CSC 320
Foundations of Computer Science

Lecture 8

Instructor: Dr. Ulrike Stege

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

This meeting will be recorded

"Please be aware our sessions are being screen-recorded to allow students who are not able to attend to watch later and will be posted in Brightspace."

Deadlines; Assessment

10%	25%	25%	40%
Quizzes	Assignments	Midterms	Final Exam
Quiz 1-8: 1% each Quiz 9: 2%	Assignment 1-5: 5% each	Midterm 1: 10% Midterm 2: 15%	

May

S	M	T	W	T	F	S
			3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

June

S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

July

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Timed quizzes (~30 min)
Review before starting quiz

Last time ....

- Pumping Lemma for regular languages
- Using the Pumping Lemma for regular languages to show that a language is nonregular
- Proof of Pumping Lemma for regular languages

Next

- Context-free languages
  - Grammars
  - Pushdown automata

Recall: Reductions

- Important concept that we will use later in the course
- What are reductions?
  - A reduction is an algorithm that transforms one problem (Problem A) into another (Problem B)
  - Certain reductions are used to show that, if A reduces to B, then problem B is *at least as difficult* as problem A in some sense

Solving x' of B will also solve x of A

Metaphor for Reduction

- Problem: Increasing brightness in a room with dirty windows

Reduce problem to window cleaning

Clean window solves problem

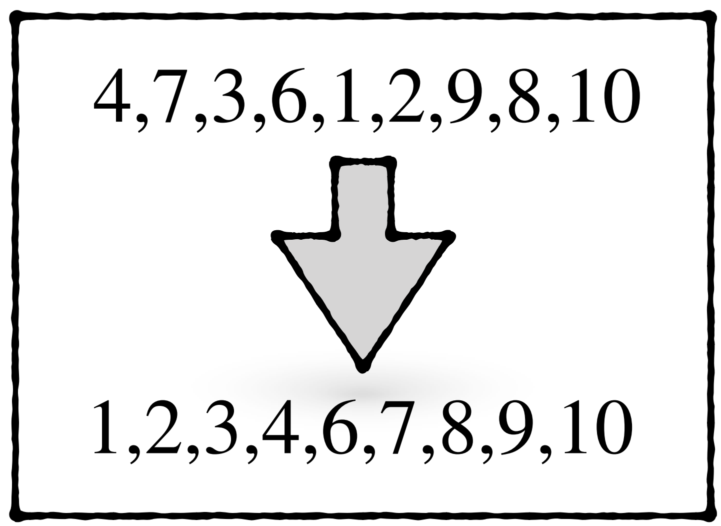
Reducing

Sorting to Convex Hull

Problem 1: Sorting

Input: integers $x_1, x_2, \ldots, x_n$

Output: $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ such that $x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_n}$ and $i_j \neq i_k$ for $j \neq k$

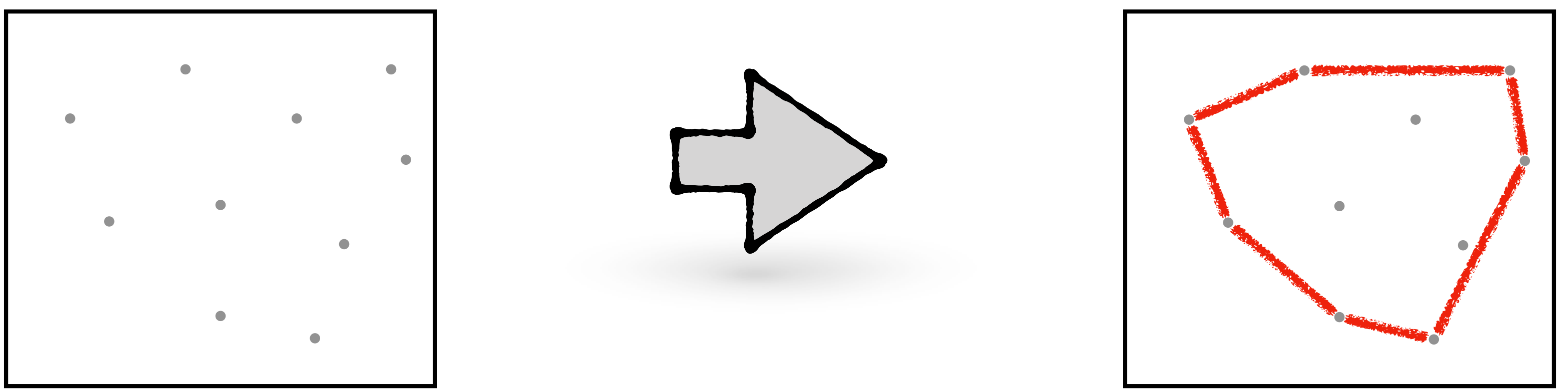


What is known about Sorting?

- **Recall**: Sorting has a *lower bound* of $\Omega(n \log n)$
- That is, there is no algorithm that sorts $x_1, x_2, \ldots, x_n$ faster

Problem 2: Convex Hull—Determining the Convex Hull points of a given point set in (counter clockwise) order

Input: Point set $S = \{(a_1^x, a_1^y), (a_2^x, a_2^y), \ldots, (a_n^x, a_n^y)\}$



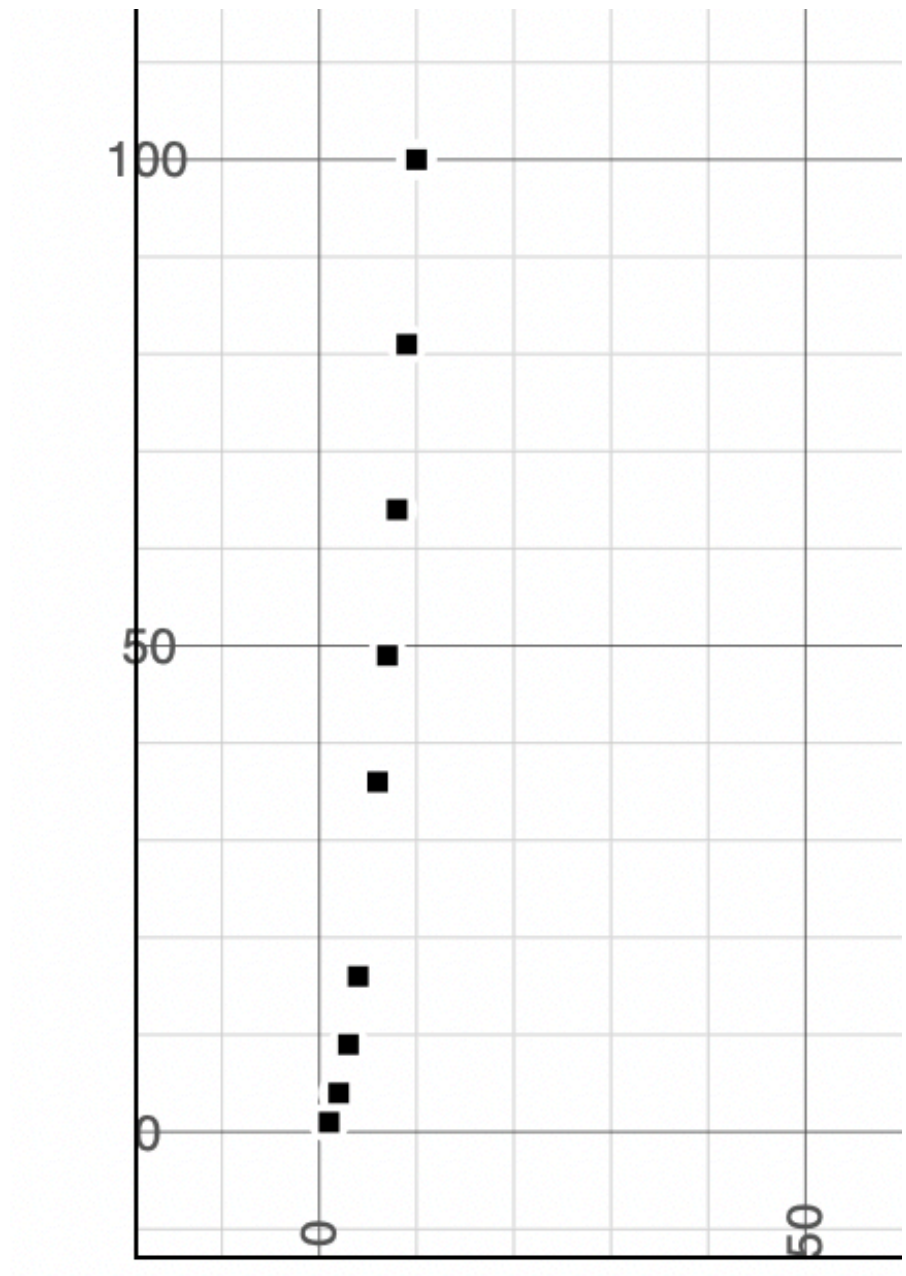
Linear(-time) Reduction from Sorting to Convex Hull

- Reduction: For Sorting input $x_1, x_2, \ldots, x_n$, create Convex Hull input $S = \{(x_1, x_1^2), (x_2, x_2^2), \ldots (x_n, x_n^2)\}$

Example

$4,7,3,6,1,2,9,8,10 \longrightarrow$

$(4,16), (7,49), (3,9), (6,36), (1,1), (2,4), (9,81), (8,64), (10,100)$



Linear(-time) Reduction from Sorting to Convex Hull

- Reduction: For Sorting input $x_1, x_2, \ldots, x_n$, create Convex Hull input $S = \{(x_1, x_1^2), (x_2, x_2^2), \ldots (x_n, x_n^2)\}$

- Proof of correctness:

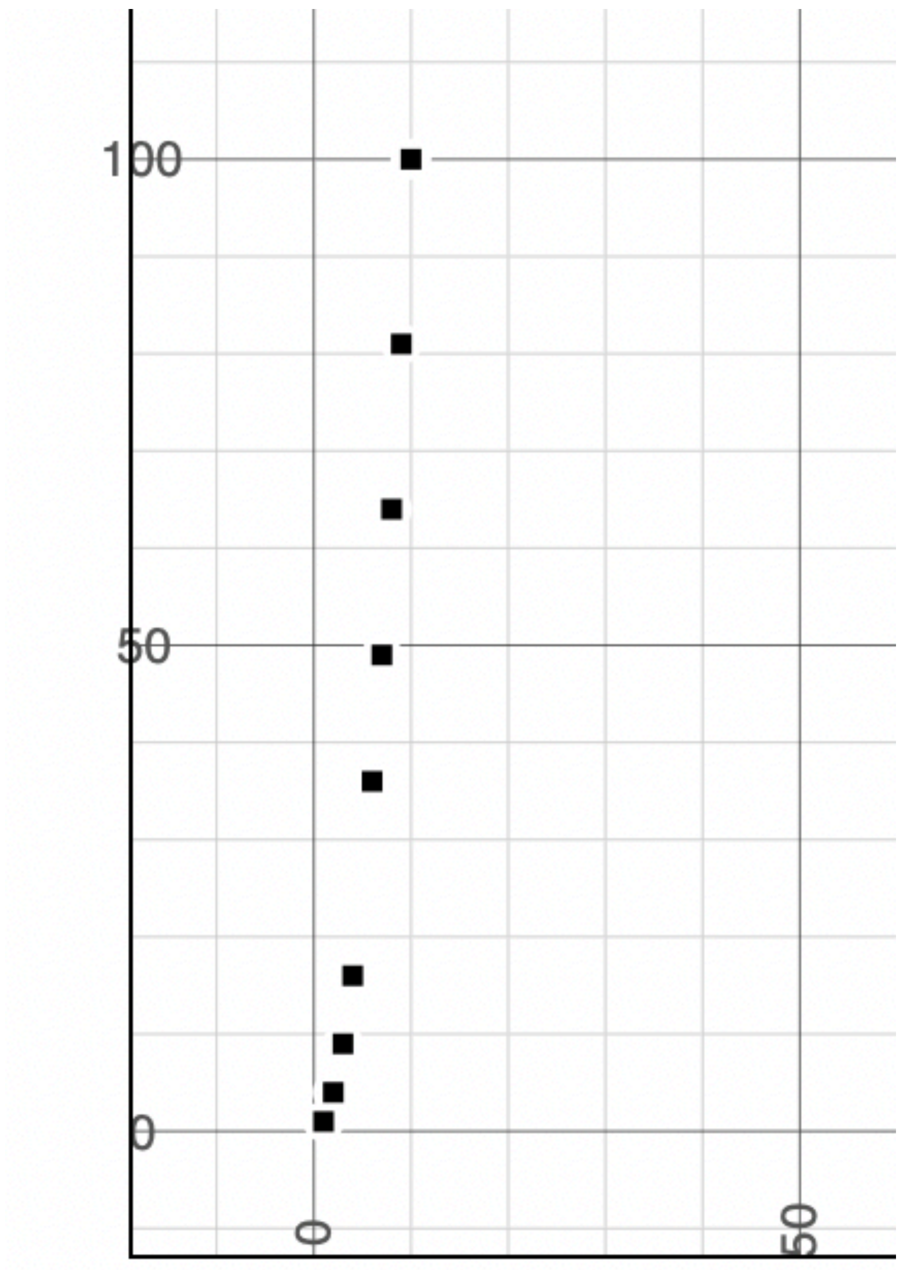
  - Claim 1. All coordinates in S lie on convex hull of S

  - Claim 2. The convex hull, organized in counter clockwise order, results in $(x_{i_1}, x_{i_1}^2), (x_{i_2}, x_{i_2}^2), \ldots, (x_{i_n}, x_{i_n}^2)$ for $x_{i_1} = \min\{x_1, x_2, \ldots, x_n\}$

Example

4,7,3,6,1,2,9,8,10 $\longrightarrow$

(4,16), (7,49), (3,9), (6,36), (1,1), (2,4), (9,81), (8,64), (10,100)



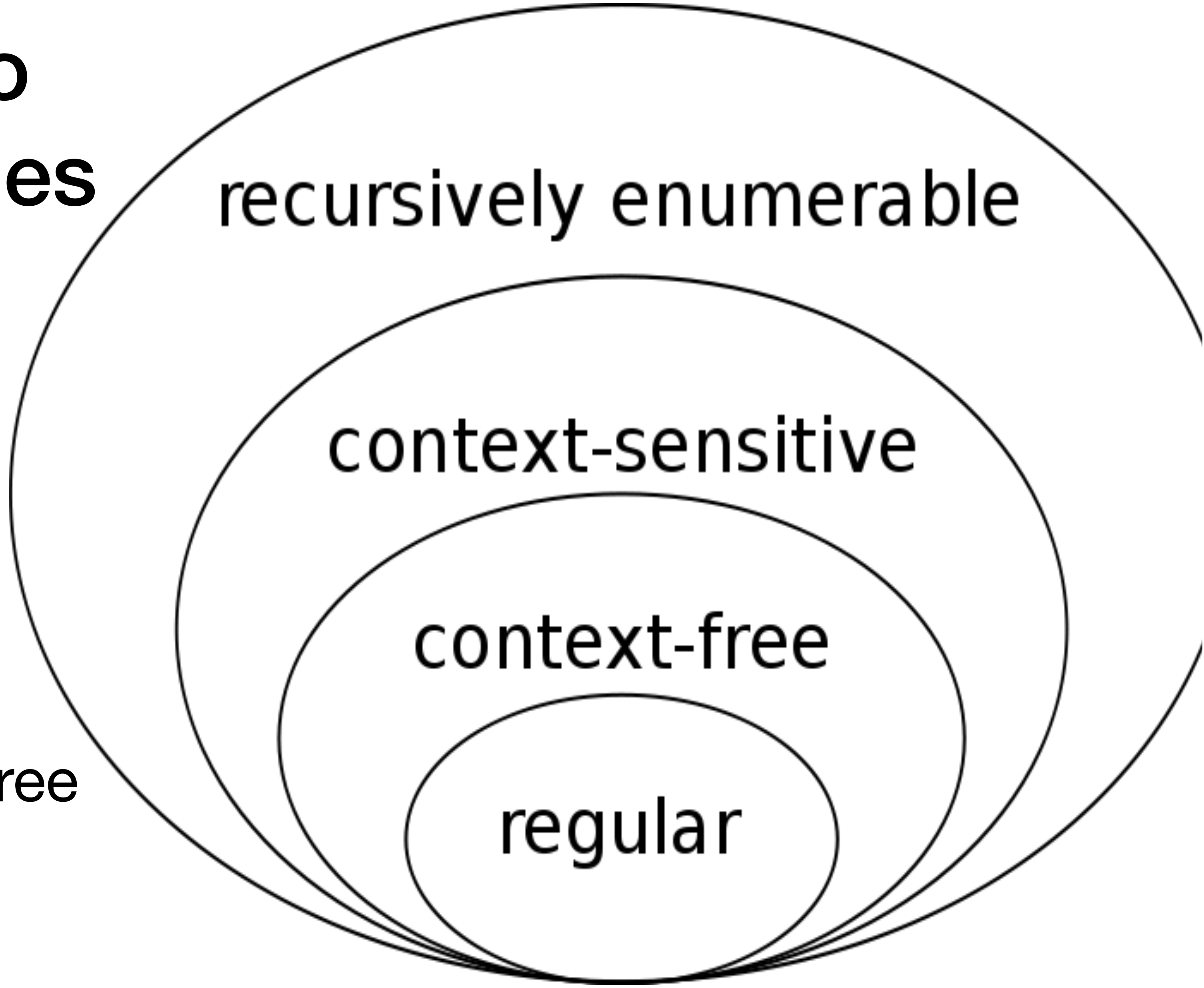
Therefore: Determining Convex Hull points of a given set of points in (counter clockwise) order is as hard as Sorting

Why? Since we can solve sorting through determining the convex hull points in counter clockwise order, we know that:

- if we could solve the Convex Hull problem in linear time, then we could solve sorting in linear time also
  - **How**? By taking the input for sorting, $x_1, x_2, \ldots, x_n$, then transforming it in linear time to input $S = \{(x_1, x_1^2), (x_2, x_2^2), \ldots (x_n, x_n^2)\}$ for the Convex Hull problem, then solving the Convex Hull problem for S, which also gives us a sorted sequences of $x_1, x_2, \ldots, x_n$

Therefore: Since Sorting requires $\Omega(n \log n)$ time, also determining Convex Hull points in order requires $\Omega(n \log n)$ time

Back to Languages



Next: context-free languages

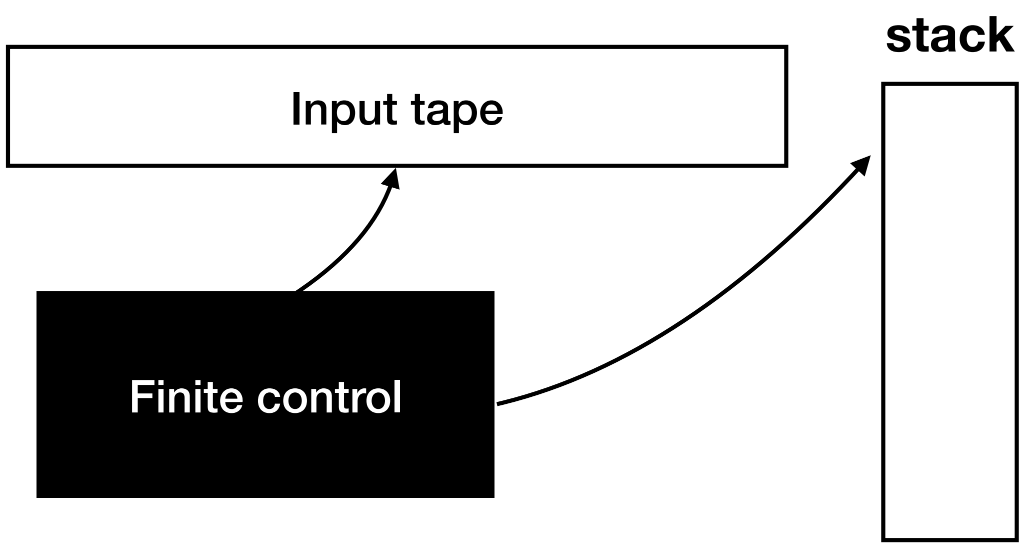
Recall: limitation of finite automata, regular expressions

- No real memory

Coming up …
Context-free languages

- Context-free grammars
- Pushdown automata

Some memory support



Context-free grammars

- More powerful method to describe languages
- First used to study (describe structure of) human languages
  - Invented by Noam Chomsky
  - relationship of terms such as *noun*, *verb*, and *preposition:* natural recursion
  - noun phrases may appear inside verb phrases and vice versa

1928—

Context-free grammars

- Computer science application:
  specification and compilation of programming languages
  - Grammar for *programming language*: reference for people learning syntax
  - Designing *compilers* and *interpreters*: first obtain grammar for language
  - *Parser:* uses grammar to extract meaning of program prior to generating compiled code

What's a grammar? Example

- Grammar G
  - $A \rightarrow 0A1$
  - $A \rightarrow B$
  - $B \rightarrow \#$
- G consists of
  - Productions/rules (substitution rules)
    - Symbol (variable), arrow, string (variables and terminals)
- Terminology: use capital letters for variables

3 (substitution) **rules**

2 **variables**: A, B

3 **terminals**: $0,1,\#$

start variable: A

Shorthand

- Grammar G
  - $A \rightarrow 0A1$
  - $A \rightarrow B$
  - $B \rightarrow \#$

G: $A \rightarrow 0A1$; $A \rightarrow B$; $B \rightarrow \#$

What does a grammar do?

- Grammar G: $A \to 0A1$; $A \to B$; $B \to \#$
- G *describes* a language L by generating each string of L as follows
  1. Write down the *start variable*
     - normally variable on the left-hand side of the top/first rule
  2. Find variable that is written down and rule that starts with that variable
     - Replace written down variable with right-hand side of that rule
  3. Repeat step 2 until no variable remains
- Example of deriving a string from G:
  - $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$
- $L(G)$: set of all strings that can be derived from G

What is $L(G)$?

- $L(G)$: set of all strings that can be derived from G
- Grammar G: $A \to 0A1$; $A \to B$; $B \to \#$
- Example of deriving a string from G:
  - $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

3 rules
2 variables: A, B
3 terminals: $0,1,\#$
start variable: A

$000\#111 \in L(G)$

$L(G) = ?$

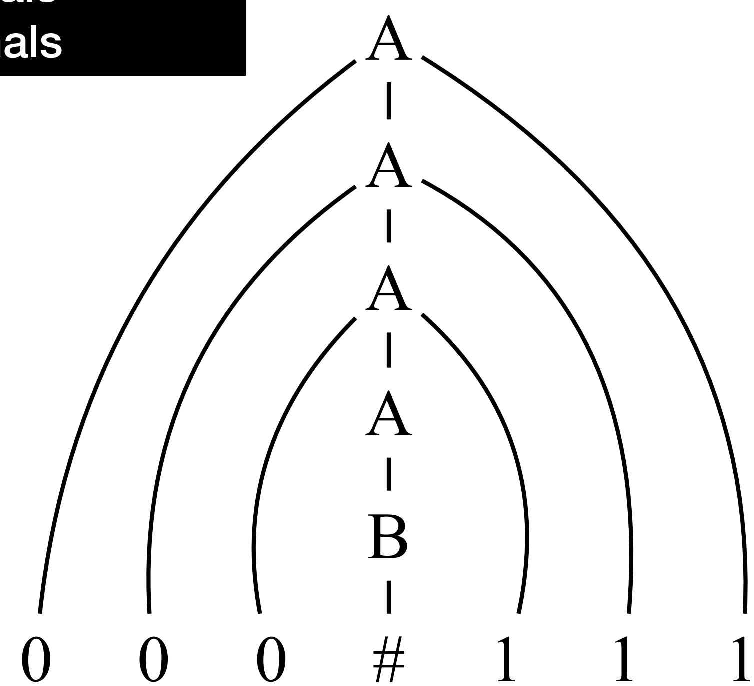
$L(G) = \{0^n\#1^n \mid n \geq 0\}$

Next: Parse Trees

- Derives strings of grammar
- Check syntax of programs
- Used also in: machine learning, natural language processing (NLP), chatGPT applications
- Syntax checking: Programming languages are often defined to be context-free (left-side of grammar rule is exactly one variable)
- Semantics verification requires also *context sensitive* analysis

Parse tree for derivation: $\mathrm{A} \Rightarrow 0\mathrm{A}1 \Rightarrow 00\mathrm{A}11 \Rightarrow 000\mathrm{A}111 \Rightarrow 000\mathrm{B}111 \Rightarrow 000\#111$

- **Parse tree:** hierarchical representation of terminals and non-terminals
- Leaves of parse tree: terminals



$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Definition: Context-Free Grammars

A **context-free grammar** is a 4-tuple (V, Σ, R, S)

- V: finite set of **variables**
- Σ: finite set of **terminals** (disjoint from V)
- R: finite set of (substitution) **rules**
  - each rule in R: a variable substituted by a string over variables and terminals
- $S \in V$: **start variable**
- The right hand side of a rule may be ϵ

More terminology

- Given grammar $G = (V, \Sigma, R, S)$
- Let u, v, and w be strings of variables and terminals, and
- let $A \rightarrow w$ be a rule of G
- Then
  - uAv **yields** uwv, written $uAv \Rightarrow uwv$
  - u **derives** v, written $u \overset{*}{\Rightarrow} v$, if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
  - The **language of grammar** G is: $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

More terminology

The class of languages described by context-free grammars is the class of **context-free languages**

Examples

Terminology: $S \to (S) \mid SS \mid \varepsilon$
short for
$S \to (S)$
$S \to SS$
$S \to \varepsilon$

- Given $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{(,)\}$, and R is given by: $S \to (S) \mid SS \mid \varepsilon$

- Then we can **derive** string (((()()(())))) as follows
    - Note: underlined symbol replaced in next derivation step

$$\underline{S} \Rightarrow (\underline{S}) \Rightarrow ((\underline{S})) \Rightarrow (((\underline{S}))) \Rightarrow (((\underline{S}S))) \Rightarrow ((((S)\underline{S}))) \Rightarrow ((((\underline{S})SS)))$$

$$\Rightarrow ((((\,)\underline{S}S))) \Rightarrow ((((\,)(S)\underline{S}))) \Rightarrow ((((\,)(\underline{S})(S)))) \Rightarrow ((((\,)(\,)(\underline{S}))))$$

$$\Rightarrow ((((\,)(\,)((\underline{S}))))) \Rightarrow ((((\,)(\,)((\,)))))$$

More examples

- Produce a grammar for language $\{1^n 0^n \mid n \geq 0\}$
- $G = (V, \Sigma, R, S)$ with
  - $V = \{S\}$
  - $\Sigma = \{0,1\}$
  - R: $S \rightarrow 1S0 \mid \epsilon$

How do we know a grammar describes a language?

- $L(G) = \{1^n 0^n \mid n \geq 0\}$
- $G = (V, \Sigma, R, S)$ with
  - $V = \{S\}; \Sigma = \{0,1\}; \ R\colon S \to 1S0 \mid \epsilon$

- If $w \in L$ then $w \in L(G)$
  - Proof by induction of length of string in language
- If $w \in L(G)$ then $w \in L$
  - Proof by induction of length of derivation

- $L(G) = \{1^n0^n \mid n \geq 0\}$
- $G = (V, \Sigma, R, S)$ with
  - $V = \{S\}; \Sigma = \{0,1\};$ R: $S \rightarrow 1S0 \mid \epsilon$

We show: If $w \in L$ then $w \in L(G)$

Proof by induction of length of string in language

Let $w = 1^n0^n$. We show $S \overset{*}{\Rightarrow} w$

$n = 0$: $w = \epsilon$

Assume, true for n

$$S \Rightarrow 1S0 \overset{*}{\Rightarrow} 11^nS0^n0 = 1^{n+1}S0^{n+1} \Rightarrow 1^{n+1}0^{n+1}$$

- $L(G) = \{1^n 0^n \mid n \geq 0\}$
- $G = (V, \Sigma, R, S)$ with
  - $V = \{S\}; \Sigma = \{0,1\};\ R$: $S \longrightarrow 1S0 \mid \epsilon$

We show: If $w \in L(G)$ then $w \in L$

Claim: If $S \overset{*}{\Rightarrow} w$ then $w = 1^{i-1}0^{i-1}$ or $w = 1^i S 0^i$ for some $i \geq 0$

Proof by induction on the length of the derivation

$n = 1$: $w = \epsilon$ or $w = 1S0$

Assume, true for n. Then $S \overset{*}{\Rightarrow} w$ with $w = 1^{i-1}0^{i-1}$ or $w = 1^i S 0^i$ for some $i \geq 0$

For derivation of length $n + 1$ we need one more step, thus only $w = 1^i S 0^i$ can be used

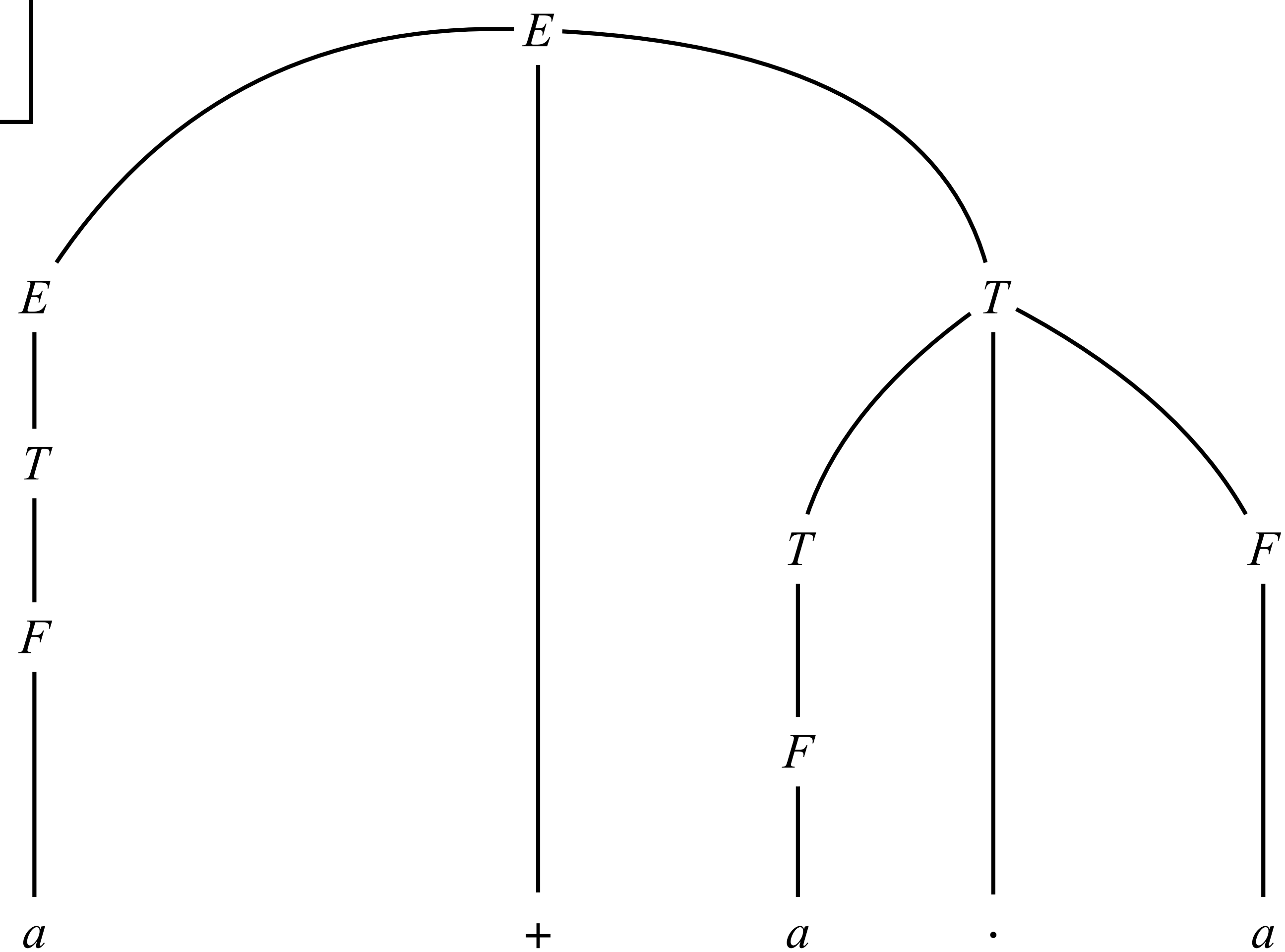
Then $1^i S 0^i \Rightarrow 1^{i+1} S 0^{i+1}$ or $1^i S 0^i \Rightarrow 1^i 0^i$

More examples (2)

- Given $G = (V, \Sigma, R, E)$ with $V = \{E, F, T\}$, $\Sigma = \{a, +, \cdot, (\ ,\ )\}$, and R is given by:
    - $E \longrightarrow E{+}T \mid T$
    - $T \longrightarrow T{\cdot}F \mid F$
    - $F \longrightarrow (E) \mid a$

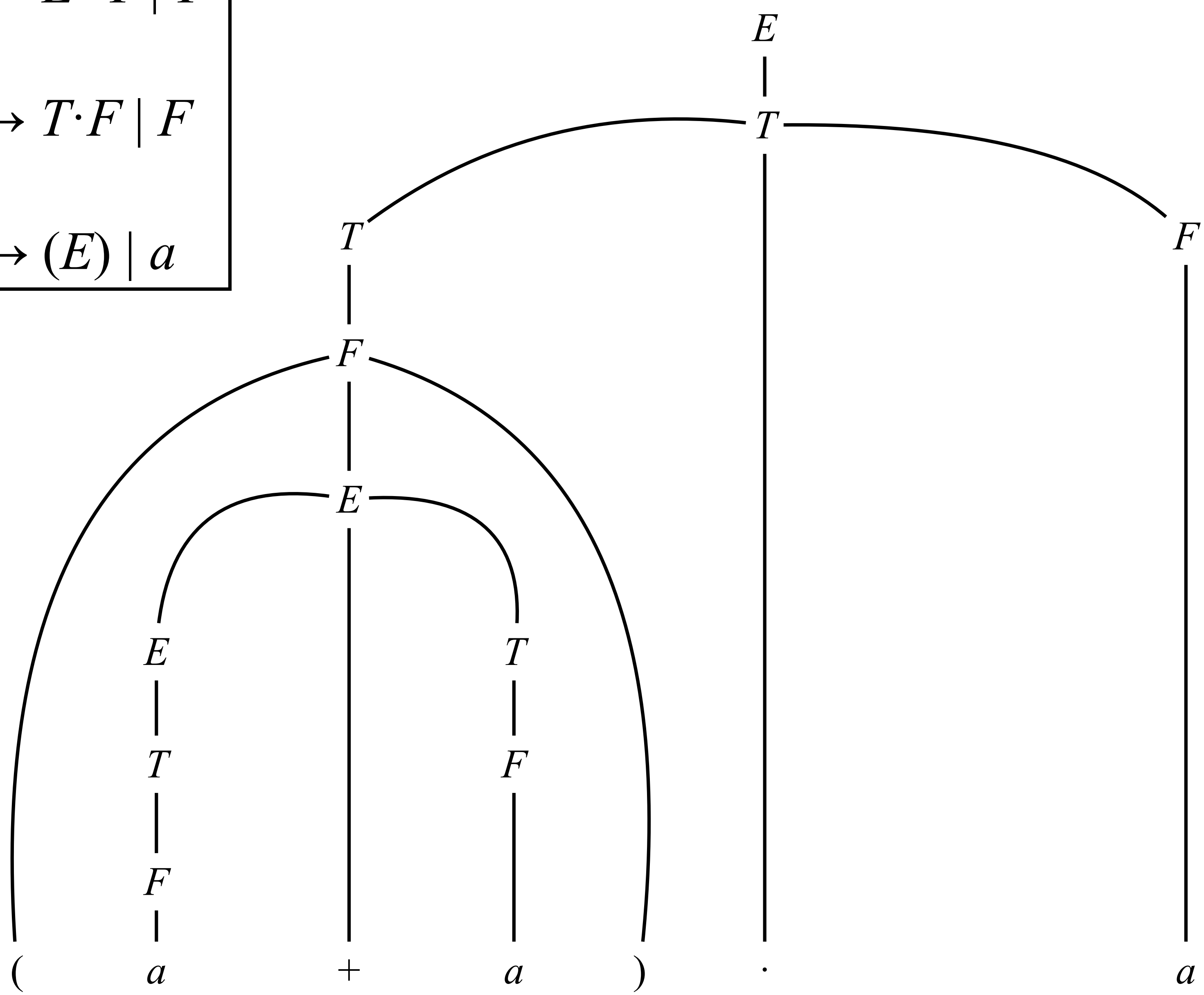
A parse tree for $a{+}a{\cdot}a$

- $E \rightarrow E{+}T \mid T$
- $T \rightarrow T{\cdot}F \mid F$
- $F \rightarrow (E) \mid a$



Parse tree for $(a+a)\cdot a$

- $E \to E{+}T \mid T$
- $T \to T{\cdot}F \mid F$
- $F \to (E) \mid a$



Leftmost derivations

- We call a derivation of string w in grammar G **leftmost derivation** if
  - at every step the *leftmost remaining variable* is replaced

Your turn

- $L = \{w \in \{0,1\}^* \mid w$ has an equal number of 0s and 1s$\}$

- Can you come up a context-free grammar G with $L(G) = L$?

More on Grammars

- Ambiguous grammars
- Inherently ambiguous languages
- Chomsky Normal Form

Ambiguous grammars

- A string w is derived **ambiguously** in context-free grammar G if it has at least two *different leftmost derivations*

- Such a grammar is called **ambiguous**

- Parsing an ambiguous strings in programming language: there are no unique instructions what code to generate! Recall: the outcomes of the different instructions can be different!

  - Eg: $a + a \cdot a$ unclear unless PEDMAS is applied additionally

  -
```
if (condition1)
    if (condition2)
        statement1;
    else
        statement2;
```

Example of an ambiguous grammar

- Given $G = (V, \Sigma, R, E)$ with $V = \{E\}$, $\Sigma = \{a, +, \cdot, (\ ,\ )\}$, and R is given by:
  - $E \rightarrow E{+}E \mid E{\cdot}E \mid (E) \mid a$

Two leftmost derivations for strings in language of $E \longrightarrow E{+}E \mid E{\cdot}E \mid (E) \mid a$

$a{+}a{\cdot}a$

- $E \Longrightarrow E{+}E \Longrightarrow a{+}E \Longrightarrow a{+}E{\cdot}E \Longrightarrow a{+}a{\cdot}E \Longrightarrow a{+}a{\cdot}a$
- $E \Longrightarrow E{\cdot}E \Longrightarrow E{+}E{\cdot}E \Longrightarrow a{+}E{\cdot}E \Longrightarrow a{+}a{\cdot}E \Longrightarrow a{+}a{\cdot}a$
- $a{+}a{\cdot}a$ is derived ambiguously in G
- Therefore G is ambiguous

Draw the different parse trees!

Language with ambiguous grammar: Often (but not always) possible to determine equivalent unambiguous grammar

We learned …

- What context-free grammars are
- What a language of a context-free grammar is
- That ambiguous grammars exists

Next…

- Chomsky Normal Form (CNF)
  - Helps dealing with ambiguity
  - Constraint grammar rules