

Part VII.

Models for Context-Free

Languages

Context-Free Grammar (CFG)

Gist: A *grammar* is based on a finite set of grammatical rules, by which it generates strings of its language.

Illustration:

Start nonterminal

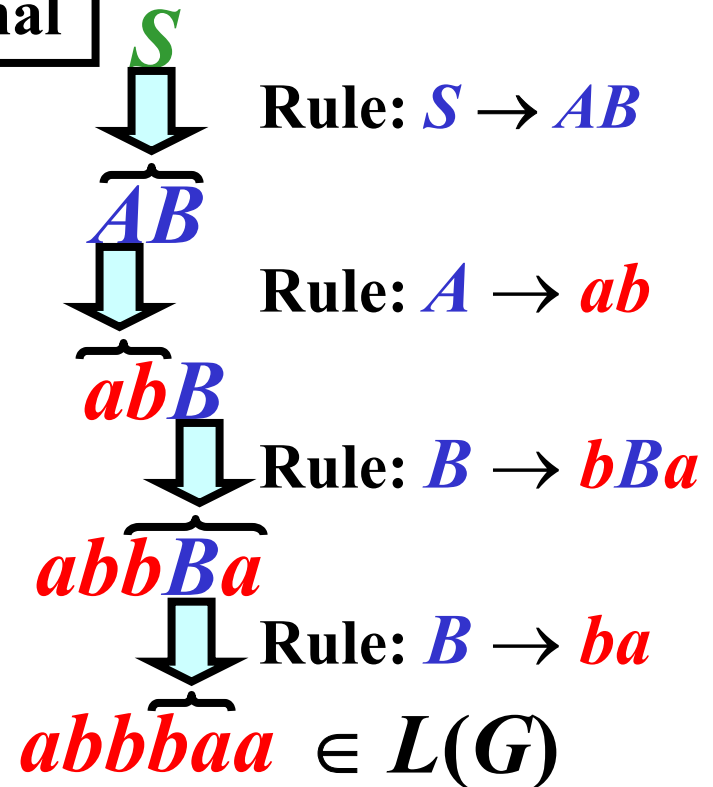
Grammar G :

Nonterminals: A, B, S

Terminals: a, b, c, d

Rules:

- $S \rightarrow AB,$
- $A \rightarrow aAb,$
- $A \rightarrow ab,$
- $B \rightarrow bBa,$
- $B \rightarrow ba$



Context-Free Grammar: Definition

Definition: A *context-free grammar* (CFG) is a quadruple $G = (N, T, P, S)$, where

- N is an alphabet of *nonterminals*
- T is an alphabet of *terminals*, $N \cap T = \emptyset$
- P is a finite set of *rules* of the form $A \rightarrow x$,
where $A \in N$, $x \in (N \cup T)^*$
- $S \in N$ is the *start nonterminal*

Mathematical Note on Rules:

- Strictly mathematically, P is a relation from N to $(N \cup T)^*$
 - Instead of $(A, x) \in P$, we write $A \rightarrow x \in P$
-
- $A \rightarrow x$ means that A can be replaced with x
 - $A \rightarrow \varepsilon$ is called *ε -rule*

Convention

- A, \dots, F, S : nonterminals
- S : the start nonterminal
- a, \dots, d : terminals
- U, \dots, Z : members of $(N \cup T)$
- u, \dots, z : members of $(N \cup T)^*$
- π : sequence of productions

A subset of rules of the form:

$$A \rightarrow x_1, A \rightarrow x_2, \dots, A \rightarrow x_n$$

can be simply written as:

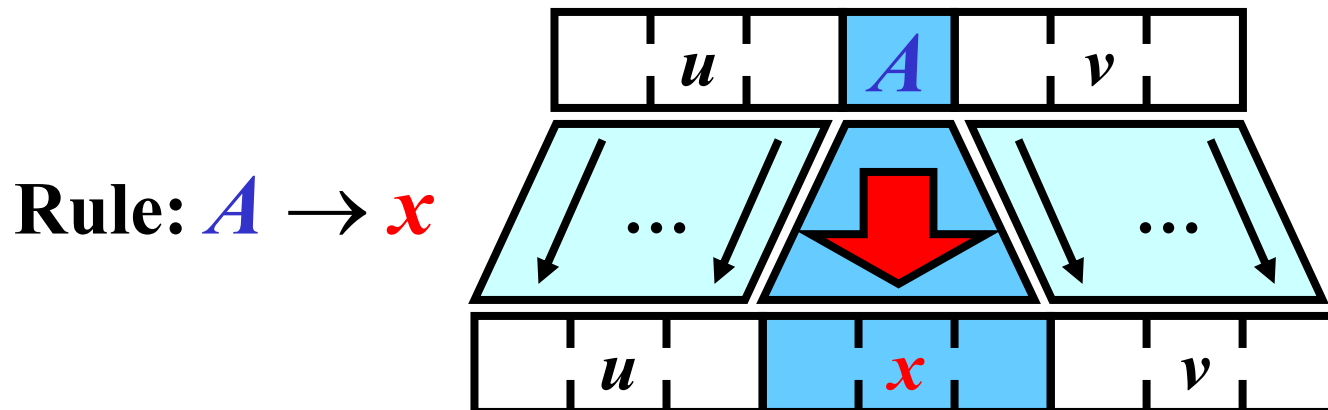
$$A \rightarrow x_1 \mid x_2 \mid \dots \mid x_n$$

Derivation Step

Gist: A change of a string by a rule.

Definition: Let $G = (N, T, P, S)$ be a CFG. Let $u, v \in (N \cup T)^*$ and $p = A \rightarrow x \in P$. Then, uAv *directly derives* uxv according to p in G , written as $uAv \Rightarrow uxv [p]$ or, simply, $uAv \Rightarrow uxv$.

Note: If $uAv \Rightarrow uxv$ in G , we also say that G makes a *derivation step* from uAv to uxv .



Sequence of Derivation Steps 1/2

Gist: Several consecutive derivation steps.

Definition: Let $u \in (N \cup T)^*$. G makes a *zero-step derivation* from u to u ; in symbols,

$$u \Rightarrow^0 u [\varepsilon] \text{ or, simply, } u \Rightarrow^0 u$$

Definition: Let $u_0, \dots, u_n \in (N \cup T)^*$, $n \geq 1$, and $u_{i-1} \Rightarrow u_i [p_i]$, $p_i \in P$, for all $i = 1, \dots, n$; that is

$$u_0 \Rightarrow u_1 [p_1] \Rightarrow u_2 [p_2] \dots \Rightarrow u_n [p_n]$$

Then, G makes n *derivation steps* from u_0 to u_n ,

$$u_0 \Rightarrow^n u_n [p_1 \dots p_n] \text{ or, simply, } u_0 \Rightarrow^n u_n$$

Sequence of Derivation Steps 2/2

If $u_0 \Rightarrow^n u_n [\pi]$ for some $n \geq 1$, then u_0 *properly derives* u_n in G , written as $u_0 \Rightarrow^+ u_n [\pi]$.

If $u_0 \Rightarrow^n u_n [\pi]$ for some $n \geq 0$, then u_0 *derives* u_n in G , written as $u_0 \Rightarrow^* u_n [\pi]$.

Example: Consider

$aAb \Rightarrow a**Bb** \quad [1: A \rightarrow **bBb**],$ and

$aa**Bbb** \Rightarrow aa**cbb** \quad [2: **B** \rightarrow **c**].$

Then, $aAb \Rightarrow^2 aa**cbb** [1 \ 2],$

$aAb \Rightarrow^+ aa**cbb** [1 \ 2],$

$aAb \Rightarrow^* aa**cbb** [1 \ 2]$

Generated Language

Gist: G generates a terminal string w by a sequence of derivation steps from S to w

Definition: Let $G = (N, T, P, S)$ be a CFG. The language generated by G , $L(G)$, is defined as

$$L(G) = \{w: w \in T^*, S \Rightarrow^* w\}$$

Illustration:

$G = (N, T, P, S)$, let $w = a_1 a_2 \dots a_n$; $a_i \in T$ for $i = 1..n$

if $S \Rightarrow \dots \Rightarrow \dots \Rightarrow \underbrace{a_1 a_2 \dots a_n}_w$ then $w \in L(G)$;

otherwise, $w \notin L(G)$

Context-Free Language (CFL)

Gist: A language generated by a CFG.

Definition: Let L be a language. L is a *context-free language* (CFL) if there exists a context-free grammar that generates L .

Example:

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

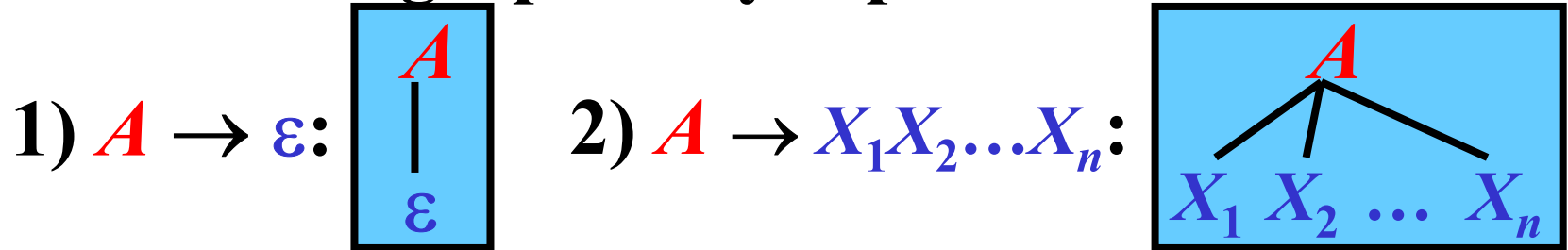
$P = \{1: S \rightarrow aSb, 2: S \rightarrow \epsilon\}$

$S \Rightarrow \epsilon$ [2] \nearrow $L(G) = \{a^n b^n: n \geq 0\}$
 $S \Rightarrow aSb$ [1] $\Rightarrow ab$ [2] \nearrow
 $S \Rightarrow aSb$ [1] $\Rightarrow aaSbb$ [1] $\Rightarrow aabb$ [2]
 \vdots

$L = \{a^n b^n: n \geq 0\}$ is a CFL.

Rule Tree

- Rule tree graphically represents a rule



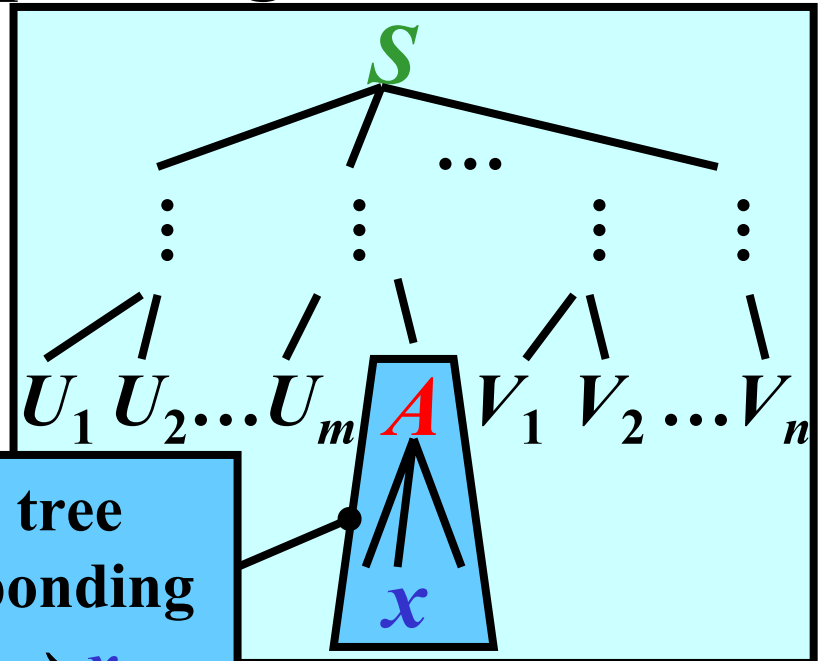
- Derivation tree corresponding to a derivation

$S \Rightarrow \dots$

\vdots

$\Rightarrow U_1 U_2 \dots U_m A V_1 V_2 \dots V_n$

$\Rightarrow U_1 U_2 \dots U_m x V_1 V_2 \dots V_n$



Rule tree
corresponding
to $A \rightarrow x$

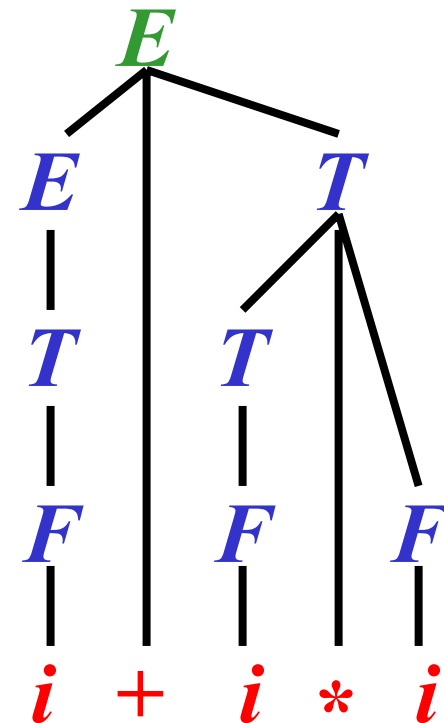
Derivation Tree: Example

$G = (N, T, P, \underline{E})$, where $N = \{\underline{E}, F, T\}$, $T = \{i, +, *, (,)\}$,
 $P = \{$ 1 $: \underline{E} \rightarrow E + T,$ 2 $: E \rightarrow T,$ 3 $: T \rightarrow T * F,$
4 $: T \rightarrow F,$ 5 $: F \rightarrow (E),$ 6 $: F \rightarrow i$ $\}$

Derivation:

$\underline{E} \Rightarrow E + \underline{T}$ [1]
 $\Rightarrow E + \underline{T} * F$ [3]
 $\Rightarrow E + \underline{F} * F$ [4]
 $\Rightarrow \underline{E} + i * F$ [6]
 $\Rightarrow T + i * \underline{E}$ [2]
 $\Rightarrow \underline{T} + i * i$ [6]
 $\Rightarrow \underline{F} + i * i$ [4]
 $\Rightarrow i + i * i$ [6]

Derivation tree:



Leftmost Derivation

Gist: During a *leftmost derivation step*, the **leftmost nonterminal is rewritten**.

Definition: Let $G = (N, T, P, S)$ be a CFG, let $u \in T^*$, $v \in (N \cup T)^*$. Let $p = A \rightarrow x \in P$ be a rule. Then, uAv directly derives uxv in the *leftmost way* according to p in G , written as

$$uAv \Rightarrow_{lm} uxv [p]$$

Note: We define \Rightarrow_{lm}^+ and \Rightarrow_{lm}^* by analogy with \Rightarrow^+ and \Rightarrow^* , respectively.

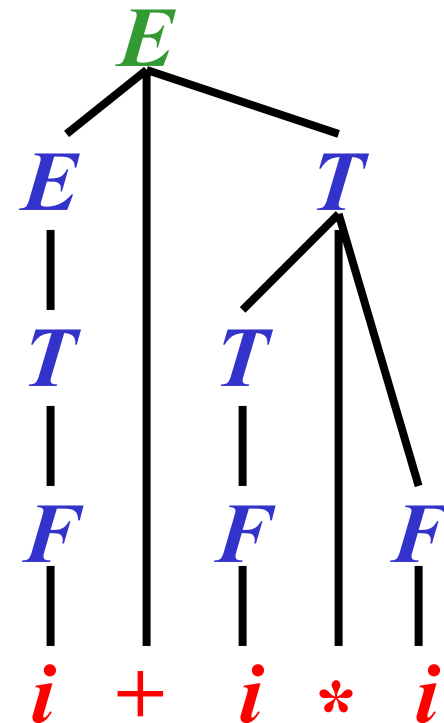
Leftmost Derivation: Example

$G = (N, T, P, \mathbf{E})$, where $N = \{\mathbf{E}, \mathbf{F}, \mathbf{T}\}$, $T = \{\mathbf{i}, +, *, (,)\}$,
 $P = \{$ 1 $: \mathbf{E} \rightarrow \mathbf{E} + \mathbf{T},$ 2 $: \mathbf{E} \rightarrow \mathbf{T},$ 3 $: \mathbf{T} \rightarrow \mathbf{T} * \mathbf{F},$
4 $: \mathbf{T} \rightarrow \mathbf{F},$ 5 $: \mathbf{F} \rightarrow (\mathbf{E}),$ 6 $: \mathbf{F} \rightarrow \mathbf{i}$ $\}$

Leftmost derivation:

$$\begin{aligned}
 \underline{\mathbf{E}} &\Rightarrow_{lm} \underline{\mathbf{E}} + \mathbf{T} && [\text{1}] \\
 &\Rightarrow_{lm} \underline{\mathbf{T}} + \mathbf{T} && [\text{2}] \\
 &\Rightarrow_{lm} \underline{\mathbf{F}} + \mathbf{T} && [\text{4}] \\
 &\Rightarrow_{lm} \mathbf{i} + \underline{\mathbf{T}} && [\text{6}] \\
 &\Rightarrow_{lm} \mathbf{i} + \underline{\mathbf{T}} * \mathbf{F} && [\text{3}] \\
 &\Rightarrow_{lm} \mathbf{i} + \underline{\mathbf{F}} * \mathbf{F} && [\text{4}] \\
 &\Rightarrow_{lm} \mathbf{i} + \mathbf{i} * \underline{\mathbf{F}} && [\text{6}] \\
 &\Rightarrow_{lm} \mathbf{i} + \mathbf{i} * \mathbf{i} && [\text{6}]
 \end{aligned}$$

Derivation tree:



Rightmost Derivation

Gist: During a *rightmost derivation step*, the **rightmost nonterminal** is rewritten.

Definition: Let $G = (N, T, P, S)$ be a CFG, let $u \in (N \cup T)^*$, $v \in T^*$. Let $p = A \rightarrow x \in P$ be a rule. Then, uAv directly derives uxv in the *rightmost way* according to p in G , written as

$$uAv \Rightarrow_{rm} uxv [p]$$

Note: We define \Rightarrow_{rm}^+ and \Rightarrow_{rm}^* by analogy with \Rightarrow^+ and \Rightarrow^* , respectively.

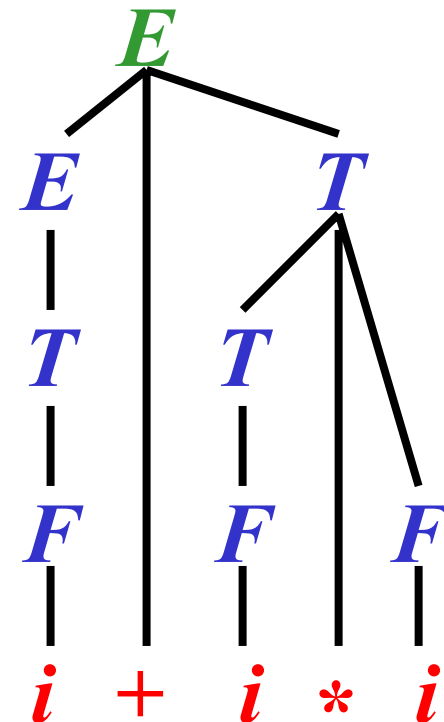
Rightmost Derivation: Example

$G = (N, T, P, \underline{E})$, where $N = \{\underline{E}, \underline{F}, \underline{T}\}$, $T = \{i, +, *, (,)\}$,
 $P = \{$ 1 $: \underline{E} \rightarrow \underline{E} + \underline{T},$ 2 $: \underline{E} \rightarrow \underline{T},$ 3 $: \underline{T} \rightarrow \underline{T} * \underline{F},$
4 $: \underline{T} \rightarrow \underline{F},$ 5 $: \underline{F} \rightarrow (\underline{E}),$ 6 $: \underline{F} \rightarrow i$ $\}$

Rightmost derivation:

$$\begin{aligned}
 \underline{E} &\Rightarrow_{rm} \underline{E} + \underline{T} && [1] \\
 &\Rightarrow_{rm} \underline{E} + \underline{T} * \underline{F} && [3] \\
 &\Rightarrow_{rm} \underline{E} + \underline{T} * i && [6] \\
 &\Rightarrow_{rm} \underline{E} + \underline{F} * i && [4] \\
 &\Rightarrow_{rm} \underline{E} + i * i && [6] \\
 &\Rightarrow_{rm} \underline{T} + i * i && [2] \\
 &\Rightarrow_{rm} \underline{F} + i * i && [4] \\
 &\Rightarrow_{rm} i + i * i && [6]
 \end{aligned}$$

Derivation tree:



Derivations: Summary

- Let $A \rightarrow x \in P$ be a rule.
-

1) Derivation:

Let $u, v \in (N \cup T)^*$: $uAv \Rightarrow uxv$

Note: Any nonterminal is rewritten

2) Leftmost derivation:

Let $u \in T^*, v \in (N \cup T)^*$: $uAv \Rightarrow_{\text{lm}} uxv$

Note: Leftmost nonterminal is rewritten

3) Rightmost derivation:

Let $u \in (N \cup T)^*, v \in T^*$: $uAv \Rightarrow_{\text{rm}} uxv$

Note: Rightmost nonterminal is rewritten

Reduction of the Number of Derivations

Gist: Without any loss of generality, we can consider only leftmost or rightmost derivations.

Theorem: Let $G = (N, T, P, S)$ be a CFG. The next three languages coincide

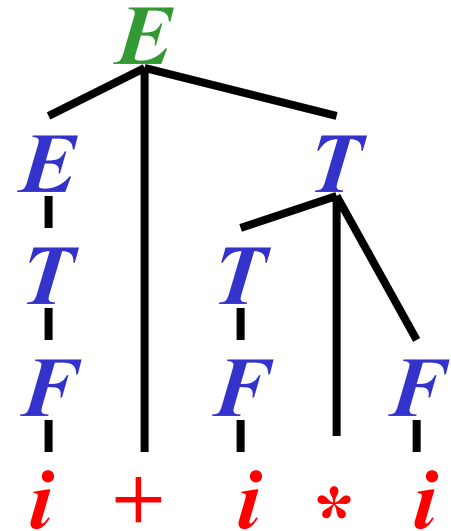
$$(1) \{w: w \in T^*, S \Rightarrow_{lm}^* w\}$$

$$(2) \{w: w \in T^*, S \Rightarrow_{rm}^* w\}$$

$$(3) \{w: w \in T^*, S \Rightarrow^* w\} = L(G)$$

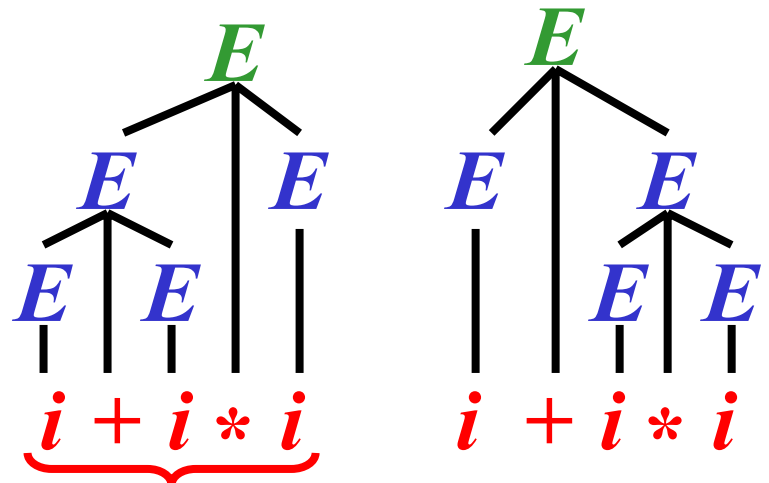
Introduction to Ambiguity

$G_{expr1} = (N, T, P, E)$, where
 $N = \{E, F, T\}$, $T = \{i, +, *, (,)\}$,
 $P = \{$
 1: $E \rightarrow E+T$, 2: $E \rightarrow T$,
 3: $T \rightarrow T*F$, 4: $T \rightarrow F$,
 5: $F \rightarrow (E)$, 6: $F \rightarrow i\}$



Theory: ☹️ × **Practice:** 😊

$G_{expr2} = (N, T, P, E)$, where
 $N = \{E\}$, $T = \{i, +, *, (,)\}$,
 $P = \{$
 1: $E \rightarrow E+E$, 2: $E \rightarrow E*E$,
 3: $E \rightarrow (E)$, 4: $E \rightarrow i\}$



Theory: 😊 × **Practice:** ☹️

Note: $L(G_{expr1}) = L(G_{expr2})$

Improper during compilation

Grammatical Ambiguity

Definition: Let $G = (N, T, P, S)$ be a CFG. If there exists $x \in L(G)$ with more than one derivation tree, then G is *ambiguous*; otherwise, G is *unambiguous*.

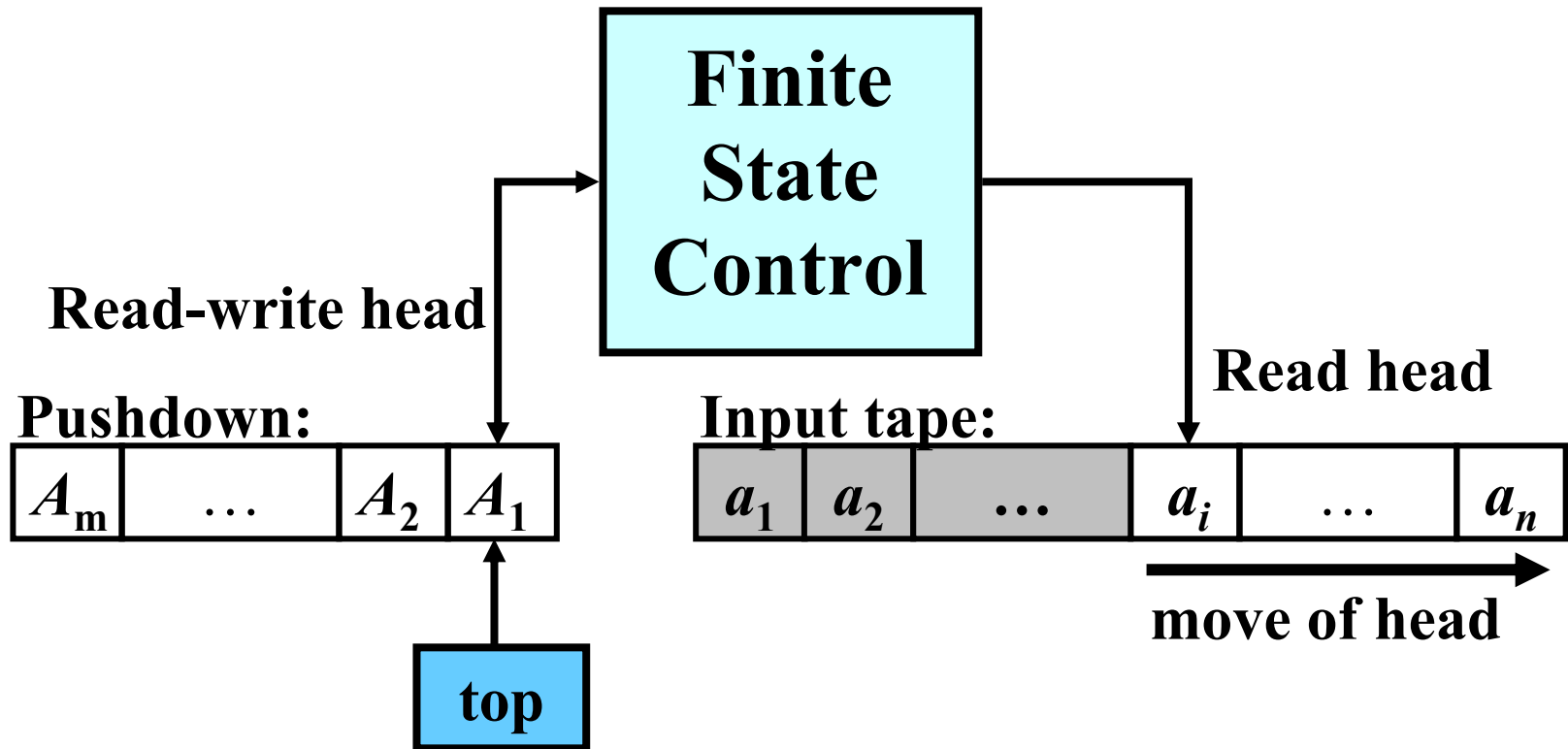
Definition: A CFL, L , is *inherently ambiguous* if L is generated by no unambiguous grammar.

Example:

- G_{expr1} is **unambiguous**, because for every $x \in L(G_{expr1})$ there exists **only one derivation tree**
- G_{expr2} is **ambiguous**, because for $i+i*i \in L(G_{expr2})$ there exist **two derivation trees**
- $L_{expr} = L(G_{expr1}) = L(G_{expr2})$ is **not inherently ambiguous** because G_{expr1} is unambiguous

Pushdown Automata (PDA)

Gist: An FA extended by a pushdown store.



Pushdown Automata: Definition

Definition: A *pushdown automaton* (PDA) is a 7-tuple $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where

- Q is a *finite set of states*
- Σ is an *input alphabet*
- Γ is a *pushdown alphabet*
- R is a *finite set of rules* of the form: $Apa \rightarrow wq$
where $A \in \Gamma, p, q \in Q, a \in \Sigma \cup \{\varepsilon\}, w \in \Gamma^*$
- $s \in Q$ is the *start state*
- $S \in \Gamma$ is the *start pushdown symbol*
- $F \subseteq Q$ is a set of *final states*

Notes on PDA Rules

Mathematical note on rules:


- Strictly mathematically, R is a relation from $\Gamma \times Q \times (\Sigma \cup \{\varepsilon\})$ to $\Gamma^* \times Q$
- Instead of $(\textcolor{green}{A}\textcolor{blue}{p}\textcolor{blue}{a}, \textcolor{green}{w}\textcolor{red}{q}) \in R$, however, we write $\textcolor{green}{A}\textcolor{blue}{p}\textcolor{blue}{a} \rightarrow \textcolor{green}{w}\textcolor{red}{q} \in R$

- **Interpretation of $\textcolor{green}{A}\textcolor{blue}{p}\textcolor{blue}{a} \rightarrow \textcolor{green}{w}\textcolor{red}{q}$:** if the current state is $\textcolor{red}{p}$, current input symbol is $\textcolor{blue}{a}$, and the topmost symbol on the pushdown is $\textcolor{green}{A}$, then M can read $\textcolor{blue}{a}$, replace $\textcolor{green}{A}$ with $\textcolor{green}{w}$ and change state $\textcolor{red}{p}$ to $\textcolor{red}{q}$.
- **Note:** if $\textcolor{blue}{a} = \varepsilon$, no symbol is read

Graphical Representation

 represents $q \in Q$

 represents the initial state $s \in Q$

 represents a final state $f \in F$

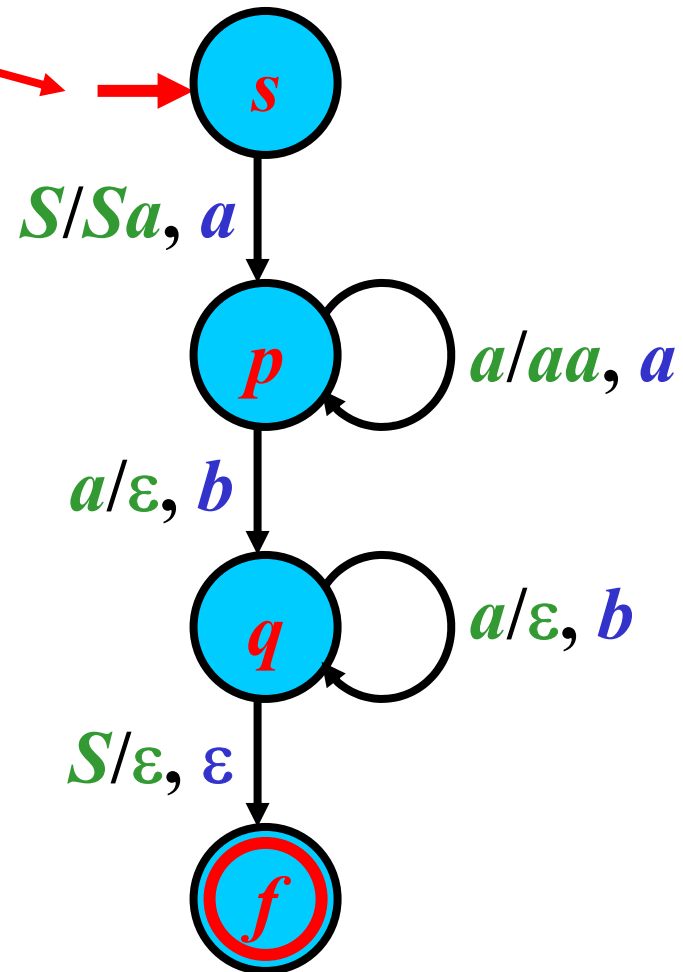
 $\xrightarrow{A/w, a}$  denotes $Apa \rightarrow wq \in R$

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

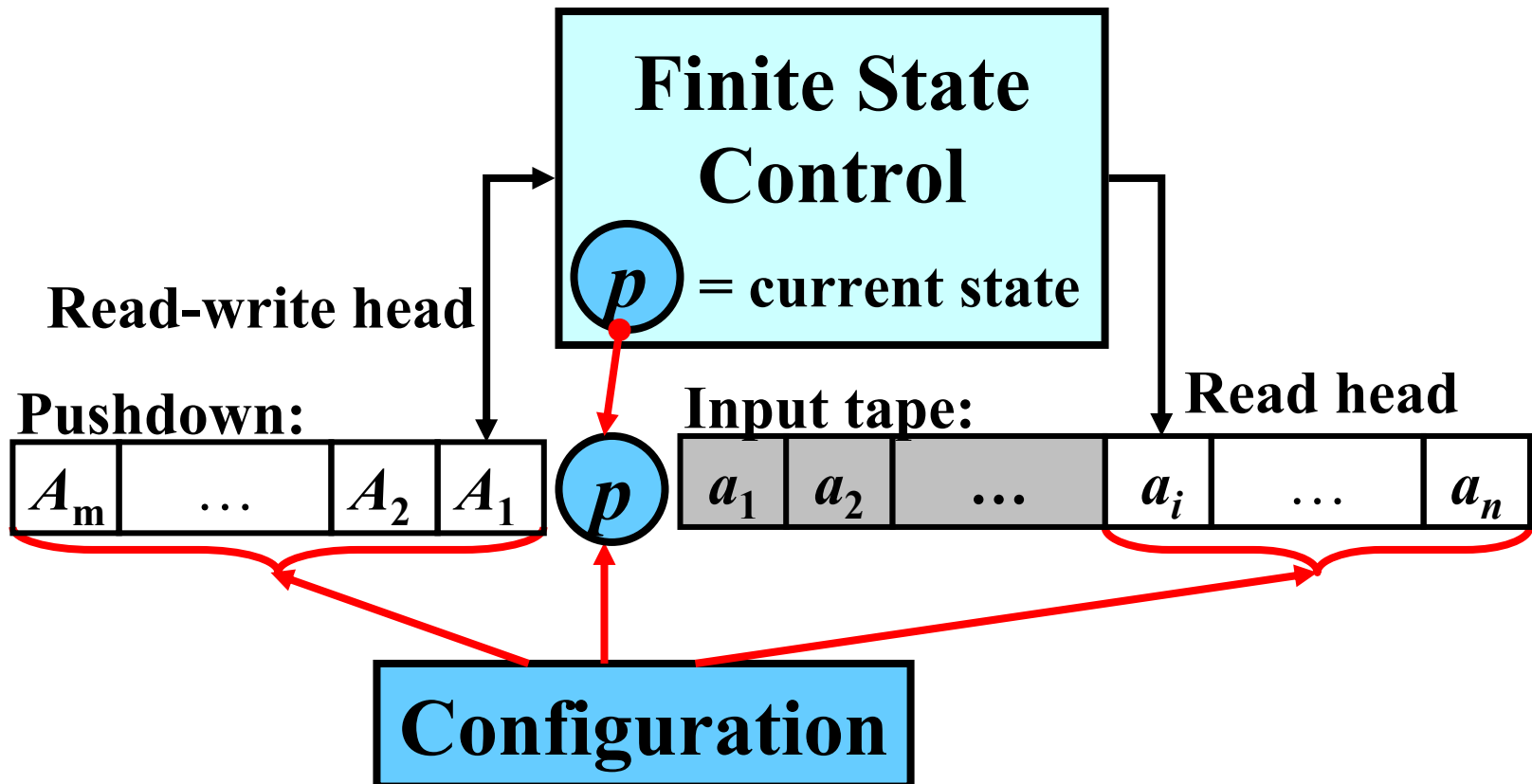
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap, \\ apa \rightarrow aap, \\ apb \rightarrow q, \\ aqb \rightarrow q, \\ Sq \rightarrow f\}$
- $F = \{f\}$



PDA Configuration

Gist: Instantaneous description of PDA

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, S, F)$ be a PDA. A configuration of M is a string $\chi \in \Gamma^* Q \Sigma^*$



Move

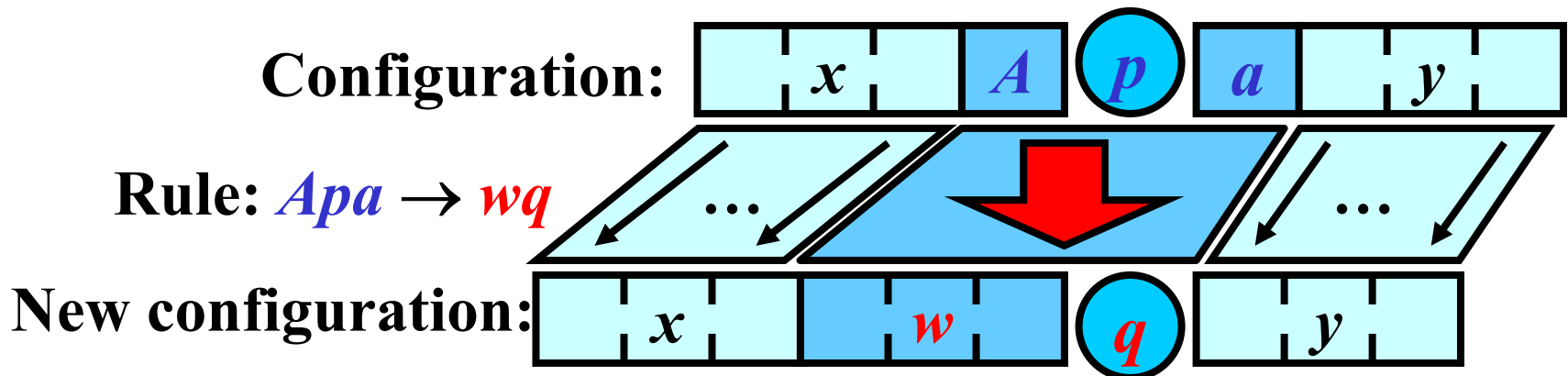
Gist: A computational step made by a PDA

Definition: Let $xApay$ and $xwqy$ be two configurations of a PDA, M , where

$x, w \in \Gamma^*$, $A \in \Gamma$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $y \in \Sigma^*$.

Let $r = Apa \rightarrow wq \in R$ be a rule. Then, M makes a move from $xApay$ to $xwqy$ according to r , written as $xApay \vdash xwqy [r]$ or, simply, $xApay \vdash xwqy$.

Note: if $a = \varepsilon$, no input symbol is read



Sequence of Moves 1/2

Gist: Several consecutive computational steps

Definition: Let χ be a configuration. M makes *zero moves* from χ to χ ; in symbols,

$$\chi \vdash^0 \chi [\varepsilon] \text{ or, simply, } \chi \vdash^0 \chi$$

Definition: Let $\chi_0, \chi_1, \dots, \chi_n$ be a sequence of configurations, $n \geq 1$, and $\chi_{i-1} \vdash \chi_i [r_i]$, $r_i \in R$, for all $i = 1, \dots, n$; that is,

$$\chi_0 \vdash \chi_1 [r_1] \vdash \chi_2 [r_2] \dots \vdash \chi_n [r_n]$$

Then M makes n moves from χ_0 to χ_n ,

$$\chi_0 \vdash^n \chi_n [r_1 \dots r_n] \text{ or, simply, } \chi_0 \vdash^n \chi_n$$

Sequence of Moves 2/2

If $\chi_0 \vdash^n \chi_n [\rho]$ for some $n \geq 1$, then
 $\chi_0 \vdash^+ \chi_n [\rho]$ or, simply, $\chi_0 \vdash^+ \chi_n$

If $\chi_0 \vdash^n \chi_n [\rho]$ for some $n \geq 0$, then
 $\chi_0 \vdash^* \chi_n [\rho]$ or, simply, $\chi_0 \vdash^* \chi_n$

Example: Consider

$AApabc \vdash ABqbc$ [1: $Apa \rightarrow Bq$], and

$ABqbc \vdash ABCrc$ [2: $Bqb \rightarrow BCrc$].

Then, $AApabc \vdash^2 ABCrc$ [1 2],

$AApabc \vdash^+ ABCrc$ [1 2],

$AApabc \vdash^* ABCrc$ [1 2]

Accepted Language: Three Types

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, S, F)$ be a PDA.

- 1) The *language that M accepts by final state*, denoted by $L(M)_f$, is defined as

$$L(M)_f = \{w: w \in \Sigma^*, Ssw \mid^* zf, \textcolor{blue}{z} \in \Gamma^*, \textcolor{red}{f} \in \textcolor{red}{F}\}$$
- 2) The *language that M accepts by empty pushdown*, denoted by $L(M)_\varepsilon$, is defined as

$$L(M)_\varepsilon = \{w: w \in \Sigma^*, Ssw \mid^* zf, \textcolor{red}{z} = \varepsilon, \textcolor{blue}{f} \in \textcolor{blue}{Q}\}$$
- 3) The *language that M accepts by final state and empty pushdown*, denoted by $L(M)_{f\varepsilon}$, is defined as

$$L(M)_{f\varepsilon} = \{w: w \in \Sigma^*, Ssw \mid^* zf, \textcolor{red}{z} = \varepsilon, \textcolor{red}{f} \in \textcolor{red}{F}\}$$

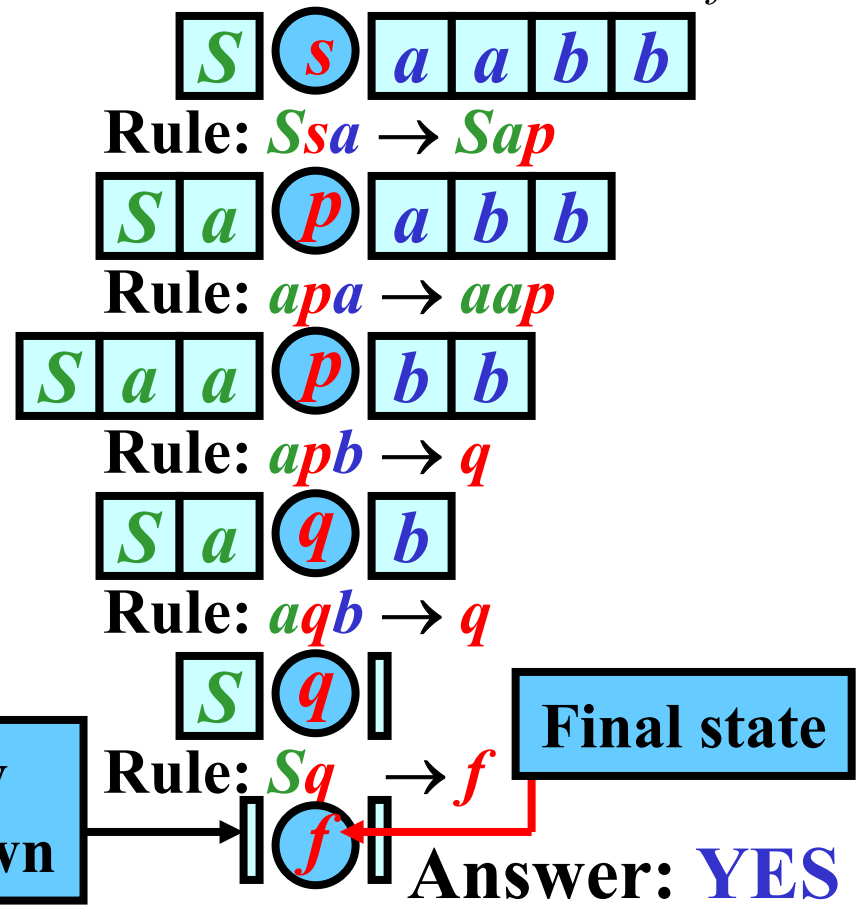
PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap, \quad apa \rightarrow aap, \quad apb \rightarrow q, \quad aqb \rightarrow q, \quad Sq \rightarrow f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$?



$Ssaabb \mid Sapabb \mid Saapbb \mid Saqb \mid Sq \mid f$

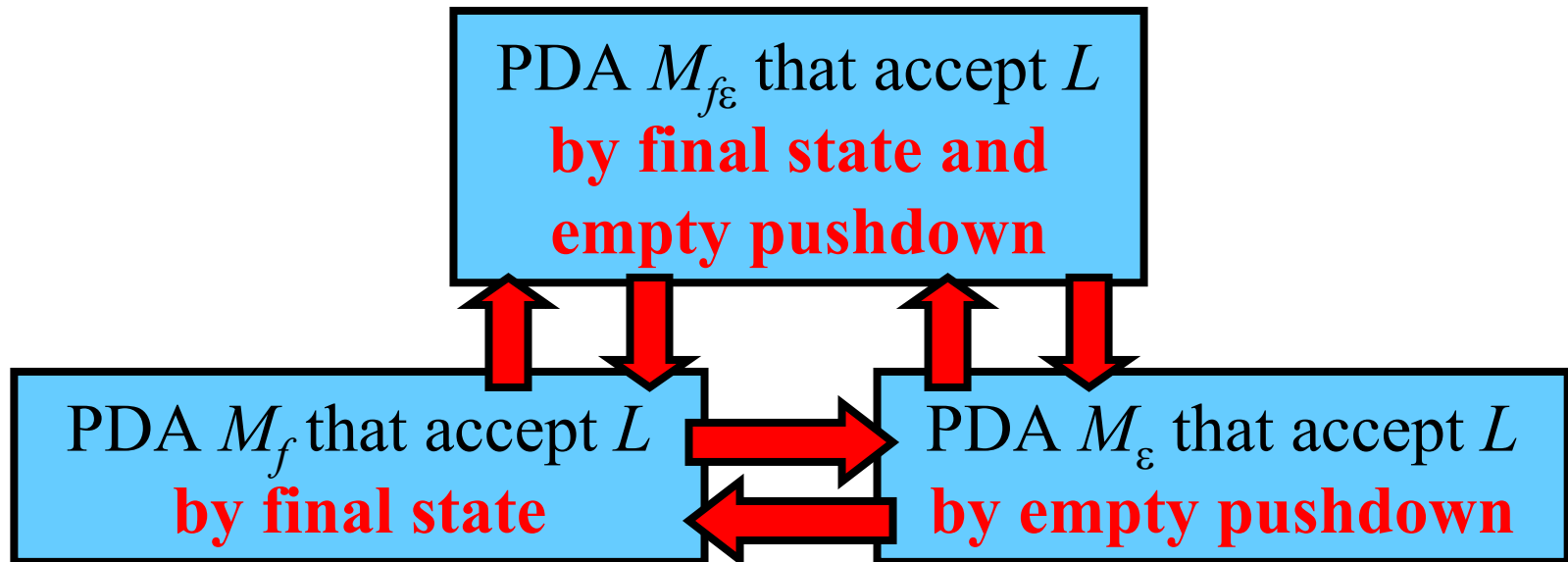
Note: $L(M)_f = L(M)_\varepsilon = L(M)_{f\varepsilon} = \{a^n b^n : n \geq 1\}$

Three Types of Acceptance: Equivalence

Theorem:

- $L = L(M_f)_f$ for a PDA $M_f \Leftrightarrow L = L(M_{f\varepsilon})_{f\varepsilon}$ for a PDA $M_{f\varepsilon}$
- $L = L(M_\varepsilon)_\varepsilon$ for a PDA $M_\varepsilon \Leftrightarrow L = L(M_{f\varepsilon})_{f\varepsilon}$ for a PDA $M_{f\varepsilon}$
- $L = L(M_f)_f$ for a PDA $M_f \Leftrightarrow L = L(M_\varepsilon)_\varepsilon$ for a PDA M_ε

Note: There exist these conversions:



Deterministic PDA (DPDA)

Gist: Deterministic PDA makes no more than one move from any configuration.

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, S, F)$ be a PDA. M is a *deterministic PDA* if for each rule $Apa \rightarrow wq \in R$, it holds that $R - \{Apa \rightarrow wq\}$ contains no rule with the left-hand side equal to Apa or Ap .

Illustration:

Configuration:



No more than one rule of the forms

$Ap \rightarrow w_1q_1$
 $Apa \rightarrow w_2q_2$

PDAs are Stronger than DPDAs

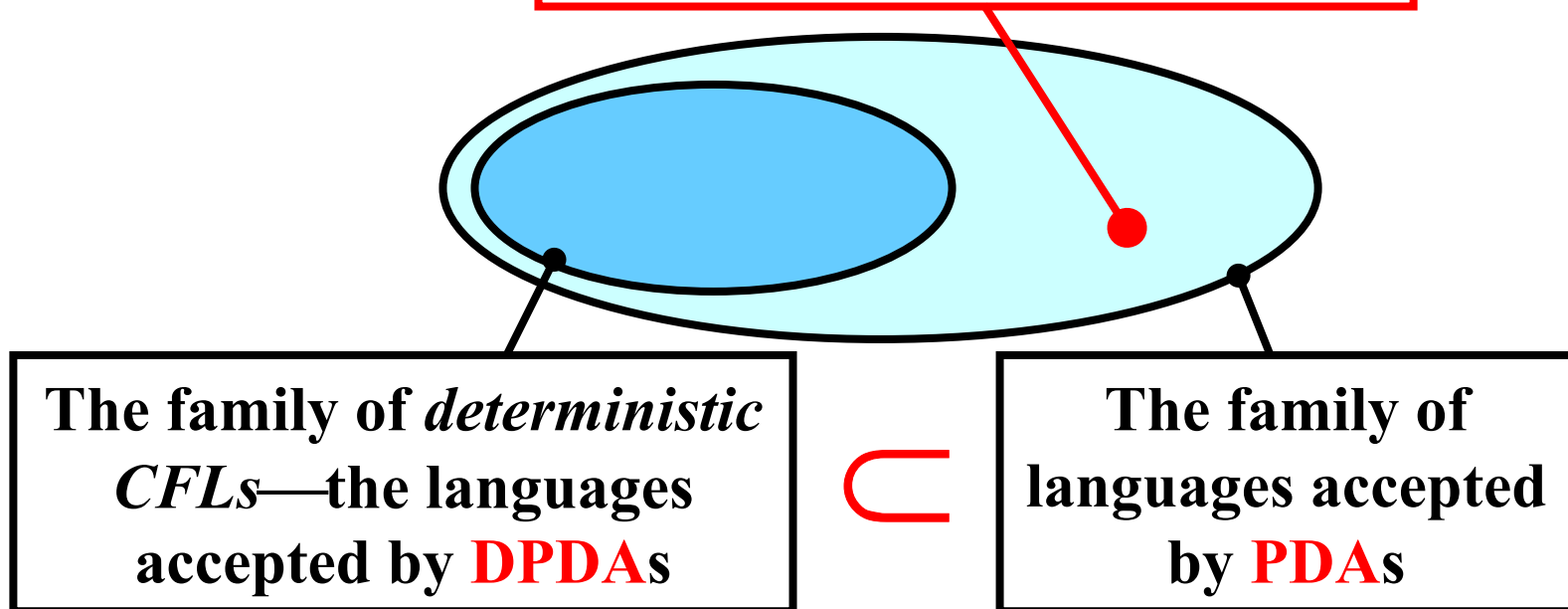
Theorem: There exists no DPDA M_{f_ε} that accepts

$$L = \{xy: x, y \in \Sigma^*, y = \text{reversal}(x)\}$$

Proof: See page 431 in [Meduna: Automata and Languages]

Illustration:

$$L = \{xy: x, y \in \Sigma^*, y = \text{reversal}(x)\}$$



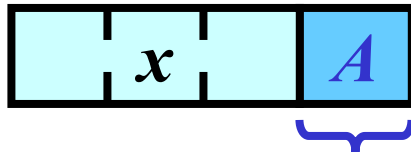
Extended PDA (EPDA)

Gist: The pushdown top of an EPDA represents a string rather than a single symbol.

Definition: *An Extended Pushdown automaton (EPDA) is a 7-tuple $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where $Q, \Sigma, \Gamma, s, S, F$ are defined as in an PDA and R is a finite set of rules of the form: $\mathbf{v}pa \rightarrow wq$, where $\mathbf{v}, w \in \Gamma^*, p, q \in Q, a \in \Sigma \cup \{\epsilon\}$*

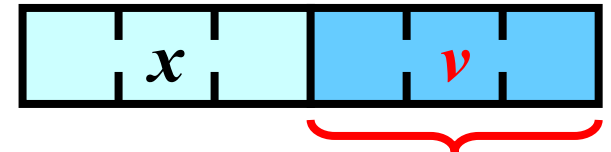
Illustration:

Pushdown of PDA:



PDA has a **single symbols** as the pushdown top

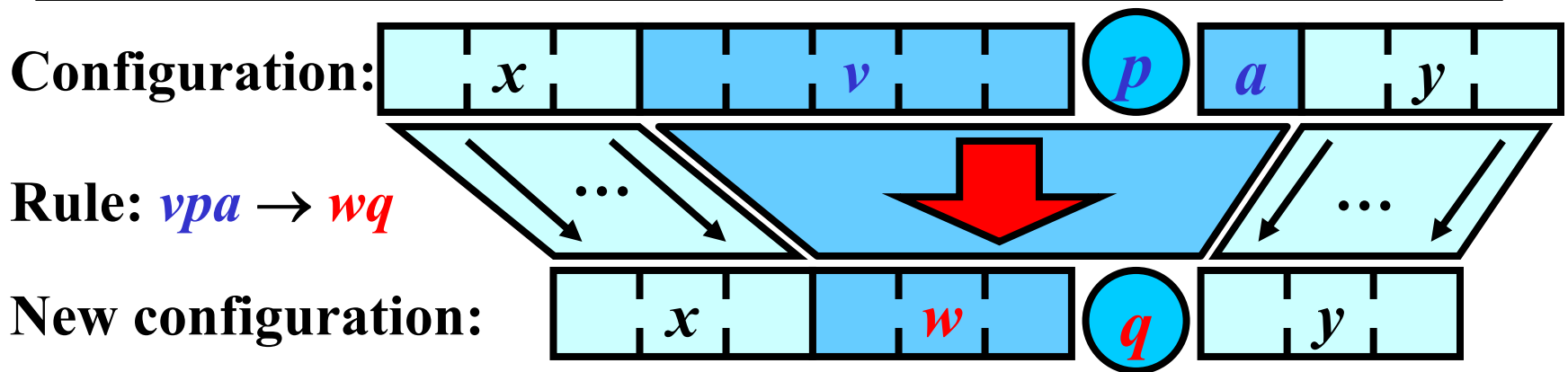
Pushdown of EPDA:



EPDA has **a string** as the pushdown top

Move in EPDA

Definition: Let $x\mathbf{v}pay$ and $xwqy$ be two configurations of an EPDA, M , where $x, \mathbf{v}, w \in \Gamma^*$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $y \in \Sigma^*$. Let $r = \mathbf{v}pa \rightarrow wq \in R$ be a rule. Then, M makes a *move* from $x\mathbf{v}pay$ to $xwqy$ according to r , written as $x\mathbf{v}pay \vdash xwqy [r]$ or $x\mathbf{v}pay \vdash xwqy$.



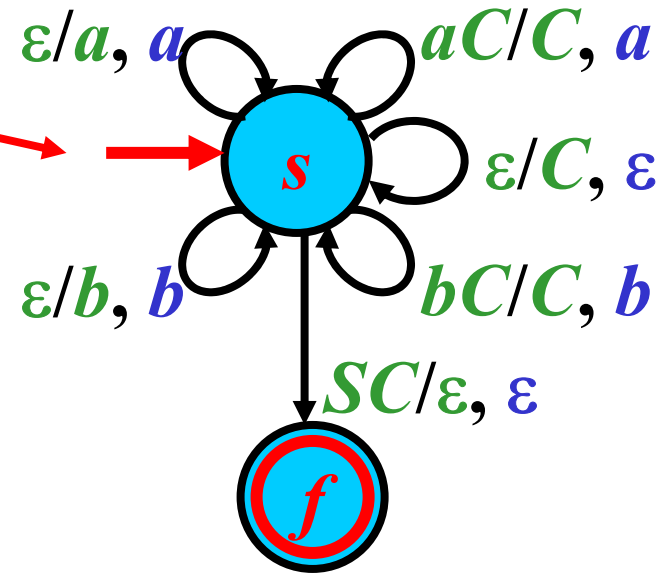
Note: \vdash^n , \vdash^+ , \vdash^* , $L(M)_f$, $L(M)_\varepsilon$, and $L(M)_{f_\varepsilon}$ are defined analogously to the corresponding definitions for PDA.

EPDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, f\};$
- $\Sigma = \{a, b\};$
- $\Gamma = \{a, b, S, C\};$
- $R = \{$
 - $sa \rightarrow as,$
 - $sb \rightarrow bs,$
 - $s \rightarrow Cs,$
 - $aCsa \rightarrow Cs,$
 - $bCsb \rightarrow Cs,$
 - $SCs \rightarrow f$
- $F = \{f\}$



Question: $abba \in L_{f\varepsilon}(M)?$

$Ssabba \mid - Sasbba \mid - Sab_sba$
 $\mid - SabCsba \mid - SaCsa$
 $\mid - SCs \mid - f$

Answer: YES

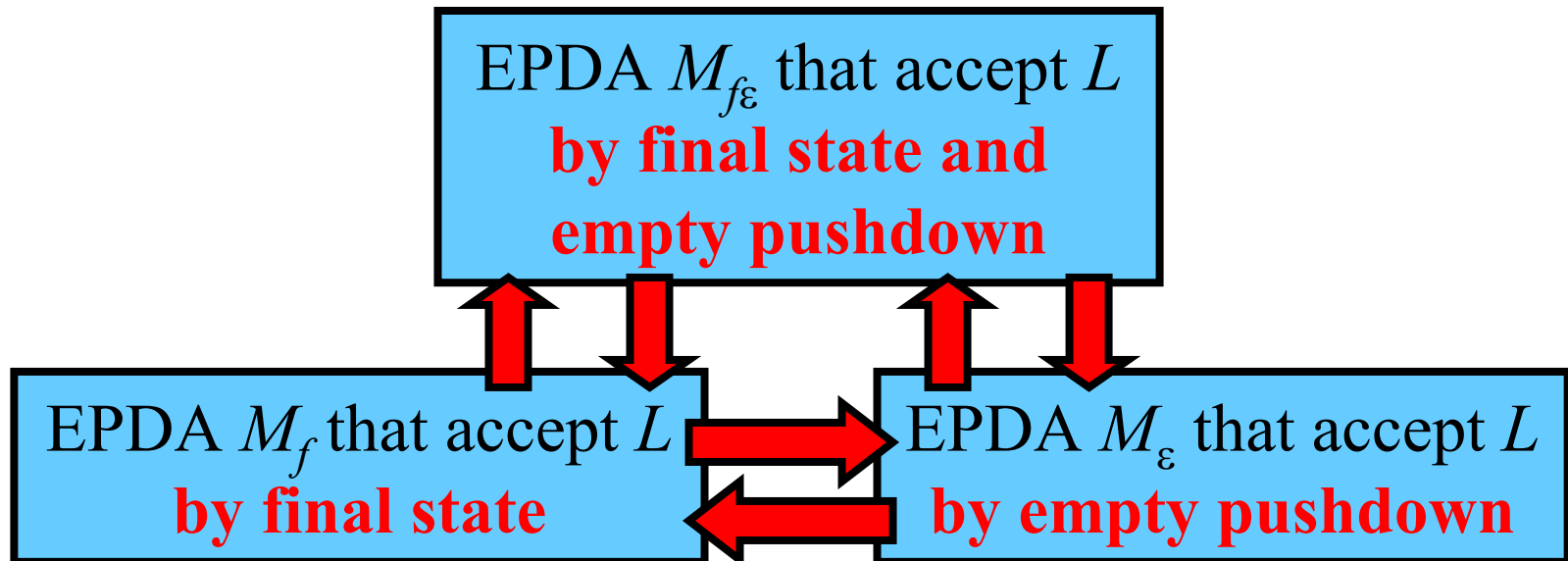
Note: $L(M)_f = L(M)_\varepsilon = L(M)_{f\varepsilon} = \{xy: x, y \in \Sigma^*, y = \text{reversal}(x)\}$

Three Types of Acceptance: Equivalence

Theorem:

- $L = L(M_f)_f$ for an EPDA $M_f \Leftrightarrow L = L(M_{f\varepsilon})_{f\varepsilon}$ for an EPDA $M_{f\varepsilon}$
- $L = L(M_\varepsilon)_\varepsilon$ for an EPDA $M_\varepsilon \Leftrightarrow L = L(M_{f\varepsilon})_{f\varepsilon}$ for an EPDA $M_{f\varepsilon}$
- $L = L(M_f)_f$ for an EPDA $M_f \Leftrightarrow L = L(M_\varepsilon)_\varepsilon$ for an EPDA M_ε

Note: There exist these conversion:

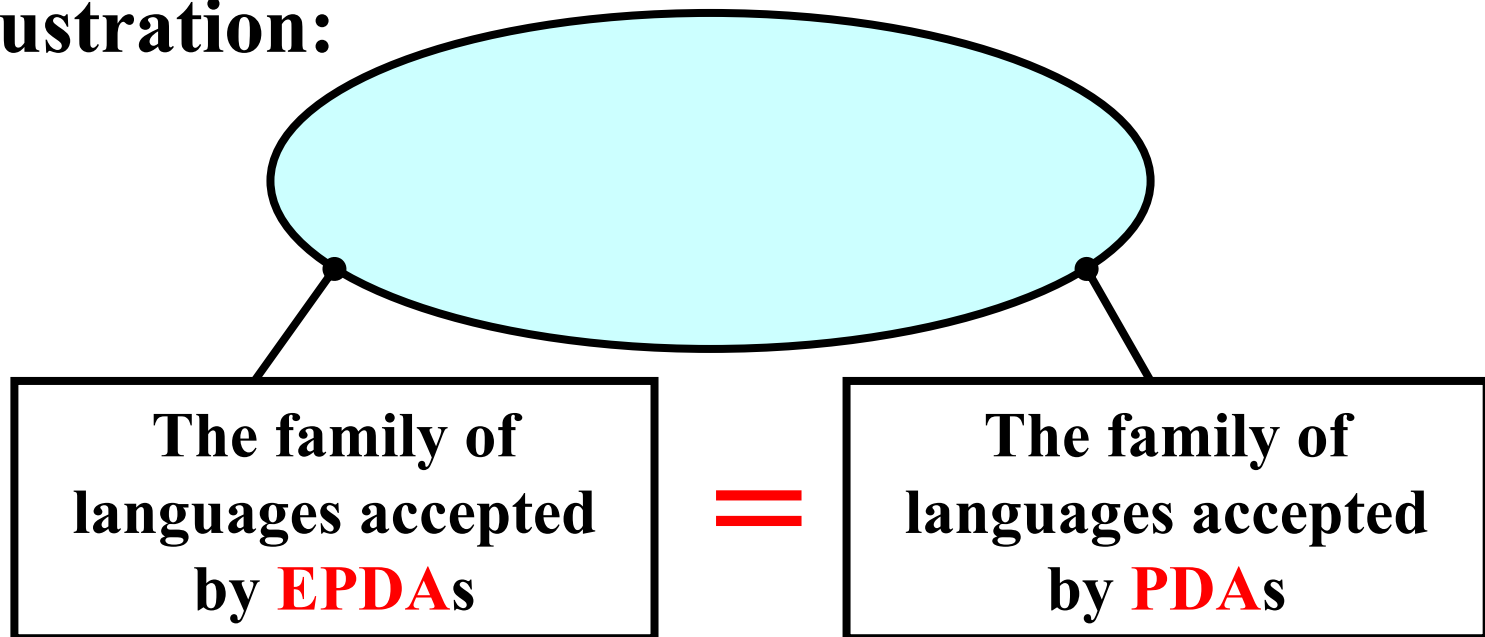


EPDAs and PDAs are Equivalent

Theorem: For every EPDA M , there is a PDA M' ,
and $L(M)_f = L(M')_f$.

Proof: See page 419 in [Meduna: Automata and Languages]

Illustration:

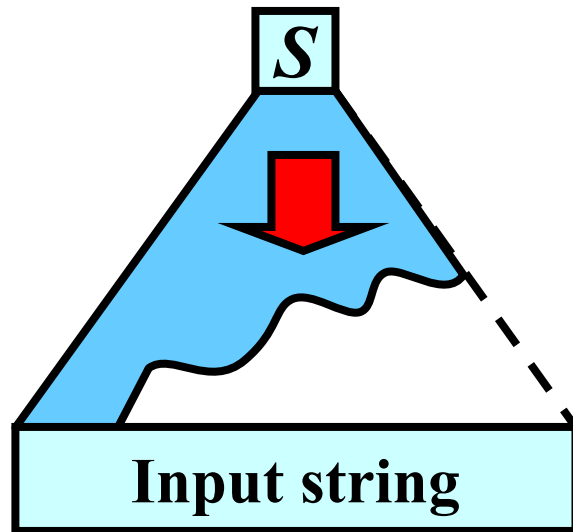


EPDAs and PDAs as Parsing Models for CFGs

Gist: An EPDA or a PDA can simulate the construction of a derivation tree for a CFG

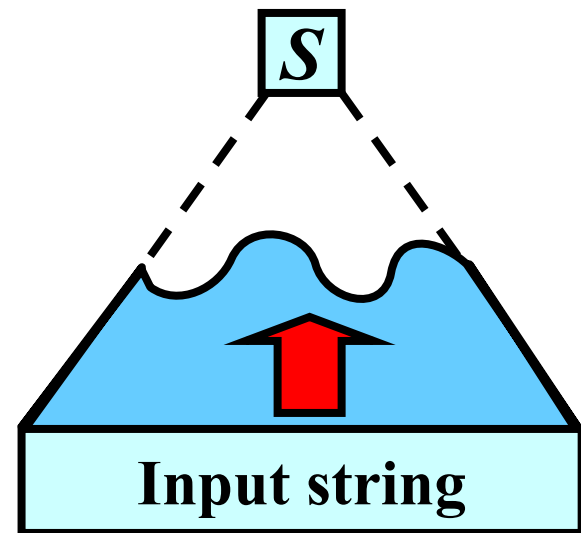
• Two basic approaches:

1) Top-Down Parsing



From S towards the input string

2) Bottom-Up Parsing

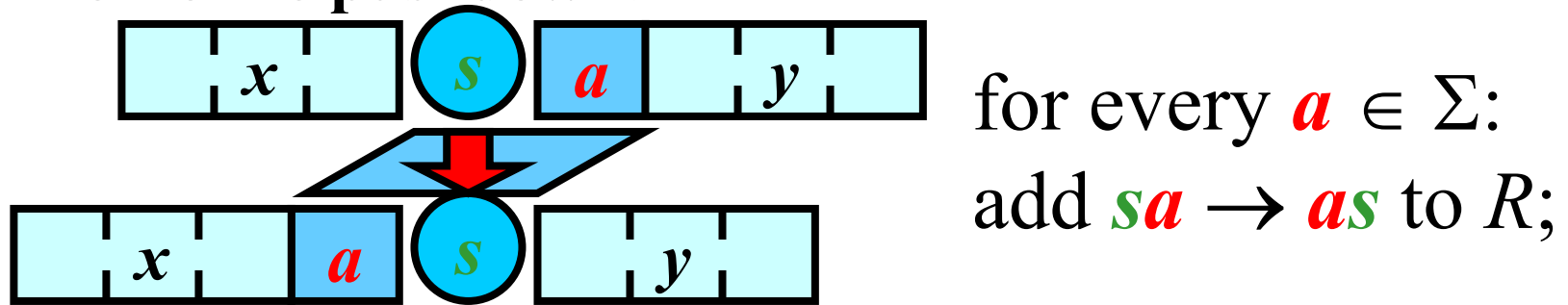


From the input string towards S

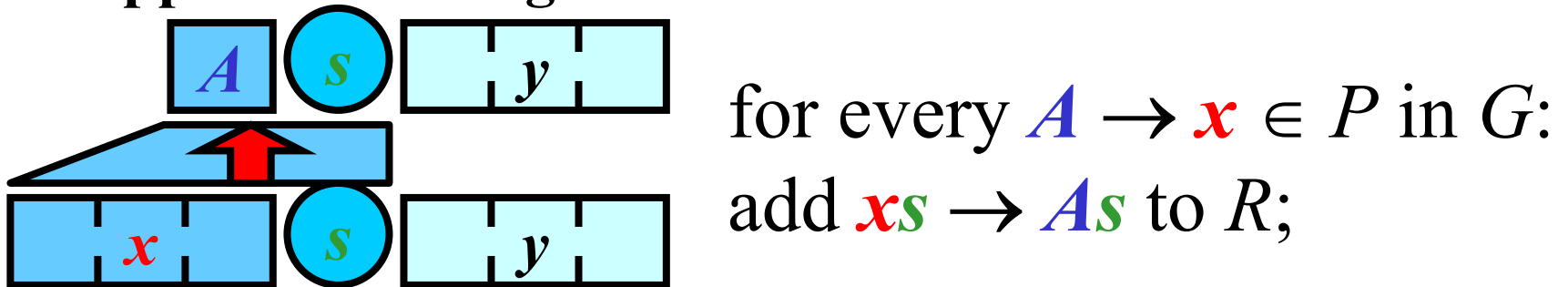
EPDAs as Models of Bottom-Up Parsers 1/2

Gist: An EPDA M underlies a bottom-up parser

1) M contains *shift* rules that copy the input symbols onto the pushdown:



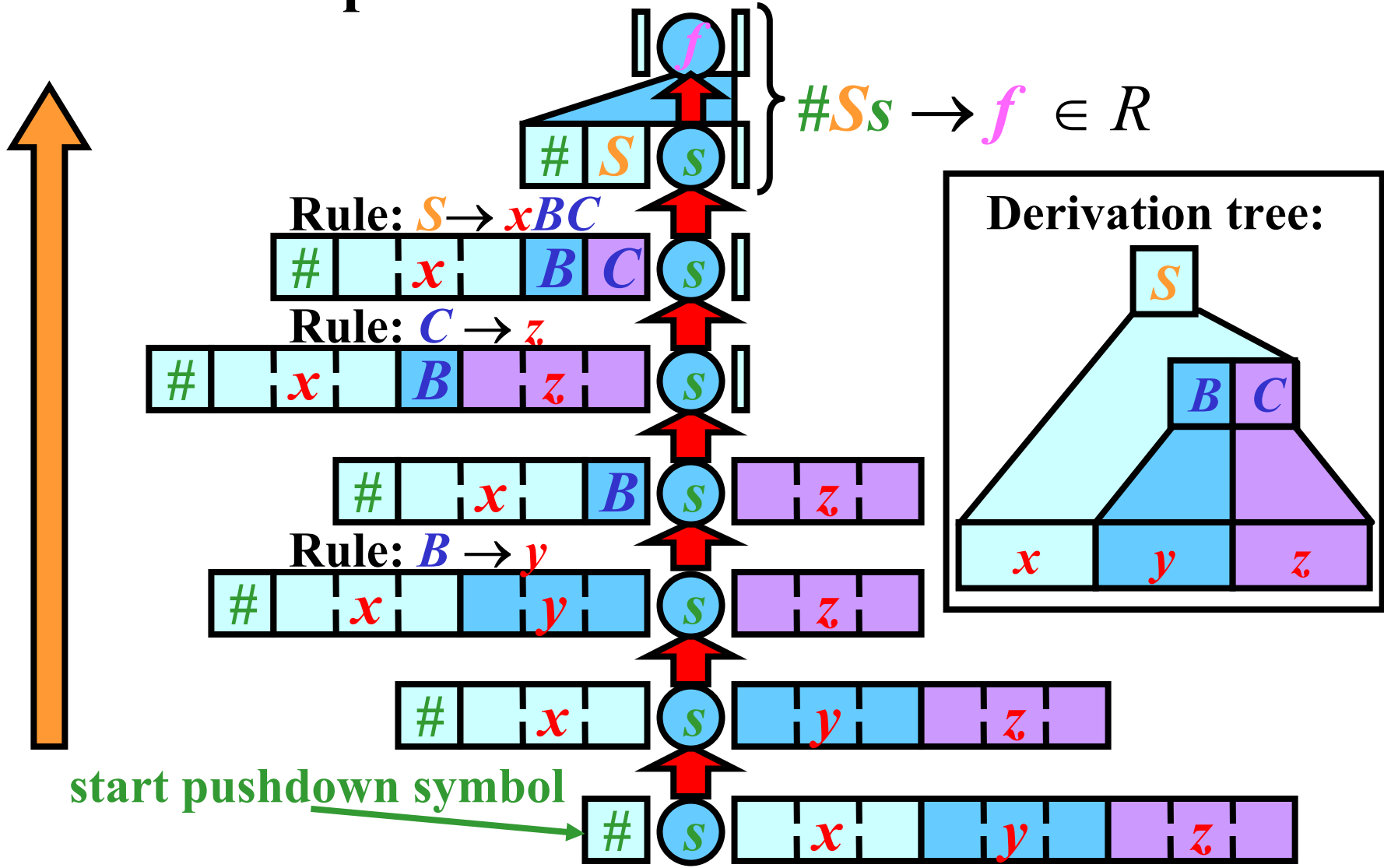
2) M contains *reduction* rules that simulate the application of a grammatical rule in reverse:



3) M also contains the rule $\#Ss \rightarrow f$ that takes M to a final state f

EPDAs as Models of Bottom-Up Parsers 2/2

Bottom-up construction of a derivation tree:



Algorithm: From CFG to EPDA

- **Input:** CFG $G = (N, T, P, S)$
 - **Output:** EPDA $M = (Q, \Sigma, \Gamma, R, s, \#, F)$; $L(G) = L(M)_f$
-
- **Method:**
 - $Q := \{s, f\};$
 - $\Sigma := T;$
 - $\Gamma := N \cup T \cup \{\#\};$
 - Construction of R :
 - for every $a \in \Sigma$, add $sa \rightarrow as$ to R ;
 - for every $A \rightarrow x \in P$, add $xs \rightarrow As$ to R ;
 - add $\#Ss \rightarrow f$ to R ;
 - $F := \{f\};$

From CFG to EPDA: Example 1/2

- $G = (N, T, P, \mathbf{S})$, where:

$$N = \{\mathbf{S}\}, T = \{(\text{, } \text{)}\}, P = \{\mathbf{S} \rightarrow (\mathbf{S}), \mathbf{S} \rightarrow (\text{ })\}$$

Objective: An EPDA M such that $L(G) = L(M)_f$

$M = (Q, \Sigma, \Gamma, R, \mathbf{s}, \#, F)$ where:

$$Q = \{\mathbf{s}, \mathbf{f}\}; \Sigma = T = \{(\text{, } \text{)}\}; \Gamma = N \cup T \cup \{\#\} = \{\mathbf{S}, (\text{, } \text{)}, \#\}$$

$$R = \underbrace{\left\{ \begin{array}{l} \text{"("} \in T \quad \text{"("} \in T \\ \downarrow \quad \downarrow \\ \mathbf{s}(\text{ } \rightarrow (\mathbf{s}, \mathbf{s}) \rightarrow \text{)s} \end{array} \right\}}_{\text{shift rules}}, \underbrace{\left\{ \begin{array}{l} \mathbf{S} \rightarrow (\mathbf{S}) \in P \quad \mathbf{S} \rightarrow (\text{ }) \in P \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ (\mathbf{S})\mathbf{s} \rightarrow \mathbf{Ss}, \quad (\text{ })\mathbf{s} \rightarrow \mathbf{Ss}, \quad \#\mathbf{Ss} \rightarrow \mathbf{f} \end{array} \right\}}_{\text{reduction rules}} \}$$

$$F = \{\mathbf{f}\}$$

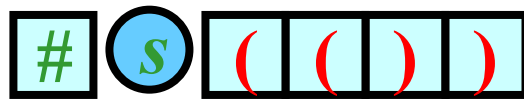
From CFG to EPDA: Example 2/2

$M = (Q, \Sigma, \Gamma, R, s, \#, F)$, where:

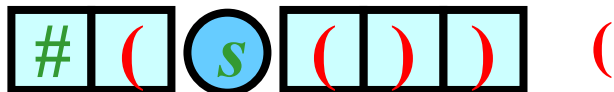
$Q = \{s, f\}$, $\Sigma = T = \{ (,) \}$, $\Gamma = \{ (,), S, \# \}$, $F = \{f\}$

$R = \{s(\rightarrow (s, s) \rightarrow)s, (S)s \rightarrow Ss, ()s \rightarrow Ss, \#Ss \rightarrow f\}$

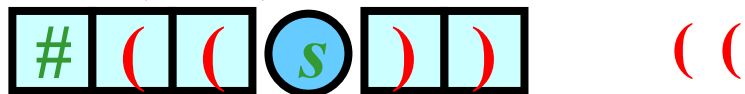
Question: $(()) \in L(M)_f?$



Rule: $s(\rightarrow (s$



Rule: $s(\rightarrow (s$



Rule: $s) \rightarrow)s$



Rule: $()s \rightarrow S$



Rule: $s) \rightarrow)s$



Rule: $(S) \rightarrow S$



Rule: $\#Ss \rightarrow f$

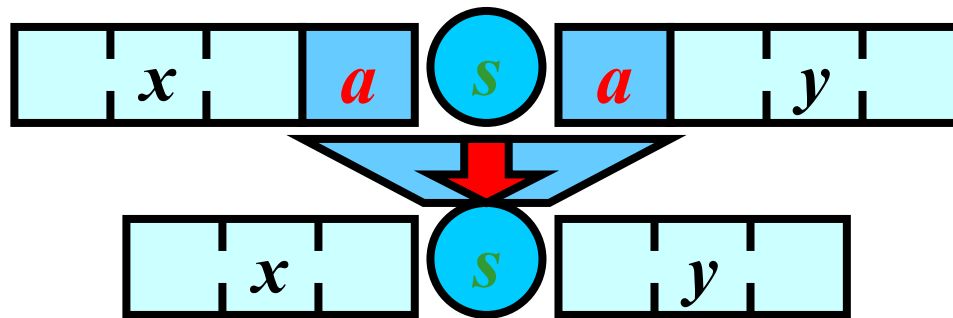


Answer: YES

PDAs as Models of Top-Down Parsers 1/2

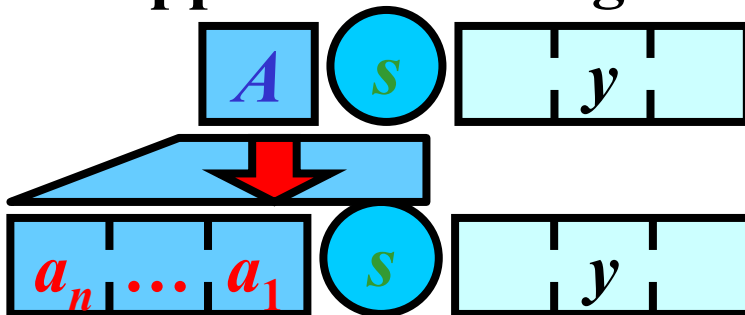
Gist: An PDA M underlies a top-down parser

1) M contains *popping* rules that pops the top symbol from the pushdown and reads the input symbol if both coincide:



for every $a \in \Sigma$:
add $asa \rightarrow s$ to R ;

2) M contains *expansion* rules that simulate the application of a grammatical rule:

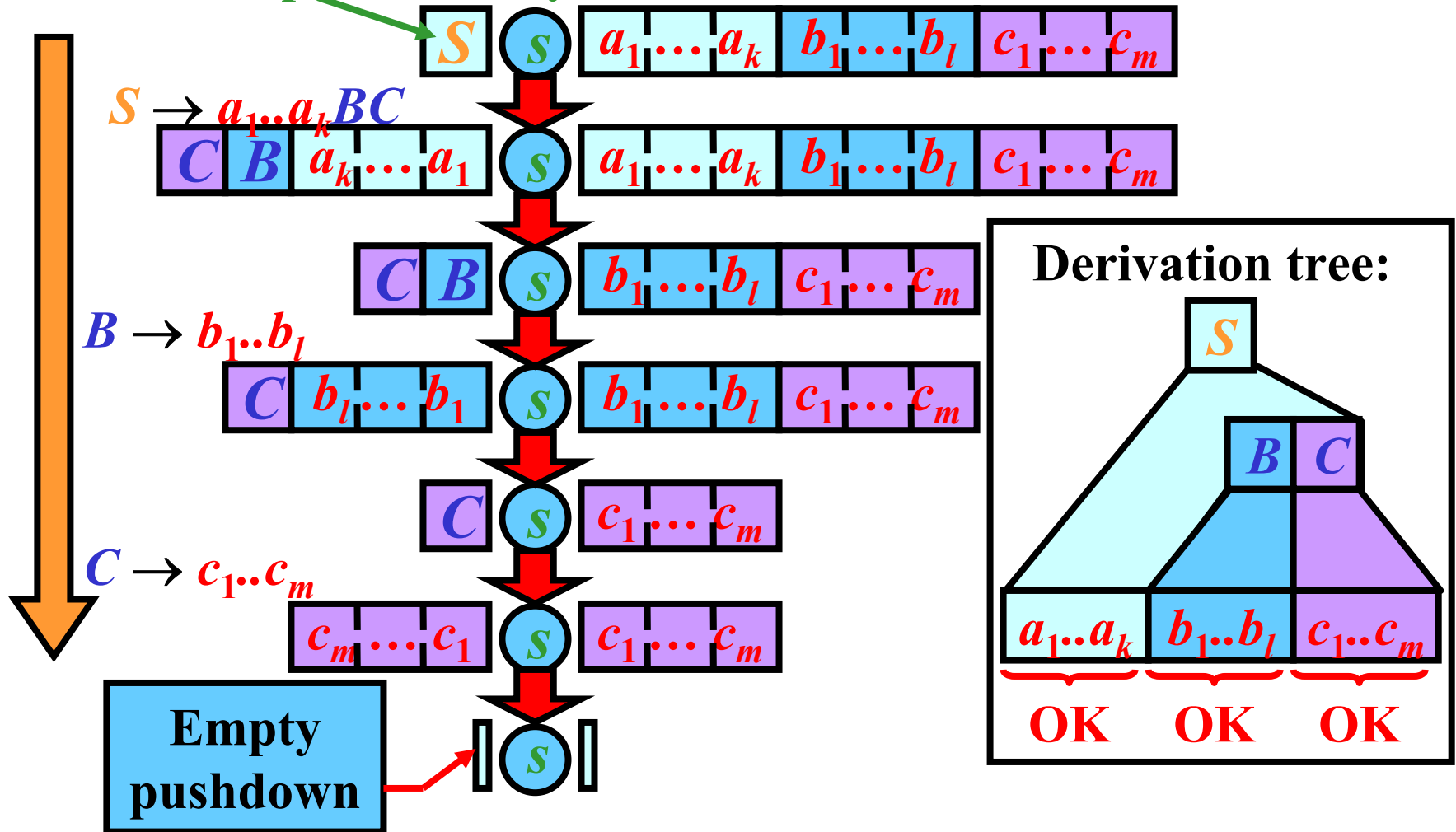


for every $A \rightarrow a_1 \dots a_n \in P$ in G ,
add $As \rightarrow \underbrace{a_n \dots a_1}_s$ to R ;
 $= \text{reversal}(a_1 \dots a_n)$

PDAs as Models of Top-Down Parsers 2/2

Top-down construction of a derivation tree:

start pushdown symbol



Algorithm: From CFG to PDA

- **Input:** CFG $G = (N, T, P, \mathbf{S})$
 - **Output:** PDA $M = (Q, \Sigma, \Gamma, R, \mathbf{s}, \mathbf{S}, F)$; $L(G) = L(M)_\varepsilon$
-
- **Method:**
 - $Q := \{\mathbf{s}\};$
 - $\Sigma := T;$
 - $\Gamma := N \cup T;$
 - Construction of R :
 - for every $\mathbf{a} \in \Sigma$, add $\mathbf{a}\mathbf{s}\mathbf{a} \rightarrow \mathbf{s}$ to R ;
 - for every $\mathbf{A} \rightarrow \mathbf{x} \in P$, add $\mathbf{A}\mathbf{s} \rightarrow \mathbf{y}\mathbf{s}$ to R ,
where $\mathbf{y} = \text{reversal}(\mathbf{x})$;
 - $F := \emptyset;$

From CFG to PDA: Example 1/2

- $G = (N, T, P, \mathbf{S})$, where:

$$N = \{\mathbf{S}\}, T = \{(\, , \,)\}, P = \{\mathbf{S} \rightarrow (\mathbf{S}), \mathbf{S} \rightarrow (\,)\}$$

Objective: An PDA M such that $L(G) = L(M)_\varepsilon$

$M = (Q, \Sigma, \Gamma, R, \mathbf{s}, \mathbf{S}, F)$ where:

$$Q = \{\mathbf{s}\}; \quad \Sigma = T = \{(\, , \,)\}; \quad \Gamma = N \cup T = \{\mathbf{S}, (\, , \,)\}$$

$$\text{"("} \in T \quad \text{"("} \in T \quad \mathbf{S} \rightarrow (\mathbf{S}) \in P \quad \mathbf{S} \rightarrow (\,) \in P$$

$$R = \underbrace{\{(\mathbf{s}(\rightarrow \mathbf{s}, \,)\mathbf{s}) \rightarrow \mathbf{s}, \quad \mathbf{S}\mathbf{s} \rightarrow \,)\mathbf{S}(\mathbf{s}, \quad \mathbf{S}\mathbf{s} \rightarrow \,)\mathbf{s}\}}_{\text{popping rules}} \quad \underbrace{\{ \mathbf{S}\mathbf{s} \rightarrow \,)\mathbf{S}(\mathbf{s}, \quad \mathbf{S}\mathbf{s} \rightarrow \,)\mathbf{s} \}}_{\text{expansion rules}}$$

$$F = \emptyset$$

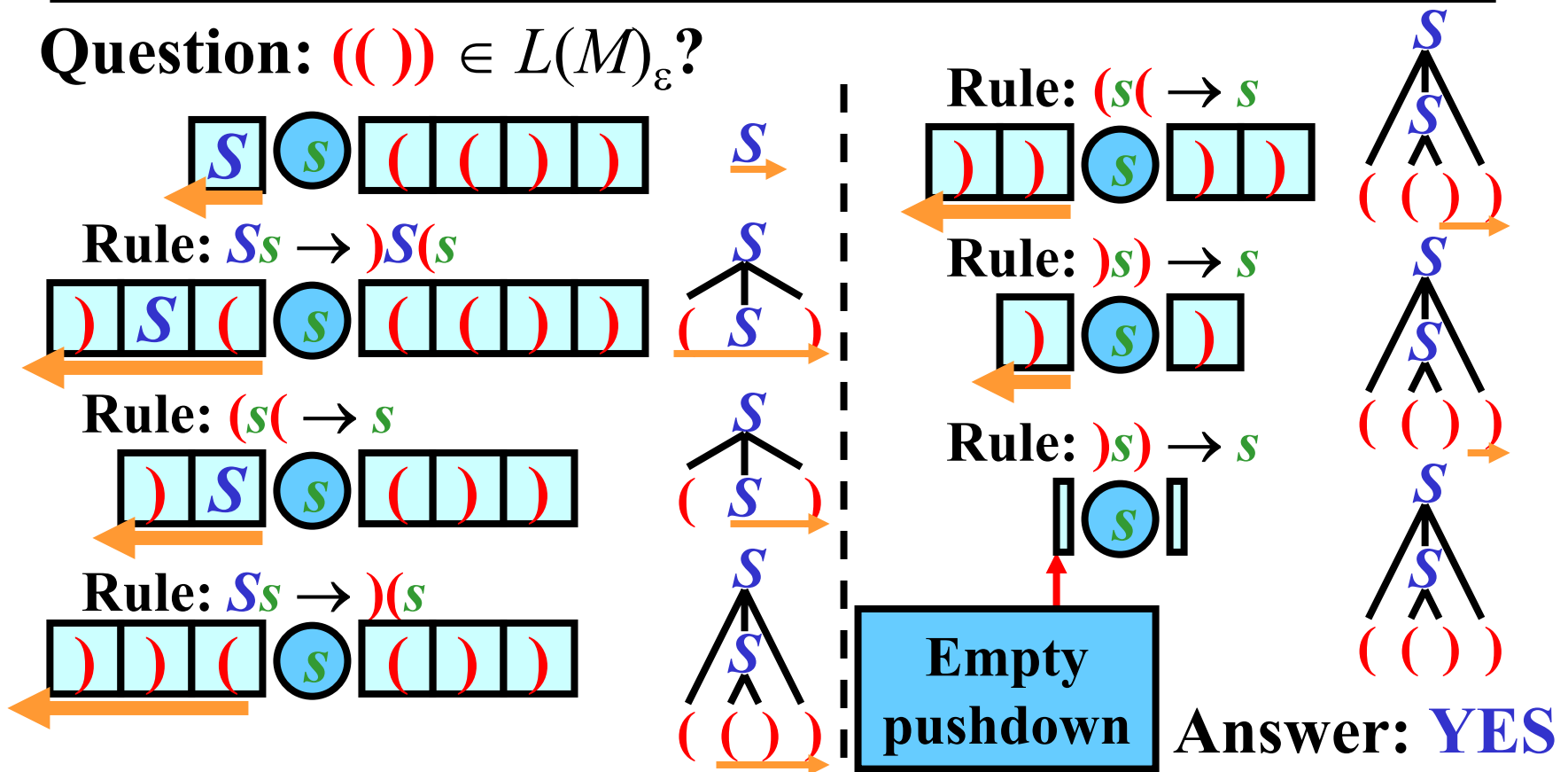
From CFG to PDA: Example 2/2

$M = (Q, \Sigma, \Gamma, R, s, S, F)$, where:

$Q = \{s\}$, $\Sigma = T = \{ (,) \}$, $\Gamma = \{ (,), S \}$, $F = \emptyset$

$P = \{ (s(\rightarrow s,)s) \rightarrow s, Ss \rightarrow)S(s, Ss \rightarrow)(s) \}$

Question: $(()) \in L(M)_\varepsilon$?



Models for Context-free Languages

Theorem: For every CFG G , there is an PDA M such that $L(G) = L(M)_\varepsilon$.

Proof: See the previous algorithm.

Theorem: For every PDA M , there is a CFG G such that $L(M)_\varepsilon = L(G)$.

Proof: See page 486 in [Meduna: Automata and Languages]

Conclusion: The fundamental models for context-free languages are

1) **Context-free grammars** 2) **Pushdown automata**