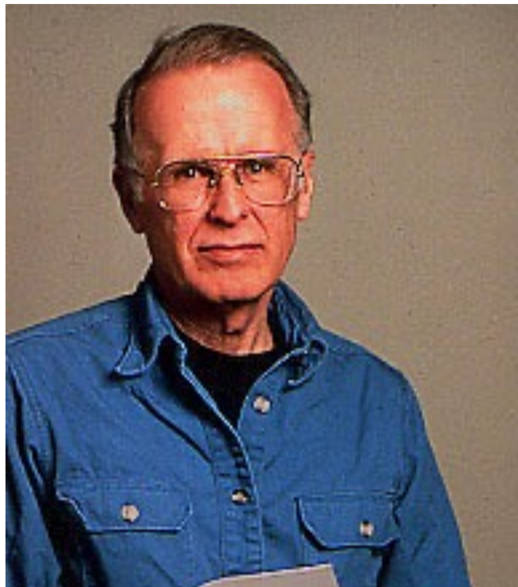
Theory of Computation Chapter 2

Context-Free Languages

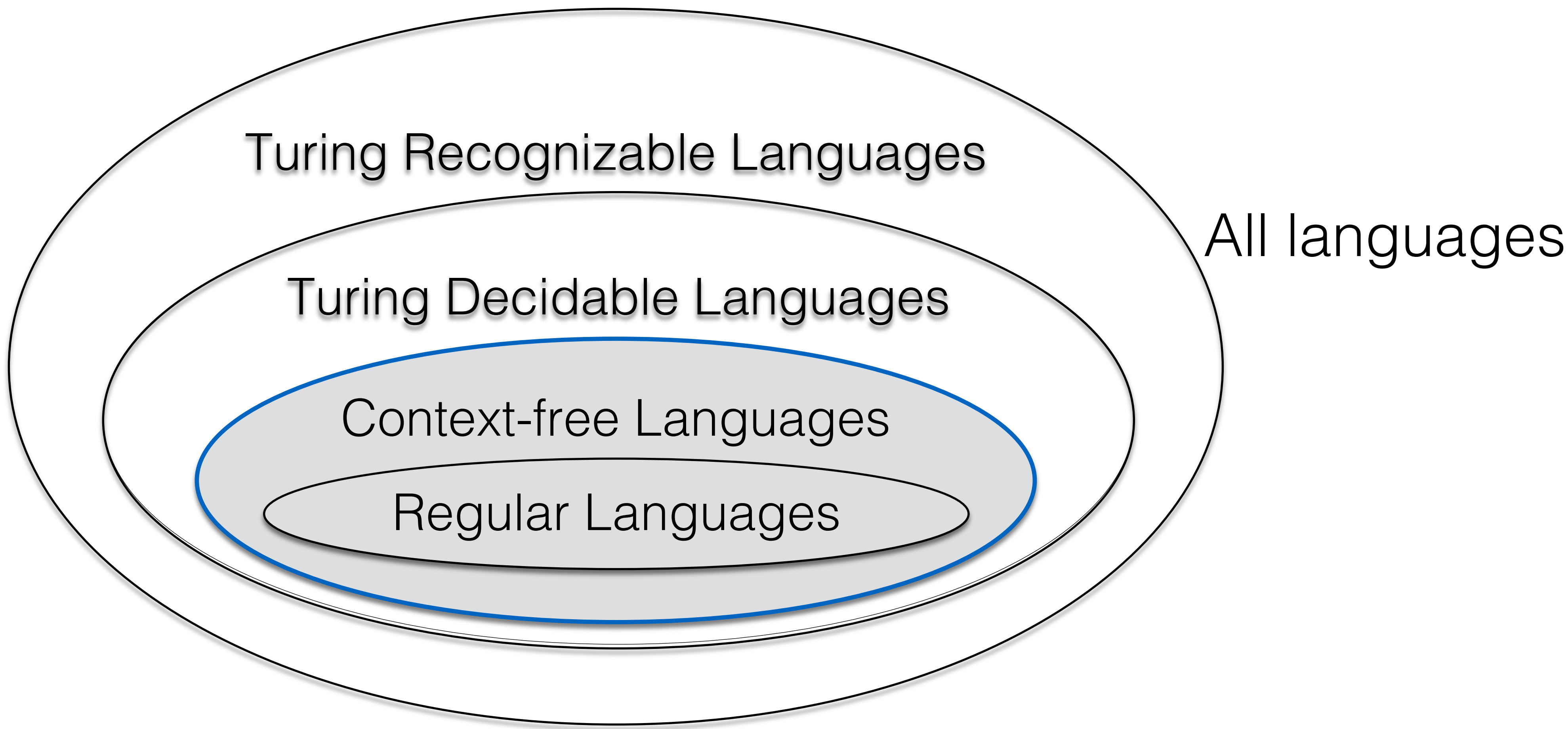
School of Engineering | Computer Science

John Backus
1924-2007

- Director of team that created Fortran, one of the first high-level programming languages (ever!) in 1957
- Fortran is still used today, in science/engineering fields
- Co-creator of Backus-Naur Form, which is used (today) to precisely describe the grammars of programming languages
- 1977 Turing Award winner



Context-Free Languages



Context-Free Languages

PDA = CFG = CNF

Closed under union, $\cup$, concatenation, $\circ$, and star, $*$.

Context-Free Languages

- Context-free languages are the next level up from regular languages

- They provide more power or expressiveness than regular languages since they have the ability to store information on a stack
- Most programming languages are Context-Free Languages

Context-Free Languages

- We describe context-free languages with context-free grammars (CFG)
- The grammar consists of a set of substitution rules of variables and terminals
- Variables are the symbols that help us derive the strings of the language, which consist of terminals
- Ex: $L(G_1) = \{0^n\#1^n \mid n \geq 0\}$
  - Grammar: $A \rightarrow 0A1$
    $A \rightarrow B$
    $B \rightarrow \#$

Context-Free Languages

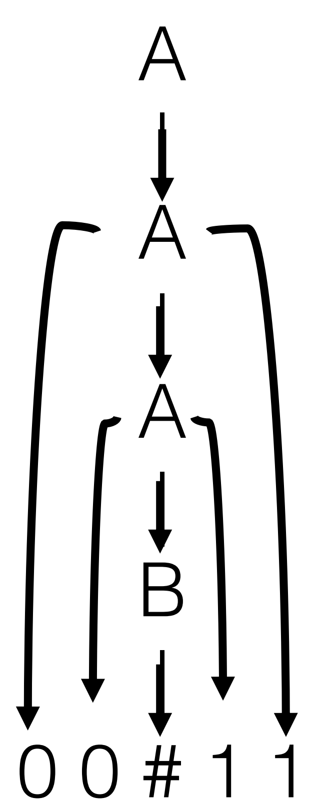
- The grammar consists of a set of substitution rules of variables and terminals
- Ex: $L(G_1) = \{0^n\#1^n \mid n \geq 0\}$
  - Grammar: $A \rightarrow 0A1$
    $A \rightarrow B$
    $B \rightarrow \#$
    - Variables: A and B
    - Terminals: 0, 1, #
    - Variable A is the <u>start variable</u> since it starts the first rule
    - There are three grammar rules listed

Context-Free Languages

- Ex: $L(G_1) = \{0^n\#1^n \mid n \geq 0\}$
  - Grammar: $A \rightarrow 0A1$
    $A \rightarrow B$
    $B \rightarrow \#$
  - To generate a string from this CFG:
    1. Start with the start variable (A)
    2. While variables are left, pick a variable and replace it with a substitution rule
    - Ex: $A \Rightarrow 0A1 \Rightarrow 0B1 \Rightarrow 0\#1$
    - Ex: $A \Rightarrow B \Rightarrow \#$
    - The sequence of substitutions to obtain a string is called a <u>derivation</u>.

Context-Free Languages

- We can also use a <u>parse tree</u> to represent the same information pictorially
- Ex: $L(G_1) = \{0^n\#1^n \mid n \geq 0\}$
  - Grammar: $A \rightarrow 0A1$
    $A \rightarrow B$
    $B \rightarrow \#$
    - Ex: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$



Context-Free Languages

- English language example:
  - <sentence> → <noun-phrase><verb-phrase>
  - <noun-phrase>→ <complex-noun>|<complex-noun><prep-phrase>
  - <verb-phrase>→ <complex-verb>|<complex-verb><prep-phrase>
  - <prep-phrase> → <prep><complex-noun>
  - <complex-noun> → <article><noun>
  - <complex-verb> → <verb>|<verb><noun-phrase>
  - <article> → a | the
  - <noun> → boy | girl | flower
  - <verb> → touches | likes | sees
  - <prep> → with

The | symbol means you can choose one derivation or the other for that rule

Context-Free Languages

<sentence> → <noun-phrase><verb-phrase>
<noun-phrase>→ <complex-noun>| <complex-noun><prep-phrase>
<verb-phrase>→ <complex-verb>| <complex-verb><prep-phrase>
<prep-phrase> → <prep><complex-noun>
<complex-noun> → <article><noun>
<complex-verb> → <verb>|<verb><noun-phrase>
<article> → a | the
<noun> → boy | girl | flower
<verb> → touches | likes | sees
<prep> → with

- Example derivation:
  - <sentence> ⇒ <noun-phrase><verb-phrase>
    ⇒ <complex-noun><verb-phrase>
    ⇒ <article><noun>< verb-phrase >
    ⇒ A <noun>< verb-phrase >
    ⇒ A boy < verb-phrase >
    ⇒ A boy <complex-verb><prep-phrase>
    ⇒ A boy <verb><prep-phrase>
    ⇒ A boy touches <prep-phrase>
    ⇒ A boy touches <prep><complex-noun>
    ⇒ A boy touches with <complex-noun>
    ⇒ A boy touches with <article><noun>
    ⇒ A boy touches with the <noun>
    ⇒ A boy touches with the flower

Context-Free Grammar

- Formal Definition of the Context-Free Grammar (CFG)
  - A CFG is a 4-tuple (V, Σ, R, S) such that:
    1. V is a finite set of variables
    2. Σ is a finite set of terminals such that $V \cap \Sigma = \emptyset$
    3. R is a finite set of substitution rules, where a rule has a variable on the left and variables/terminals on the right
    4. $S \in V$ is the start variable

Context-Free Terminology

- Context-Free Grammar (CFG) Terminology
  - $u \Rightarrow v$ means u yields v
    - applying one substitution rule to u gives v
  - $u \Rightarrow^* v$ means u derives v
    - Either:
      1. $u = v$, or
      2. there is a sequence of strings $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ such that $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
  - The language of G, $L(G) = \{x \in \Sigma^* \mid S \Rightarrow^* x\}$

Context-Free Grammar

- Context-Free Grammar (CFG) Example 2.3
  - G = ({S}, {a, b}, R, S) where the set of rules,

    R is: S → aSb | SS | ε
  - L(G) = {set of all strings of the same number of a's and b's (where no prefix has more b's than a's)}
    - Ex: ε, ab, aabb, abab, aababb
  - Can think of a and b as (and), so L(G) = {set of all strings of balanced parentheses
    - Ex: from above - ε, (), (( )), ( )( ), (( )( ))
    - But not aabaabb or (( )(( )) or abba ())(

School of Engineering | Computer Science

Context-Free Grammar

- Context-Free Grammar (CFG) Example 2.4
  - $G = (V, \Sigma, R, A)$ where $V = \{A, B, C\}$, $\Sigma = \{a, +, \times, (,)\}$
    - Rules R: $A \rightarrow A + B \mid B$
      $B \rightarrow B \times C \mid C$
      $C \rightarrow (A) \mid a$
    - String $a + a \times a$

Context-Free Grammar

- Context-Free Grammar (CFG) Example 2.4
  - $G = (V, \Sigma, R, A)$ where $V = \{A, B, C\}$, $\Sigma = \{a, +, \times, (,)\}$
    - Rules R: $A \rightarrow A + B \mid B$
      $B \rightarrow B \times C \mid C$
      $C \rightarrow (A) \mid a$
    - String $a + a \times a$

$$
\begin{aligned}
A &\Rightarrow A + B \\
&\Rightarrow A + B \times C \\
&\Rightarrow B + B \times C \\
&\Rightarrow C + B \times C \\
&\Rightarrow a + B \times C
\end{aligned}
\qquad
\begin{aligned}
&\Rightarrow a + B \times C \\
&\Rightarrow a + C \times C \\
&\Rightarrow a + a \times C \\
&\Rightarrow a + a \times a
\end{aligned}
$$

School of Engineering | Computer Science

Context-Free Grammar

- Ex 2.4 cont.
  - String $a + a \times a$ (parse tree)

$$
\boxed{\begin{aligned}
A &\rightarrow A + B \mid B \\
B &\rightarrow B \times C \mid C \\
C &\rightarrow (A) \mid a
\end{aligned}}
$$

Context-Free Grammar

- Ex 2.4 cont.
  - String a + a × a (parse tree)

$$
\begin{aligned}
A &\rightarrow A + B \mid B \\
B &\rightarrow B \times C \mid C \\
C &\rightarrow (A) \mid a
\end{aligned}
$$

$$
\begin{array}{c}
A \\
\downarrow \\
A + B \\
\downarrow \\
A + B \times C \\
\downarrow \\
B + B \times C \\
\downarrow \\
C + B \times C \\
\downarrow \\
a + B \times C \\
\downarrow \\
C \times C \\
\downarrow \\
a \times C \\
\downarrow \\
a
\end{array}
$$

a + a × a

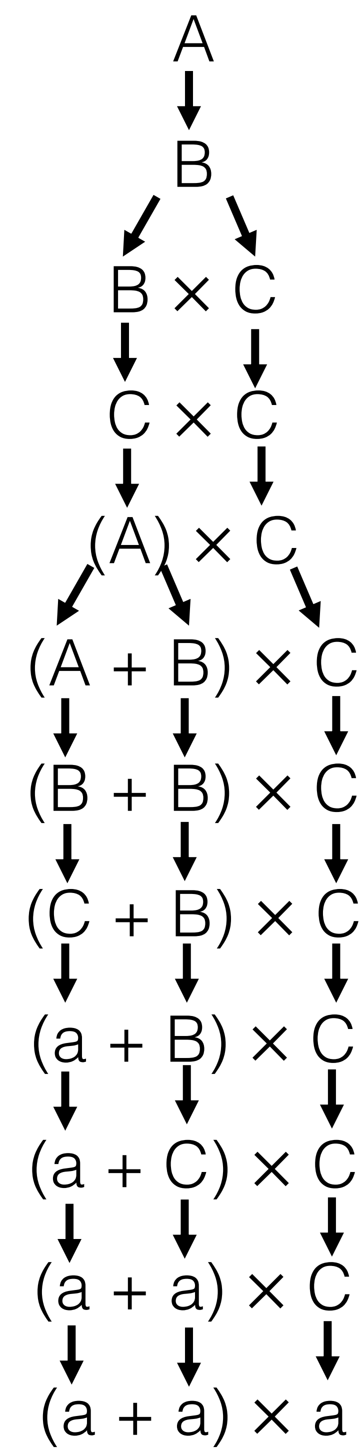
Context-Free Grammar

- Ex 2.4 cont.
  - String (a + a) × a

$A \rightarrow A + B \mid B$
$B \rightarrow B \times C \mid C$
$C \rightarrow (A) \mid a$

Context-Free Grammar

- Ex 2.4 cont.
  - String (a + a) × a
    - $$\begin{aligned} A &\Rightarrow B \\ &\Rightarrow B \times C \\ &\Rightarrow C \times C \\ &\Rightarrow (A) \times C \\ &\Rightarrow (A + B) \times C \\ &\Rightarrow (B + B) \times C \\ &\Rightarrow (C + B) \times C \\ &\Rightarrow (a + B) \times C \\ &\Rightarrow (a + C) \times C \\ &\Rightarrow (a + a) \times C \\ &\Rightarrow (a + a) \times a \end{aligned}$$

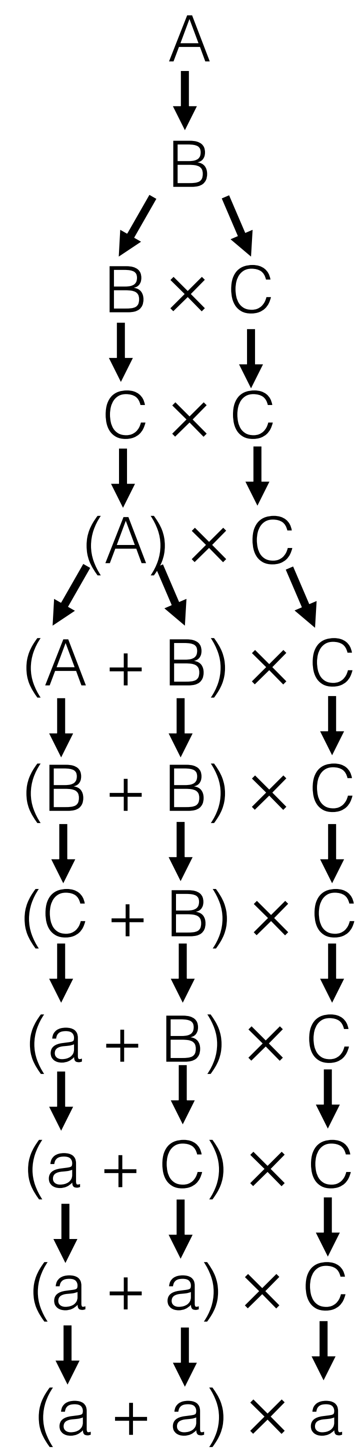


$A \rightarrow A + B \mid B$
$B \rightarrow B \times C \mid C$
$C \rightarrow (A) \mid a$

VCU School of Engineering | Computer Science

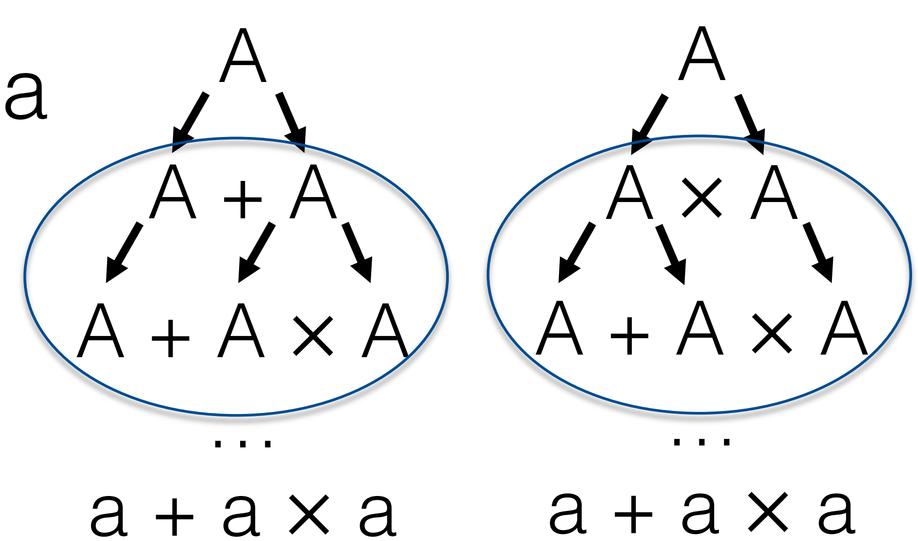
Context-Free Grammar

- Ex 2.4 cont.
  - $A \rightarrow A + B \mid B$
    $B \rightarrow B \times C \mid C$
    $C \rightarrow (A) \mid a$
  - There is just one way to derive the strings from this language
    - Ex: String $(a + a) \times a$ has only one possible parse tree, so does the string $a + a \times a$ or any other string from this language
  - The language is <u>unambiguous</u>



Ambiguity in Grammars

- Definition 2.7:
  - A string w is ambiguously derived in CFG G if w has at least two different left-most derivations (where the left-most variable is always replaced first)
  - G is ambiguous if it generates some strings ambiguously
  - Note: some languages are inherently ambiguous
    - Ex: $A \rightarrow A + A \mid A \times A \mid A \mid a$
      - String: $a + a \times a$ has two different parse trees



Creating CFGs

- Give a Context-Free Grammar for the language L = {w | w is odd and the middle symbol is 0} where Σ = {0, 1}

Creating CFGs

- Give a Context-Free Grammar for the language L = {w | w is odd and the middle symbol is 0} where Σ = {0, 1}

$$S \rightarrow 0 \mid 0S0 \mid 1S1 \mid 0S1 \mid 1S0$$

Creating CFGs

- Give a Context-Free Grammar for the language L = {w | w is even} where $\Sigma = \{0, 1\}$

Creating CFGs

- Give a Context-Free Grammar for the language L = {w | w is even} where $\Sigma = \{0, 1\}$

$$S \rightarrow \varepsilon \mid 0S0 \mid 1S1 \mid 0S1 \mid 1S0$$

Try It

1. Give a CFG for the language L = {w | w starts and ends with the same symbol} $\Sigma = \{0, 1\}$

2. Show the derivation or parse tree of the following string, 011001, using the grammar G:

$$S \rightarrow TSV \mid VST \mid \varepsilon$$

$$T \rightarrow 0$$

$$V \rightarrow 1$$

Is the grammar above ambiguous? Why or why not?

Try It

1. Give a CFG for the language L = {w | w starts and ends with the same symbol} where $\Sigma = \{0, 1\}$

$A \rightarrow 0B0 \mid 1B1 \mid 0 \mid 1$
$B \rightarrow 0B \mid 1B \mid \varepsilon$

Try It

2. Show the derivation or parse tree of the following string, 011001, using the grammar G:

$$S \rightarrow TSV \mid VST \mid \varepsilon$$

$$T \rightarrow 0$$

$$V \rightarrow 1$$

Is the grammar above ambiguous? Why or why not?

- No. There is only one possible parse tree for any string in the language.

$$\begin{array}{c} S \\ \downarrow \\ TSV \\ \downarrow \\ 0SV \\ \downarrow \\ 0VSTV \\ \downarrow \\ 01STV \\ \downarrow \\ 01VSTTV \\ \downarrow \\ 011STTV \\ \downarrow \\ 011\varepsilon TTV \\ \downarrow \\ 011\varepsilon 0TV \\ \downarrow \\ 011\varepsilon 00V \\ \downarrow \\ 011\varepsilon 001 \\ \\ 011001 \end{array}$$