Automata theory and its applications

Lecture 3-4: Chomsky hierarchy-Context sensitive and free languages

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences

October 24, 2012

Outline

Context-sensitive grammar

Context-sensitive grammar (CSG)

A formal grammar $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ of the production rules of the form

$$\alpha A \beta \to \alpha \gamma \beta \text{ such that } \gamma \neq \varepsilon.$$

Note that for every CSG G, $\varepsilon \notin L(G)$.

Noncontracting grammar (NCG)

A formal grammar $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ of the production rules of the form

$$\alpha \to \beta \text{ such that } |\alpha| \leqslant |\beta|.$$

CSG $\equiv$ NCG

Left as homework.

1. Warm-up exercise: First transform the following NCG into a CSG,

$$L = \{a^n b^n c^n \mid n \geqslant 1\} : S \to aSBc \mid abc,\ cB \to Bc,\ bB \to bb.$$

2. Prove that each NCG can be transformed into an equivalent CSG.

Linear-bounded automata (LBA)

Intuitively, a LBA is a (single-tape) nondeterministic TM using linear space.

Formally, a LBA is a nondeterministic TM s.t.

- Its input alphabet includes two special symbols # and \$, the left and right endmarkers,
- The LBA has no moves left from # or right from \$, nor may print another symbol over # or \$.

  The language accepted by a linear-bounded TM M,

$$\{w \mid w \in (\Sigma - \{\#, \$\})^*, q_0 \# w \$ \vdash_M^* \alpha q \beta \text{ for some } q \in F\}$$

From LBA to CSG

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a LBA. The CSG G

- first nondet. generates a finite word

$$\left[\begin{array}{c} a_1 \\ q_0\#a_1 \end{array}\right]\left[\begin{array}{c} a_2 \\ a_2 \end{array}\right]\cdots\left[\begin{array}{c} a_{n-1} \\ a_{n-1} \end{array}\right]\left[\begin{array}{c} a_n \\ a_n\$ \end{array}\right],$$

  with the intention that $a_1 \ldots a_n$ is the input of M.

- then G simulates the computation of M over $a_1 \ldots a_n$, by rewriting the second components of the generated word.

From LBA to CSG

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a LBA. The CSG G

- first nondet. generates a finite word

$$\begin{bmatrix} a_1 \\ q_0\#a_1 \end{bmatrix}\begin{bmatrix} a_2 \\ a_2 \end{bmatrix}\cdots\begin{bmatrix} a_{n-1} \\ a_{n-1} \end{bmatrix}\begin{bmatrix} a_n \\ a_n\$ \end{bmatrix},$$

Formally, $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ is defined as follows.

- $S = A_1$,
- $\mathcal{P}$ includes the following rules,
  - $A_1 \to [a, q_0\#a]A_2$, $A_1 \to [a, q_0\#a\$]$, $A_2 \to [a, a]A_2$, $A_2 \to [a, a\$]$.
  - If $\delta(q, \#) = (p, \#, R)$, then $[a, q\#\alpha] \to [a, \#p\alpha]$.
  - If $\delta(q, X) = (p, Y, R)$ for $X, Y \in \Sigma\backslash\{\#, \$\}$, then
    for each $a, b \in \Sigma\backslash\{\#, \$\}$, $\beta \in \Sigma \cup \#\Sigma \cup \Sigma\$ \cup \#\Sigma\$$,
    $[a, qX][b, \beta] \to [a, Y][b, p\beta]$, $[a, \#qX][b, \beta] \to [a, \#Y][b, p\beta]$,
    $[a, \#qX\$)] \to [a, \#Yp\$]$.
  - Symmetrically for the transition $\delta(q, \$) = (p, \$, L)$ and $\delta(q, X) = (p, Y, L)$.
  - For each $a, b \in \Sigma\backslash\{\#, \$\}$, $q \in F$, and $\alpha_1\alpha_2, \alpha \in \Sigma \cup \#\Sigma \cup \Sigma\$ \cup \#\Sigma\$$,
    $[a, \alpha_1 q\alpha_2] \to a$, $[a, \alpha]b \to ab$, $b[a, \alpha] \to ba$.

From CSG to LBA

Recalling: From Type-0 grammar to TM

Let $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a Type-0 grammar.
Construct a nondet. TM M to recognize the language $L(G)$.

- M has two tapes.
- The input of M (say w) is in tape 1.
- M simulates the derivation relation of G in tape 2 by repeating the following procedure.
  1. It nondet. chooses a position i in tape 2 and a production rule $\alpha \to \beta$.

     If α appears from position i in tape 2, then α is replaced by β in tape 2. Some shifting over of the symbols on tape 2 should be done if $|\alpha| \neq |\beta|$.

  2. M compares the sequence of symbols in tape 2 with the sequence in tape 1, to see whether w has been generated by G. If so, then M accepts.

From CSG to LBA

Recalling: From Type-0 grammar to TM

Let $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a Type-0 grammar.
Construct a nondet. TM M to recognize the language $L(G)$.

- M has two tapes.
- The input of M (say w) is in tape 1.
- M simulates the derivation relation of G in tape 2 by repeating the following procedure.

From CSG to LBA

G is a CSG $\Rightarrow$ G is noncontracting ($\alpha \to \beta$ s.t. $|\alpha| \leqslant |\beta|$) $\Rightarrow$
The space of tape 2 used by M is bounded by $|w|$.
Therefore, M can be simulated by

a single-tape nondeterministic TM M' using linear space.

Properties of CSL: A bird's-eye view

Closure properties

Theorem. CSL are closed under all Boolean operations.

Decision problems

- The nonemptiness problem for CSG is undecidable.
- The membership problem for CSG is PSPACE-complete.

Outline

Outline

Definition

A CFG is a formal grammar $(\mathcal{N}, \Sigma, \mathcal{P}, S)$ such that

each production rule in $\mathcal{P}$ *is of the form* $A \to \alpha$ *with* $A \in \mathcal{N}$ *and* $\alpha \in (\mathcal{N} \cup \Sigma)^*$.

Example: CFG for arithmetic expressions,

$$E \to E + E \quad E \to E * E$$
$$E \to (E) \qquad E \to id$$

Derivation:

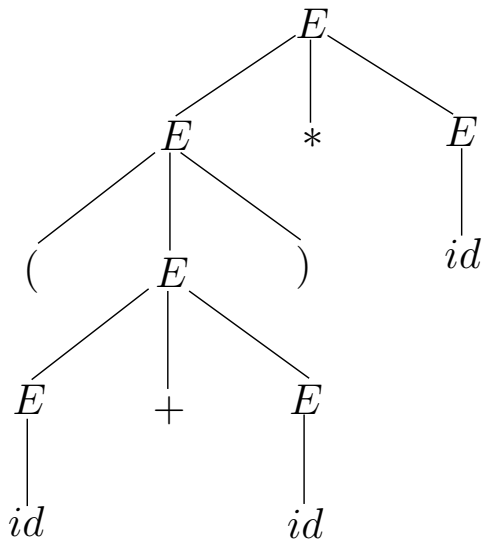
$$E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E \Rightarrow (id + id) * id.$$

Derivation trees

A derivation tree for a CFG $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$: A labeled tree s.t.

- every vertex has a label from $\mathcal{N} \cup \Sigma \cup \{\varepsilon\}$,
- the root is labeled by S,
- every internal vertex is labeled by symbols from $\mathcal{N}$,
- for every vertex labeled by $A \in \mathcal{N}$,
  if its children are labeled by $X_1, \ldots, X_k$ (from left to right),
  then $A \to X_1 \ldots X_k$ is a production rule of G.

Example: Derivation tree for $(id + id) * id$



Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.

$$E$$

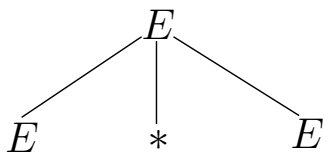
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



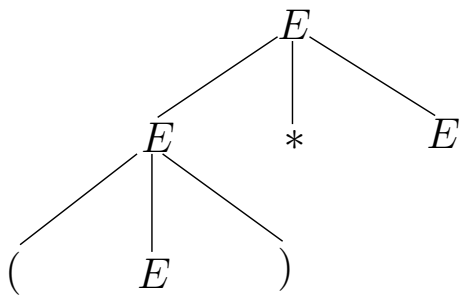
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



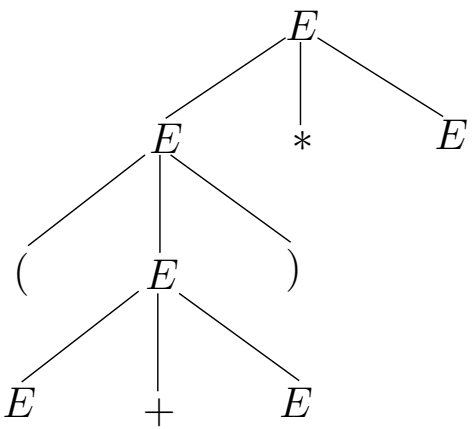
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



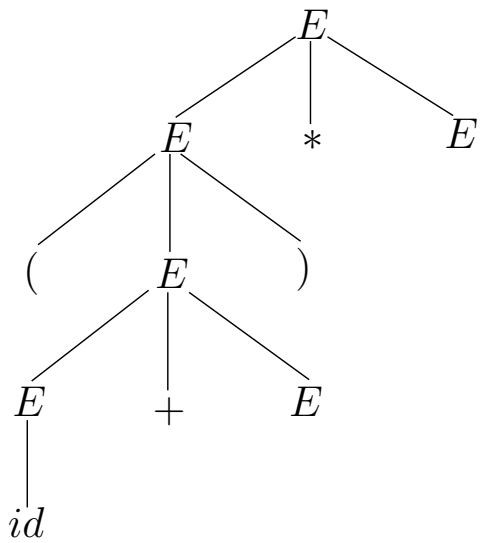
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



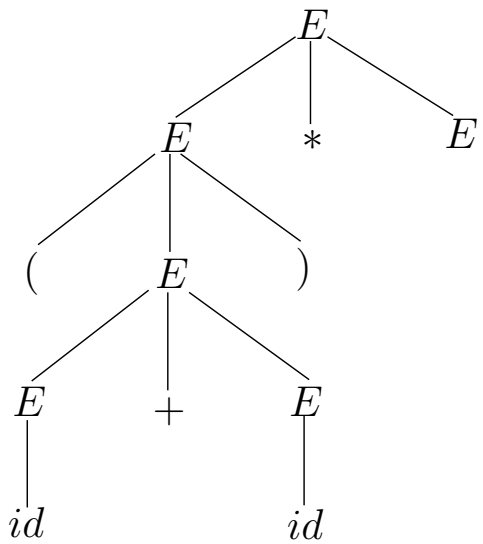
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



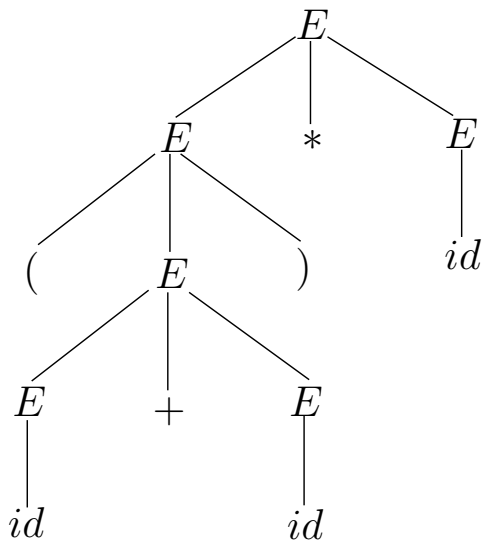
Leftmost derivation

Leftmost derivation:

Each step in the derivation is applied to the leftmost nonterminal.

Roughly speaking, leftmost derivation corresponds to

a preorder DFS traversal of the derivation tree.



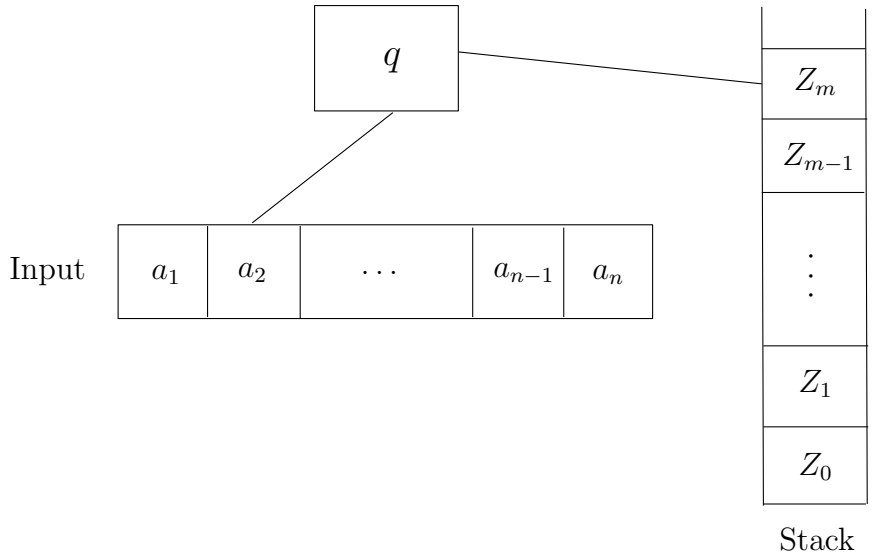
Outline

Definition

Intuitively, a pushdown automaton is a TM with the infinite tape organized as a stack.

Formally, a PDA M is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

- Q: Finite set of states,
- Σ: The input alphabet,
- Γ: The stack alphabet,
- $q_0 \in Q$: The initial state,
- $Z_0 \in \Gamma$: The start symbol,
- $F \subseteq Q$: The set of final states,
- δ: A mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$.



Instantaneous configuration (IC)

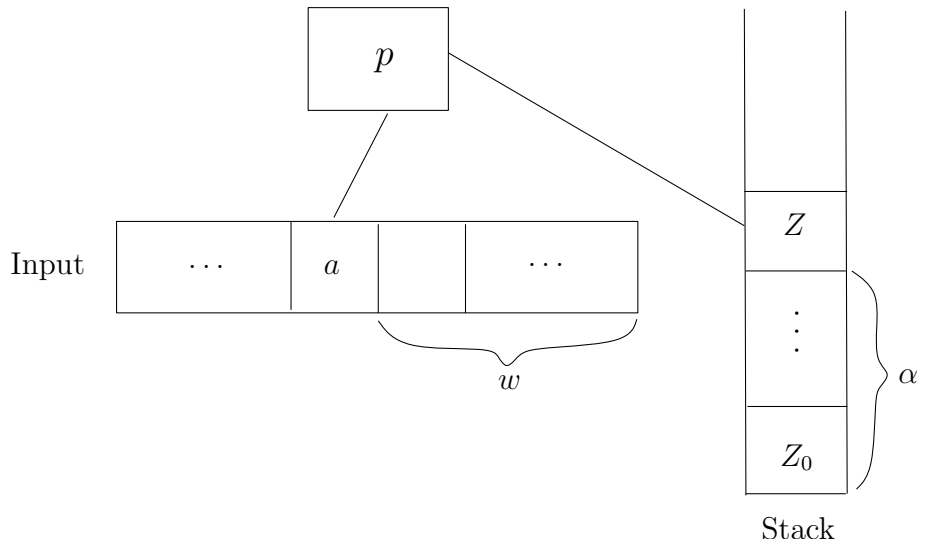
IC and $\vdash_M$

An instantaneous configuration (IC) is a tuple (q, w, γ) such that

- $q \in Q$,
- $w \in \Sigma^*$ (the suffix of the input not read yet),
- $\gamma \in \Gamma^*$ (the content of the stack, with the topmost symbol leftmost).

The $\vdash_M$ relation between ICs is defined as follows:

If $(p, \beta) \in \delta(q, a, Z)$ *(where* $a \in \Sigma \cup \{\varepsilon\}$*), then* $(q, aw, Z\alpha) \vdash_M (p, w, \beta\alpha)$.



Instantaneous configuration (IC)

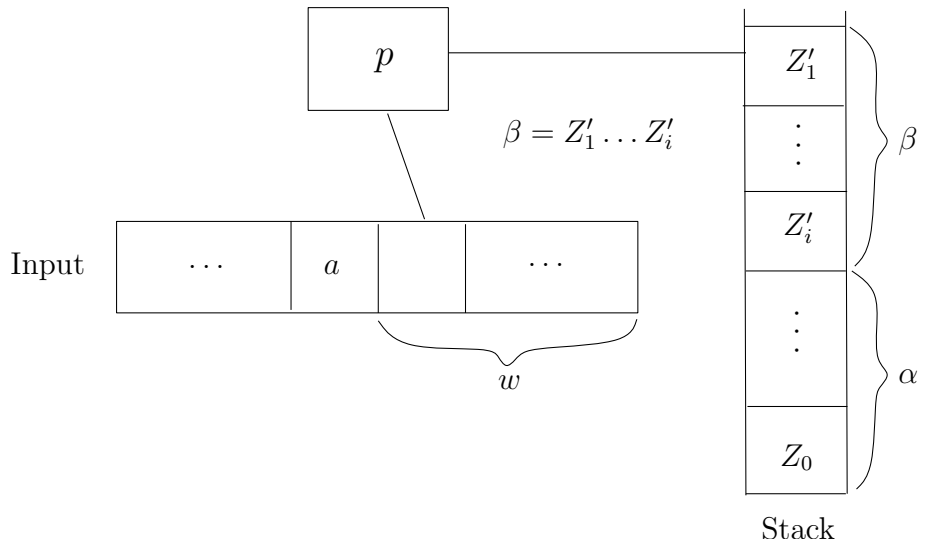
IC and $\vdash_M$

An instantaneous configuration (IC) is a tuple (q, w, γ) such that

- $q \in Q$,
- $w \in \Sigma^*$ (the suffix of the input not read yet),
- $\gamma \in \Gamma^*$ (the content of the stack, with the topmost symbol leftmost).

The $\vdash_M$ relation between ICs is defined as follows:

If $(p, \beta) \in \delta(q, a, Z)$ *(where* $a \in \Sigma \cup \{\varepsilon\}$*), then* $(q, aw, Z\alpha) \vdash_M (p, w, \beta\alpha)$.



Accepted languages

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA.

- Define $L(M)$, the language accepted by final state, as follows,

$$\{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_M^* (p, \varepsilon, \gamma) \text{ for } p \in F, \gamma \in \Gamma^*\}.$$

- Define $N(M)$, the language accepted by empty stack, as follows,

$$\{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_M^* (p, \varepsilon, \varepsilon) \text{ for } p \in Q\}.$$

Example: PDA $M = (\{q_0, q_1\}, \{0, 1\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \varnothing)$ accepting $\{ww^R \mid w \in \{0, 1\}^*\}$ by empty stack.

$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$	$\delta(q_0, 1, Z_0) = (q_0, BZ_0)$
$\delta(q_0, 0, A) = (q_0, AA)$	$\delta(q_0, 1, A) = (q_0, BA)$
$\delta(q_0, 0, B) = (q_0, AB)$	$\delta(q_0, 1, B) = (q_0, BB)$
$\delta(q_0, 0, A) = (q_1, \varepsilon)$	$\delta(q_0, 1, B) = (q_1, \varepsilon)$
$\delta(q_1, 0, A) = (q_1, \varepsilon)$	$\delta(q_1, 1, B) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$	$\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$

Equivalence of the two accepting means

From final state to empty stack

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA.
The goal: Define a PDA M' such that $N(M') = L(M)$.
The intuition:

> *if M accepts an input by a final state q, then from q, M' enters q_e, empties the stack, and accepts.*

In addition, M' includes a new start symbol Z_0' to deal with the following situation,

> *M reads all the input, enters a non-final state, and the stack is empty.*

$M' = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', \varnothing)$ is defined as follows.
δ' includes all the elements of δ and the transitions

- $\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0')\}$,
- $\delta'(q, \varepsilon, X) = \delta(q, \varepsilon, X) \cup \{(q_e, X)\}$ for $q \in F$ and $X \in \Gamma$,
- $\delta'(q_e, \varepsilon, X) = \{(q_e, \varepsilon)\}$ for every $X \in \Gamma \cup \{Z_0'\}$.

Equivalence of the two accepting means

From empty stack to final state

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ be a PDA.
The goal: Define a PDA M' such that $L(M') = N(M)$.
The intuition of M':

> *Add a new start symbol Z_0' to M, if Z_0' becomes the topmost symbol of the stack, then M' enters a final state.*

$M' = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', \{q_f\})$ is defined as follows.
δ' includes all the elements of δ and the transitions

- $\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0')\}$,
- $(q_f, \varepsilon) \in \delta'(q, \varepsilon, Z_0')$ for every $q \in Q$.

Outline

From CFG to PDA

The intuition

Use the stack to store the words reached in the leftmost derivation.

From CFG to PDA

The intuition

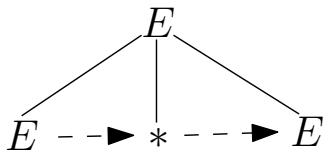
Use the stack to store the words reached in the leftmost derivation.

$$E$$

From CFG to PDA

The intuition

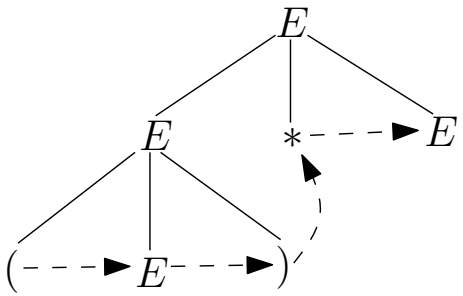
Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA

The intuition

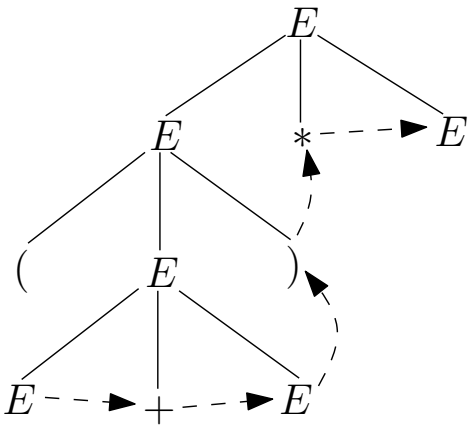
Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA

The intuition

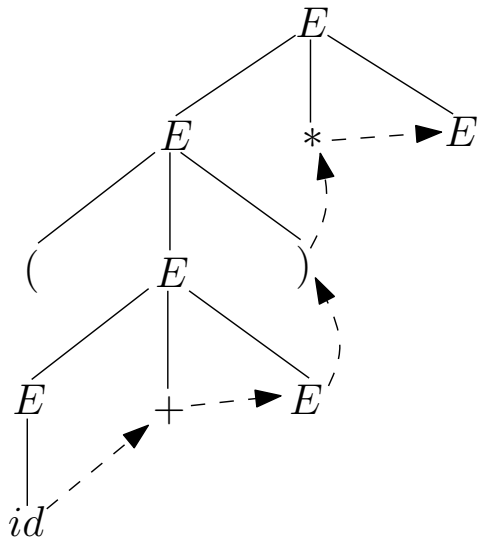
Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA

The intuition

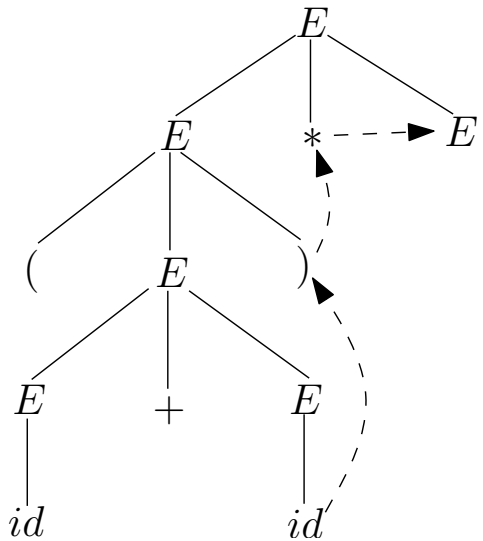
Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA

The intuition

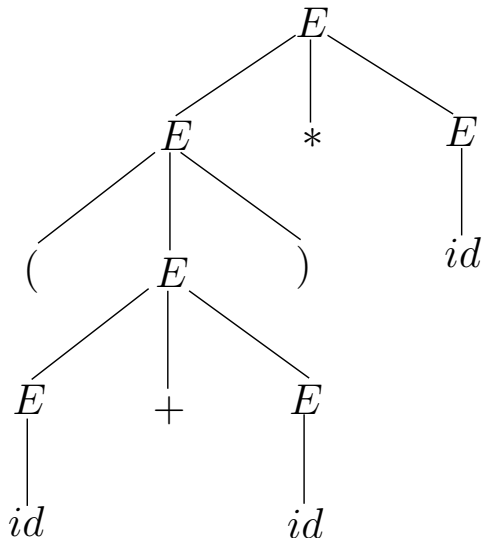
Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA

The intuition

Use the stack to store the words reached in the leftmost derivation.



From CFG to PDA: continued

The construction

Let $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a CFG.
Define a PDA $M = (\{q\}, \Sigma, \Sigma \cup \mathcal{N}, \delta, q, S, \varnothing)$ as follows.

- $\delta(q, \varepsilon, A) = \{(q, \alpha)\}$ for each production rule $A \to \alpha$,
- $\delta(q, a, a) = \{(q, \varepsilon)\}$ for every $a \in \Sigma$.

The correctness of the construction $N(M) = L(G)$

Claim.

$\forall x \in \Sigma^*, \alpha \in (\mathcal{N} \cup \Sigma)^*$,
$S \Rightarrow^* x\alpha$ *by leftmost derivation in* G *iff* $(q, x, S) \vdash_M^* (q, \varepsilon, \alpha)$.

Proof of the claim.
Induction on the derivation or computation steps.

From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

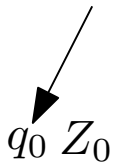
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

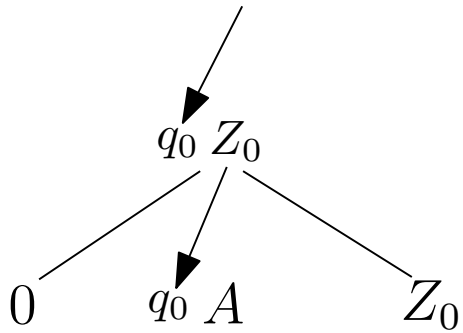
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

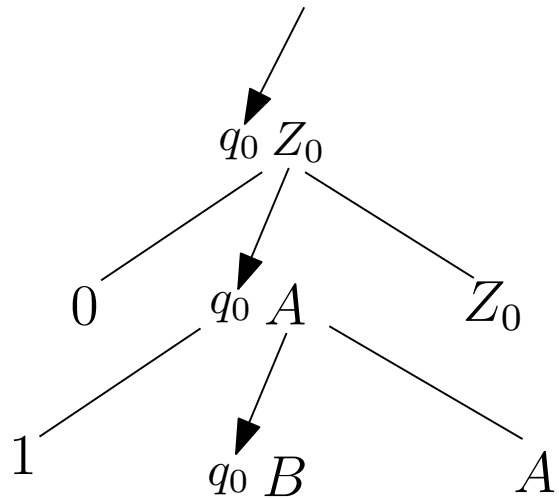
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

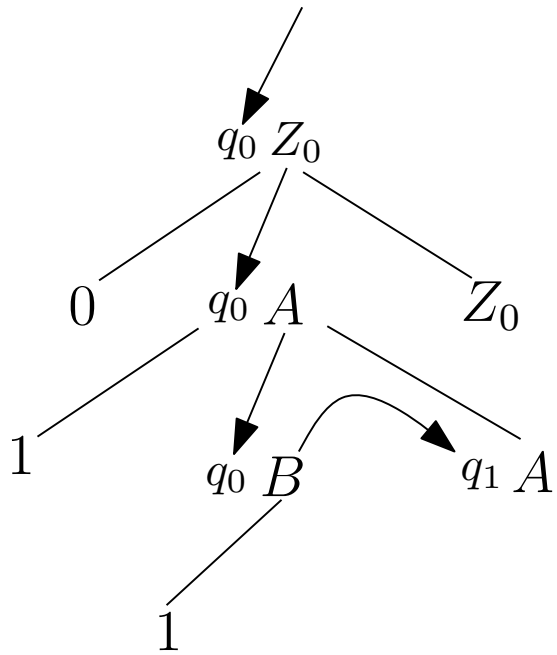
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

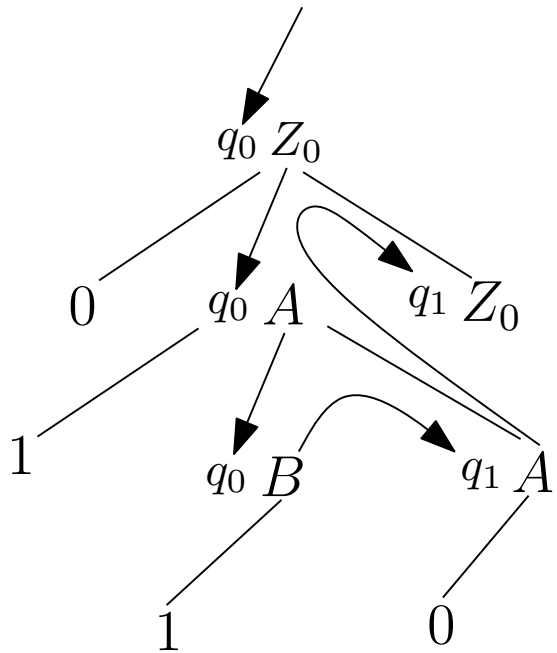
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

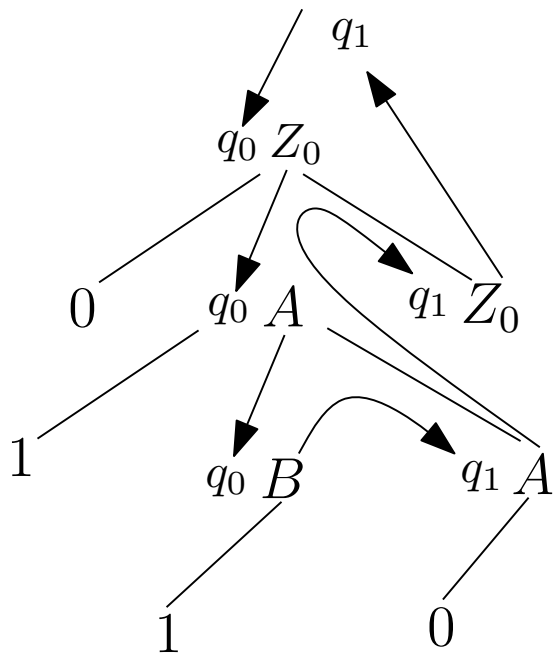
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

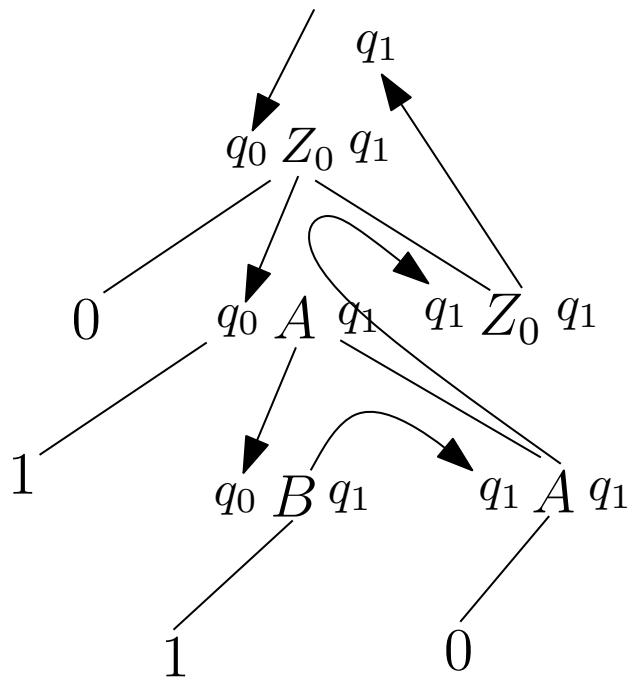
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG

An illustration of the intuition:

Analyze the computation tree of PDA M for $\{ww^R \mid w \in \{0,1\}^*\}$ over the input 0110.

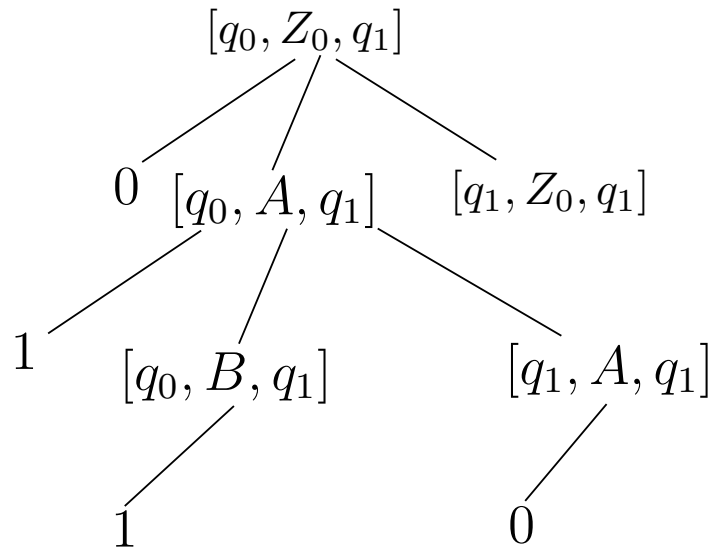
$$\delta(q_0, 0, Z_0) = (q_0, AZ_0)$$

$$\delta(q_0, 1, A) = (q_0, BA)$$

$$\delta(q_0, 1, B) = (q_1, \varepsilon)$$

$$\delta(q_1, 0, A) = (q_1, \varepsilon)$$

$$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$$



From PDA to CFG: continued

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ be a PDA.
The goal: Define a CFG $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ such that $L(G) = N(M)$.

- $\mathcal{N} = Q \times \Gamma \times Q \cup \{S\}$.
- $\mathcal{P}$ is defined as follows.
  - $S \to [q_0, Z_0, q]$ for every $q \in Q$.
  - If $(p, B_1 \dots B_k) \in \delta(q, a, X)$ (where $a \in \Sigma \cup \{\varepsilon\}$), then $[q, X, q_{k+1}] \to a[q_1, B_1, q_2] \dots [q_k, B_k, q_{k+1}]$ such that $q_1 = p$.

The correctness of the construction

Claim. $[q, A, p] \Rightarrow^* x$ iff $(q, x, A) \vdash_M^* (p, \varepsilon, \varepsilon)$.

Outline

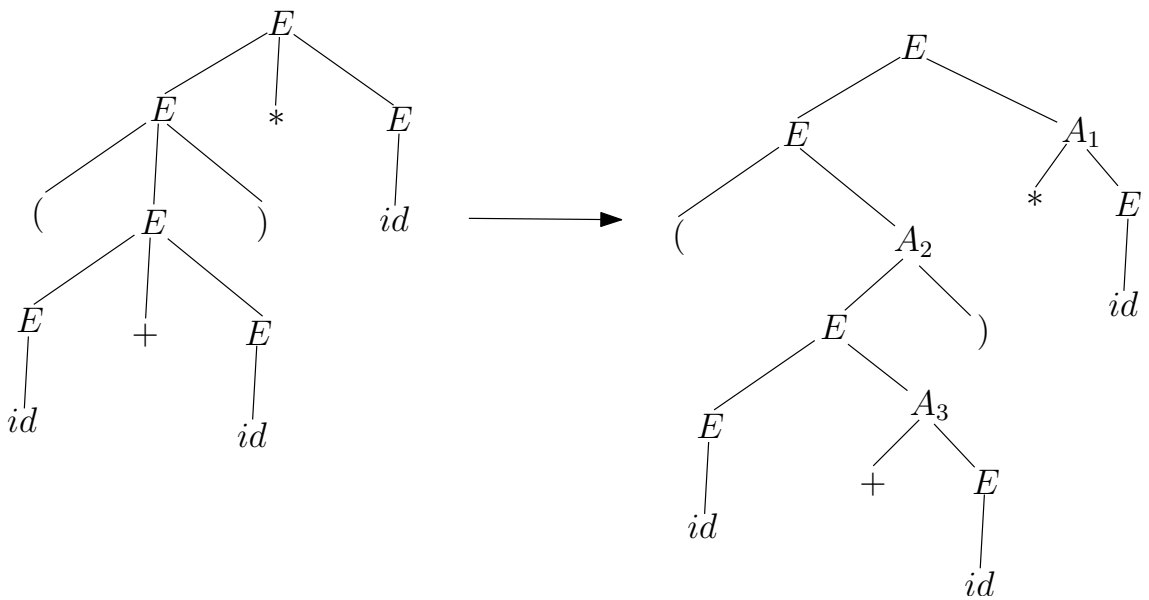
Outline

Binary normal form (2NF)

Theorem. For every CFG G, a CFG G' can be computed in linear time satisfying that

- $L(G) = L(G')$,
- G' contains only rules of the form $A \to \alpha$ such that $|\alpha| \leqslant 2$.

The intuition.



Binary normal form (2NF)

Theorem. For every CFG G, a CFG G' can be computed in linear time satisfying that

- $L(G) = L(G')$,
- G' contains only rules of the form $A \to \alpha$ such that $|\alpha| \leqslant 2$.

The intuition.

$$E \to E + E \quad E \to E * E$$
$$E \to (E) \quad E \to id$$

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E$$
$$E \to (A_2 \quad A_2 \to E) \quad E \to id$$

Binary normal form (2NF)

Theorem. For every CFG G, a CFG G' can be computed in linear time satisfying that

- $L(G) = L(G')$,
- G' contains only rules of the form $A \to \alpha$ such that $|\alpha| \leqslant 2$.

The intuition.

$$E \to E + E \quad E \to E * E$$
$$E \to (E) \quad E \to id$$

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E$$
$$E \to (A_2 \quad A_2 \to E) \quad E \to id$$

Formally, a 2NF can be obtained from a CFG by the following procedure.

Replace every rule $A \to X_1 \dots X_k$ *(where* $k \geqslant 3$*) by the following rules*

$$A \to X_1C_1, C_1 \to X_2C_2, \dots, C_{k-2} \to X_{k-1}X_k$$

where $C_1, \dots, C_{k-2}$ are new nonterminal symbols.

Outline

Pumping lemma

Pumping lemma. For every CFL L, there is a natural number $n \geqslant 1$ such that for every string $z \in L$ with $|z| \geqslant n$, z can be decomposed into $uvwxy$ satisfying that

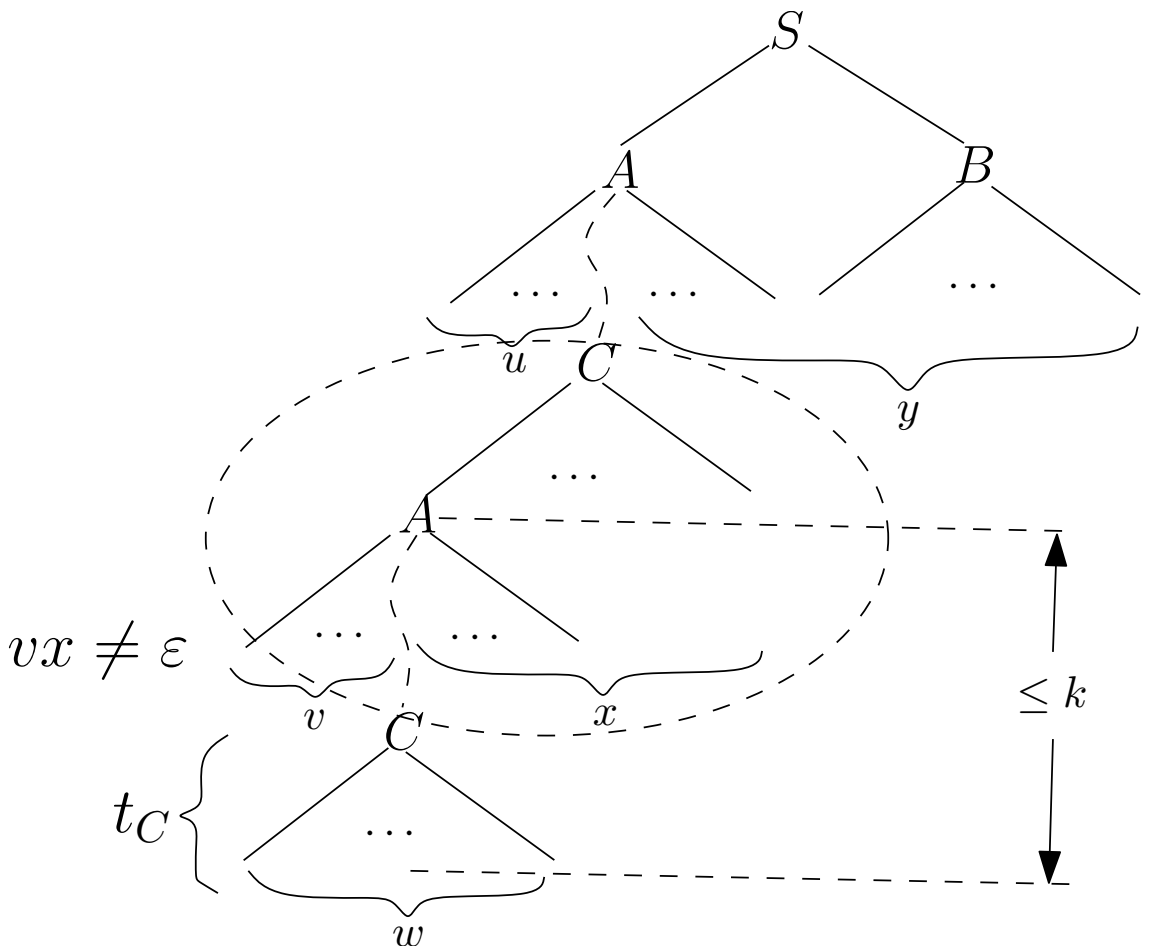
- $|vx| \geqslant 1$,
- $|vwx| \leqslant n$,
- for every $i \in \mathbb{N}$, $uv^iwx^iy \in L$.

Let L be a CFL and $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a CFG in 2NF s.t. $L = L(G)$. Let k be the number of nonterminals in G.

Pumping lemma: continued

The idea of the proof.

Analyzing the structure of the derivation trees of CFGs.

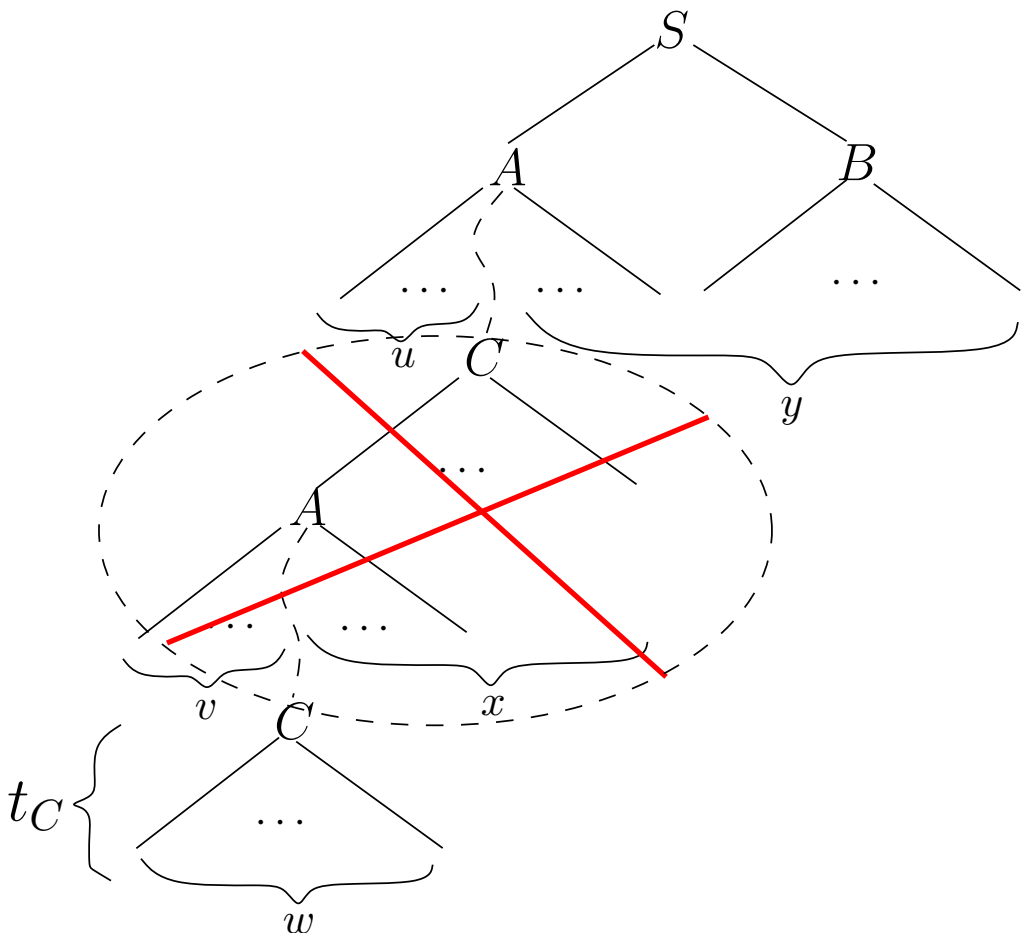


Pumping lemma: continued

The idea of the proof.

Analyzing the structure of the derivation trees of CFGs.

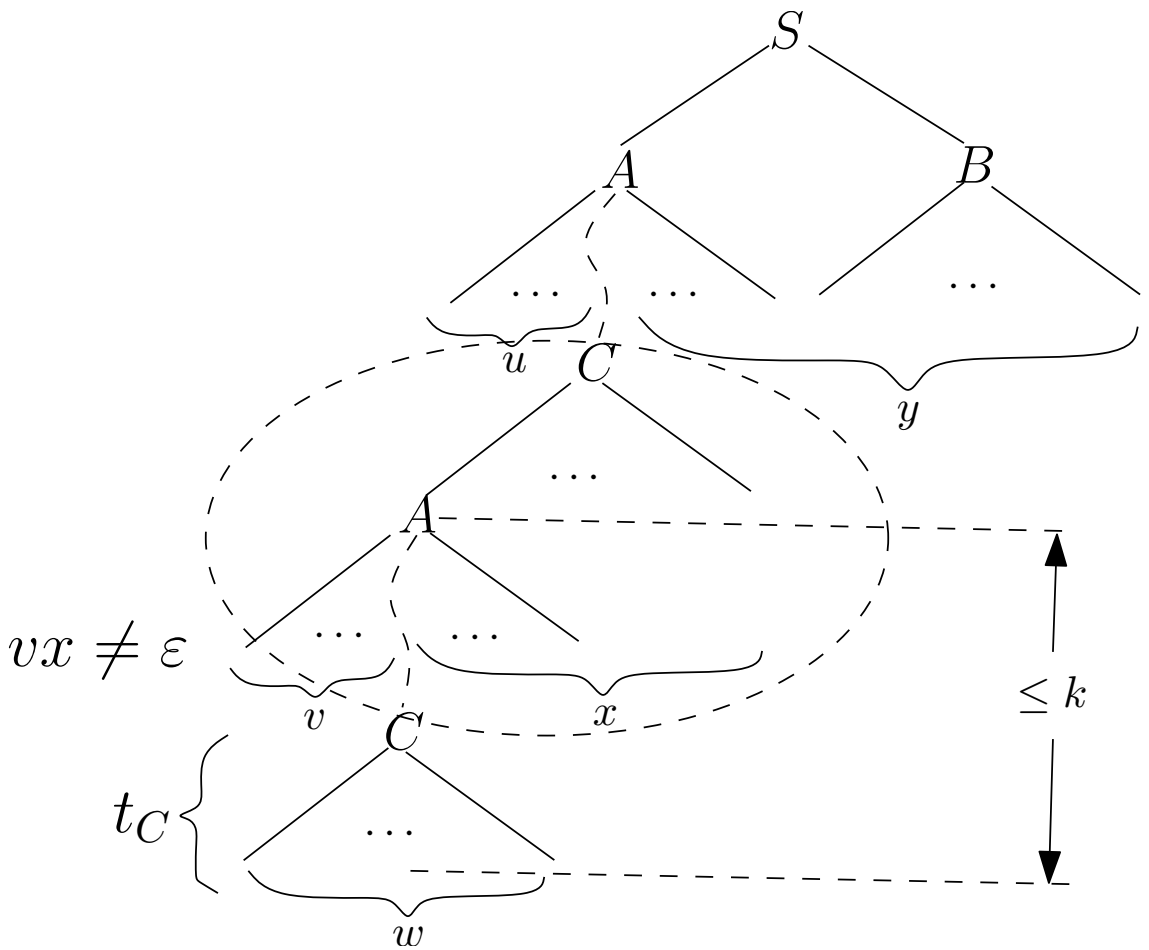
If $vx = \varepsilon$, then consider the reduced derivation tree instead



Pumping lemma: continued

The idea of the proof.

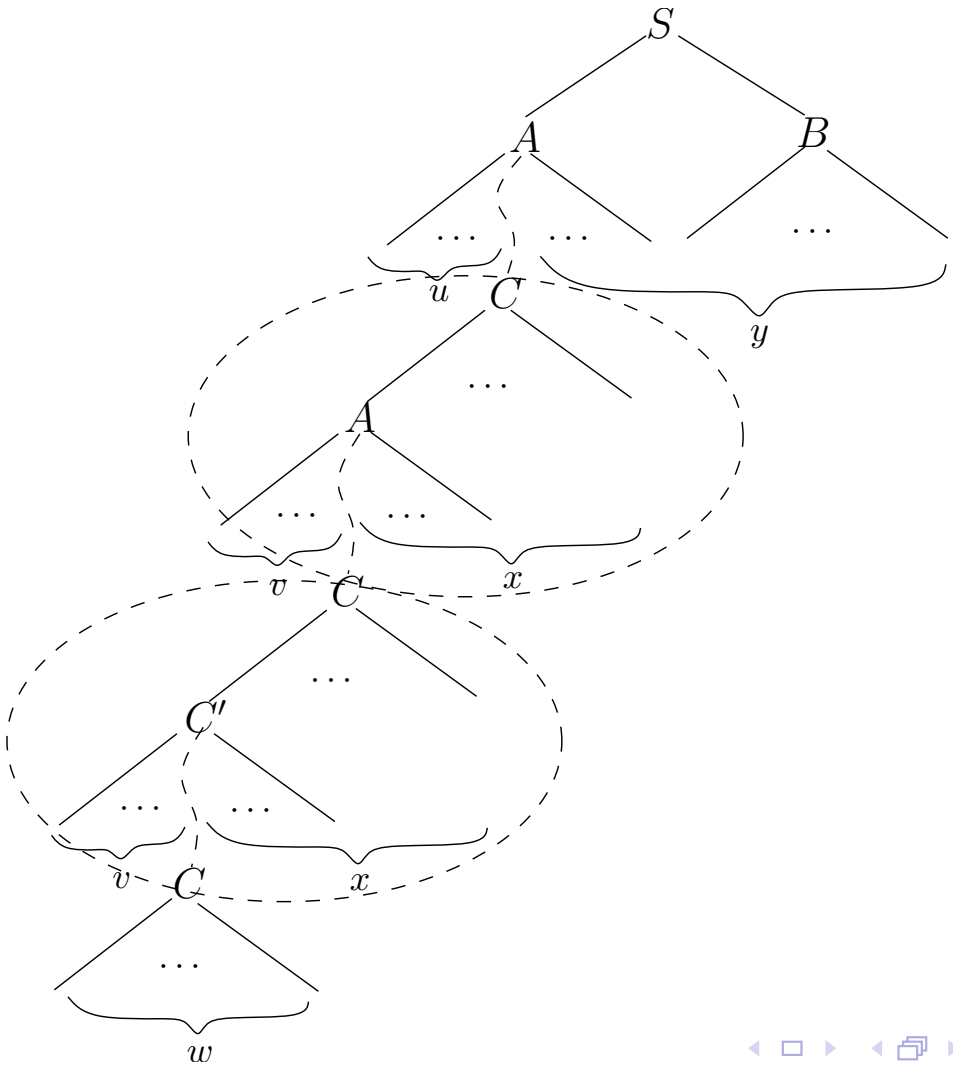
Analyzing the structure of the derivation trees of CFGs.



Pumping lemma: continued

The idea of the proof.

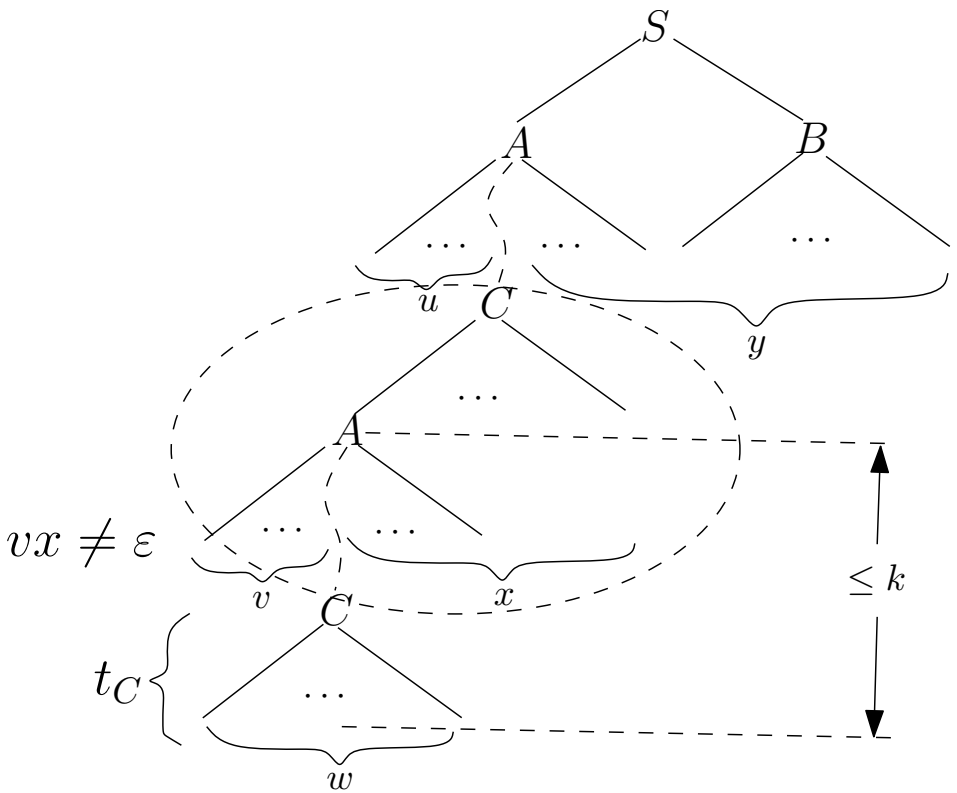
Analyzing the structure of the derivation trees of CFGs.



Pumping lemma: continued

The idea of the proof.

Analyzing the structure of the derivation trees of CFGs.



$|vwx|$: # leaves of a binary tree of depth $\leqslant k+1$ (depth 0 for root): $\leqslant 2^{k+1}$.

Set $n := 2^{k+1}$.

Applications of pumping lemma

Proposition. $L = \{a^i b^i c^i \mid i \geqslant 1\}$ is not a CFL.

To the contrary, suppose that L is a CFL.

Then there is a CFG G such that $L = L(G)$.

So, there is $n \geqslant 1$ satisfying the conditions in the pumping lemma.

Consider $z = a^n b^n c^n$.

Then $z = uvwxy$ s.t. $|vx| \geqslant 1$ and $|vwx| \leqslant n$. $\exists$ the following situations

- $vwx = a^i$ or $vwx = b^i$ or $vwx = c^i$ for some $i : 1 \leqslant i \leqslant n$.

  Consequences: $uv^2wx^2y \notin L$, *a contradiction.*

- $vwx = a^i b^j$ or $vwx = b^i c^j$ for some $i, j : 1 \leqslant i, j, i + j \leqslant n$.

  Consequences: $uv^2wx^2y \notin L$, *a contradiction.*

Corollary. $CFL \subset CSL$.

Homework.

Prove that $\{a^i b^j c^i d^j \mid i, j \geqslant 1\}$ is not a CFL, using pumping lemma.

Outline

Closure properties

Theorem. The class of CFLs is

- closed under union,
- not closed under intersection,
- closed under intersection with regular languages,
- not closed under complementation.

Union:

Suppose $G_1 = (\mathcal{N}_1, \Sigma, \mathcal{P}_1, S_1)$ and $G_2 = (\mathcal{N}_2, \Sigma, \mathcal{P}_2, S_2)$.

Then G: Add the rules $S \to S_1, S \to S_2$.

It is easy to see that $L(G) = L(G_1) \cup L(G_2)$.

Closure properties

Theorem. The class of CFLs is

- closed under union,
- not closed under intersection,
- closed under intersection with regular languages,
- not closed under complementation.

Intersection:

$L_1 = \{a^i b^i c^j \mid i, j \geqslant 1\}$ and $L_2 = \{a^j b^i c^i \mid i, j \geqslant 1\}$ are both CFL.

On the other hand, $L_1 \cap L_2 = \{a^i b^i c^i \mid i \geqslant 1\}$ is not a CFL.

Closure properties

Theorem. The class of CFLs is

- closed under union,
- not closed under intersection,
- closed under intersection with regular languages,
- not closed under complementation.

Intersection with regular languages:

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA for L_1,
- $\mathcal{A} = (Q', \Sigma, \delta', q_0', F')$ be a FSA for L_2.

Then $L_1 \cap L_2 = L(M'')$ such that

$$M'' = (Q \times Q', \Sigma, \Gamma, \delta'', (q_0, q_0'), Z_0, F \times F'),$$

where

- $\delta''((q, q'), a, X) = \{((p, p'), \gamma) \mid (p, \gamma) \in \delta(q, a, X), \delta'(q', a) = p'\}$ for $a \in \Sigma$,
- $\delta''((q, q'), \varepsilon, X) = \{((p, q'), \gamma) \mid (p, \gamma) \in \delta(q, \varepsilon, X)\}$.

Closure properties

Theorem. The class of CFLs is

- closed under union,
- not closed under intersection,
- closed under intersection with regular languages,
- not closed under complementation.

Complementation:

If the class of CFLs is closed under complementation, then

from the fact that $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$,

the class of CFLs is closed under intersection, a contradiction.

Outline

Outline

Emptiness

Let G be a CFG.

- $T_0(G) = \Sigma \cup \{A \mid \exists \alpha \in \Sigma^*.A \to \alpha\}$.
- $T_{i+1}(G) = T_i(G) \cup \{A \mid A \to X_1 \dots X_k, \forall j : 1 \leqslant j \leqslant k.X_j \in T_i(G)\}$.

Define $T(G) = T_i(G)$ such that $T_i(G) = T_{i+1}(G)$.

Claim. $L(G) \neq \varnothing$ iff $S \in T(G)$.

Theorem. The emptiness of CFG can be solved in PTIME.

Outline

Universality

Universality problem (A special case of inclusion problem):

Given a CFG G, whether $L(G) = \Sigma^*$.

Theorem. The universality problem of CFG is undecidable.

Proof idea: Reduction from emptiness problem of TM.

Lemma. Emptiness problem of TM is undecidable.

Reduction from halting problem:
From $\langle M, w\rangle$, construct M' as follows,

M': *Over an input $x \in \{0,1\}^*$, simulate the computation of M on w. If the computation halts, then accepts.*

The following fact holds.

- If the computation of M over w halts, then $L(M') = \{0,1\}^*$,
- otherwise, $L(M') = \varnothing$.

$$\langle M, w\rangle \in L_h \text{ iff}$$
$$\text{the computation of } M \text{ over } w \text{ halts iff}$$
$$L(M') \neq \varnothing.$$

Universality

Universality problem (A special case of inclusion problem):

Given a CFG G, whether $L(G) = \Sigma^*$.

Theorem. The universality problem of CFG is undecidable.

Proof idea: Reduction from emptiness problem of TM.

Valid computation of a TM M

A valid computation of $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ is

$$C_1\$C_2^R\$C_3\$C_4^R\$\ldots\$C_n$$

such that

- for every i, C_i is an IC of M,
- C_1 is of the form q_0x for some $x \in \Sigma^*$,
- for every i, $C_i \vdash_M C_{i+1}$,
- $C_n = \alpha q \beta$ such that $q \in F$, $\alpha, \beta \in \Gamma^*$.

The set of invalid computations is the complement of the valid computations wrt. $(Q \cup \Gamma \cup \{\$\})^*$.

Universality: Continued

Theorem. The set of invalid computations of a TM M is a CFL.
Let L_I be the set of invalid computations of M.

A word $w \in (Q \cup \Gamma \cup \$)^*$ belongs to L_I iff one of the following conditions hold,

1. w is not of the form $C_1\$C_2\$C_3 \ldots \$C_n$ such that each $C_i \in \Gamma^* Q \Gamma^*$,
2. C_1 is not initial, i.e. is not of the form $q_0\Sigma^*$,
3. there is an odd i such that $C_i \nvdash_M C_{i+1}^R$,
4. there is an even i such that $C_i^R \nvdash_M C_{i+1}$,
5. C_n is not final, namely, $C_n \notin \Gamma^* F \Gamma^*$.

Universality: Continued

Theorem. The set of invalid computations of a TM M is a CFL.
Let L_I be the set of invalid computations of M.

A word $w \in (Q \cup \Gamma \cup \$)^*$ belongs to L_I iff one of the following conditions hold,

1. w is not of the form $C_1\$C_2\$C_3 \ldots \$C_n$ such that each $C_i \in \Gamma^* Q \Gamma^*$,
   Regular
2. C_1 is not initial, i.e. is not of the form $q_0\Sigma^*$,
   Regular
3. there is an odd i such that $C_i \not\vdash_M C_{i+1}^R$,
   Verified by a pushdown automaton
4. there is an even i such that $C_i^R \not\vdash_M C_{i+1}$,
   Verified by a pushdown automaton
5. C_n is not final, namely, $C_n \notin \Gamma^* F \Gamma^*$.
   Regular

Universality: Continued

Theorem. The set of invalid computations of a TM M is a CFL.
Let L_I be the set of invalid computations of M.

A word $w \in (Q \cup \Gamma \cup \$)^*$ belongs to L_I iff one of the following conditions hold,

1. w is not of the form $C_1\$C_2\$C_3 \ldots \$C_n$ such that each $C_i \in \Gamma^* Q \Gamma^*$,
   Regular
2. C_1 is not initial, i.e. is not of the form $q_0\Sigma^*$,
   Regular
3. there is an odd i such that $C_i \nvdash_M C_{i+1}^R$,
   Verified by a pushdown automaton
4. there is an even i such that $C_i^R \nvdash_M C_{i+1}$,
   Verified by a pushdown automaton
5. C_n is not final, namely, $C_n \notin \Gamma^* F \Gamma^*$.
   Regular

Claim. $L_I = (Q \cup \Gamma \cup \$)^*$ *iff* $L(M) = \varnothing$.

Outline

Cocke-Younger-Kasami (CYK) algorithm

Membership problem:

Given a CFG $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ *and a word* $w \in \Sigma^*$, *is* $w \in L(G)$*?*

Theorem. The membership problem for CFG G can be solved in PTIME.

W.l.o.g. assume that G is in 2NF.
Let $n = |w| \geqslant 1$.

The intuition:

Bottom-up computation of the derivation tree through dynamic programming.

The CFG G

$$\begin{array}{cccc} E \to EA_3 & A_3 \to +E & E \to EA_1 & A_1 \to *E \\ E \to (A_2 & A_2 \to E) & E \to aI & E \to bI \\ I \to 0I & I \to 1I & I \to \varepsilon & \end{array}$$

Question: $(a + b0) * a1 \in L(G)$?

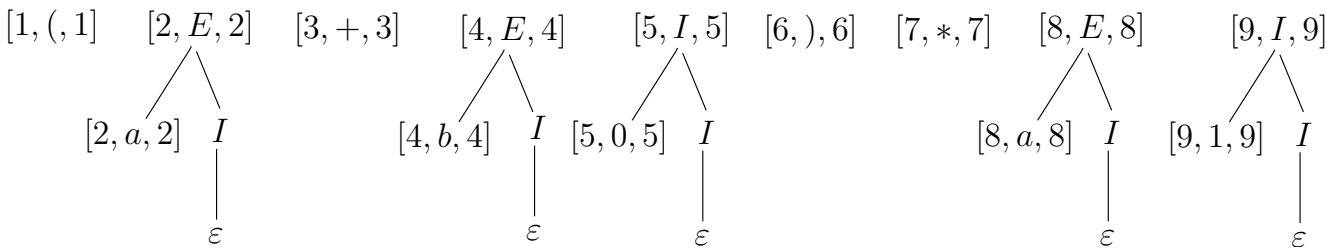
Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

(	a	$+$	b	0	)	$*$	a	1
1	2	3	4	5	6	7	8	9



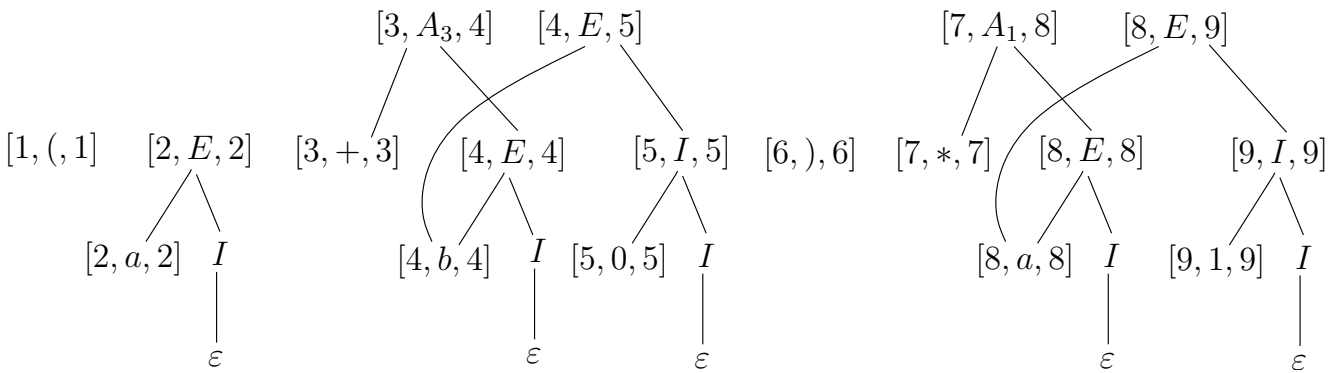
Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

(	a	+	b	0	)	$*$	a	1
1	2	3	4	5	6	7	8	9



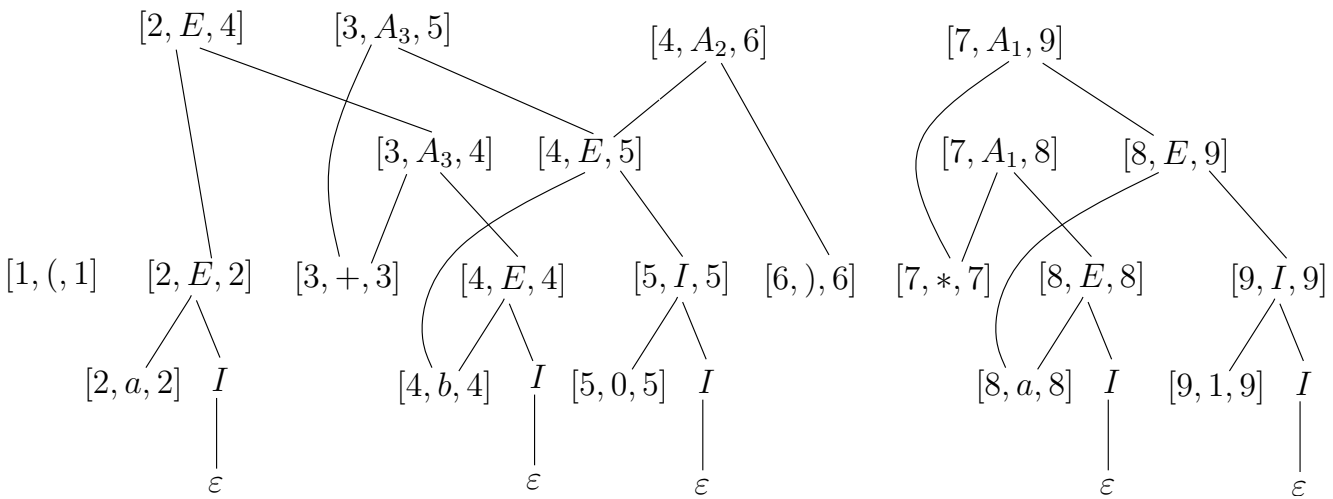
Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

(	a	+	b	0	)	$*$	a	1
1	2	3	4	5	6	7	8	9



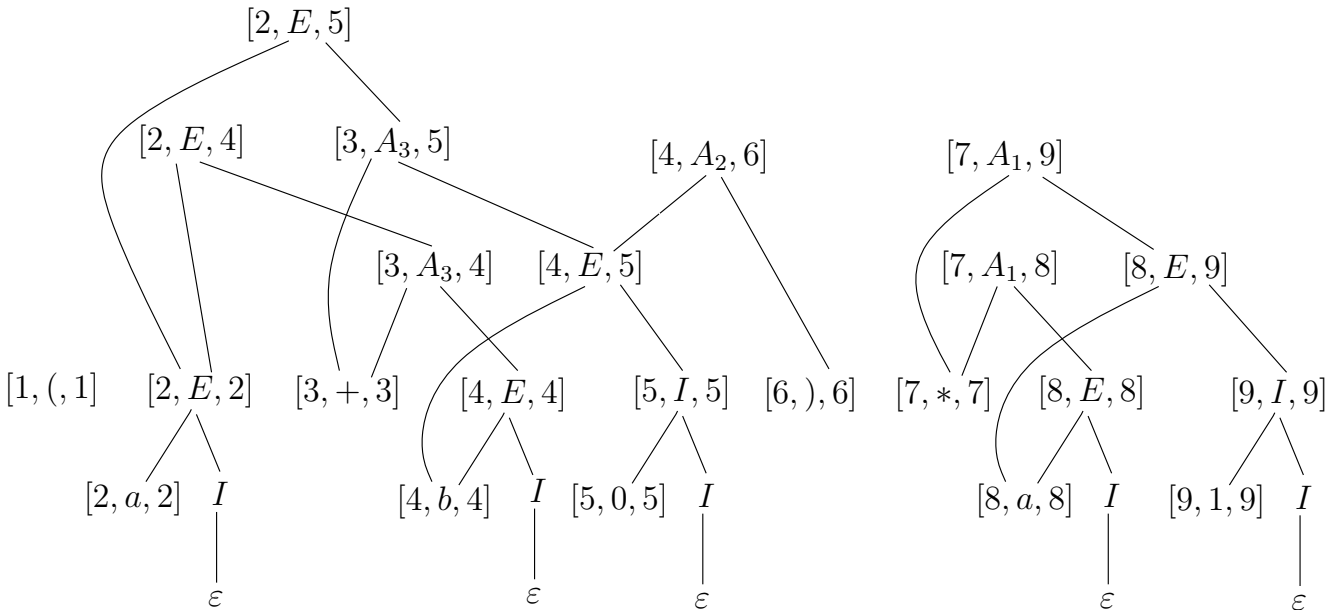
Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

(	a	+	b	0	)	$*$	a	1
1	2	3	4	5	6	7	8	9



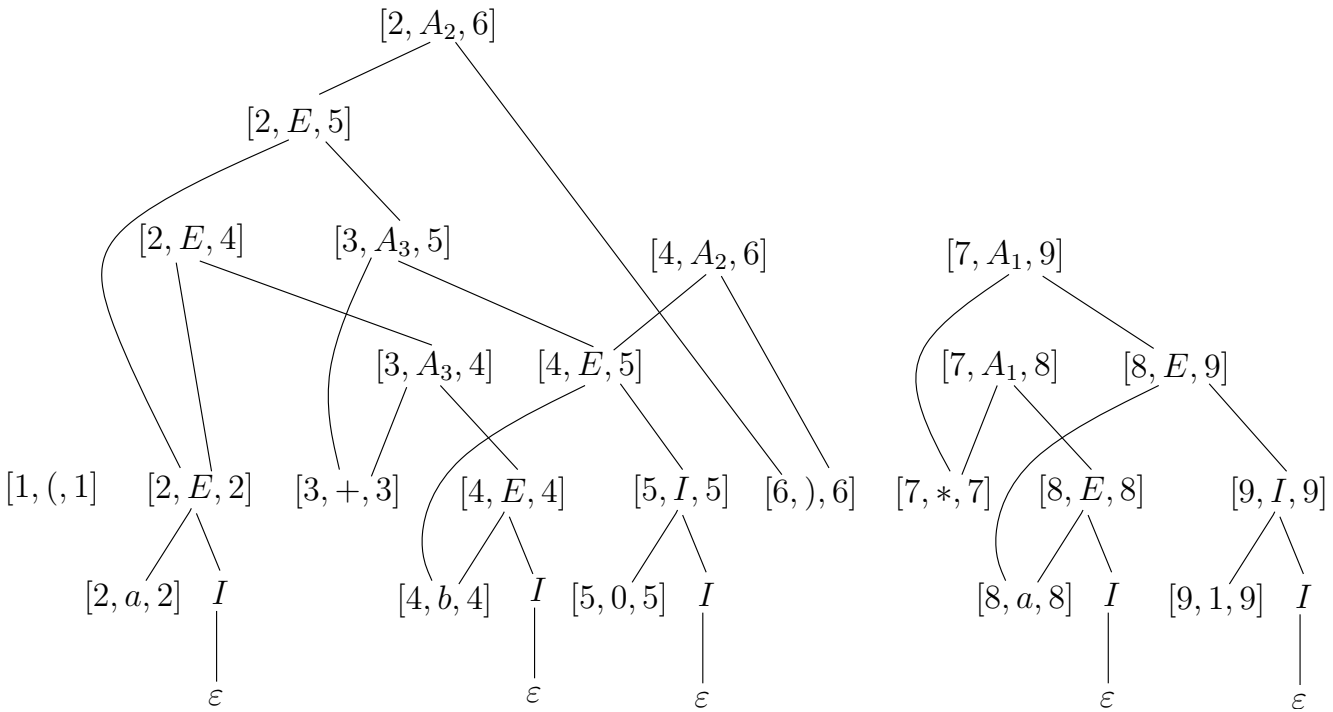
Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

(	a	+	b	0	)	$*$	a	1
1	2	3	4	5	6	7	8	9

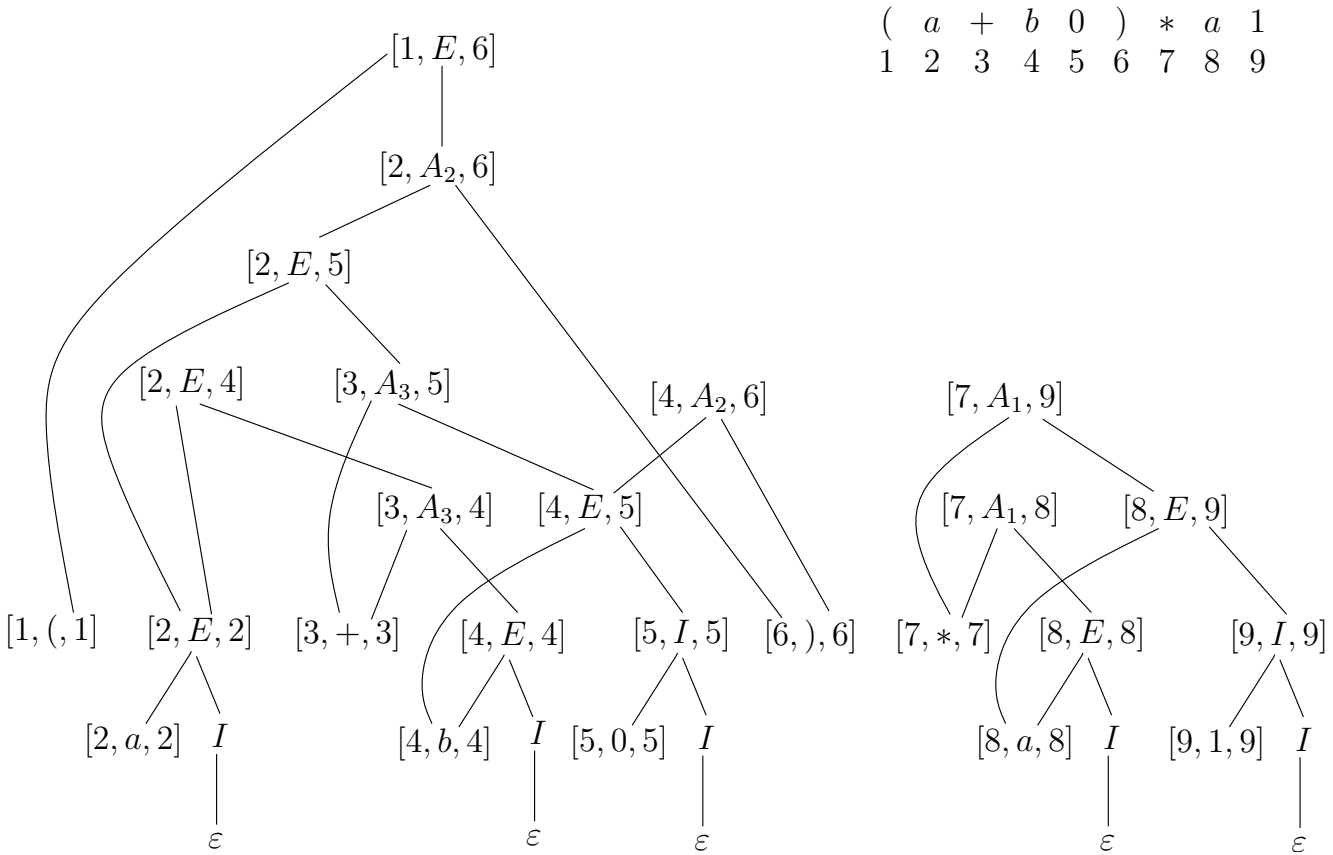


Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$

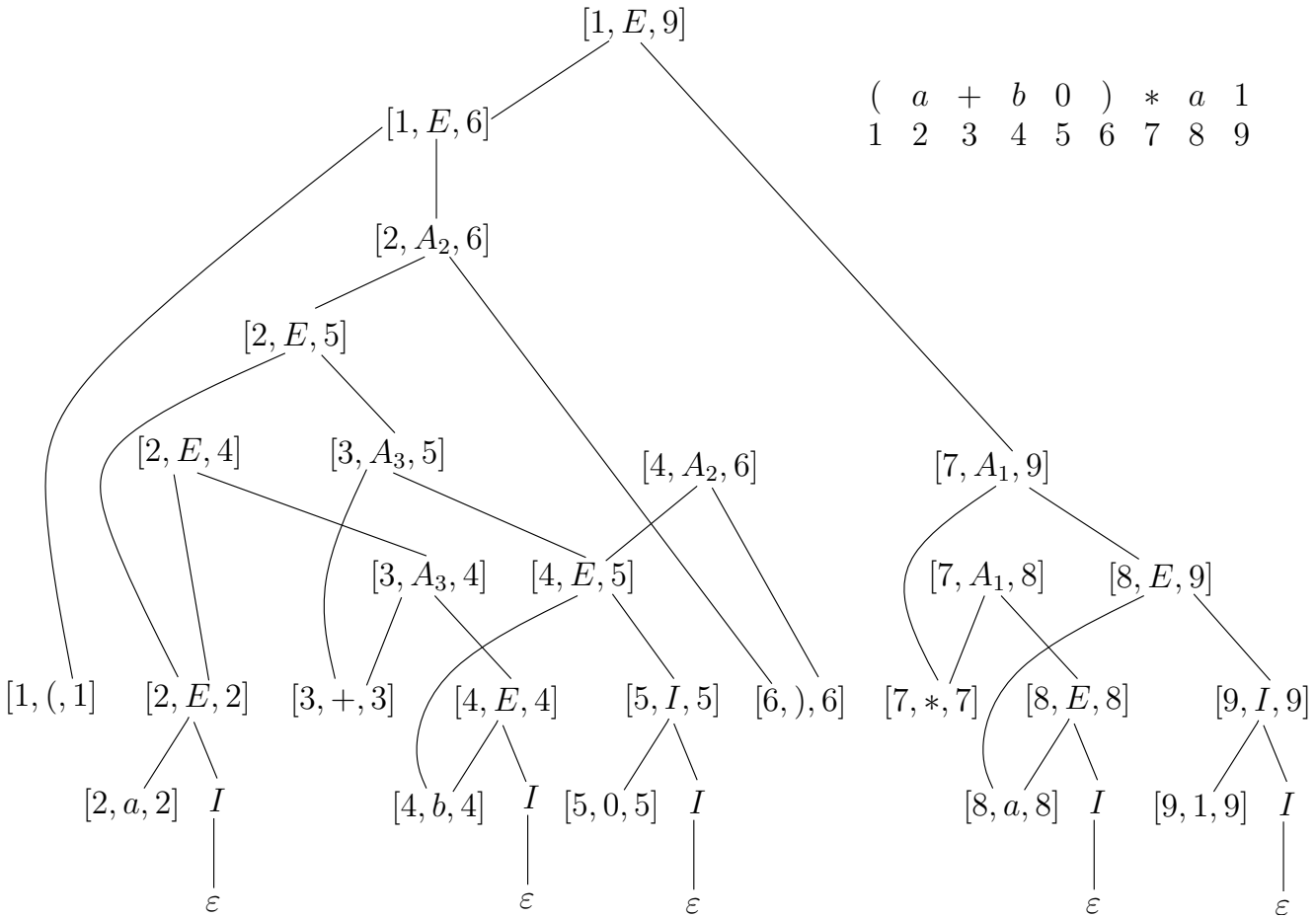


Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition:

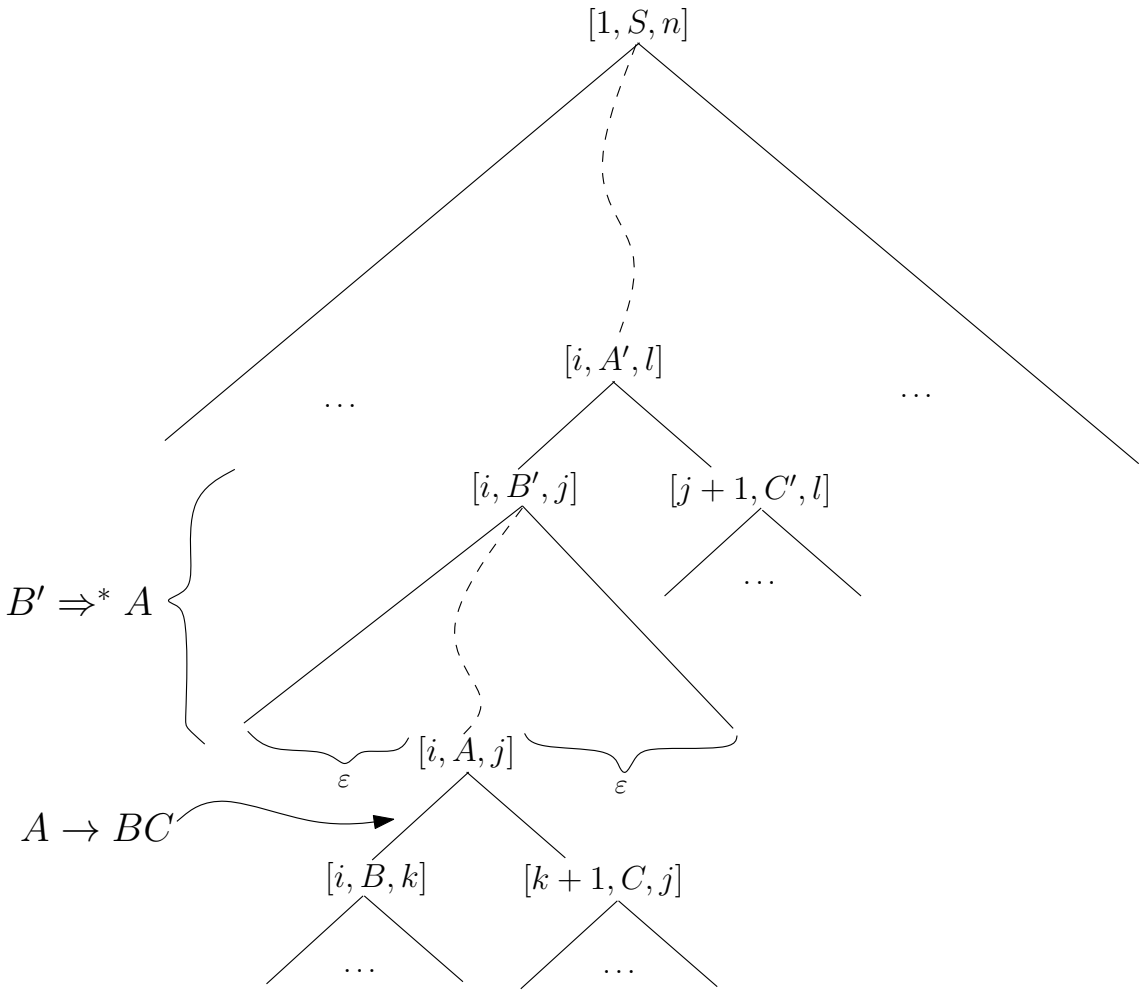
The CFG G:

$$E \to EA_3 \quad A_3 \to +E \quad E \to EA_1 \quad A_1 \to *E \quad E \to (A_2 \quad A_2 \to E)$$
$$E \to aI \quad E \to bI \quad I \to 0I \quad I \to 1I \quad I \to \varepsilon$$



Cocke-Younger-Kasami (CYK) algorithm: continued

The intuition: Suppose G is a CFG in 2NF and $w \in \Sigma^*$.



Cocke-Younger-Kasami (CYK) algorithm: continued

Nullable nonterminals $N(G)$

Nonterminals A s.t. $A \Rightarrow^* \varepsilon$.
The set of nullable nonterminals, $N(G)$, is computed as follows.

- $N_0(G) = \{A \mid A \to \varepsilon\}$,
- $N_{i+1}(G) =$
  $N_i(G) \cup \{A \mid \exists B \in N_i(G).A \to B\} \cup \{A \mid \exists B, C \in N_i(G).A \to BC\}$.

Example: CFG G

$$\begin{array}{cccc} E \to EA_3 & A_3 \to +E & E \to EA_1 & A_1 \to *E \\ E \to (A_2 & A_2 \to E) & E \to aI & E \to bI \\ I \to 0I & I \to 1I & I \to \varepsilon & \end{array}$$

$N(G) = \{I\}$.

Cocke-Younger-Kasami (CYK) algorithm: continued

Unit derivation relation $U(G)$

The unit derivation relation of G, $U(G)$, is defined as

$$\{(A, \gamma) \mid A \Rightarrow^* \gamma, \gamma \in \mathcal{N} \cup \Sigma\}.$$

$U(G)$ is computed as follows.

- $U_0(G) = \left\{ (A, \gamma) \,\middle|\, \begin{array}{c} \gamma \in \mathcal{N} \cup \Sigma, \text{ either } A \to \gamma, \\ \text{or } \exists B \in N(G). A \to B\gamma \vee A \to \gamma B \end{array} \right\}$,
- $U_{i+1}(G) = U_i(G) \cup \left\{ (A, \gamma) \,\middle|\, \begin{array}{c} \gamma \in \mathcal{N} \cup \Sigma, \\ \exists B \in \mathcal{N}. (A, B) \in U_i(G), (B, \gamma) \in U_i(G) \end{array} \right\}$.

For a subset $\mathcal{T} \subseteq \mathcal{N} \cup \Sigma$, define

$$U(\mathcal{T}) = \mathcal{T} \cup \{A \mid \exists \gamma \in \mathcal{T}, (A, \gamma) \in U(G)\}.$$

Example: $U(G) = \{(I, 0), (I, 1), (E, a), (E, b)\}$.

Cocke-Younger-Kasami (CYK) algorithm: continued

Define $V_{i,j} := \{A \mid A \Rightarrow^* w[i,j]\}$ for $i, j : 1 \leqslant i \leqslant j \leqslant n$.
Then $V_{i,j}$'s are computed inductively as follows.

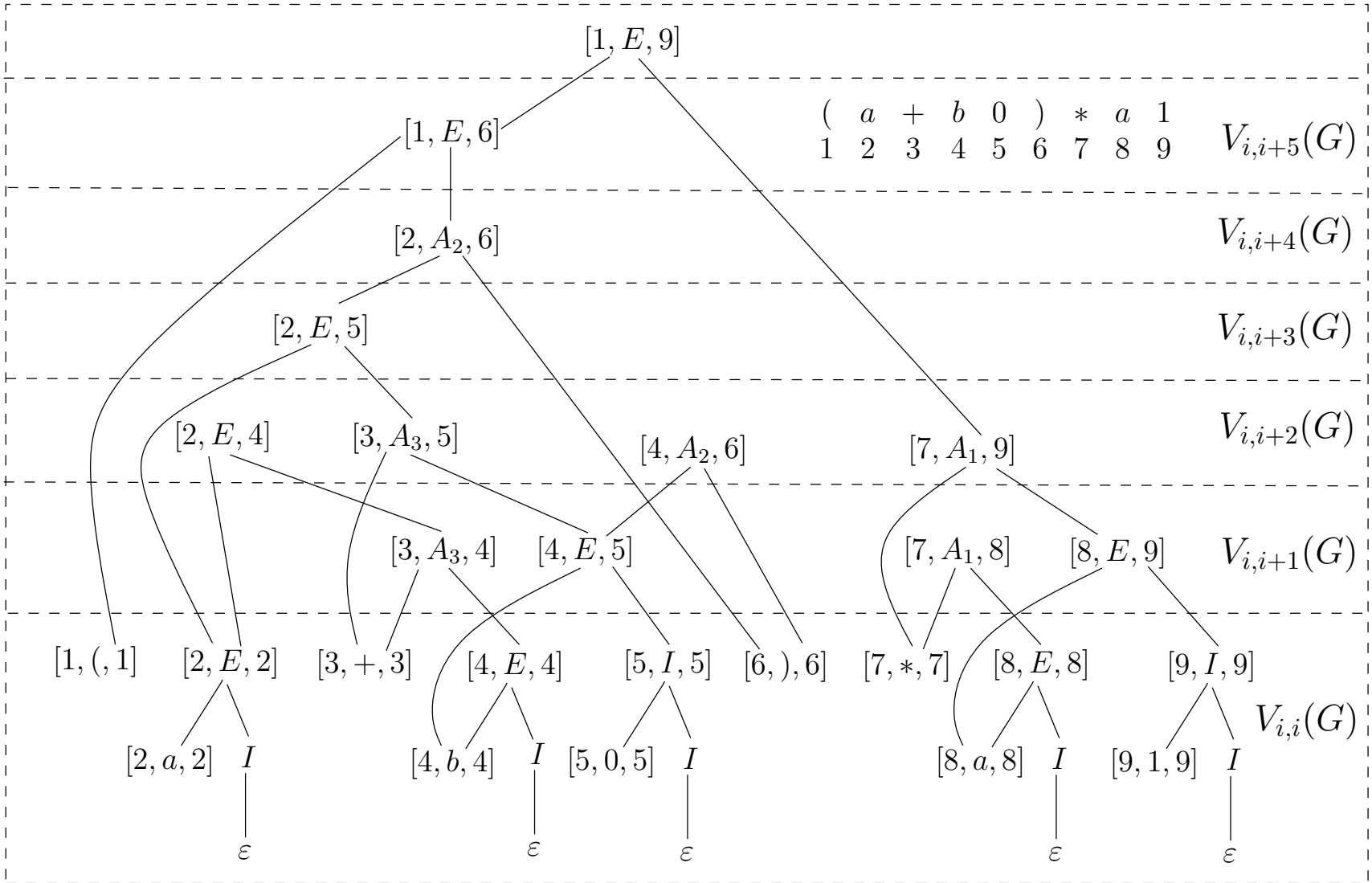
- $V_{i,i} = U(\{w_i\})$,
- For $i, j : 1 \leqslant i < j \leqslant n$,

$$V'_{i,j} = \{A \mid \exists k : i \leqslant k \leqslant j, B \in V_{i,k}, C \in V_{k+1,j}. \text{ s.t. } A \to BC\}.$$

- For $i, j : 1 \leqslant i < j \leqslant n$, $V_{i,j} = U(V'_{i,j})$.

Claim. $w \in L(G)$ *iff* $S \in V_{1,n}$.

Cocke-Younger-Kasami (CYK) algorithm: continued



Summary

1. Equivalence of CSG and LBA
2. Equivalence of CFG and pushdown automata
   - Context-free grammar (CFG)
   - Pushdown automata (PDA)
   - Equivalence of CFG and PDA
3. Properties of CFL
   - Normal form of CFG
   - Pumping lemma
   - Closure properties
4. Decision problems of CFG
   - Emptiness
   - Inclusion
   - Membership

Finite state automata