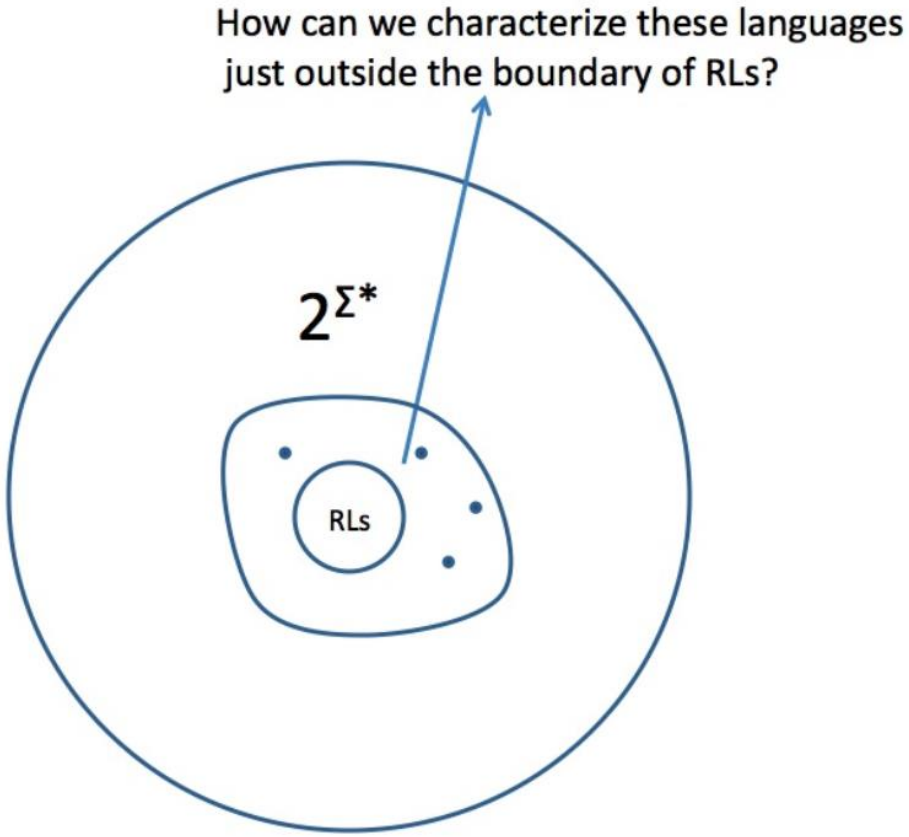
Formal languages and automata

Context Free Languages

https://t.me/fla_uog

mh.olyaee@gmail.com

Where are we?


How can we characterize these languages
just outside the boundary of RLs?
2^{Σ^*}
RLs

Example1

- We showed that $L = \{a^n b^n \mid n \geq 0\}$ was not regular.
  - No DFA
  - No Regular Expression
- How can we describe such languages?
- Remember: the description has to be finite!

Example1

- Consider $L = \{a^n b^n \mid n \geq 0\}$ again.
- How can we generate such strings?
  - Remember DFAs did recognition, not generation.
- Consider the following inductive way to generate elements of L
  - Basis: ϵ is in the language
  - Recursion: If the string w is in the language, then so is the string awb.
- $\epsilon \rightarrow ab \rightarrow aabb \cdots \rightarrow a^{55}b^{55} \cdots$
- Looks like we have simple and finite length process to generate all the strings in L
- How can we generalize this kind of description?

Example 2

- Consider $L = \{w \mid n_a(w) = n_b(w)\}$.
- Now consider the following inductive way to generate elements of L
  - Basis: ϵ is in the language
  - Recursion 1: If the string w is in the language, then so are awb and bwa
  - Recursion 2: If the strings w and v are in the language, so is wv.
- The first recursion rules makes sure that the a's and b's are generated in the same number (regardless of order)
- The second recursion takes any two strings each with equal number of a's and b's and generates a new such string by concatenating them.

Grammars

- Grammars provide the generative mechanism to generate all strings in a language.
- A grammar is essentially a collection of substitution rules, called productions
- Each production rule has a left-hand-side and a right-hand-side.

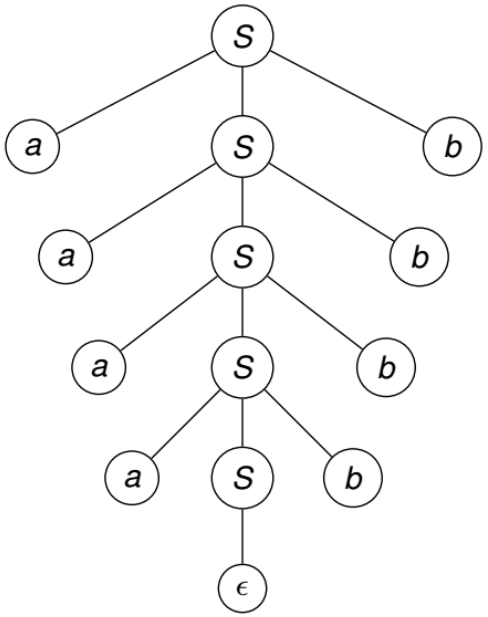
Grammars-An example

- Consider once again $L = \{a^n b^n \mid n \geq 0\}$
- Basis: ϵ is in the language
  - Production: $S \rightarrow \epsilon$
- Recursion: If w is in the language, then so is the string awb.
  - Production: $S \rightarrow aSb$
- S is called a variable or a nonterminal symbol
- a, b etc., are called terminal symbols
- One variable is designated as the start variable or start symbol.

How does a grammar work?

- Consider the set of rules $R = \{S \to \epsilon, S \to aSb\}$
- Start with the start variable S
- Apply the following until all remaining symbols are terminal.
  - Choose a production in R whose left-hand sides matches one of the variables.
  - Replace the variable with the rule's right hand side.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$
- The string $aaaabbbb$ is in the language L
- The sequence of rule applications above is called a derivation.

Parse Trees



The terminals concatenated from left to right give us the string.

- Derivations can also be represented with a parse tree.
- The leaves constitute the yield of the tree.
- Terminal symbols can occur only at the leaves.
- Variables can occur only at the internal nodes.

A grammar for English

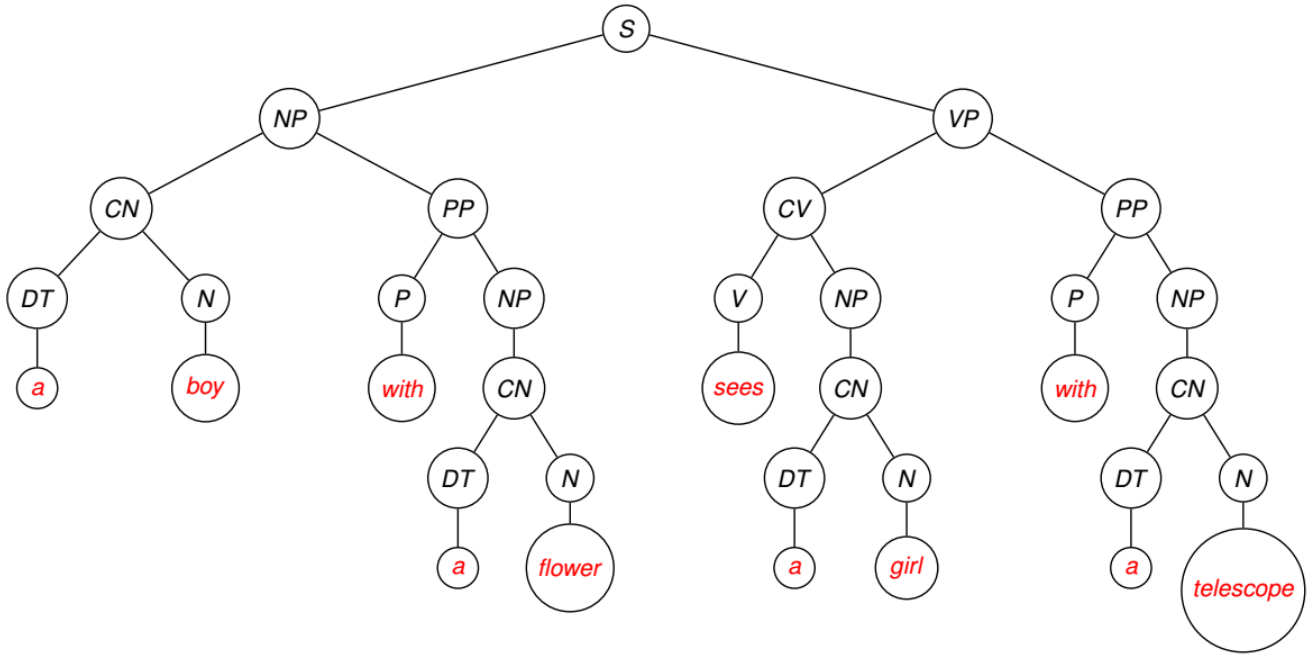
$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow CN \mid CN\ PP \\
VP &\rightarrow CV \mid CV\ PP \\
PP &\rightarrow P\ NP \\
CN &\rightarrow DT\ N \\
CV &\rightarrow V \mid V\ NP \\
DT &\rightarrow \text{a} \mid \text{the} \\
N &\rightarrow \text{boy} \mid \text{girl} \mid \text{flower} \mid \text{telescope} \\
V &\rightarrow \text{touches} \mid \text{likes} \mid \text{sees} \mid \text{gives} \\
P &\rightarrow \text{with} \mid \text{to}
\end{aligned}
$$

Nomenclature:

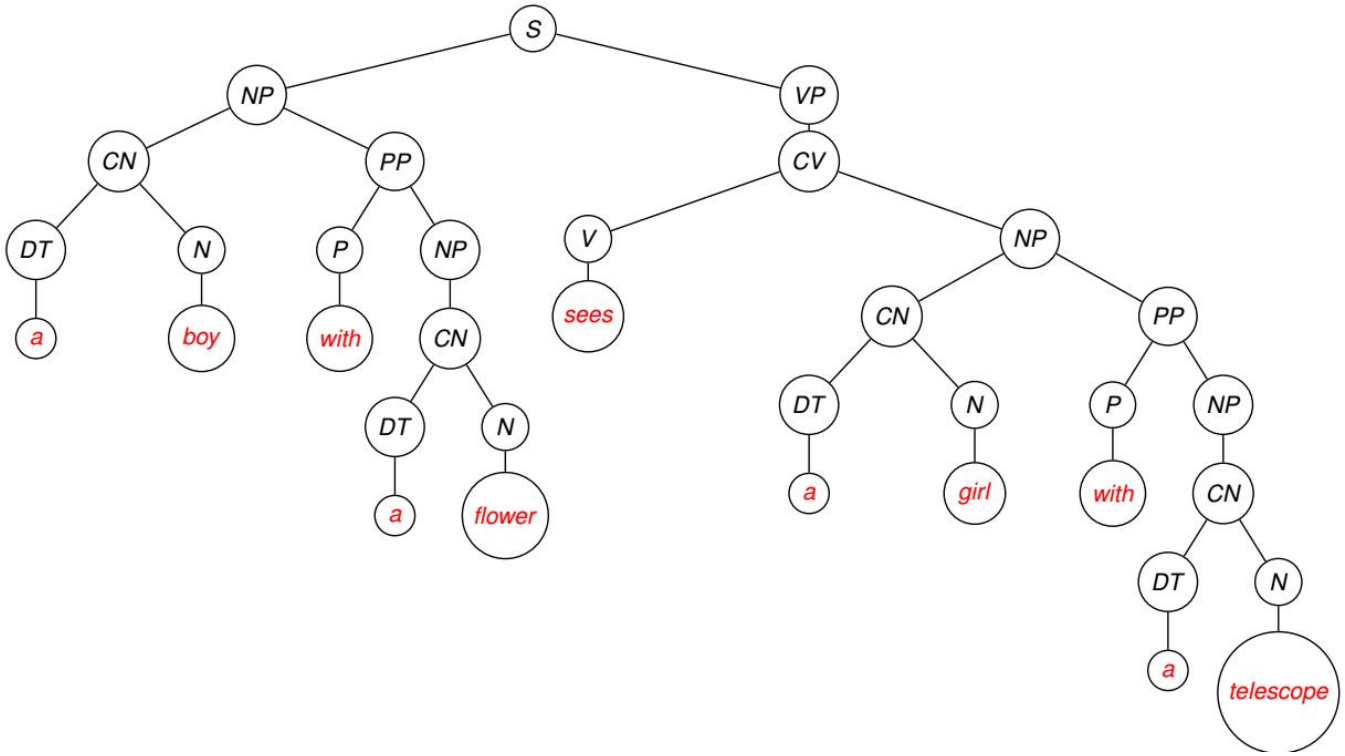
- *S*: Sentence
- *NP*: Noun Phrase
- *CN*: Complex Noun
- *PP*: Prepositional Phrase
- *VP*: Verb Phrase
- *CV*: Complex Verb
- *P*: Preposition
- *DT*: Determiner

$$
\begin{array}{rcl}
S & \to & NP\ VP \\
NP & \to & CN \mid CN\ PP \\
VP & \to & CV \mid CV\ PP \\
PP & \to & P\ NP \\
CN & \to & DT\ N \\
CV & \to & V \mid V\ NP \\
DT & \to & \text{a} \mid \text{the} \\
N & \to & \text{boy} \mid \text{girl} \mid \text{flower} \mid \\
 & & \text{telescope} \\
V & \to & \text{touches} \mid \text{likes} \mid \\
 & & \text{sees} \mid \text{gives} \\
P & \to & \text{with} \mid \text{to}
\end{array}
$$

$$
\begin{array}{rcl}
S & \Rightarrow & NP\ VP \\
 & \Rightarrow & CN\ PP\ VP \\
 & \Rightarrow & DT\ N\ PP\ VP \\
 & \Rightarrow & \text{a}\ N\ PP\ VP \\
 & \Rightarrow & \ldots \\
 & \Rightarrow & \text{a boy with a flower}\ VP \\
 & \Rightarrow & \text{a boy with a flower}\ CV\ PP \\
 & \Rightarrow & \ldots \\
 & \Rightarrow & \text{a boy with a flower sees a girl} \\
 & & \text{with a telescope}
\end{array}
$$


S
NP
VP
CN
PP
CV
PP
DT
N
P
NP
V
NP
P
NP
a
boy
with
CN
sees
CN
with
CN
DT
N
DT
N
DT
N
a
flower
a
girl
a
telescope

Alternate Parse tree



Structural Ambiguity

- A set of rules can assign multiple structures to the same string.
- Which rule one chooses determines the eventual structure.
  - $VP \rightarrow CV \mid CV\ PP$
  - $CV \rightarrow V \mid V\ NP$
  - $NP \rightarrow CN \mid CN\ PP$
  - $\cdots$ [$_{VP}$ [$_{CV}$ sees [$_{NP}$ a girl] [$_{PP}$ with a telescope]].
  - $\cdots$ [$_{VP}$ [$_{CV}$ sees] [$_{NP}$ [$_{CN}$ a girl] [$_{PP}$ with a telescope]].
    - (Not all brackets are shown!)

Grammar definition(Remember)

- A Grammar is a 4-tuple $G = (V, \Sigma, R, S)$ where
  - V is a finite set of variables
  - Σ is a finite set of terminals, disjoint from V.
  - R is a set of rules of the $X \to Y$
  - $S \in V$ is the start variable

Context Free Grammar

- In general $X \in (V \cup \Sigma)^+$ and $Y \in (V \cup \Sigma)^*$
- A context-free grammar is a grammar where all rules have $X \in V$ (remember $V \subset (V \cup \Sigma)^+$)
  - The substitution is independent of the context V appears in.
- The right hand side of the rules can be any combination of variables and terminals, including ϵ (hence $Y \in (V \cup \Sigma)^*$).

$$V \rightarrow (V \cup \Sigma)^*$$

Designing Context free grammars

- Consider once again the language $L = \{w \mid n_a(w) = n_b(w)\}$.
- The grammar for this language is $G = (\{S\}, \{a, b\}, R, S)$ with R as follows:
  1. $S \to aSb$
  2. $S \to bSa$
  3. $S \to SS$
  4. $S \to \epsilon$
- From now we will only list the productions, the others will be implicit.
- We will also combine productions with the same left-hand side using | symbol.
- $S \to aSb \mid bSa \mid SS \mid \epsilon$

Some Examples

$L = \{a^n b^{2n} \mid n \geq 0\}$

Some Examples

$L = \{a^n b^{2n} | n \geq 0\}$

$S \rightarrow aSbb|\varepsilon$

Some Examples

- $L = \{a^n b^m c^m d^{2n} \mid m, n \geq 0\}$

Some Examples

- $L = \{a^n b^m c^m d^{2n} \mid n \geq 0\}$

$S \rightarrow aSdd|aAdd|\varepsilon$

$A \rightarrow bAc|\varepsilon$

Some Examples

$L = \{a^n b^m \mid m \leq n \leq 2m\}$

Some Examples

$L = \{a^n b^m \mid m \leq n \leq 2m\}$

$S \rightarrow aSb|aaSb|\varepsilon$

Some Examples

$L = \{a^n b^m c^n | n, m > 0\}$

Some Examples

$L = \{a^n b^m c^n | n, m > 0\}$

$S \rightarrow aSc | aAc$

$A \rightarrow bA | b$

Some Examples

$L = \{a^n b^m c^n | n \geq 0, m > 0\}$

Some Examples

$L = \{a^n b^m c^n | n \geq 0, m > 0\}$

$S \rightarrow aSc|A$
$A \rightarrow Ab|b$

Some Examples

$$L = \{a^n b^m c^{2n} c^m | n, m > 0\}$$

Some Examples

$L = \{a^n b^m c^m c^{2n} | n, m > 0\}$

$S \rightarrow aScc | aAcc$

$A \rightarrow bAc | bc$

Homework1

- $L = \{a^n b^m | n \neq m\}$
- $L = \{a^n b^m c^k | n = m \ or \ m = k\}$
- $L = \{a^n b^m c^k | n > m + k\}$

Summary

- Describing nonregular languages
- Grammars as finite descriptions of infinite sets
- Context-free Grammars and context-free languages
- Derivations and parse trees
- Ambiguity
- Writing grammars

Grammar for arithmetic expressions

- $L \to a \mid b \mid \cdots \mid z$ (letters)
- $D \to 0 \mid \cdots \mid 9$ (digits)
- $V \to L \mid V\ L \mid V\ D$ (variables)
- $N \to D \mid N\ D$ (positive numbers)
- $F \to V \mid N \mid (E)$ (factors)
- $T \to F \mid T * F \mid T/F$ (terms)
- $E \to T \mid E + T \mid E - T$ (expressions)
- E is the start symbol.

Let us generate $(v23 + 456) * k23/(a - b * 34)$ as an exercise.

Exhaustive search parsing

- Suppose w is a given string and you want to determine that w belongs to L or not.
- Example:
- $S \rightarrow SS|aSb|bSa|\varepsilon$
- w=aabb

Round1

- $S \rightarrow SS$
- $S \rightarrow aSb$
- $S \rightarrow bSa$
- $S \rightarrow \varepsilon$
- w=aabb

- Extending $S \rightarrow SS$:
  - $S \rightarrow SS \rightarrow SSS$
  - $S \rightarrow SS \rightarrow aSbS$
  - $S \rightarrow SS \rightarrow bSaS$
  - $S \rightarrow SS \rightarrow S$
-

- Extending $S \rightarrow aSb$:
  - $S \rightarrow aSb \rightarrow aSSb$
  - $S \rightarrow aSb \rightarrow aaSbb$
  - $S \rightarrow aSb \rightarrow abSab$
  - $S \rightarrow aSb \rightarrow ab$
- We can eliminate some of these extensions in the next round!
- w will be obtained in the next round as:
  - $S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$

Weaknesses

- Low performance
- Infinite loop
  - In previous example, let w as *abb*

Weaknesses

- Low performance
- Infinite loop
  - In previous example, let w as abb
  - Sometimes it is possible to solve this problem by re-writing the grammar such as below:
  - $S \rightarrow SS|aSb|bSa|ab|ba$

Ambiguity

- Remember a boy with a flower sees a girl with a telescope?
- We say that a grammar generates a string ambiguously, if the string has two different parse trees (not just two different derivations)
- A derivation of a string *w* in a grammar *G* is a leftmost derivation if at every step, the leftmost remaining variable is the one replaced.

Definition

A string w is derived ambiguously in context-free grammar G if it has two or more different leftmost derivations. Grammar G is ambiguous if it generates some string ambiguously.

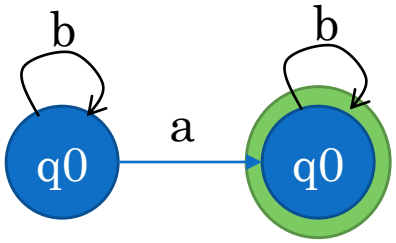
- Sometimes an ambiguous grammar can be transformed into an unambiguous grammar for the same language.
- Some context-free grammars can be generated only by ambiguous grammars. These are known as inherently ambiguous languages.
  - $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$

Example

- $L = \{b^n a b^m | n, m \geq 0\}$

Example

- $L = \{b^n a b^m | n, m \geq 0\}$



- $S \rightarrow bS|aA$
- $A \rightarrow bA|\varepsilon$

Example

- $S \rightarrow bS|Sb|a$

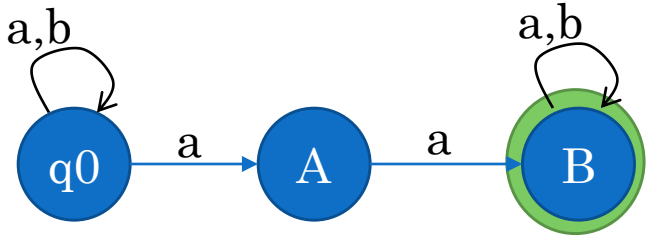
Example

- $S \rightarrow bS|Sb|a$
- w=bbab
- $S \rightarrow bS \rightarrow bSb \rightarrow bbSb \rightarrow bbab$
- $S \rightarrow Sb \rightarrow bSb \rightarrow bbSb \rightarrow bbab$

آیا یک گرامر منظم می تواند مبهم باشد؟

آیا یک گرامر منظم می تواند مبهم باشد؟

- بله! وقتی که از روی یک NFA بدست آمده باشد.



$$S \rightarrow aS|bS|aA$$
$$A \rightarrow aB$$
$$B \rightarrow aB|bB|\varepsilon$$

آیا یک زبان منظم می تواند ذاتا مبهم باشد؟

Grammar Transformation

- Some types of productions cause problems in some uses of grammars.
- ϵ-productions: $A \rightarrow \epsilon$
  - Intermediate sentential forms in a derivation get shorter and this has computational implications.
- Unit productions: $A \rightarrow B$.
  - Such a rule does not achieve much except for lengthening the derivation sequence.
  - There may be inadvertent "infinite loops": e.g., if $A \stackrel{*}{\Rightarrow} A$

- Removing useless symbols
  - Symbols which cannot reach to terminals
  - Symbols which cannot be accessible from S

Removing ε

- If $\epsilon \in L$, then we can not do much. $S \to \epsilon$ is needed for this.
- For all rules of the type $A \to \epsilon$ and A is not the start symbol, we proceed as follows:
- For occurrence of an A on the right-hand side of a rule, we add a rule with that occurence deleted.
  - For a rule like $R \to uAv$, we add the rule $R \to uv$ (either u or v not ϵ)
  - For a rule like $R \to A$, we add $R \to \epsilon$, unless we removed $R \to \epsilon$ earlier.
  - For a rule with multiple occurences of A, we add one rule for each combination. $R \to uAvAw$ would add $R \to uvAw$, $R \to uAvw$, and $R \to uvw$.

Removing ε

- Consider

$$
\begin{array}{lcl}
S & \rightarrow & ASA \mid aB \\
A & \rightarrow & B \mid S \\
B & \rightarrow & b \mid \epsilon
\end{array}
$$

Removing ε

- Consider

$$\begin{aligned} S &\rightarrow ASA \mid aB \\ A &\rightarrow B \mid S \\ B &\rightarrow b \mid \epsilon \end{aligned}$$

- Add a new start symbol S_0

$$\begin{aligned} \mathbf{S_0} &\rightarrow \mathbf{S} \\ S &\rightarrow ASA \mid aB \\ A &\rightarrow B \mid S \\ B &\rightarrow b \mid \epsilon \end{aligned}$$

- Remove $B \rightarrow \epsilon$

$$\begin{aligned} \mathbf{S_0} &\rightarrow \mathbf{S} \\ S &\rightarrow ASA \mid aB \mid \mathbf{a} \\ A &\rightarrow B \mid S \mid \epsilon \\ B &\rightarrow b \end{aligned}$$

- Remove $A \rightarrow \epsilon$

$$\begin{aligned} \mathbf{S_0} &\rightarrow \mathbf{S} \\ S &\rightarrow ASA \mid aB \mid a \mid \\ &\quad\ \mathbf{SA} \mid \mathbf{AS} \mid \mathbf{S} \\ A &\rightarrow B \mid S \\ B &\rightarrow b \end{aligned}$$

Removing unit productions

- To remove a unit rule like $A \to B$,
  - We first add to the grammar a rule $A \to u$ whenever $B \to u$ is in the grammar, unless this is a unit rule previously removed.
  - We then delete $A \to B$, from the grammar.
- We repeat these until we eliminate all unit rules.

- After ϵ-rule removal

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S \\ A &\rightarrow B \mid S \\ B &\rightarrow b \end{aligned}$$

- Remove $S \rightarrow S$

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \\ A &\rightarrow B \mid S \\ B &\rightarrow b \end{aligned}$$

- Remove $S_0 \rightarrow S$

$$\begin{aligned} S_0 &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \\ S &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \\ A &\rightarrow B \mid S \\ B &\rightarrow b \end{aligned}$$

- After $S_0 \to S$ removal

$$\begin{aligned} S_0 &\to ASA \mid aB \mid a \mid SA \mid AS \\ S &\to ASA \mid aB \mid a \mid SA \mid AS \\ A &\to B \mid S \\ B &\to b \end{aligned}$$

- Remove $A \to B$

$$\begin{aligned} S_0 &\to ASA \mid aB \mid a \mid SA \mid AS \\ S &\to ASA \mid aB \mid a \mid SA \mid AS \\ A &\to b \mid S \\ B &\to b \end{aligned}$$

- Remove $A \to S$

$$\begin{aligned} S_0 &\to ASA \mid aB \mid a \mid SA \mid AS \\ S &\to ASA \mid aB \mid a \mid SA \mid AS \\ A &\to b \mid ASA \mid aB \mid a \mid SA \mid AS \\ B &\to b \end{aligned}$$

Example

- $S \rightarrow AC|BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow CF|b$
- $C \rightarrow cC|D$
- $D \rightarrow aD|BD|C$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$

Example

- $S \rightarrow AC|BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow CF|b$
- $C \rightarrow cC|D$
- $D \rightarrow aD|BD|C$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$

Which symbols directly access to terminals?

Example

- $S \rightarrow AC|BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow CF|b$
- $C \rightarrow cC|D$
- $D \rightarrow aD|BD|C$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$

Which symbols directly access to terminals?

Example

- $S \rightarrow AC|BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow CF|b$
- $C \rightarrow cC|D$
- $D \rightarrow aD|BD|C$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$

Find the other symbols which can access to terminals by B and F.

Example

- $S \rightarrow AC|BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow CF|b$
- $C \rightarrow cC|D$
- $D \rightarrow aD|BD|C$
- $E \rightarrow aA|BSA$ by A!
- $F \rightarrow bB|b$

Find the other symbols which can access to terminals by B and F.

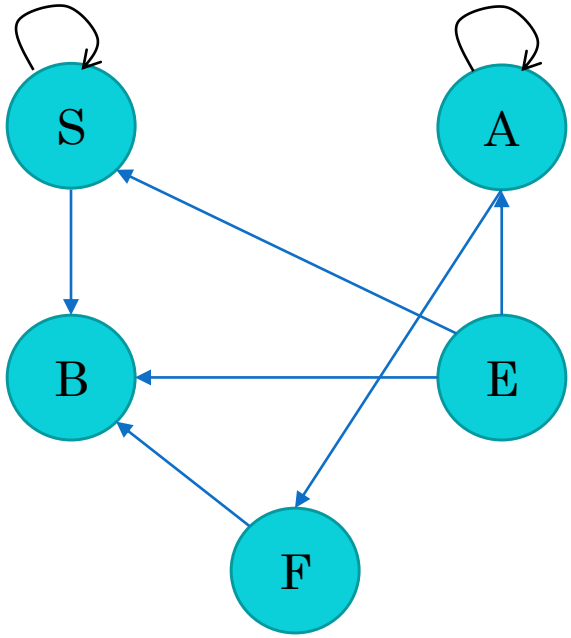
Example

- $S \rightarrow BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow b$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$

Which symbols can be reached from S?

Example

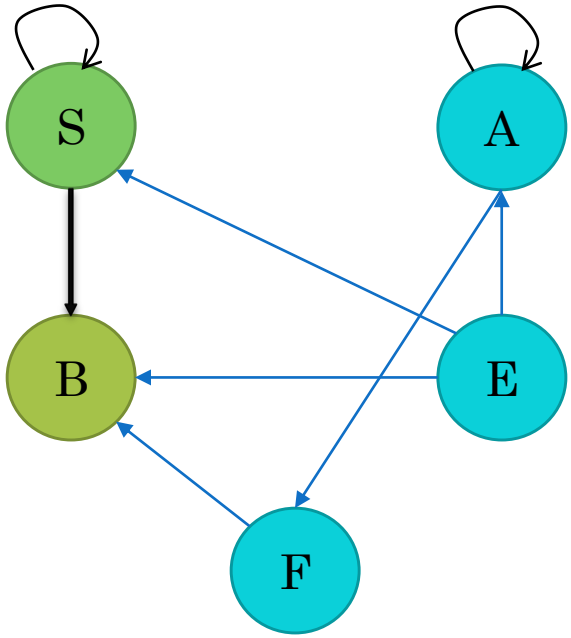
- $S \rightarrow BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow b$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$



Which symbols can be reached from S?

Example

- $S \rightarrow BS|B$
- $A \rightarrow aA|aF$
- $B \rightarrow b$
- $E \rightarrow aA|BSA$
- $F \rightarrow bB|b$



Finally the grammar decreases to:

- $S \rightarrow BS|B$
- $B \rightarrow b$

Chomsky Normal Form

- CFGs in certain standard forms are quite useful for some computational problems.

Chomsky Normal Form

A context-free grammar is in Chomsky normal form(CNF) if every rule is either of the form

$$A \to BC \text{ or } A \to a$$

where a is a terminal and A, B, C are variables – except B and C may not be the start variable. In addition, we allow the rule $S \to \epsilon$ if necessary.

- Grammar after ϵ and unit production removal

$$
\begin{aligned}
S_0 &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
S &\rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
A &\rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS \\
B &\rightarrow b
\end{aligned}
$$

- Remove $S_0 \rightarrow ASA$ and add $S_0 \rightarrow AA_1$ and $A_1 \rightarrow SA$
- Remove $S \rightarrow ASA$ and add $S \rightarrow AA_1$ ($A_1 \rightarrow SA$ already added)
- Remove $A \rightarrow ASA$ and add $A \rightarrow AA_1$ ($A_1 \rightarrow SA$ already added)
- Replace $S_0 \rightarrow aB$ with $S_0 \rightarrow UB$ and $U \rightarrow a$
- Replace $S \rightarrow aB$ with $S \rightarrow UB$ ($U \rightarrow a$ already added)
- Replace $A \rightarrow aB$ with $A \rightarrow UB$ ($U \rightarrow a$ already added)

- Final grammar in Chomsky normal form

$$
\begin{aligned}
S_0 &\rightarrow AA_1 \mid UB \mid a \mid SA \mid AS \\
S &\rightarrow AA_1 \mid UB \mid a \mid SA \mid AS \\
A &\rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS \\
A_1 &\rightarrow SA \\
U &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Example2

- $S \rightarrow aABC|a$
- $A \rightarrow aA|a$
- $B \rightarrow bcB|bc$
- $C \rightarrow cC|c$

Greibach Normal Form (GNF):

- Is a type of normal form for context-free grammars. In GNF, all production rules have the form:
- $A \rightarrow a\alpha$
- $S \rightarrow \varepsilon$
- where A is a non-terminal symbol($A \in V^*$), a is a terminal symbol, and α is a string of non-terminals.
- The right-hand side can start with a single terminal symbol followed by non-terminals.

Greibach Normal Form (GNF):

- $G1$:
  - $S \rightarrow aAB|bB$
  - $A \rightarrow bBA|b$
  - $B \rightarrow bB|b$

- $G2$:
  - $S \rightarrow aSB|b$
  - $B \rightarrow bBB|b$