CS 320
Computer Language Processing
Review Exercises

April 02, 2025

2016 **Exercise 1** Let $\Sigma = \{a, b\}$ for distinct a, b. Let L_1, L_2, L range over subsets of Σ^* (languages). Remember that for languages, concatenation is given by:

$$L_1 L_1 = \{u_1 u_2 \mid u_1 \in L_1 \wedge u_2 \in L_2\}$$

We that a language L left-cancels if and only if for every L_1, L_2:

$$LL_1 = LL_2 \implies L_1 = L_2$$

1. Does $L = $ left-cancel?

2. Does $L = \epsilon$ left-cancel?

3. Give a regular expression describing an infinite language L that left-cancels.

4. Give a context-free grammar for another language L that left-cancels, but is not regular.

2016 **Exercise 2** Consider the grammar:

$$
\begin{aligned}
decl &::= varDecl \mid funDecl \\
varDecl &::= type \; \texttt{ID}; \\
funDecl &::= type \; \texttt{ID} \; (optIDs); \\
optIDs &::= \epsilon \mid IDs \\
IDs &::= \texttt{ID} \mid IDs \; \texttt{ID} \\
type &::= \texttt{int} \mid type\texttt{*}
\end{aligned}
$$

1. Compute nullable and first for each non-terminal of the grammar above.

2. Explain why the grammar is not LL(1).

3. Give an LL(1) grammar describing the same sequences of tokens as the previous grammar.

2022 **Exercise 3** Consider the following grammar with non-terminals S and A and terminals **EOF**, $(,), [,$ and $]$:

$$S ::= A \textbf{ EOF}$$
$$A ::= (A)\ A \mid A\ [A] \mid \epsilon$$

1. Choose all true statements about the grammar above:

   (a) "$[]()([])$" is accepted by the grammar.
   (b) The grammar is LL(1).
   (c) The grammar is ambiguous.
   (d) nullable$(A) = true$.
   (e) nullable$(S) = true$.

2. Choose the correct option:

   (a) first$(S) = \{\textbf{EOF}\}$
   (b) first$(S) = \{(, [\}$
   (c) first$(S) = \{(,), \textbf{EOF}\}$
   (d) first$(S) = \{(, [, \textbf{EOF}\}$
   (e) first$(S) = \{(,), [,], \textbf{EOF}\}$

3. Choose the correct option:

   (a) follow$(A) = \{),]\}$
   (b) follow$(A) = \{),], \textbf{EOF}\}$
   (c) follow$(A) = \{(, [,),]\}$
   (d) follow$(A) = \{(, [,], \textbf{EOF}\}$
   (e) follow$(A) = \{(, [,),], \textbf{EOF}\}$

Exercise 4 Complete 2022 (on the next page) the type derivation for the body of the function `f`.

```scala
def f(x: Int, u: Int, v: Int): Int = {
  if (x < u) {
    u
  }
  else if (v < x) {
    v
  }
  else {
    x
  }
}
```

You may use the following type rules:

$$\frac{(x, T) \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\Gamma \vdash e_1 : Int \qquad \Gamma \vdash e_2 : Int}{\Gamma \vdash e_1 + e_2 : Int} \qquad \frac{\Gamma \vdash e_1 : Int \qquad \Gamma \vdash e_2 : Int}{\Gamma \vdash e_1 * e_2 : Int}$$

$$\frac{\Gamma \vdash e_1 : Bool \qquad \Gamma \vdash e_2 : Bool}{\Gamma \vdash e_1 \&\& e_2 : Bool} \qquad \frac{\Gamma \vdash e_1 : Bool \qquad \Gamma \vdash e_2 : Bool}{\Gamma \vdash e_1 || e_2 : Bool}$$

$$\frac{\Gamma \vdash e_1 : Int \qquad \Gamma \vdash e_2 : Int}{\Gamma \vdash e_1 < e_2 : Bool} \qquad \frac{\Gamma \vdash e_1 : Bool \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash if\ e_1\ then\ e_2\ else\ e_3 : T}$$

$$
\cfrac{
\cfrac{(x,Int)\in\Gamma}{\Gamma\vdash x:Int} \qquad
\cfrac{(u,Int)\in\Gamma}{\Gamma\vdash u:Int}
}{\Gamma\vdash x<u:Bool}
\qquad
\cfrac{(u,Int)\in\Gamma}{\Gamma\vdash u:Int}
\qquad
\cfrac{
\cfrac{
\cfrac{(\,,\ )\in\Gamma}{\Gamma\vdash\ :\ } \quad
\cfrac{(\,,\ )\in\Gamma}{\Gamma\vdash\ :\ }
}{\Gamma\vdash} \quad
\cfrac{(\,,\ )\in\Gamma}{\Gamma\vdash\ :\ } \quad
\cfrac{(\,,\ )\in\Gamma}{\Gamma\vdash\ :\ }
}{\Gamma\vdash\ :\ }
$$

$$\Gamma\vdash if\ (x<u)\ then\ u\ else\ if\ (v<x)\ v\ else\ x : Int$$

4

2022, contd   **Exercise 5**   For which of the following expressions does type unification succeed? For the $+$ operator, assume the type rules as in the previous question.

1. $x \Rightarrow y \Rightarrow y(z \Rightarrow 6) + y(7)$

2. $g \Rightarrow f \Rightarrow x \Rightarrow g(f(x))$

3. $x \Rightarrow y \Rightarrow ((z \Rightarrow y), y)$

4. $g \Rightarrow f \Rightarrow x \Rightarrow g(f(x)) + f(g(x)) + x$

2022, contd   **Exercise 6**   Consider a programming language with pairs and the usual typing rules, as in the lecture. Apply the unification algorithm on the following function:

```
def swap(t) = {
  (t._2, t._1)
}
```

assuming the following type variables assigned to tree nodes:

$$((t : \tau)._2 : \tau_1, (t : \tau)._1 : \tau_2) : \tau_3$$

Write each step of the unification algorithm, mentioning what rules of the algorithm you are applying. We provide you with the initial step:

$$\tau = (\tau_{10}, \tau_1)$$
$$\tau = (\tau_2, \tau_{20})$$
$$\tau_3 = (\tau_1, \tau_2)$$

Write down an expression for the argument and return types of `swap` in terms of the type variables τ_1 and τ_2.