[S25 e2]

$$
\begin{aligned}
S &-> SvS \mid SrS \mid T \\
T &-> cT \mid Td \mid D \\
D &-> d \mid fSf
\end{aligned}
$$

(a) Derive `dvddrd` using a **leftmost** derivation (do not draw a tree). If it is not possible, derive `fdvdf` using a **rightmost** derivation [8 pts]

$S \rightarrow SvS \rightarrow TvS \rightarrow DvS \rightarrow dvS \rightarrow$
$dvSrS \rightarrow dvTrS \rightarrow dvTdrS \rightarrow dvddrS \rightarrow$
$dvddrT \rightarrow dvddrD \rightarrow dvddrd$

(b) Is this grammar ambiguous? [2 pts]

Y Yes N No

[F18 e2]

1. [6 pts] Write a CFG that is equivalent to the regular expression $(wp)^+g^*$

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow wpA \mid wp \\
B &\rightarrow Bg \mid \epsilon
\end{aligned}
$$

[S25 f]

Which Context Free Grammar(s) are equivalent to the regex: `c*(ab)+d`?

Select all that apply.

- $U -> MDC$; $D -> abD \mid ab$; $M -> cM \mid \epsilon$; $C -> d \mid \epsilon$
- A: $U -> MUD \mid abU$; $M -> cM \mid \epsilon$; $D -> d \mid \epsilon$
- D: $U -> MUD \mid abU$; $M -> c \mid M \mid \epsilon$; $D -> d \mid \epsilon$
- C: $U -> MDC$; $D -> abD \mid ab$; $M -> cM \mid \epsilon$; $C -> dC \mid \epsilon$

[S18 e2]

B. [11 pts] Consider the following CFG, in which **p** and **q** are terminals, and A and B are nonterminals.

a. [4 pts] Which of the following strings are accepted? Circle them.

A -> **pAq** | B
B -> **pB** | B**q** | **pq**

Circle: **pppqqq** **pqpq** **pppq** **p**

b. [3 pts] Give a regular expression that accepts the same strings as the CFG. If this is not possible, explain why.

c. [4 pts] Show that the CFG is ambiguous.

[S18 q3]

$s^y e^z$, where $z = 2y + 1$ and $y \geq 0$

$T \rightarrow sTee \mid e$

[F18 e2]

2. [6 pts] Create a CFG that generates all strings of the form $a^x b^y a^z$, where $y = x + z$ and $x, y, z \geq 0$.

$S \rightarrow TR$

$T \rightarrow aTb \mid \epsilon$ x

$R \rightarrow bTa \mid \epsilon$ z

$$\dfrac{\dfrac{}{A;\,3 \Rightarrow 3} \qquad \dfrac{\dfrac{}{A;\,6 \Rightarrow 6} \quad \dfrac{}{A;\,4 \Rightarrow 4} \quad 10 \text{ is } 6+4}{A;\,\mathrm{op2}\ 6\ 4 \Rightarrow 10} \qquad 30 \text{ is } 3*10}{\mathrm{op1}\ 3\ \mathrm{op2}\ 6\ 4 \Rightarrow 30}$$

$$\dfrac{A;\,3 \Rightarrow 3 \qquad \dfrac{A;\,6 \Rightarrow 6 \quad A;\,4 \Rightarrow 4 \quad 10 \text{ is } 6+4}{A;\,6\ 4\ \mathrm{op4} \Rightarrow 10} \qquad \dfrac{}{30 \text{ is } 3*10}}{A;\,(6\ 4\ \mathrm{op4}\ 3\ \mathrm{op3}) \Rightarrow 30}$$

$$\dfrac{\dfrac{}{G \vdash \mathrm{true} : \mathrm{bool}} \qquad \dfrac{\dfrac{G,\,x:\mathrm{bool}\ (x) = \mathrm{bool}}{G,\,x:\mathrm{bool} \vdash x : \mathrm{bool}} \quad \dfrac{}{G,\,x:\mathrm{bool} \vdash \mathrm{false} : \mathrm{bool}}}{G,\,x:\mathrm{bool} \vdash x \mathrel{<>} \mathrm{false} : \mathrm{bool}}}{G \vdash \mathrm{let}\ x = \mathrm{true}\ \mathrm{in}\ x \mathrel{<>} \mathrm{false} : \mathrm{bool}}$$

Type Checking

$$\frac{}{G \vdash \text{true} : \mathit{bool}} \qquad \frac{}{G \vdash \text{false} : \mathit{bool}} \qquad \frac{}{G \vdash n : \mathit{int}}$$

$$\frac{G \vdash e_1 : \mathit{int} \qquad G \vdash e_2 : \mathit{int}}{G \vdash e_1 + e_2 : \mathit{int}}$$

$$\frac{G \vdash e_1 : \mathit{bool} \qquad G \vdash e_2 : \mathit{bool}}{G \vdash e_1 \;\&\&\; e_2 : \mathit{bool}}$$

$$\frac{G \vdash e_1 : \mathit{bool} \qquad G \vdash e_2 : t \qquad G \vdash e_3 : t}{G \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t}$$

$$\frac{G, x{:}t_1 \vdash e : t_2}{G \vdash \text{fun } (x : t_1) \to e : t_1 \to t_2}$$

$$\frac{}{G \vdash x : G(x)}$$

$$\frac{G \vdash e_1 : t_1 \qquad G, x{:}t_1 \vdash e_2 : t_2}{G \vdash \text{let } x = e_1 \text{ in } e_2 : t_2}$$

Example 1a: Use type checking to prove that this is well-typed:

```
let a = true in a && false
```

Example 1b: Use type checking to prove that this is well-typed:

```
fun (z:int) -> z + 1
```

\

Type Inference

$$\frac{}{G \vdash \text{true} : \mathit{bool} \dashv \{\}} \qquad \frac{}{G \vdash \text{false} : \mathit{bool} \dashv \{\}} \qquad \frac{}{G \vdash n : \mathit{int} \dashv \{\}}$$

$$\frac{G \vdash e_1 : t_1 \dashv C_1 \qquad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1 + e_2 : \mathit{int} \dashv \{t_1{:}t_2,\, t_1{:}\mathit{int}\} \cup C_1 \cup C_2}$$

$$\frac{G \vdash e_1 : t_1 \dashv C_1 \qquad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1 \mathbin{\&\&} e_2 : \mathit{bool} \dashv \{t_1{:}t_2,\, t_1{:}\mathit{bool}\} \cup C_1 \cup C_2}$$

$$\frac{G \vdash e_1 : t_1 \dashv C_1 \qquad G \vdash e_2 : t_2 \dashv C_2 \qquad G \vdash e_3 : t_3 \dashv C_3}{G \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t_2 \dashv \{t_1{:}\mathit{bool},\, t_2{:}t_3\} \cup C1 \cup C_2 \cup C_3}$$

$$\frac{\text{make}(t_x) \qquad G, x{:}t_x \vdash e : t \dashv C_1}{G \vdash \text{fun } x \to e : t_x \to t \dashv C_1}$$

(make creates some new, not yet used 'a/'b/'c-style type)

$$\frac{}{G \vdash x : G(x) \dashv \{\}}$$

$$\frac{G \vdash e_1 : t_1 \dashv C_1 \qquad G, x{:}t_1 \vdash e_2 : t_2 \dashv C_2}{G \vdash \text{let } x = e_1 \text{ in } e_2 : t_2 \dashv C_1 \cup C_2}$$

$\{t_x : int, t_x : int\}$ $\cup \{\} \cup \{\}$

Example 2a: Use type inference to prove that this is well-typed:

~~`let a = true in a && false`~~

fun z → z+1

C1 C2

G, z:tx ⊢ z : tx ⊣ {} G, z:tx ⊢ 1 : int ⊣ {}

tx G, z:tx ⊢ z+1 : ___ ⊣ ___ {tx:int, tx:

G ⊢ fun z -> z+1 : ___ ⊣ ___

Example 2b: Use type inference to prove that this is well-typed:

```
fun z -> z + 1
```

\

Example(s) with just let bindings and &&:

type checking:

ex: let a = true in a && false

$$\dfrac{\dfrac{}{G \vdash true : \underline{bool}} \qquad \dfrac{\dfrac{}{G, a:bool \vdash a : \underline{bool}} \qquad \dfrac{}{G, a:bool \vdash false : \underline{bool}}}{G, a : bool \vdash a \,\&\&\, false : \underline{bool}}}{G \vdash let\ \underset{x}{a} = \underset{e1}{true}\ in\ \underbrace{a \,\&\&\, false}_{e2} : \underset{t2}{\underline{bool}}}$$

final type: bool , final answer: (well typed)

type inference:

ex: let a = true in a && false

$$\dfrac{\dfrac{}{G \vdash true : \underline{bool} \dashv \underline{\{\}}} \qquad \dfrac{\dfrac{}{G, a:bool \vdash a : \underline{bool} \dashv \underline{\{\}}} \qquad \dfrac{}{G, a:bool \vdash false : \underline{bool} \dashv \underline{\{\}}}}{G, a:bool \vdash a \,\&\&\, false : \underline{bool} \dashv \{\underline{bool} : \underline{bool}, \underline{bool} : bool\} \cup \underline{\{\}} \cup \underline{\{}}}{G \vdash let\ a = true\ in\ a \,\&\&\, false : \underline{bool} \dashv \underline{\{\}} \cup \underline{\{bool : bool}, bool : bool\}}$$

final type: bool, w/ constraint list {bool:bool, bool:bool}

constraint list has no contradictions, unify w/ "bool"

to get: type is bool

final answer: (well typed)

Example(s) with functions and +:

type checking:

ex: fun (z : int) → z + 1

$$\dfrac{\dfrac{\dfrac{}{G, z:int \vdash z : \underline{int}} \qquad \dfrac{}{G, z:int \vdash 1 : \underline{int}}}{G, z:int \vdash z+1 : \underline{int}}}{G \vdash \text{fun } (\underset{x}{z} : \underset{t_1}{int}) \to \underset{e}{z+1} : \underline{int} \to \underline{int}}$$

final type: int → int

final answer: ~~int → int~~ (well typed)

type inference:

ex: fun z → z + 1

$$\dfrac{\dfrac{\dfrac{}{G, z:'a \vdash z : \underline{'a} \dashv \underline{\{\}}} \qquad \dfrac{}{G, z:'a \vdash 1 : \underline{int} \dashv \underline{\{\}}}}{G, z:'a \vdash z+1 : \underline{int} \dashv \underline{\{'a : int, 'a : int\}} \cup \underline{\{\}} \cup \underline{\{\}}}}{G \vdash \text{fun } z \to z+1 : \underline{'a} \to \underline{int} \dashv \underline{\{'a : int, 'a : int\}}}\ \text{make}('a)$$

final type: 'a → int, w/ constraint list {'a : int, 'a : int}

Constraint list has no contradictions, unify w "'a → int"

to get: type is (int → int)

final answer: well typed

$((\lambda x.(\lambda y.\ x\ y))\ z)\ a$

x – y

$(\lambda y.\ z\ y)\ a$ (fun x => ....) z

y [x:=z]

$z\ a$

Beta-Normal Form!

$(\lambda x.\ x\ y)\ ((\lambda a.\ a)\ b)$

Eager Eval

• reduce arg first

$(\lambda x.\ x\ y)\ (b)$

$b\ y$

=

Lazy Eval

• apply arg as is

$((\lambda a.\ a)\ b)\ y$

$b\ y$

$(\lambda a.\ b)\ ((\lambda x.\ x\ x)(\lambda x.\ x\ x))$

Lazy

b

Eager $(\lambda a.\ b)\ ((\lambda x.\ xx)(\lambda x.\ xx))$

inf loop

$(\lambda x.\ a\ x\ (\lambda x.\ x))\ z$

x ⊗

$a\ z\ (\lambda x.\ x)$

$(\lambda y.\ a\ y\ (\lambda x.\ x))$

fun x -> (...

fun x -> (

x

)

x

)

$(\lambda x.(\lambda x.(\lambda y.x\ y))\ x)\ x \rightarrow (\lambda a.(\lambda b.(\lambda y.b\ y))\ a)\ x$

$(\lambda x.(\lambda y.x\ y))\ x$

$(\lambda b.(\lambda y.b\ y))\ x$

$\lambda y.x\ y$

$(\lambda y.x\ y)$

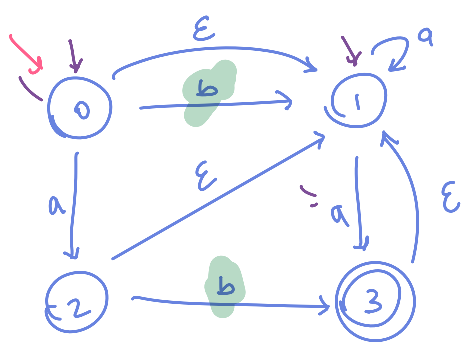
$(\lambda x.\lambda y.y\ x)\ y \rightarrow (\lambda a.\lambda b.b\ a)\ y$

$(\lambda y.y\ y)$

$(\lambda b.b\ y)$

$\emptyset$

$(\lambda x.E)\ E$



Final	states	a	b
	0,1	1,2,3	1
X	1,2,3	1,3	1,3
	1	1,3	∅
X	1,3	1,3	~~∅~~