x_2 \ x_3. \ x_2 \ (r_1 \ x_1 \ x_2 \ x_3) \\ \overline{\text{Zero}} &\equiv \lambda x_1 \ x_2 \ x_3. \ x_3\end{aligned}$$

- (c) Write the lambda-calculus match function for your datatype.

Solution:

$$\overline{\text{match}_{\text{integer}}} \equiv \lambda x_1 \ x_2 \ x_3 \ e. \ e \ x_1 \ x_2 \ x_3$$

Problem 3. (0 points)

Consider the following datatype

```
type jab = Ber of jab | Wock of jab | Ky
```

- (a) Write the generalized fold function for this type.

Solution:

```
let rec fold ber wock ky v =  
  match v with  
  | Ber j -> ber (fold ber wock ky j)  
  | Wock j -> wock (fold ber wock ky j)  
  | Ky -> ky
```

- (b) Write the type of this function.

Solution: `fold : ('a -> 'a) -> ('a -> 'a) -> 'a -> jab -> 'a`

- (c) Write a CPS version of your fold function.

Solution:

```
let rec foldk ber wock ky v k =  
  match v with  
  | Ber j -> foldk ber wock ky j (fun r -> ber r k)  
  | Wock j -> foldk ber wock ky j (fun r -> wock r k)  
  | Ky -> k ky
```

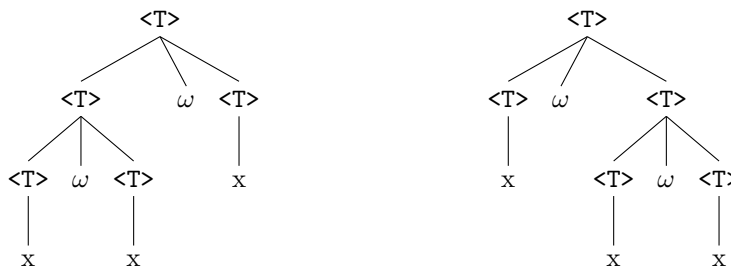
Problem 4. (0 points)

Consider the following grammar:

$$\langle T \rangle ::= \langle T \rangle \omega \langle T \rangle \mid + \langle T \rangle \mid [\langle T \rangle] \mid \{ \langle T \rangle \} \mid x \mid zx$$

- (a) Demonstrate that this grammar is ambiguous.

Solution: The string $x\omega x\omega x$ has the following two parse trees and is thus ambiguous.



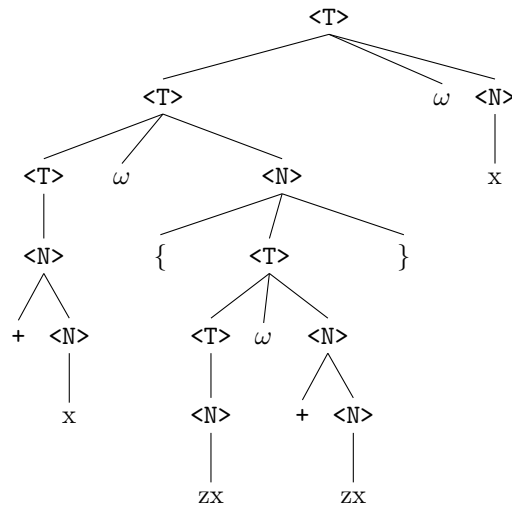
- (b) Disambiguate the grammar. $+$ should bind more tightly than ω and ω should associate to the left.

Solution: $\langle T \rangle ::= \langle T \rangle \omega \langle N \rangle \mid \langle N \rangle$
 $\langle N \rangle ::= x \mid xz \mid +\langle N \rangle \mid [\langle E \rangle] \mid \{\langle E \rangle\}$

- (c) Using your disambiguated grammar give a parse tree for:

$+x\omega\{zx\omega+zx\}\omega x$

Solution:



Problem 5. (0 points)

Reduce the following term, showing intermediate reductions, using the specified scheme

$$((\lambda u. \lambda r. u \ r) (\lambda z. \lambda x. z \ x)) ((\lambda w. w \ w) (\lambda y. y))$$

(a) eager evaluation

Solution:

$$\begin{aligned} & ((\lambda u. \lambda r. u \ r) (\lambda z. \lambda x. z \ x)) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) ((\lambda y. y) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) (\lambda y. y) \\ \equiv_{\beta} & (\lambda z. \lambda x. z \ x) (\lambda y. y) \\ \equiv_{\beta} & \lambda x. (\lambda y. y) \ x \end{aligned}$$

(b) lazy evaluation

Solution:

$$\begin{aligned} & ((\lambda u. \lambda r. u \ r) (\lambda z. \lambda x. z \ x)) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda z. \lambda x. z \ x) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & \lambda x. ((\lambda w. w \ w) (\lambda y. y)) \ x \end{aligned}$$

(c) unrestricted $\alpha\beta$ -reduction

Solution:

$$\begin{aligned} & ((\lambda u. \lambda r. u \ r) (\lambda z. \lambda x. z \ x)) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) ((\lambda w. w \ w) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) ((\lambda y. y) (\lambda y. y)) \\ \equiv_{\beta} & (\lambda r. (\lambda z. \lambda x. z \ x) \ r) (\lambda y. y) \\ \equiv_{\beta} & (\lambda z. \lambda x. z \ x) (\lambda y. y) \\ \equiv_{\beta} & \lambda x. (\lambda y. y) \ x \\ \equiv_{\beta} & \lambda x. x \end{aligned}$$

Problem 6. (0 points)

Assuming that x and y are integer-valued variables, give a proof in Hoare logic of the following triple:

$$\{y = a\} \text{ if } x > y \text{ then } x := x - 1 \text{ else } x := y \text{ fi}; y := x \{y \geq a\}$$

Solution:

$$\frac{\dagger \quad \overline{\{x \geq a\} \ y := x \ \{y \geq a\}} \text{ ASSIGN}}{\{y = a\} \text{ if } x > y \text{ then } x := x - 1 \text{ else } x := y \text{ fi}; y := x \ \{y \geq a\}} \text{ SEQ}$$

where

$$\dagger \equiv \frac{\frac{y = a \wedge \neg x > y \implies x - 1 \geq a \quad \star \text{ PRE}}{\{y = a \wedge x > y\} x := x - 1 \{x \geq a\}} \quad \frac{y = a \wedge \neg x > y \implies y \geq a \quad \diamond \text{ PRE}}{\{y = a \wedge \neg x > y\} x := y \{x \geq a\}} \text{ PRE}}{\{y = a\} \text{ if } x > y \text{ then } x := x - 1 \text{ else } x := y \text{ fi} \ \{x \geq a\}} \text{ IF}$$

$$\star \equiv \overline{\{x - 1 \geq a\} x := x - 1 \{x \geq a\}} \text{ ASSIGN}$$

$$\diamond \equiv \overline{\{y \geq a\} x := y \{x \geq a\}} \text{ ASSIGN}$$