Theory of Computer Science – SCJ 3203

Context-Free Grammar

Sazali Abd Manaf
Mohd Soperi Mohd Zahid

Definition 3.1.1: Context-Free Grammars

A context-free grammar is a quadruple

(V, Σ, P, S) where:

- V is a finite set of variables
- Σ (the alphabet) is a finite set of terminal symbols , where $V \cap \Sigma = \phi$
- P is a finite set of rules (production rules) written as:

$A \rightarrow \alpha$ for $A \in V$, $\alpha \in (v \cup \Sigma)^*$.

- S is the start symbol, $S \in V$

Context-Free Grammars

- Terminal symbols – elements of the alphabet
- Variables or nonterminals – additional symbols used in production rules
- Variable S (start symbol) initiates the process of generating acceptable strings.

Context-Free Grammars

- A rule is an element of the set $V \times (V \cup \Sigma)^*$.
- An A rule:

  $[A,w]$ or $A \rightarrow w$

- A null rule or lambda rule:

  $A \rightarrow \lambda$

Definition 3.1.2

Let cfg $G = (V, \Sigma, P, S)$
and $v \in (V \cup \Sigma)^*$.

The set of strings derivable from v is defined recursively as follows:

1. Basis: v is derivable from v
2. Recursive step:
   if $u=xAy$ is derivable from v,
   and $A \rightarrow w \in P$,
   then xwy is derivable from v
3. Closure: Precisely those strings constructed from v by finitely many applications of the recursive step are derivable from v

Context-Free Grammars

- Grammars are used to generate strings of a language.
- An A rule can be applied to the variable A whenever and wherever it occurs.
- No limitation on applicability of a rule – it is context free

Cfg – generating strings

- Generating a string:
  - Transform a string by applying 1 rule
- example:

P: $S \rightarrow uAv$

$A \rightarrow w$

We can derived string *uvw* as:

$S \Rightarrow uAv \Rightarrow uwv$

Cfg – generating strings

- Example 2:
  G = ({S}, {a,b}, P, S)
  P: $S \rightarrow aS \mid bS \mid \lambda$
- The following strings can be derived:
  $S \Rightarrow \lambda$
  $S \Rightarrow aS \Rightarrow a\lambda \Rightarrow a$
  $S \Rightarrow bS \Rightarrow b\lambda \Rightarrow b$
  $S \Rightarrow aS \Rightarrow aaS \Rightarrow aa\lambda \Rightarrow aa$
  $S \Rightarrow aS \Rightarrow abS \Rightarrow ab\lambda \Rightarrow ab$
  $S \Rightarrow bS \Rightarrow baS \Rightarrow ba\lambda \Rightarrow ba$
  $S \Rightarrow aS \Rightarrow abS \Rightarrow abbS \Rightarrow abb\lambda \Rightarrow abb$

Cfg – generating strings

- Example 2:

  $G = (\{S\}, \{a,b\}, P, S)$

  P: $S \rightarrow aS \mid bS \mid \lambda$

- The language above can also be defined using regular expression:

  $L(G) = (a+b)^*$

Cfg – generating strings

- Example 3:
  G = ({S,A}, {a,b}, P, S)
  P: S → AA
  A → AAA | bA | Ab | a
- The following strings can be derived:
  S => AA
  S => aA [A → a]
  S => aAAA [A → AAA]
  S => abAAA [A → bA]
  S => abaAA [A → a]
  S => abaAbA [A → Ab]
  S => abaabA [A → a]
  S => ababaa [A → a]

Cfg – generating strings

- A string w is derivable from v if there is a finite sequence of rule applications that transforms *v* to *w.*

  $v \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w$

- $v \Rightarrow^* w$ *means*

  w is derivable from *v*

Definition 3.1.3

Let $G = (V, \Sigma, P, S)$ be a cfg

1. A string $w \in (V \cup \Sigma)^*$ is a sentencial form of G if there is a derivation $S \Rightarrow^* w$ in G
2. A string $w \in \Sigma^*$ is a sentence of G
   if there is a derivation $S \Rightarrow^* w$ in G
3. The language of G, denoted by L(G), is the set $\{w \in \Sigma^* \mid S \Rightarrow^* w\}$

Context-Free Grammars

- Sentencial forms are the strings derivable from start symbol of the grammar.
- Sentences are forms that contain only terminal symbols.
- A set of strings over ∑ is context-free language if there is a context-free grammar that generates it.

Definition 3.1.4

Let G = (V, Σ, P, S) be a cfg

And $S\ast_G\Rightarrow w$ a derivation.

The derivation tree DT of $S\ast_G\Rightarrow w$ is an ordered tree that can be built iteratively as:

1. Initialize DT with root S.
2. If A => $x_1 x_2 \ldots x_n$ with $x_i \in (V \cup \Sigma)$ is the rule in derivation of string *uAv,* then add $x_1, x_2, \ldots, x_n$ as the children of A in the tree.
3. If A => λ is the rule in derivation applied to string *uAv*, then add λ as the only child of A in tree.

Example:

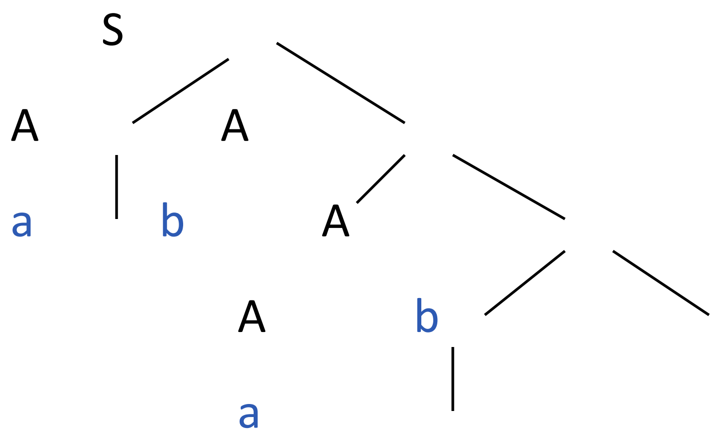
G = ({S,A}, {a, b}, P, S)

P: S → AA

A → AAA | bA | Ab | a

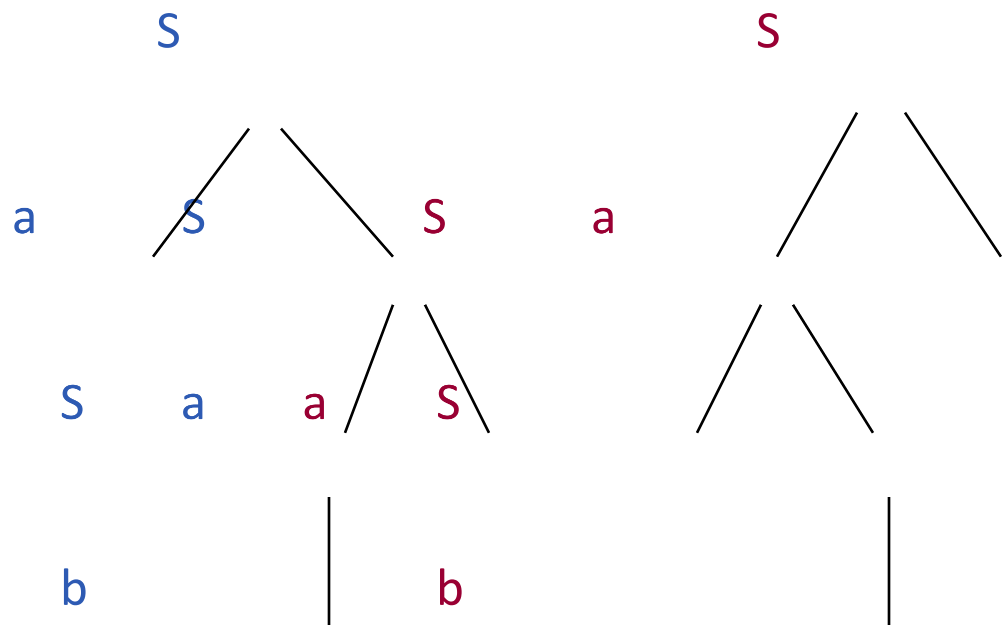
The derivation tree for

S=>AA=>aA=>abA=>abAb=>abab is:



Context-Free Grammars

S=>aS=>aSa=>aba S=>Sa=>aSa=>aba



Context-Free Grammars

- A cfg G is ambiguous if there exist >1 DT for n, where n $\in$L(G).
- Example:

G = ({S}, {a, b}, P, S)

P: S → aS | Sa | b

the string aba can be derived as:

S=>aS=>aSa=>aba

or

S=>Sa=>aSa=>aba

Example of Grammars and Languages

Example 3.2.1

- Let G be the grammar given by the production

  $S \rightarrow aSa \mid aBa$

  $B \rightarrow bB \mid b$

- Then $L(G) = \{a^n b^m a^n \mid n>0, m>0\}$

Example 3.2.2

- Let $L(G) = \{a^n b^m c^m d^{2n} \mid n \geq 0, m > 0\}$
- Then the production rules for this grammar is:

  $S \rightarrow aSdd \mid A$

  $A \rightarrow bAc \mid bc$

Example 3.2.3

- A string w is a palindrome if $w=w^R$
- The set of palindrome over {a,b} can be derived using rules:

$$S \rightarrow a \mid b \mid \lambda$$

$$S \rightarrow aSa \mid bSb$$

Example 3.2.5

- Consider the grammar:

  $S \rightarrow abScB \mid \lambda$

  $B \rightarrow bB \mid b$

- The language of this grammar consists of the set:

  $\{(ab)^n (cb^{m_n})^n \mid n \geq 0, m_n > 0\}$

Example 3.2.9

- Grammar for even-length strings over {a,b}:

$S \rightarrow aE \mid bE \mid \lambda$

$E \rightarrow aS \mid bS$

Example 3.2.12

- Consider the grammar:

$$S \rightarrow bS \mid cS \mid aB \mid \lambda$$

$$B \rightarrow aB \mid cS \mid bC \mid \lambda$$

$$C \rightarrow aB \mid bS \mid bC \mid \lambda$$

Regular Grammars

Regular Grammars

- Regular grammars play prominent role in lexical analysis and parsing of programming languages.
- Regular grammars are obtained by placing restrictions on the form of the right hand side of the rules.

Definition 3.3.1

A **regular grammar** is a cfg in which each rule has one of the following form:

1. $A \rightarrow a$
2. $A \rightarrow aB$
3. $A \rightarrow \lambda$

where A, B $\in$ V, and a $\in$ Σ

Regular Grammars

- There is at most ONE variable in a sentential form – the rightmost symbol in the string.
- Each rule application adds ONE terminal to the derived string.derivation is terminated by rules:
  - **A** $\rightarrow$ ***a*** *OR* **A** $\rightarrow$ λ

Example 3.3.1

- Consider the grammar:

G: $S \rightarrow abSA \mid \lambda$

$A \rightarrow Aa \mid \lambda$

- The equivalent regular grammar:

G_r: $S \rightarrow aB \mid \lambda$

$B \rightarrow bS \mid bA$

$A \rightarrow aA \mid \lambda$

Example 3.3.2

Syntax of Pascal in Backus-Naur Form

<*assign*> → <*var*> := <*exp*>

<*var*> → A | B | C

<*exp*> → <*var*>+<*exp*> |

<*var*> - <*exp*> |

(<*exp*>) | <*var*>*<*exp*> |

<*var*>

Example 3.3.3

Is A := B*(A+C) Syntactically correct?

<assign> → <var> := <expr>

A := <expr>

A := <var>*<expr>

A := B*<expr>

A := B*(<var>+<expr>)

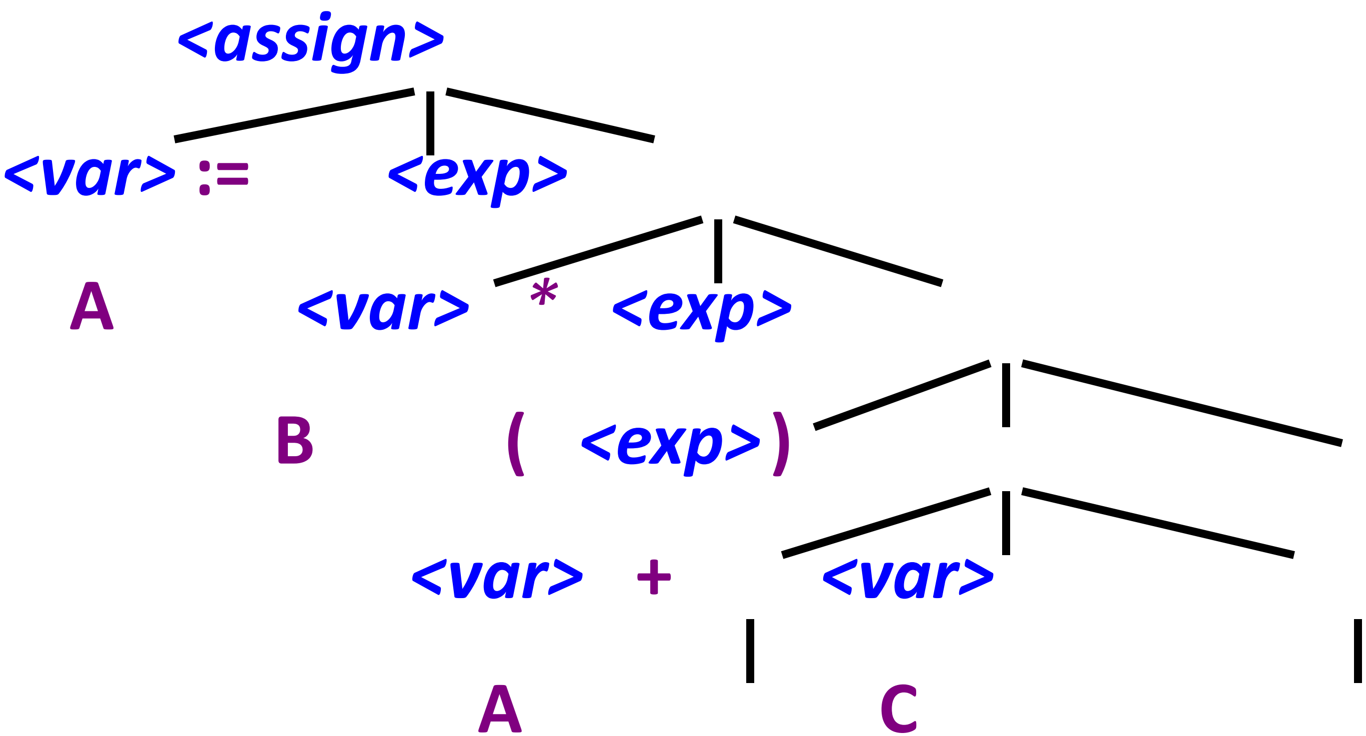
A := B*(A+<expr>)

A := B*(A+<var>)

A := B*(A+C)

Example 3.3.4

Is A := B*(A+C) Syntactically correct?



RE → RG

- Consider the regular expression:

  a^+b^*

- The regular grammar is:

  $S \rightarrow aS \mid aR$

  $R \rightarrow bR \mid \lambda$

RE → RG → RL

- The regular language L = a^+b^* can be defined as:

  L = (V, Σ, P, S)

  where:

  V = {S,R}

  Σ = {a,b}

  P: $S \rightarrow aS \mid aR$

  $R \rightarrow bR \mid \lambda$

Non-Regular Language

- The language L = a^+b^*

  can also be defined as:

  L = (V, Σ, P, S)

  where:

  V = {S,A,B}

  Σ = {a,b}

  P: S → AB

  A → aA | a

  B → bB | λ

  non-regular grammar

Derivation

- A terminal string is in the language of the grammar if it can be derived from the start symbol S using the rules of the grammar:
- Example:

$$S \rightarrow AASB \mid AAB$$

$$A \rightarrow a$$

$$A \rightarrow bbb$$

Derivation

Derivation	**Rule Applied**	
S ==> AASB	S → AASB	
==> AAAASBB	S → AASB	
==> AAAAAASBBB	S → AASB	==>
AAAAAAAABBBB	S → AAB	==>
aaaaaaaaBBBB	A → a	
==> aaaaaaaabbbbbbbbb	B → bbb	

Derivation

- Let G be the grammar :

  $S \rightarrow aS \mid aA$

  $A \rightarrow bA \mid \lambda$

- The derivation of aabb is as shown:

  $S \Rightarrow aS$

  $\Rightarrow aaA$

  $\Rightarrow aabA$

  $\Rightarrow aabbA$

  $\Rightarrow aabb\lambda \Rightarrow aabb$

Example

- Let G be the grammar :

  $S \rightarrow aS \mid bS \mid \lambda$

- The derivation of aabb is as shown:

  S ➔ aS

  ➔ abS

  ➔ abbS

  ➔ abbaS

  ➔ abbaaS ➔ abbaa

➔ a^*b^*

Example

- Let G be the grammar :

  S → aSb | ab

- The derivation of aabb is as shown:

  S → aSb

  → aaSbb

  → aaaSbbb

  → aaaaSbbbb

  → aaaabbbb

→ **$a^n b^n$**

Selected Exercises

Exercise 1

Let G be the grammar:

$S \rightarrow SAB \mid \lambda$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid \lambda$

1. Give a leftmost derivation of aabaaabb. Build the derivation tree.
2. Give another derivation tree of aabaaabb which is different from that of (a).
3. Give a regular expression for L(G).

Exercise 2

For each of the following context free grammar, use set notation to define the language generated by the grammar.

- $S \rightarrow aaSB \mid \lambda$
  $B \rightarrow bB \mid b$
- $S \rightarrow aSbb \mid A$
  $A \rightarrow cA \mid c$
- $S \rightarrow abSdc \mid A$
  $A \rightarrow cdAba \mid \lambda$

Exercise 2 (cont.)

For each of the following context free grammar, use set notation to define the language generated by the grammar.

- $S \rightarrow aSb \mid A$
  $A \rightarrow cAd \mid cBd$
  $B \rightarrow aBb \mid ab$

- $S \rightarrow aSB \mid aB$
  $B \rightarrow bb \mid b$

References