CIS 500
Software Foundations
Fall 2006

November 6

Some Hints

Hints

- The exam will potentially cover everything in the course so far, but will focus on material we've seen since the first midterm.
- There *will* be a question that is also a one-star exercise from the book.
- There *will* be a question similar to problem 6 from midterm 1 ("Which properties remain true if we change one fo the type systems we've studied in the following way...?")
- There *will* be (at least) one question based on one of the proofs in chapter 15.
- For PhD students, there *will* be a question involving subtyping and references.

Review

What are the types of these expressions?

- λx:Bool→Bool. x (x (x (true)))

What are the types of these expressions?

- λx:Bool→Bool. x (x (x (true)))
- (λx:Bool. λy:Bool→Bool. true) false (λz:Bool→Bool. true)

What are the types of these expressions?

- `λx:Bool→Bool. x (x (x (true)))`
- `(λx:Bool. λy:Bool→Bool. true)`
  `false (λz:Bool→Bool. true)`
- `(λx:Bool. λy:Bool. error)`
  `false false false false false`

What are the types of these expressions?

- `λx:Bool→Bool. x (x (x (true)))`
- `(λx:Bool. λy:Bool→Bool. true)`
  `false (λz:Bool→Bool. true)`
- `(λx:Bool. λy:Bool. error)`
  `false false false false false`
- `(λx:Bool. λy:Bool. true)`
  `false false false false false`

What are the types of these expressions?

- `λx:Bool→Bool. x (x (x (true)))`
- `(λx:Bool. λy:Bool→Bool. true)`
  `false (λz:Bool→Bool. true)`
- `(λx:Bool. λy:Bool. error)`
  `false false false false false`
- `(λx:Bool. λy:Bool. true)`
  `false false false false false`
- ```
  try
    (if (λx:Bool. x) error
       then (error false)
       else error)
  with
    λy:Bool→Bool. y
  ```

For reference: Typing rules for exceptions

$$\Gamma \vdash \texttt{error} : \texttt{T} \qquad \text{(T-Error)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{T} \qquad \Gamma \vdash \texttt{t}_2 : \texttt{T}}{\Gamma \vdash \texttt{try } \texttt{t}_1 \texttt{ with } \texttt{t}_2 : \texttt{T}} \qquad \text{(T-Try)}$$

Give the result of evaluation and the final store after each of these expressions is evaluated to a normal form starting in the empty store.

- ```
  let x = ref 0 in
  let y = ref 1 in
  y:=3
  !x
  ```

Give the result of evaluation and the final store after each of these expressions is evaluated to a normal form starting in the empty store.

- ```
  let x = ref 0 in
  let y = ref 1 in
  y:=3
  !x
  ```
- ```
  let x = ref 0 in
  let y = ref 1 in
  let x = y in !x
  ```

Give the result of evaluation and the final store after each of these expressions is evaluated to a normal form starting in the empty store.

- ```
  let x = ref 0 in
  let y = ref 1 in
  y:=3
  !x
  ```
- ```
  let x = ref 0 in
  let y = ref 1 in
  let x = y in !x
  ```
- ```
  let x = ref 0 in
  let y = x in
  let x = y in !x
  ```

For reference: Evaluation rules for references

$$\frac{l \notin dom(\mu)}{\texttt{ref } v_1 \mid \mu \longrightarrow l \mid (\mu, l \mapsto v_1)} \quad \text{(E-RefV)}$$

$$\frac{\mu(l) = v}{!l \mid \mu \longrightarrow v \mid \mu} \quad \text{(E-DerefLoc)}$$

$$l{:=}v_2 \mid \mu \longrightarrow \texttt{unit} \mid [l \mapsto v_2]\mu \quad \text{(E-Assign)}$$

(Plus several congruence rules.)

Which of the following functions *could* evaluate to 42 when applied to a single argument and evaluated with a store of the appropriate type?

- λx:Ref Nat. !x+1

Which of the following functions *could* evaluate to 42 when applied to a single argument and evaluated with a store of the appropriate type?

- λx:Ref Nat. !x+1
- λx:Ref Nat. x

Which of the following functions *could* evaluate to 42 when applied to a single argument and evaluated with a store of the appropriate type?

- λx:Ref Nat. !x+1
- λx:Ref Nat. x
- λx:Ref Nat. (x:=3; l_1:=42; !l_1)

Which of the following functions *could* evaluate to 42 when applied to a single argument and evaluated with a store of the appropriate type?

- λx:Ref Nat. !x+1
- λx:Ref Nat. x
- λx:Ref Nat. (x:=3; l_1:=42; !l_1)
- λf:Unit$\rightarrow$Unit. (l_1:=3; f unit; !l_1)
  ```

Preservation and progress for chapter 13

- The preservation and progress proofs for $\lambda_{\to}$ with references are just sketched in TAPL.
- Working out the details for yourself is an excellent exercise
- A question based on this proof may appear on the final exam, but will *not* appear on the coming midterm

Subtyping

For each of the following pairs of terms, say whether the one on the left is a subtype of the one on the right, a supertype, equivalent, or incomparable.

- ({}→{})→Top and Top→Top

Subtyping

For each of the following pairs of terms, say whether the one on the left is a subtype of the one on the right, a supertype, equivalent, or incomparable.

- ({}→{})→Top and Top→Top

- (Top→Top)→{}→{} and (Top→{})→Top

Subtyping

For each of the following pairs of terms, say whether the one on the left is a subtype of the one on the right, a supertype, equivalent, or incomparable.

- ({}→{})→Top and Top→Top
- (Top→Top)→{}→{} and (Top→{})→Top

- {a:Top, b:{c:Top,d:Top}} and {b:{d:Top,c:Top},a:Top}

Subtyping

For each of the following pairs of terms, say whether the one on the left is a subtype of the one on the right, a supertype, equivalent, or incomparable.

- ({}→{})→Top and Top→Top
- (Top→Top)→{}→{} and (Top→{})→Top
- {a:Top, b:{c:Top,d:Top}} and {b:{d:Top,c:Top},a:Top}

- <l:Top,m:{n:Top}> and <m:{n:Top,o:Top}>

Subtyping

For each of the following pairs of terms, say whether the one on the left is a subtype of the one on the right, a supertype, equivalent, or incomparable.

- ({}→{})→Top and Top→Top
- (Top→Top)→{}→{} and (Top→{})→Top
- {a:Top, b:{c:Top,d:Top}} and {b:{d:Top,c:Top},a:Top}
- <l:Top,m:{n:Top}> and <m:{n:Top,o:Top}>

- <>→Top and {}→Top

Subtyping

Draw a subtyping derivation for the following statement:

$$\texttt{(Top}\to\texttt{\{x:Nat\})}\to\texttt{\{x:Nat,y:Nat\}} \quad <: \quad \texttt{(\{\}}\to\texttt{\{\})}\to\texttt{\{y:Nat\}}$$

For reference: Subtyping rules

$$\texttt{S} <: \texttt{S} \qquad \text{(S-Refl)}$$

$$\frac{\texttt{S} <: \texttt{U} \qquad \texttt{U} <: \texttt{T}}{\texttt{S} <: \texttt{T}} \qquad \text{(S-Trans)}$$

$$\{\texttt{l}_i\texttt{:T}_i\ ^{i\in 1..n+k}\} <: \{\texttt{l}_i\texttt{:T}_i\ ^{i\in 1..n}\} \qquad \text{(S-RcdWidth)}$$

$$\frac{\text{for each } i \quad \texttt{S}_i <: \texttt{T}_i}{\{\texttt{l}_i\texttt{:S}_i\ ^{i\in 1..n}\} <: \{\texttt{l}_i\texttt{:T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdDepth)}$$

$$\frac{\{\texttt{k}_j\texttt{:S}_j\ ^{j\in 1..n}\} \text{ is a permutation of } \{\texttt{l}_i\texttt{:T}_i\ ^{i\in 1..n}\}}{\{\texttt{k}_j\texttt{:S}_j\ ^{j\in 1..n}\} <: \{\texttt{l}_i\texttt{:T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdPerm)}$$

$$\frac{\texttt{T}_1 <: \texttt{S}_1 \qquad \texttt{S}_2 <: \texttt{T}_2}{\texttt{S}_1\to\texttt{S}_2 <: \texttt{T}_1\to\texttt{T}_2} \qquad \text{(S-Arrow)}$$

$$\texttt{S} <: \texttt{Top} \qquad \text{(S-Top)}$$

Ascription as a derived form

- Someone asked to work exercise 11.4.1 part 2 today.
- But the solution is somewhat technical and would take too much time to discuss in detail.
- This exercise is not needed for the exam.

The Hints, again

- The exam will potentially cover everything in the course so far, but will focus on material we've seen since the first midterm.
- There *will* be a question that is also a one-star exercise from the book.
- There *will* be a question similar to problem 6 from midterm 1 ("Which properties remain true if we change one fo the type systems we've studied in the following way...?")
- There *will* be (at least) one question based on one of the proofs in chapter 15.
- For PhD students, there *will* be a question involving subtyping and references.