

# CS421 Summer 2006 Final Study Guide

CS 421 — Programming Languages and Compilers  
Summer Term 2006

## 1. Type Derivations:

- (a) Provide a full type derivation for the following term. Feel free to write on the paper sideways – the trees tend to be wider than they are tall. You can also write out type environments separately and just reference them in the rules (so you could say  $\Gamma_1 = \{x : \text{int}\}$  and then just reference  $\Gamma_1$  in the derivation) – this is especially useful if you use the same environment in multiple places. Typing rules are on page 11.

```
let f = fun a -> (a + 9) in (f 4)
```

- (b) Provide a full type derivation for the following term. Feel free to write on the paper sideways – the trees tend to be wider than they are tall. You can also write out type environments separately and just reference them in the rules (so you could say  $\Gamma_1 = \{x : \text{int}\}$  and then just reference  $\Gamma_1$  in the derivation) – this is especially useful if you use the same environment in multiple places. Typing rules are on page 11.

```
let rec f = fun a -> (a < 9) in (if (f 4) then 3 else 5)
```

**2. Lambda Calculus**

(a) Reduce the following term using lazy evaluation.

$$(\lambda a b. a b)(\lambda x y. x)((\lambda t. t)(\lambda u. u))(\lambda c. c c c)$$

(b) Reduce the same term using eager evaluation.

- (c) Recall our definition of pairs in the  $\lambda$  calculus – a pair can be represented as a term  $\lambda x.x \mathbf{a} \mathbf{b}$  where  $x$  is the constructor tag (here we only have one constructor, so we only need one tag) and  $\mathbf{a}$  and  $\mathbf{b}$  are the two elements in the pair. Provide a  $\lambda$  abstraction `swappair` that takes a pair and returns a new pair with the element swapped: applying this to pair  $(a, b)$  will return pair  $(b, a)$ .

3. **Context-Free Grammars, Derivations, and Parse Trees:** The following exercises make use of this grammar for arithmetic expressions. Here, the start symbol is  $E$ ,  $\text{id}$  refers to identifiers made up of one lowercase character ( $\mathbf{a, b, \dots, y, z}$ ), and  $\text{num}$  refers to any integer:

$$E \rightarrow E + T$$

$$T \rightarrow T * F$$

$$F \rightarrow \text{id}$$

$$E \rightarrow E - T$$

$$T \rightarrow T / F$$

$$F \rightarrow \text{num}$$

$$E \rightarrow T$$

$$T \rightarrow F$$

$$F \rightarrow (E)$$

- (a) Show a leftmost derivation for the following term:

$$y/3$$

- (b) Show a rightmost derivation for the same term:

- (c) Provide a parse tree for the same term:

$$E \rightarrow E + T$$

$$E \rightarrow E - T$$

$$E \rightarrow T$$

$$T \rightarrow T * F$$

$$T \rightarrow T / F$$

$$T \rightarrow F$$

$$F \rightarrow \mathbf{id}$$

$$F \rightarrow \mathbf{num}$$

$$F \rightarrow (E)$$

(d) Show a leftmost derivation for the following term:

$$y/(4 + 3)$$

(e) Show a rightmost derivation for the same term:

(f) Provide a parse tree for the same term:

**4. Semantics:**

- (a) Derivations: Using natural semantics (rules are on pages 14–15), provide a derivation for the following term:

$$\text{if } (i \leq n) \text{ then } (n := n + 1) \text{ else } (i := i + 1)$$

Assume the starting  $\sigma = \{i \mapsto 1, n \mapsto 3\}$ . Feel free to turn the page sideways, or to number parts of the derivation and provide them separately (so you can say #1 somewhere in the derivation and then elsewhere provide the definition for the #1 part of the derivation tree).

- (b) Derivations: Using transition semantics (rules are on pages 12–13), provide a derivation for the following term:

`if (i ≤ n) then (n := n + 1) else (i := i + 1)`

Assume the starting  $\sigma = \{i \mapsto 1, n \mapsto 3\}$ . Feel free to turn the page sideways, or to number parts of the derivation and provide them separately (so you can say #1 somewhere in the derivation and then elsewhere provide the definition for the #1 part of the derivation tree).

- (c) Semantics Rules: Say you are given a new language construct for a **for** statement. This adds the following to the grammar (remember, *Com* is a command, something that can change the state):

$$\text{Com } c ::= \mathbf{for } X := a_0 \mathbf{to } a_1 \mathbf{do } c_0 \mathbf{done}$$

This is similar to a **while** statement, but the iteration space is constrained by the initial values specified in the loop and a new name ( $X$ ) is introduced into scope which can be referenced in the loop body ( $c_0$ ). The semantics should work as follows:

- i. Expression  $a_0$  is evaluated until it yields a number  $n_0$
- ii. Expression  $a_1$  is evaluated until it yields a number  $n_1$
- iii. Name  $X$  is added to the environment and assigned the value  $n_0$
- iv. While the numeric value assigned to  $X$  is less than or equal to  $n_1$ , execute command  $c_0$  and then add one to  $X$

The name  $X$  should not be available once the loop ends. Define the necessary **transition** semantics rules for this construct. You can reference the transition semantics rules for the IMP language, on pages 12–13, for guidance on how the rules should look.

- (d) Semantics Rules: Define the necessary **natural** semantics rules for the **for** construct defined above. You can reference the natural semantics rules for the IMP language, on pages 14–15, for guidance on how the rules should look.

**Rules for type derivations:****Constants**

$$\frac{}{\vdash n : \text{int}} \text{(assuming } n \text{ is an int)}$$

$$\frac{}{\vdash \text{true} : \text{bool}}$$

$$\frac{}{\vdash \text{false} : \text{bool}}$$
**Variables**

$$\frac{}{\Gamma \vdash x : \tau} \text{if } (x : \tau) \in \Gamma$$
**Arithmetic Operators**

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 \oplus e_2 : \text{int}} \quad (\oplus \in \{+, -, *, /, \dots\})$$
**Relational Operators**

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 \sim e_2 : \text{bool}} \quad (\sim \in \{<, >, \leq, \geq, =, \neq, \dots\})$$
**Booleans**

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash e_1 \ \&\& \ e_2 : \text{bool}}$$

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash e_1 \ || \ e_2 : \text{bool}}$$

$$\frac{\Gamma \vdash e_1 : \text{bool}}{\Gamma \vdash ! e_1 : \text{bool}}$$
**If**

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \text{if } e_1 \ \text{then } e_2 \ \text{else } e_3 : \tau}$$
**Function Abstraction**

$$\frac{\Gamma \cup [x : \tau_1] \vdash e : \tau_2}{\Gamma \vdash \text{fun } x \rightarrow e : \tau_1 \rightarrow \tau_2}$$
**Function Application**

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2}$$
**Let**

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \cup [x : \tau] \vdash e_2 : \tau'}{\Gamma \vdash \text{let } x = e_1 \ \text{in } e_2 : \tau'}$$
**Letrec**

$$\frac{\Gamma \cup [x : \tau] \vdash e_1 : \tau \quad \Gamma \cup [x : \tau] \vdash e_2 : \tau'}{\Gamma \vdash \text{let rec } x = e_1 \ \text{in } e_2 : \tau'}$$

**Transition Semantics, Dynamic Semantics Rules:**

<i>Aexp</i>	$a ::=$	$n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$
<i>Bexp</i>	$b ::=$	<b>true</b> $\mid$ <b>false</b> $\mid$ $a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$
<i>Com</i>	$c ::=$	<b>skip</b> $\mid$ $X := a \mid c_0; c_1 \mid$ <b>if</b> $b$ <b>then</b> $c_0$ <b>else</b> $c_1 \mid$ <b>while</b> $b$ <b>do</b> $c$

**Identifiers**

$$\langle X, \sigma \rangle \rightarrow \sigma(X)$$

**Constants**

$$\langle n, \sigma \rangle \rightarrow n$$

$$\langle \mathbf{true}, \sigma \rangle \rightarrow \mathbf{true}$$

$$\langle \mathbf{false}, \sigma \rangle \rightarrow \mathbf{false}$$

**Arithmetic Operators,  $op \in \{+, -, \times\}$** 

$$\frac{\langle a_0, \sigma \rangle \rightarrow \langle a'_0, \sigma \rangle}{\langle a_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle a'_0 \text{ op } a_1, \sigma \rangle}$$

$$\langle a_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle a'_0 \text{ op } a_1, \sigma \rangle$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow \langle a'_1, \sigma \rangle}{\langle n_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle n_0 \text{ op } a'_1, \sigma \rangle}$$

$$\langle n_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle n_0 \text{ op } a'_1, \sigma \rangle$$

$$\langle n_0 \text{ op } n_1, \sigma \rangle \rightarrow n, \text{ where } n = n_0 \text{ op}_{int} n_1$$

**Relational Operators,  $op \in \{=, \leq\}$** 

$$\frac{\langle a_0, \sigma \rangle \rightarrow \langle a'_0, \sigma \rangle}{\langle a_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle a'_0 \text{ op } a_1, \sigma \rangle}$$

$$\langle a_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle a'_0 \text{ op } a_1, \sigma \rangle$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow \langle a'_1, \sigma \rangle}{\langle n_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle n_0 \text{ op } a'_1, \sigma \rangle}$$

$$\langle n_0 \text{ op } a_1, \sigma \rangle \rightarrow \langle n_0 \text{ op } a'_1, \sigma \rangle$$

$$\langle n_0 \text{ op } n_1, \sigma \rangle \rightarrow b, \text{ where } b = (n_0 \text{ op}_{int} n_1)$$

**Logical Operators ( $\wedge, \vee$  and  $\neg$  are similar)**

$$\frac{\langle b_0, \sigma \rangle \rightarrow \langle b'_0, \sigma \rangle}{\langle b_0 \wedge b_1, \sigma \rangle \rightarrow \langle b'_0 \wedge b_1, \sigma \rangle}$$

$$\langle b_0 \wedge b_1, \sigma \rangle \rightarrow \langle b'_0 \wedge b_1, \sigma \rangle$$

$$\frac{\langle b_1, \sigma \rangle \rightarrow \langle b'_1, \sigma \rangle}{\langle \mathbf{true} \wedge b_1, \sigma \rangle \rightarrow \langle b'_1, \sigma \rangle}$$

$$\langle \mathbf{true} \wedge b_1, \sigma \rangle \rightarrow \langle b'_1, \sigma \rangle$$

$$\langle \mathbf{false} \wedge b_1, \sigma \rangle \rightarrow \mathbf{false}$$

**Skip**

$$\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma$$

## Assignment

$$\frac{\langle a, \sigma \rangle \rightarrow \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \rightarrow \langle X := a', \sigma \rangle}$$

$$\langle X := n, \sigma \rangle \rightarrow \sigma[n/X]$$

## Sequencing

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle c'_0, \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \rightarrow \langle c'_0; c_1, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \langle c_1, \sigma' \rangle}$$

## If

$$\frac{\langle b, \sigma \rangle \rightarrow \langle b', \sigma \rangle}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma \rangle \rightarrow \langle \text{if } b' \text{ then } c_0 \text{ else } c_1, \sigma \rangle}$$

$$\langle \text{if true then } c_0 \text{ else } c_1, \sigma \rangle \rightarrow \langle c_0, \sigma \rangle$$

$$\langle \text{if false then } c_0 \text{ else } c_1, \sigma \rangle \rightarrow \langle c_1, \sigma \rangle$$

## While

$$\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \langle \text{if } b \text{ then } (c; \text{while } b \text{ do } c) \text{ else skip}, \sigma \rangle$$

**Natural Semantics, Dynamic Semantics Rules:**

<i>Aexp</i>	$a ::=$	$n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$
<i>Bexp</i>	$b ::=$	$\mathbf{true} \mid \mathbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$
<i>Com</i>	$c ::=$	$\mathbf{skip} \mid X := a \mid c_0; c_1 \mid \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1 \mid$ $\mathbf{while } b \mathbf{ do } c$

**Identifiers**

$$\langle X, \sigma \rangle \Downarrow \sigma(X)$$

**Constants**

$$\langle n, \sigma \rangle \Downarrow n$$

$$\langle \mathbf{true}, \sigma \rangle \Downarrow \mathbf{true}$$

$$\langle \mathbf{false}, \sigma \rangle \Downarrow \mathbf{false}$$

**Arithmetic Operators**

$$\frac{\langle a_0, \sigma \rangle \Downarrow n_0 \quad \langle a_1, \sigma \rangle \Downarrow n_1 \quad n = n_0 \text{ op}_{int} n_1}{\langle a_0 \text{ op } a_1, \sigma \rangle \Downarrow n} \quad (op \in \{+, -, \times\})$$

**Relational Operators**

$$\frac{\langle a_0, \sigma \rangle \Downarrow n_0 \quad \langle a_1, \sigma \rangle \Downarrow n_1 \quad b = n_0 \text{ op}_{int} n_1}{\langle a_0 \text{ op } a_1, \sigma \rangle \Downarrow b} \quad (op \in \{=, \leq\})$$

**Logical Operators**

$$\frac{\langle b_0, \sigma \rangle \Downarrow \mathbf{false}}{\langle b_0 \wedge b_1, \sigma \rangle \Downarrow \mathbf{false}}$$

$$\langle b_0 \wedge b_1, \sigma \rangle \Downarrow \mathbf{false}$$

$$\frac{\langle b_0, \sigma \rangle \Downarrow \mathbf{true} \quad \langle b_1, \sigma \rangle \Downarrow b}{\langle b_0 \wedge b_1, \sigma \rangle \Downarrow b}$$

$$\langle b_0 \wedge b_1, \sigma \rangle \Downarrow b$$

$$\frac{\langle b_0, \sigma \rangle \Downarrow \mathbf{true}}{\langle b_0 \vee b_1, \sigma \rangle \Downarrow \mathbf{true}}$$

$$\langle b_0 \vee b_1, \sigma \rangle \Downarrow \mathbf{true}$$

$$\frac{\langle b_0, \sigma \rangle \Downarrow \mathbf{false} \quad \langle b_1, \sigma \rangle \Downarrow b}{\langle b_0 \vee b_1, \sigma \rangle \Downarrow b}$$

$$\langle b_0 \vee b_1, \sigma \rangle \Downarrow b$$

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{false}}{\langle \neg b, \sigma \rangle \Downarrow \mathbf{true}}$$

$$\langle \neg b, \sigma \rangle \Downarrow \mathbf{true}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{true}}{\langle \neg b, \sigma \rangle \Downarrow \mathbf{false}}$$

$$\langle \neg b, \sigma \rangle \Downarrow \mathbf{false}$$

**Skip**

$$\{\mathbf{skip}, \sigma\} \Downarrow \sigma$$

Assignment
------------

$$\frac{\langle a, \sigma \rangle \Downarrow n}{\langle X := a, \sigma \rangle \Downarrow \sigma[n/X]}$$

Sequencing
------------

$$\frac{\langle c_0, \sigma \rangle \Downarrow \sigma' \quad \langle c_1, \sigma' \rangle \Downarrow \sigma''}{\langle c_0; c_1, \sigma \rangle \Downarrow \sigma''}$$

If
----

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{true} \quad \langle c_0, \sigma \rangle \Downarrow \sigma'}{\langle \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1, \sigma \rangle \Downarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{false} \quad \langle c_1, \sigma \rangle \Downarrow \sigma'}{\langle \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1, \sigma \rangle \Downarrow \sigma'}$$

While
-------

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{false}}{\langle \mathbf{while } b \mathbf{ do } c, \sigma \rangle \Downarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{true} \quad \langle c, \sigma \rangle \Downarrow \sigma' \quad \langle \mathbf{while } b \mathbf{ do } c, \sigma' \rangle \Downarrow \sigma''}{\langle \mathbf{while } b \mathbf{ do } c, \sigma \rangle \Downarrow \sigma''}$$