Decidable properties of CFL

Several undecidable are there …

- Membership question.
- Empty?
- Infinite/not ?

Membership question

- Given the CFG G and string w, we ask is $w \in L(G)$?
- There is a $O(n^3)$ algorithm where $|w| = n$, which is called the CYK algorithm.
  - This is a parsing technique whereby one can create the parse tree if the string is in the language.
  - Since this works for any CFG (not restricted to a subclass), this is one of universal parsers.

- If $w = \epsilon$, we verify to find whether S is nullable or not.
- Else, we convert the CFG in to CNF first.
- With CNF form the parse tree is a binary tree.
- And the string w can be derived in exactly $2|w| - 1$ steps.
- The parse tree will have exactly this many variables.

- We can list all possible derivations having $2|w| - 1$ steps.
- We verify whether, any, gave the string.
- But, this is an exponential time algorithm.

- There is a much more efficient technique based on the idea of “dynamic programming”.
- This is called the CYK algorithm.
- Also called the table-filling or tabulation algorithm.

---

[3] It is named after three people, each of whom independently discovered essentially the same idea: J. Cocke, D. Younger, and T. Kasami.

CYK algorithm

- Let $w = a_1 a_2 \cdots a_n$ be the given string.
- We fill a table, as shown, for example when $w = a_1 a_2 \cdots a_5$

The table entry X_{ij} is the set of variables A such that $A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j$.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

- If $S \in X_{1n}$ then $S \overset{*}{\Rightarrow} w$
- To find X_{1n} we need to fill the table, in a bottom-up fashion.

The table entry X_{ij} is the set of variables A such that $A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j$.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

–	$\{B\}$	$\{B\}$		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

$$
\begin{array}{lcl}
S & \rightarrow & AB \mid BC \\
A & \rightarrow & BA \mid a \\
B & \rightarrow & CC \mid b \\
C & \rightarrow & AB \mid a
\end{array}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

–	{S,A,C}			
–	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	*a*	*a*	*b*	*a*

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

{S,A,C}				
–	{S,A,C}			
–	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	*a*	*a*	*b*	*a*

Parse tree

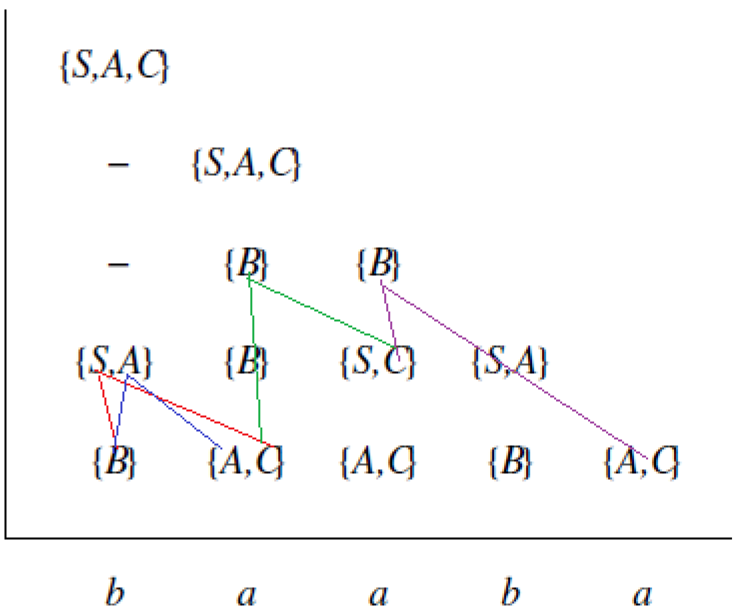
- Parse tree can be found by keeping track of some side information.

We shall test for membership in $L(G)$ the string $\boxed{baaba}$

$$\boxed{\begin{array}{lcl} S & \rightarrow & AB \mid BC \\ A & \rightarrow & BA \mid a \\ B & \rightarrow & CC \mid b \\ C & \rightarrow & AB \mid a \end{array}}$$

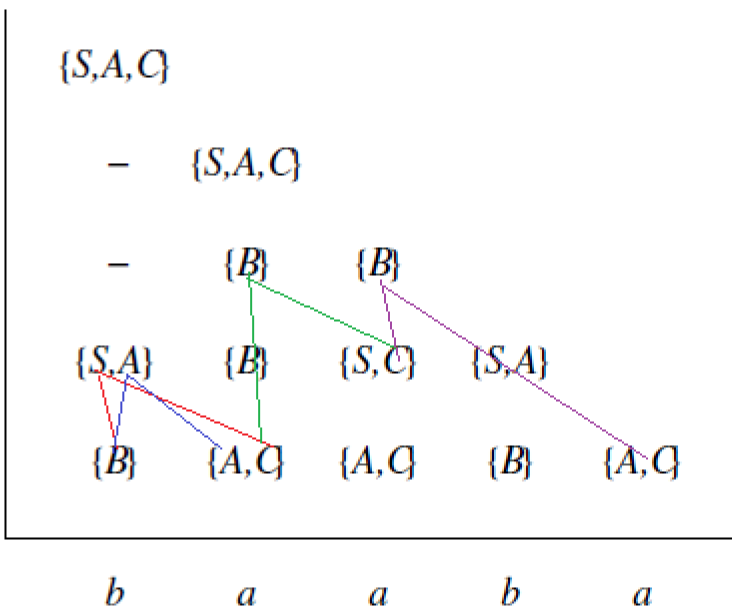
We shall test for membership in $L(G)$ the string $\boxed{baaba}$

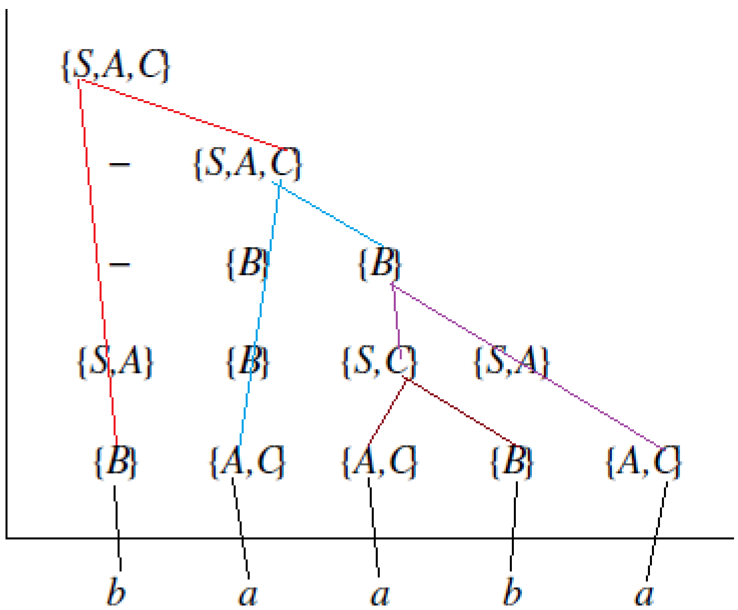
$$
\begin{array}{|lcl|}
\hline
S & \rightarrow & AB \mid BC \\
A & \rightarrow & BA \mid a \\
B & \rightarrow & CC \mid b \\
C & \rightarrow & AB \mid a \\
\hline
\end{array}
$$



We shall test for membership in $L(G)$ the string $\boxed{baaba}$

$$
\boxed{\begin{array}{lcl}
S & \rightarrow & AB \mid BC \\
A & \rightarrow & BA \mid a \\
B & \rightarrow & CC \mid b \\
C & \rightarrow & AB \mid a
\end{array}}
$$





Time complexity of CYK

- $O(n^3)$

Empty ?

- This is easy.
- Is S generating ?

Infinite or not?

- **Algorithm**.

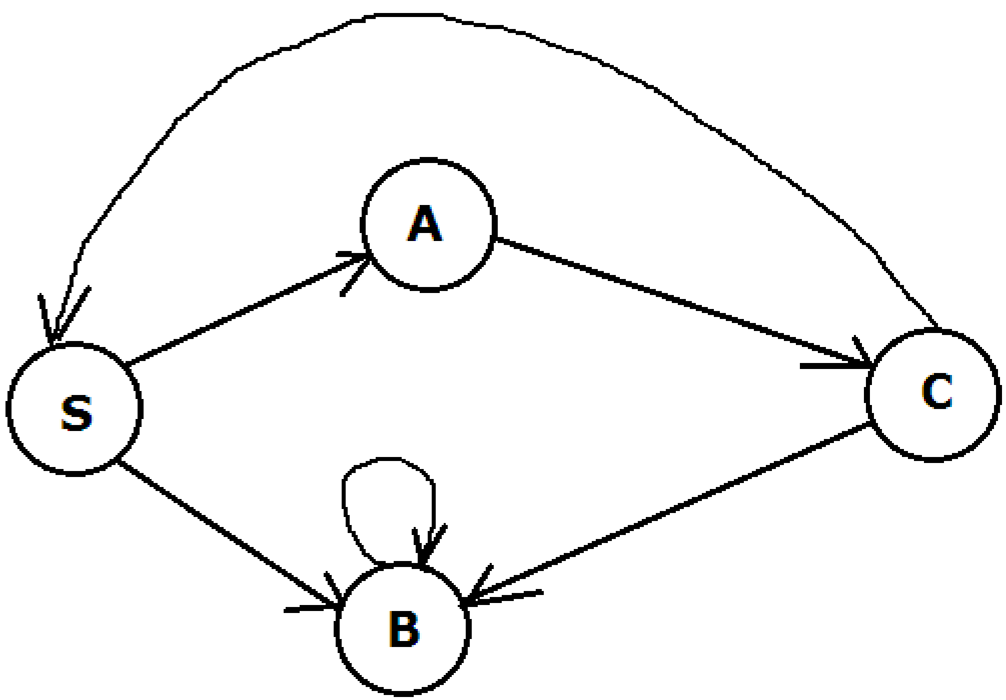
1. Remove useless symbols.
2. Remove unit and ϵ productions
3. Create dependency graph for variables
4. If there is a loop in the dependency graph then the language is infinite, else not.

Example

- $S \to AB, A \to aCb|a, B \to bB|b, C \to cBS$

Example

- $S \rightarrow AB, A \rightarrow aCb|a, B \rightarrow bB|b, C \rightarrow cBS$



Some undecidable properties ☹

- Let G_1 and G_2 be two CFGs.
- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1)$ is regular ?
- Is $L(G_1) \subseteq L(G_2)$?
- Is $L(G_1) = L(G_2)$?
- Is $L(G_1) \cap L(G_2) = \phi$?
- Is $L(G_1)$ ambiguous? (inherent ambiguity)
- Is G_1 ambiguous?