$$\left(\underbrace{(\lambda x.(y\ (\lambda y.((x\ y)(\lambda x.(x}_{①} \quad \underbrace{(\lambda z.\ z)}_{②} \right) \underbrace{(\lambda z.\ w)}_{③}$$

$$((a\ \ b)\ \ c)$$

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

① Parens for λ body
② F. A paren (L.A)

$$(a\ \ b)$$

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

$$(\lambda x.\ y\ \ \lambda m.\ x\ \ m\ \ \ \lambda n.\ n\ \ m)\ (\lambda z.z)\ (\lambda z.w)$$

t t

$$(\lambda x. (\lambda y. y\, a)\, x)\, ((\lambda x. x)\, \underline{(\lambda y. y\, b)})$$

Eager / CBV

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

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

$$(\lambda y. y\, b)\, \underline{a}$$

$$\underline{\underline{a\, b}}$$

$$(\cancel{\lambda x. x})\, \underbrace{((\lambda x. x x)\, (\lambda x. x x))}_{\text{infinite}}$$

$\text{true} = \lambda x.\lambda y.x$

$\text{false} = \lambda x.\lambda y.y$

$\text{not} = (\lambda x.x\ \text{false}\ \text{true})$

$\text{or} = (\lambda x.\lambda y.\ x\ \text{true}\ y)$

$\text{not}\ (\text{or}\ \text{false}\ \text{true}) \Rightarrow \text{false}$

$\text{not}\ ((\lambda x.\lambda y.\underline{x}\ \text{true}\ y)\ \text{false}\ \text{true})$

$\text{not}\ ((\lambda y.\ \text{false}\ \text{true}\ y)\ \text{true})$

$\text{not}\ (\text{false}\ \text{true}\ \text{true})$

$\text{not}\ ((\lambda x.\lambda y.y)\ \text{true}\ \text{true})$

$\text{not}\ ((\lambda y.y)\ \text{true})$

$\text{not}\ \text{true}$

$(\lambda x.x \text{ false true}) \text{ true}$

true false true

$((\lambda x.\lambda y.x) \text{ false}) \text{ true}$

$(\lambda y.\text{false}) \text{ true}$

false

Problem 1: Context Free Grammars

Consider the following Grammar:

```
S -> S1S2
A -> 12A|A31|T
T -> 1|2|3
```

(a) Is this grammar ambiguous?

(A) Yes (B) No

(b) Derive the string "313312" using the above CFG

Prove that the following grammar is ambiguous:

```
S → bS | Sb | T
T → Sa | Sb | Sc | ϵ
```

Prove that the following grammar is ambiguous

```
S -> bS | Sb | T
T -> Sa | Sb | Sc | ϵ
```

Consider the following Grammars:

Grammar 1

$$S \rightarrow aSb \mid aaSb \mid aaaSb \mid \epsilon$$

Grammar 2

$$S \rightarrow AAASB \mid \epsilon$$
$$A \rightarrow a \mid \epsilon$$
$$B \rightarrow b$$

Grammar 3

$$S \rightarrow ASB$$
$$A \rightarrow aA \mid \epsilon$$
$$B \rightarrow bbbB \mid \epsilon$$

(a) Which of the following grammars describe strings of $a^x b^y$, $x < 3y$? Select all that apply.

Grammar 1 Grammar 2 Grammar 3 None

(b) Prove that Grammar 2 is ambiguous

(c) Draw the abstract syntax tree that would be generated by parsing the following string with the given CFG using a leftmost derivation.

String: "1 * 2 + 3"

CFG:

S -> M * S | M

M -> M + N | N

N -> 1 | 2 | 3 | (N), where n is any number

$$\frac{}{A;\ \text{true} \Rightarrow \text{true}} \quad \frac{}{A;\ \text{false} \Rightarrow \text{false}}$$

CM

$$\frac{A;\ e_1 \Rightarrow \text{true}}{A;\ (\text{not } e_1) \Rightarrow \text{false}} \quad \frac{A;\ e_1 \Rightarrow \text{false}}{A;\ (\text{not } e_1) \Rightarrow \text{true}}$$

$$\frac{A;\ e_1 \Rightarrow \text{true} \qquad A;\ e_2 \Rightarrow v_1}{A;\ (\text{if } e_1 \text{ then } e_2 \text{ else } e_3) \Rightarrow v_1}$$

$$\frac{A;\ e_1 \Rightarrow \text{false} \qquad A;\ e_3 \Rightarrow v_1}{A;\ (\text{if } e_1 \text{ then } e_2 \text{ else } e_3) \Rightarrow v_1}$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_3 \text{ is } v_1 \mid\mid v_2}{A;\ (e_1 \mid\mid e_2) \Rightarrow v_3}$$

Fill in the following derivation:

$$\frac{\dfrac{\dfrac{}{A;\ \#3 \Rightarrow \#3} \quad \dfrac{}{A;\ \#4 \Rightarrow \#4} \quad \#5 \text{ is} \#1}{A;\ \#1 \Rightarrow \#5} \qquad \dfrac{}{A;\ \#2 \Rightarrow \#2}}{\text{if } \underbrace{true||false}_{e1} \text{ then } \underbrace{true}_{e2} \text{ else } \underbrace{\text{not} false}_{e3} \Rightarrow \#6}$$

$$\frac{\dfrac{\dfrac{}{A; true \to true} \quad \dfrac{}{A; false \to false} \quad true \text{ is } true||}{A; true || false \Rightarrow true} \qquad \dfrac{}{A; true \to true}}{\text{if } true||false \text{ then } true \text{ else not} f \Rightarrow true}$$

$\Downarrow$ e2 $\quad$ 2T

$$\frac{}{A;\ n \Rightarrow n} \qquad \frac{A(x) = v}{A;\ x \Rightarrow v}$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A, x : v_1;\ e_2 \Rightarrow v_2}{A;\ \text{let } x = e_1 \text{ in } e_2 \Rightarrow v_2}$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_3 \text{ is } v_1 \wedge v_2}{A;\ e_1 \wedge e_2 \Rightarrow v_3}$$

$$\frac{\dfrac{}{A;\ \text{"cmsc"} \Rightarrow \text{"cmsc"}} \qquad \dfrac{\dfrac{\dfrac{(\#4)}{A, x : \text{"cmsc"}, y : \text{"330"};\ x \Rightarrow \text{"cmsc"}} \quad \dfrac{(\#5)}{A, x : \text{"cmsc"}, y : \text{"330"};\ y \Rightarrow \text{"330"}} \quad (\#6)}{A, x : \text{"cmsc"}, y : \text{"330"};\ (\#3) \Rightarrow \text{"cmsc330"}} \quad (\#2)}{A, x : \text{"cmsc"};\ \text{let } y = \text{"330" in } x \wedge y \Rightarrow \text{"cmsc330"}}}{A;\ (\#1) \text{ in let } y = \text{"330" in } x \wedge y \Rightarrow \text{"cmsc330"}}$$

#1: let $\underline{x}$ = $\underline{\text{"cmsc"}}$

#2: A, x: "cmsc"; "330" ⇒ "330"

#3: x ∧ y

#4: A, x: "cmsc", y: "330" (x) = "cmsc"

#5: A, x: "cmsc", y: "330" (y) = "330"

#6: "cmsc330" is "cmsc" ∧ "330"

cM

$$\frac{}{\text{Barbie} \to \text{barbie}} \qquad \frac{}{\text{Ken} \to \text{ken}}$$

$$\frac{E1 \to v_1 \qquad E_2 \to v_2 \qquad v_3 \text{ is ken if } v_1 \text{ is barbie and } v_2 \text{ is barbie}}{E1 \heartsuit E2 \to v_3}$$

$$\frac{E1 \to v_1 \qquad E_2 \to v_2 \qquad v_3 \text{ is barbie if } v_1 \text{ is barbie and } v_2 \text{ is ken}}{E1 \heartsuit E2 \to v_3}$$

$$\frac{E_1 \to v_1 \qquad E_2 \to v_2 \qquad v_3 \text{ is barbie if } v_1 \text{ is ken and } v_2 \text{ is ken}}{E1 \text{ 💔 } E_2 \to v_3}$$

$$\frac{E1 \to v_1 \qquad E_2 \to v_2 \qquad v_3 \text{ is ken if } v_1 \text{ is ken and } v_2 \text{ is barbie}}{E1 \heartsuit E2 \to v_3}$$

$$\dfrac{\dfrac{}{\text{Barbie} \to \text{barbie}} \qquad \dfrac{\dfrac{}{\text{Barbie} \to \text{barbie}} \qquad \dfrac{\dfrac{}{\text{Ken} \to \text{ken}} \quad \dfrac{}{\text{Ken} \to \text{ken}} \quad v_3 \text{ is barbie}}{\text{Ken 💔 Ken} \to \text{barbie}} \qquad v_3 \text{ is ken}}{\text{Barbie} \heartsuit (\text{Ken 💔 Ken}) \to \text{ken}} \qquad v_3 \text{ is barbie}}{\text{Barbie} \heartsuit (\text{Barbie} \heartsuit (\text{Ken 💔 Ken})) \to \text{barbie}}$$

$$\frac{A, var: val(var) = val}{A, var: val;\ var \Rightarrow val}$$

ZT

$$\frac{\dfrac{A, var: val(x) \Rightarrow val}{A, var: val;\ var \Rightarrow val} \qquad A, var: val;\ e1 \Rightarrow v_2 \qquad A, var: v_2;\ e_2 \Rightarrow v_3}{A,\ var: val;\ \text{EXCHANGE}\ var\ \text{for}\ e1\ \text{in}\ e2 \Rightarrow v_3}$$

$$\frac{A;\ e1 \Rightarrow v_1 \qquad v_1\ \text{is yap}}{A;\ \text{slayp?}\ e1 \Rightarrow \text{yap}}$$

$$\frac{A;\ e1 \Rightarrow v_1 \qquad v_1\ \text{is slay}}{A;\ \text{slayp?}\ e1 \Rightarrow \text{slay}} \qquad \frac{}{A, x: \text{slay};\ \text{slay} \Rightarrow \text{slay}}$$

$$\frac{\dfrac{\#2}{\#1} \qquad \dfrac{\dfrac{}{\#4}}{A, X: \text{yap};\quad \#3} \qquad \dfrac{\dfrac{\#7\ = \text{slay}}{A, x: \text{slay};\ \#6} \qquad \text{slay is slay}}{\#5}}{A, x: \text{yap};\ \text{EXCHANGE}\ x\ \text{for slayp? slay in slayp?}\ X \Rightarrow \text{slay}}$$

#1: $A, x: \text{yap};\ X \Rightarrow \text{yap}$

#2: $A, X: \text{yap}\ (X) = \text{yap}$

#3: $\text{slayp? slay} \Rightarrow \text{slay}$

#4: $\text{slay} \Rightarrow \text{slay}$

#5: $A, X: \text{slay};\ \text{Slayp?}\ X \Rightarrow$ "slay"

#6: $X \Rightarrow$ "slay"

#7: $A, X: \text{slay}\ (X)$

A, x: Yap (x) ⇒ Yap
A, X: Yap: X ⇒ Yap

A, x: Yap; Slay ⇒ Slay   Slay is slay
A, X: Yap; slaup? slay ⇒ slay

A, x: Slay (x) = slay
A, x: slay; x ⇒ slay   Slay is Slay
A, X: slay; slayp? slay ⇒ slay

A, X: Yap ; EXCHANGE x for slayp? slay in slayp? slay ⇒ slay

A, x: Yap(x) = yap
A, X: yap: X ⇒ yap

A, x: Yap; slay ⇒ slay   slay is slay
A, x: yap; slayp? slay ⇒ slay

A, x: slay (x) = slay
A, x: slay; x ⇒ slay   Slay is slay
A, x: slay; slayp? x ⇒ slay

A, X: Yap; EXCHANGE X for slayp? slay in slayp? X ⇒ slay

🤸‍♀️

Exam 2 Review - OCaml & Opsem

Date: @November 7, 2023

Context Free Grammars

- Any regular expression can be expressed as a string that is in the following set
    - S → Non terminal
    - others are terminals

$$
\begin{aligned}
S \longrightarrow \quad & \epsilon \\
& |\sigma \\
& |S\,S \\
& |S|S \\
& |S^* \\
& |(S)
\end{aligned}
$$

- **Example**

$$
\begin{aligned}
S &\rightarrow \quad NP\ VP \\
NP &\rightarrow \quad \text{pronoun} \\
&\quad\quad |\text{proper\_noun} \\
&\quad\quad |\text{det } AN \\
AN &\rightarrow \quad \text{adj } AN \\
&\quad\quad |\text{noun} \\
VP &\rightarrow \quad \text{verb} \\
&\quad\quad |\text{verb } NP
\end{aligned}
$$

- Terminals → pronoun, proper_noun, det, adj, noun, verb
- Non-terminals → S, NP, VP, AN
- Production → a production tells us all the things a non-terminal can be
    - S → N P V P
    - AN → adj AN | noun

Designing Grammars

- Base Cases
    - ∅ → the empty set, the language contains no strings
    - ε → regex accepts an empty string, the CFG should have a single production
        - S → ε
    - σ → The regular expression is a single character, CFG should have a single production
        - S → σ
- Recursive Definitions
    - Concatenation (Union) → To concatenate two strings together, we can just push them together with either non-terminals or just the string you would expect

S → ab

S → AB

A → a

B → b

    - Branching (Or) → to branch, we can use the same symbol we used in regex : |

      S → hello | hi

    - Kleene Closure (Star) → To allow repeated values, we initialize the recursive property these sets have

      S → aS | ε (same thing as a*)

- Not Supported by Regular Expressions
    - CFG’s support more than regular expressions
    - We can check balanced parentheses like this:

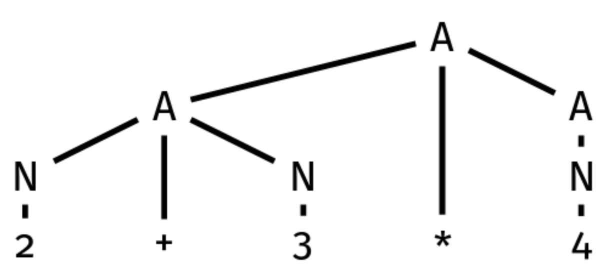
      Balanced parentheses surrounding “a” → S→(S)|a

      Palindromes of “a”, “b”, “c” → S→ aSa | bSb | cSc | ε

- Abstract Syntax Trees

$$
\begin{aligned}
A &\longrightarrow \quad A+A \mid A-A \mid A*A \mid A/A \mid (A) \mid N \\
N &\longrightarrow \quad number
\end{aligned}
$$

    - 2 + 3 * 4



Operational Semantics

- Operational semantics is a way to help describe the meaning of a statement in a programming language. It helps us to determine the correctness of a programming statement

- Axiom: things that are basic enough that we don’t need to prove it

$$\frac{}{n \Rightarrow n}$$

- Target language: a language that the operational semantics is describing
- Environment: a mapping from variables to values.
  - [x : 3, y:4]
  - how to look up a variable in our environment?
    - if we want to evaluate V into a value, we need to look up that value in the environment

$$\frac{A(x) \Rightarrow v}{A; x \Rightarrow v}$$

$$\frac{A, x : 4; (x) \Rightarrow 4}{A, x : 4; x \Rightarrow 4}$$