

## 5. Parsing

### **parser**

*language recognizer*

*check if syntactically correct*

*text transformer*

*produce internal representation*

*parse tree, rule string(parse)*

### **pushdown automaton**

*homomorphism: sentence  $\rightarrow$  left / right parse*

### 5.1 Pushdown Automaton

$M = (Q \cup \Sigma \cup \{\$, \mid\}, \Gamma)$  is a rewriting system.

$M = (Q, \Sigma, \Gamma, \gamma_s, F, \$, \mid)$  is a **pushdown automaton**

(1) **stack alphabet**  $Q$ ,

(2) **input alphabet**  $\Sigma$  where  $Q \cap \Sigma \neq \emptyset$ ,

(4) **initial stack content**  $\gamma_s \in Q^*$ ,

(5) **set of final stack contents**  $F \subseteq Q^*$ ,

(6) **end marker**  $\$ \notin Q \cup \Sigma$ ,

(7) **delimiter**  $\mid \notin Q \cup \Sigma$ , and

(3) **set of actions**  $\Gamma$ ,

*action*  $\xi \in \Gamma$ : a pair of configuration strings

$$\xi = \alpha \mid xy \rightarrow \beta \mid y \in \Gamma,$$

$$\alpha, \beta \in Q^*, x, y \in \Sigma^*.$$

**configuration**(*instantaneous description*) of  $M$

$\$ \gamma \mid w \$$ ,

$\gamma \in Q^*$  *stack string*

$\$ \gamma : 1$  *stack topmost symbol*

$w \in \Sigma^*$  *remaining input string*

$1 : w$  *current input symbol*

Let  $\$ \delta \alpha \mid xyz \$$  be a configuration. Then

$\$ \delta \alpha \mid xyz \$ \Rightarrow \$ \delta \beta \mid yz \$$ , iff  $\alpha \mid xy \rightarrow \beta \mid y \in \Gamma$ .

$\$ \gamma_s \mid w \$$  *initial configuration for  $w$*

$\$ \gamma \mid \$$ ,  $\gamma \in F$  *final configuration*

*no applicable action* *error configuration*

A **computation**(or **process**) of  $M$  on  $w$

any derivation in  $M$  from initial configuration for  $w$

**accepting computation**

a computation ends with final configuration.

**nonaccepting computation**

a computation ends with error configuration.

$M$  **accepts**  $w$ , if it has an accepting computation on  $w$ .

$M$  **halts correctly** on  $w$ , if it accepts  $w$ .

$M$  **halts incorrectly** on  $w$ ,

if every computation on  $w$  is nonaccepting.

$M$  **loops forever** on  $w$ , otherwise.

i.e., if it has an arbitrary long computation on  $w$   
but no accepting computation.

*language accepted(recognized, or described) by  $M$*

$$L(M) = \{w \in \Sigma^* \mid \$\gamma_s \mid w\$ \Rightarrow^* \$\gamma \mid \$, \gamma \in F\}.$$

A pushdown automaton  $M$  is **ambiguous**,  
if **two** accepting computations on some sentence  
 $M$  is **unambiguous**, otherwise.

$M$  is **nondeterministic**, if it has some computation  
to which **two** actions are applicable.

$M$  is **deterministic**, if it is not nondeterministic.

**Fact 5.1**  $M$  is nondeterministic if and only if it has  
distinct actions

$$\alpha \mid x \rightarrow \alpha' \mid x', \beta \mid y \rightarrow \beta' \mid y' \in \Gamma,$$

$\alpha:k = \beta$ , or  $\alpha = \beta:k$ ; and  $l:x = y$  or  $x = l:y$ .  
one of  $\alpha, \beta$  is **suffix** of other, and  
one of  $x, y$  is **prefix** of other,  
provided that  $M$  is **useful**(?).

**Fact 5.2** Any **deterministic pushdown automaton** is  
**unambiguous** provided that no action is applicable  
to any of the accepting configurations.

**dpda**  $\Rightarrow$  **unambiguous**

no action on accepting configuration

**Fact 5.3** Any fa  $M$  can be transformed in time  $O(|M|)$  into an equivalent pda  $M'$  which is **unambiguous** (respectively **deterministic**), iff  $M$  is. Also  $M'$  has **bounded stack** (stack contents of every configuration in any computation of  $M'$  consists of single symbol only).

**Proof**  $\Gamma = \{q \mid x \rightarrow p \mid / qx \rightarrow p\}$ .

*pda: at least as descriptive and succinct as fa*

**Proposition 5.4** There exist a constant  $c > 0$  and an infinite sequence of regular languages  $L_1, L_2, \dots$  such that each  $L_n$  is accepted by some deterministic pda of size  $O(n^3)$  but any fa accepting  $L_n$  must have size at least  $O(2^n)$ .

*dpda: exponentially more succinct than nfa*

A pda is **normal-form** if its action are of forms:  
 $\alpha \mid x \rightarrow \beta \mid$  where  $|\alpha| \leq 2$ ,  $|x| \leq 1$ .

**Proposition 5.5** Any pda  $M$  can be transformed into an equivalent **normal-form** pda  $M'$ . Moreover  $M'$  is **unambiguous**(respectively **deterministic**), iff  $M$  is.

Let  $G = (N, \Sigma, P, S)$  be a context-free grammar.

The **predictive machine** for  $G$  is a pda  $M_P^G(N \cup \Sigma, \Sigma, \Gamma_P, S, \{\varepsilon\}, \$, \mid\}$ , where  $\Gamma_P$  are of form:

(pa)  $A \mid \rightarrow \omega^R \mid$   $A \in N$ , "**produce** by  $A \rightarrow \omega \in P$ ",

(sa)  $a \mid a \rightarrow \mid$   $a \in \Sigma$ , "**shift**  $a \in \Sigma$ ".

stack content is the **prediction** of remaining input

$\$ \gamma \mid w \$$   $\gamma^R \Rightarrow^* w$

$A \mid \rightarrow \omega^R \mid$  **guessing**  $A \rightarrow \omega \in P$

$a \mid a \rightarrow \mid$  **verifying**  $a \in \Sigma$

initial configuration  $\$ S \mid w \$$

initial prediction  $S$ :  $S$  derives  $w$

final configuration  $\$ \mid \$$

final prediction  $\varepsilon$ : remaining input is empty

**Lemma 5.6** For any  $G$ , the language accepted by the **predictive machine**  $M$  for  $G$  is the language generated by  $G$ . Moreover for any sentence, there is a **bijec-tive** correspondence between **leftmost** derivation in  $G$  and accepting computation of predictive machine  $M$  on  $w$ .

**Theorem 5.7** Any grammar  $G$  can be transformed in time  $O(|G|)$  into an equivalent **normal-form** pda  $M$ . Moreover  $M$  is **unambiguous**, iff  $G$  is.

**Proposition 5.8** Any normal-form pda  $M$  can be transformed in time  $O(|M| \cdot |V|^3)$  into an equivalent **context-free** grammar  $G$ . Moreover  $G$  is **unambigu-ous**, iff  $M$  is.

**Theorem 5.9** Any language is **context-free** iff it is the language accepted by some pda.

$\exists$  **inherently ambiguous context-free languages**

$\therefore \forall$  **algorithm: pda  $\rightarrow$  unambiguous pda**

A language is **deterministic**, if it is accepted by some deterministic machine.

$L_{match}$  is deterministic. (p157)

$L_{pal} = \{w \in \{0, 1\}^* \mid w^R = w\}$  **palindromes**

$G_{pal} = (\{S\}, \{0, 1\}, \{S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}, S)$ .

guessing center  $S \rightarrow \varepsilon$  even length

$S \rightarrow 0 \mid 1$  odd length

no pda can guess center of palindrome deterministically

**Proposition 5.10**  $L_{pal}$  is not deterministic, but

$L_{cpal} = \{wcw^R \mid w \in \{0, 1\}^*\}$  is deterministic

**palindromes with center marker  $c$**

### **Theorem 5.11**

#### **(Characterization of Context-free Languages)**

The following statements are logically equivalent.

- (1)  $L$  is the language generated by some **context-free** grammar.
- (2)  $L$  is the language generated by some **canonical two-form** grammar.
- (3)  $L$  is the language generated by some **Chomsky normal-form** grammar.
- (4)  $L$  is the language accepted by some **pda**.
- (5)  $L$  is the language accepted by some **normal-form pda**.

Moreover, the descriptions of  $L$  can be transformed into an equivalent descriptions to the other classes.

Let  $M = (Q, \Sigma, \Gamma, \gamma_s, F, \$, |)$ . Then

$Time_M(w) = \min\{Time_M(\$ \gamma_s | w \$, \gamma | \$) / \gamma \in F\}$ .

$Space_M(w) = \min\{Space_M(\$ \gamma_s | w \$, \gamma | \$) / \gamma \in F\}$ .

where  $Time_M(\phi_1, \phi_2) / Space_M(\phi_1, \phi_2)$  denote the time/space complexity for deriving  $\phi_2$  from  $\phi_1$  in  $M$ .

**time/space complexity of accepting  $w$  in  $M$ .**

$Time_M(w) / Space_M(w)$

$M$  accepts  $w$  in time  $t$ , if  $t \geq Time_M(w)$ .

$M$  accepts  $w$  in space  $s$ , if  $s \geq Space_M(w)$ .

$M$  runs in time  $T(n)$  (respectively in space  $S(n)$ ), if  $M$  accepts every sentence of length  $n$  in time  $T(n)$  (respectively in space  $S(n)$ ).

For example, for  $M_{match}$

$Time_{M_{match}}(0^n 1^n) = 3n + 1$

$Space_{M_{match}}(0^n 1^n) = 2n + 4$

$M_{match}$  runs in time  $3n + 1$  and in space  $2n + 4$ .



## 5.2 Left and Right Parsers

*left parse of  $w$  in  $G$ :  $\pi \in P^*$ .*

$$S \xrightarrow[lm]{\pi} w \text{ in } G$$

*right parse of  $w$  in  $G$ :  $\pi^R \in P^*$ .*

$$S \xrightarrow[rm]{\pi} w \text{ in } G$$

*left(right) parser for a grammar  $G$*

*A (possibly nondeterministic) RAM program  
recognizes the sentence in  $L(G)$   
produces at least one left(right) parse  
for each sentence*

*left parse      leftmost derivation*

*top-down manner*

*right parse    reversal of rightmost derivation*

*bottom-up manner*

*$M$  is a **pushdown transducer** with **output alphabet**  $\Delta$   
and **output effect**  $\tau$ , written  $(M, \tau)$ , if  $M$  is a pda and  
 $\tau$  is a homomorphism from  $\Gamma^*$  to  $\Delta^*$  where  $\Gamma$  is a set  
of action of  $M$ .*

$M(Q, \Sigma, \Gamma, \gamma_s, F)$       pda

$M(Q, \Sigma, \Gamma, \Delta, \tau, \gamma_s, F)$       pdt

$(M, \tau)$

Let  $\theta \in \Gamma^*$  be an action string of sentence  $w$ . Then pushdown transducer  $(M, \tau)$  **produce output**  $\sigma$  for  $w$ , if  $\tau(\theta) = \sigma \in \Delta^*$ .

A pdt  $M = (Q, \Sigma, \Gamma, \Delta, \tau, \gamma_s, F)$  is a **left/right parser** for a grammar  $G = (N, \Sigma', P, S)$ , if

- (1)  $\Sigma = \Sigma'$ ,
- (2)  $L(M) = L(G)$ ,
- (3)  $\Delta = P$ ,
- (4)  $\forall \pi \in P^*$ ,  $\pi$ : left/right parse of  $G$ .

Let  $G = (N, \Sigma, P, S)$  be a context-free grammar.

$M = (Q, \Sigma, \Gamma, P, \tau, \gamma_s, F)$  is a left/right parser for  $G$ .

or  $(M, \tau)$  is a left/right parser for  $G$  in short.

**produce-shift parser**  $(M(V, \Sigma, \Gamma, S, \{\varepsilon\}, \$, |), \tau)$

(pa)  $\tau(A | \rightarrow \omega^R |) = A \rightarrow \omega \in P$ .

(sa)  $\tau(a | a \rightarrow |) = \varepsilon, a \in \Sigma$ .

### **Proposition**

Produce-shift parser is indeed a left parser.

**Lemma 5.12, 5.13**

produce-shift parser  $\Rightarrow$  linear left parser

**Lemma 5.14, 5.15**

produce-shift parser  $\Leftarrow$  linear left parser

**Theorem 5.16**

produce-shift parser  $\Leftrightarrow$  linear left parser

**Lemma 5.12** Let  $G = (N, \Sigma, P, S)$  be a grammar and  $M = (V, \Sigma, \Gamma, P, \tau, S, \{\varepsilon\})$  be a **produce-shift** parser.

If  $\$ \gamma \mid xy \$ \xrightarrow{\theta} \$ \delta \mid y \$$ ,  $\theta \in \Gamma^*$  in  $M$ , then

$$\gamma^R \xrightarrow[lm]{\tau(\theta)} x\delta^R \text{ in } G, \text{ and } |\theta| = |\tau(\theta)| + |x|.$$

**Proof** Induction on the length of action string  $\theta$ .

i)  $\theta = \varepsilon$ .  $x = \varepsilon$ ,  $\gamma = \delta$ , and  $\tau(\varepsilon) = \varepsilon$ .

ii)  $\theta = r\theta'$ .

ii.1)  $r = A \mid \rightarrow \omega^R \mid \in \Gamma$

$$\begin{aligned} \$ \gamma \mid xy \$ &= \$ \gamma' A \mid xy \$ \xrightarrow{r} \$ \gamma' \omega^R \mid xy \$ \xrightarrow{\theta'} \$ \delta \mid y \$ \\ &(\gamma' \omega^R)^R \xrightarrow[lm]{\tau(\theta')} x\delta^R \text{ in } G, |\theta'| = |\tau(\theta')| + |x| \text{ IH.} \end{aligned}$$

$$\gamma^R = (\gamma' A)^R = A \gamma'^R \xrightarrow[lm]{A \rightarrow \omega} \omega \gamma'^R = (\gamma' \omega^R)^R.$$

$$\therefore \gamma^R \xrightarrow[lm]{\tau(r) \cdot \tau(\theta')} (= \xrightarrow[lm]{\tau(\theta)}) x\delta^R \text{ in } G, \text{ and}$$

$$|\theta| = 1 + |\theta'| = 1 + |\tau(\theta')| + |x| = |\tau(\theta)| + |x|.$$

ii.2)  $r = a \mid a \rightarrow \mid \in \Gamma$

$$\begin{aligned} \$ \gamma \mid xy \$ &= \$ \gamma' a \mid ax'y \$ \xrightarrow{r} \$ \gamma' \mid x'y \$ \xrightarrow{\theta'} \$ \delta \mid y \$ \\ &\gamma'^R \xrightarrow[lm]{\tau(\theta')} x'\delta^R \text{ in } G, |\theta'| = |\tau(\theta')| + |x'| \text{ IH.} \end{aligned}$$

$$\gamma^R = (\gamma' a)^R = a \gamma'^R.$$

$$\therefore \gamma^R \xrightarrow[lm]{\tau(r) \cdot \tau(\theta')} (= \xrightarrow[lm]{\tau(\theta')}) x\delta^R \text{ in } G, \text{ and}$$

$$|\theta| = 1 + |\theta'| = |\tau(\theta')| + 1 + |x'| = |\tau(\theta)| + |x|$$

**Lemma 5.13** Let  $M$  be a *produce-shift* parser for  $G$ .

- (1)  $L(M) \subseteq L(G)$ ,
- (2)  $\forall \theta$ : actions in  $M$ ,  $\tau(\theta)$  is a *left parse* of  $w$ ,
- (3)  $\text{Time}_G(w) \leq \text{Time}_M(w) - |w|$ .

**Lemma 5.14** Let  $G = (N, \Sigma, P, S)$  be a grammar and  $M = (V, \Sigma, \Gamma, P, \tau, S, \{\varepsilon\})$  be a *produce-shift* parser.

If  $\gamma^R \xrightarrow[lm]{\pi} x\delta^R$  in  $G$ ,  $\delta^R = \varepsilon$  or  $1:\delta^R \in N$ , then

$$\begin{aligned} \$\gamma \mid xy\$ \xrightarrow{\theta} \$\delta \mid y\$, \theta \in \Gamma^* \text{ and} \\ \tau(\theta) = \pi, |\theta| = |\pi| + |x|. \end{aligned}$$

**Proof** Induction on the length of rule string  $\pi$ .

i)  $\pi = \varepsilon$ .  $\gamma^R = x\delta^R$ .

$$\begin{aligned} \$\gamma \mid xy\$ = \$\delta x^R \mid xy\$ \xrightarrow{\theta} \$\delta \mid y\$, \text{ and} \\ |\theta| = |x|, \tau(\theta) = \pi = \varepsilon. \end{aligned}$$

ii)  $\pi = \pi' \cdot A \rightarrow \omega$ ,  $\pi' \in P^*$ ,  $\exists \theta'$ ,  $\tau(\theta') = \pi'$ .

$$\gamma^R \xrightarrow[lm]{\pi'} x'\delta'^R = x'A\delta'' \xrightarrow[lm]{A \rightarrow \omega} x'\omega\delta'' = x'z\delta^R = x\delta^R,$$

where  $|\theta'| = |\pi'| + |x'|$ ,  $\delta'^R = A\delta''$ ,  $\omega\delta'' = z\delta^R$ ,  $x'z = x$

$$\begin{aligned} \therefore \$\gamma \mid xy\$ = \$\gamma \mid x'zy\$ \xrightarrow{\theta'} \$\delta' \mid zy\$ = \$\delta''^R A \mid zy\$ \\ \xrightarrow{r} \$\delta''^R \omega^R \mid zy\$ = \$(\omega\delta'')^R \mid zy\$ = \$\delta z^R \mid zy\$ \\ \xrightarrow{\theta''} \$\delta \mid y\$, \text{ where } |\theta''| = |z|, \tau(\theta'') = \varepsilon. \end{aligned}$$

$$\begin{aligned} \therefore |\theta| = |\theta'| + 1 + |\theta''| = |\pi'| + |x'| + 1 + |z| = |\pi| + |x| \\ \tau(\theta) = \pi' \cdot A \rightarrow \omega \cdot \varepsilon = \pi' \cdot A \rightarrow \omega = \pi. \end{aligned}$$

**Lemma 5.15** *Let  $M$  be a produce-shift parser for  $G$ .*

- (1)  $L(G) \subseteq L(M)$ ,
- (2)  $\forall \pi$ : *left parse of  $w$  in  $G$ ,  $\tau(\theta) = \pi$  in  $M$ ,*
- (3)  $\text{Time}_M(w) \leq \text{Time}_G(w) + |w|$ .

**Theorem 5.16** *Let  $M$  be a produce-shift parser  $M$  for  $G$ . Then*

- (1)  *$M$  is a left parser for  $G$ .*
- (2)  $\forall w \in L(G)$ ,  *$M$  produces all left parses of  $w$ .*
- (3)  $\text{Time}_M(w) = \text{Time}_G(w) + |w|$ .

*produce-shift parser runs in time linear in the length of the sentence.*

Let  $G = (N, \Sigma, P, S)$  be a context-free grammar.

The **shift-reduce parser**  $(M, \tau)$  for  $G$  is with  $M(V, \Sigma, \Gamma, \varepsilon, \{S\}, \$, |)$ ,

$$\begin{array}{ll} (ra) & \omega | \rightarrow A | \in \Gamma \quad \text{"reduce by } A \rightarrow \omega \in P\text{"} \\ (sa) & | a \rightarrow a | \in \Gamma \quad \text{"shift } a \in \Sigma\text{"} \end{array}$$

$$\begin{array}{l} \tau(\omega | \rightarrow A |) = A \rightarrow \omega, \\ \tau(| a \rightarrow a |) = \varepsilon. \end{array}$$

**Theorem 5.21** Let  $M$  be a shift-reduce parser  $M$  for  $G$ . Then

- (1)  $M$  is a **right parser** for  $G$ .
- (2)  $\forall w \in L(G)$ ,  $M$  produces **all right parses** of  $w$ .
- (3)  $\text{Time}_M(w) = \text{Time}_G(w) + |w|$ .

**shift-reduce parser** runs in time linear in the length of the sentence.

**Lemma 5.17, 5.18**

shift-reduce parser  $\Rightarrow$  linear right parser

**Lemma 5.19, 5.20**

shift-reduce parser  $\Leftarrow$  linear right parser

**Theorem 5.21**

shift-reduce parser  $\Leftrightarrow$  linear right parser

**Lemma 5.17** Let  $G = (N, \Sigma, P, S)$  be a grammar and  $M = (V, \Sigma, \Gamma, P, \tau, \{\varepsilon\}, S)$  be a **shift-reduce** parser. If  $\$ \gamma \mid xy \$ \xrightarrow{\theta} \$ \delta \mid y \$$ ,  $\theta \in \Gamma^*$  in  $M$ , then

$$\delta \xrightarrow[rm]{\tau(\theta)^R} \gamma x \text{ in } G, \text{ and } |\theta| = |\tau(\theta)| + |x|.$$

**Proof** Induction on the length of action string  $\theta \in \Gamma^*$ .

i)  $\theta = \varepsilon$ .  $x = \varepsilon$ ,  $\gamma = \delta$ , and  $\tau(\varepsilon) = \varepsilon$ .

ii)  $\theta = r\theta'$ .

ii.1)  $r = \omega \mid \rightarrow A \mid \in \Gamma$

$$\begin{aligned} \$ \gamma \mid xy \$ &= \$ \gamma'' \omega \mid xy \$ \xrightarrow{r} \$ \gamma'' A \mid xy \$ = \$ \gamma' \mid xy \$ \\ &\xrightarrow{\theta'} \$ \delta \mid y \$, \text{ and } |\theta'| = |\tau(\theta')| + |x|. \end{aligned}$$

$$\delta \xrightarrow[rm]{\tau(\theta')^R} \gamma' x = \gamma'' A x \xrightarrow[rm]{A \rightarrow \omega} \gamma'' \omega x = \gamma x.$$

$$\therefore \delta \xrightarrow[rm]{(\tau(r\theta'))^R} \gamma x \text{ in } G, \text{ and}$$

$$|\theta| = 1 + |\theta'| = 1 + |\tau(\theta')| + |x| = |\tau(\theta)| + |x|.$$

ii.2)  $r = \mid a \rightarrow a \mid \in \Gamma$

$$\begin{aligned} \$ \gamma \mid xy \$ &= \$ \gamma \mid ax'y \$ \xrightarrow{r} \$ \gamma a \mid x'y \$ = \$ \gamma' \mid x'y \$ \\ &\xrightarrow{\theta'} \$ \delta \mid y \$, \text{ and } |\theta'| = |\tau(\theta')| + |x'|. \end{aligned}$$

$$\delta \xrightarrow[rm]{\tau(\theta')^R} \gamma' x' = \gamma a x' = \gamma x$$

$$\therefore \delta \xrightarrow[rm]{(\varepsilon \cdot \tau(\theta'))^R} \gamma x \text{ in } G, \text{ and}$$

$$|\theta| = 1 + |\theta'| = |\tau(\theta')| + 1 + |x'| = |\tau(\theta)| + |x|$$

**Lemma 5.18** Let  $M$  be a *shift-reduce* parser for  $G$ .

$$(1) L(M) \subseteq L(G),$$

(2)  $\forall \theta$ : actions in  $M$ ,  $\tau(\theta)$  is a **right parse** of  $w$ ,

$$(3) \text{Time}_G(w) \leq \text{Time}_M(w) - |w|.$$

**Lemma 5.19** Let  $M$  be a *shift-reduce* parser for  $G$ . If

$\delta \xrightarrow[rm]{\pi^R} \gamma x$  in  $G$ ,  $\gamma = \varepsilon$  or  $\gamma:l \in N$ , then

$$\begin{aligned} \$\gamma \mid xy\$ \xrightarrow{\theta} \$\delta \mid y\$, \theta \in \Gamma^* \text{ and} \\ \tau(\theta) = \pi, |\theta| = |\pi| + |x|. \end{aligned}$$

**Proof** Induction on the length of rule string  $\pi \in P^*$ .

i)  $\pi = \varepsilon$ .  $\delta = \gamma x$ .

$$\begin{aligned} \$\gamma \mid xy\$ \xrightarrow{\theta} \$\gamma x \mid y\$ = \$\delta \mid y\$, \text{ and} \\ |\theta| = |x|, \tau(\theta) = \pi = \varepsilon. \end{aligned}$$

ii)  $\pi = A \rightarrow \omega \cdot \pi'$ ,  $\pi' \neq \varepsilon$ ,  $\exists \theta'$ ,  $\tau(\theta') = \pi'$ .

$$\delta \xrightarrow[rm]{\pi^R} \gamma' x' = \delta'' A x' \xrightarrow[rm]{A \rightarrow \omega} \delta'' \omega x' = \gamma x$$

where  $|\theta'| = |\pi'| + |x'|$ ,  $\gamma' = \delta'' A$ ,  $\delta'' \omega = \gamma z$ ,  $z x' = x$

$$\begin{aligned} \therefore \$\gamma \mid xy\$ = \$\gamma \mid zx'y\$ \xrightarrow{\theta''} \$\gamma z \mid x'y\$ = \$\delta'' \omega \mid x'y\$ \\ \xrightarrow{r} \$\delta'' A \mid x'y\$ = \$\gamma' \mid x'y\$ \xrightarrow{\theta'} \$\delta \mid y\$. \end{aligned}$$

where  $|\theta''| = |z|$ ,  $\tau(\theta'') = \varepsilon$ .

$$\therefore |\theta| = |\theta'| + 1 + |\theta''| = |\pi'| + |x'| + 1 + |z| = |\pi| + |x|.$$

**Lemma 5.20** Let  $M$  be a *shift-reduce* parser for  $G$ .

$$(1) L(G) \subseteq L(M),$$

(2)  $\forall \pi$ : **right parse** of  $w$  in  $G$ ,  $\tau(\theta) = \pi$  in  $M$ ,

$$(3) \text{Time}_M(w) \leq \text{Time}_G(w) + |w|.$$



A grammar  $G = (N, \Sigma, P, S)$  is in **Greibach normal-form**, if its rules are of forms:

- (1)  $A \rightarrow a\beta$ ,  $a \in \Sigma$ ,  $\beta \in (V \setminus \{S\})^*$  where  $V = N \cup \Sigma$ ,
- (2)  $S \rightarrow \varepsilon$ .

**one symbol lookahead**

$$\begin{array}{l} A \mid \rightarrow \beta^R a \mid \\ S \mid \rightarrow \mid \\ a \mid a \rightarrow \mid \end{array}$$

$$\begin{array}{l} A \mid a \rightarrow \beta^R a \mid a \\ S \mid \$ \rightarrow \mid \$ \\ a \mid a \rightarrow \mid \end{array}$$

A Greibach normal-form grammar that has no pair of distinct rules of the form,

$$A \rightarrow a\beta_1, A \rightarrow a\beta_2 \in P \Rightarrow \beta_1 = \beta_2.$$

is called **simple grammar**(or **s-grammar**)

*predictive parser for simple grammar is deterministic.*

**produce shift parser**

$$A \mid \rightarrow \omega_1^R \mid \text{ or } A \mid \rightarrow \omega_2^R \mid \text{ produce-produce conf.}$$

$$A \mid \rightarrow \omega^R \mid \text{ or } a \mid a \rightarrow \mid \text{ no shift-produce conf.}$$

$$a \mid a \rightarrow \mid \text{ or } b \mid b \rightarrow \mid \text{ no shift-shift conf.}$$

**shift-reduce parser**

$$\mid a \rightarrow a \mid \text{ or } \omega \mid \rightarrow A \mid \text{ shift-reduce conflict}$$

$$\omega_1 \mid \rightarrow A_1 \mid \text{ or } \omega_2 \mid \rightarrow A_2 \text{ reduce-reduce conflict}$$

$$\mid a \rightarrow a \mid \text{ or } \mid b \rightarrow b \mid \text{ no shift-shift conf.}$$

Let  $G = (N, \Sigma, P, S)$  be a context-free grammar.

Then **produce-shift**(or **predictive; top-down**) **machine** for  $G$  is a pda  $M_P(V, \Sigma, \Gamma_P, P, \tau, S, \{\varepsilon\}, \$, |)$ , where  $\Gamma_P$  and  $\tau$  are of form:

(pa)  $A | \rightarrow \omega^R | \in \Gamma_P$  "**produce** by  $A \rightarrow \omega \in P$ ",

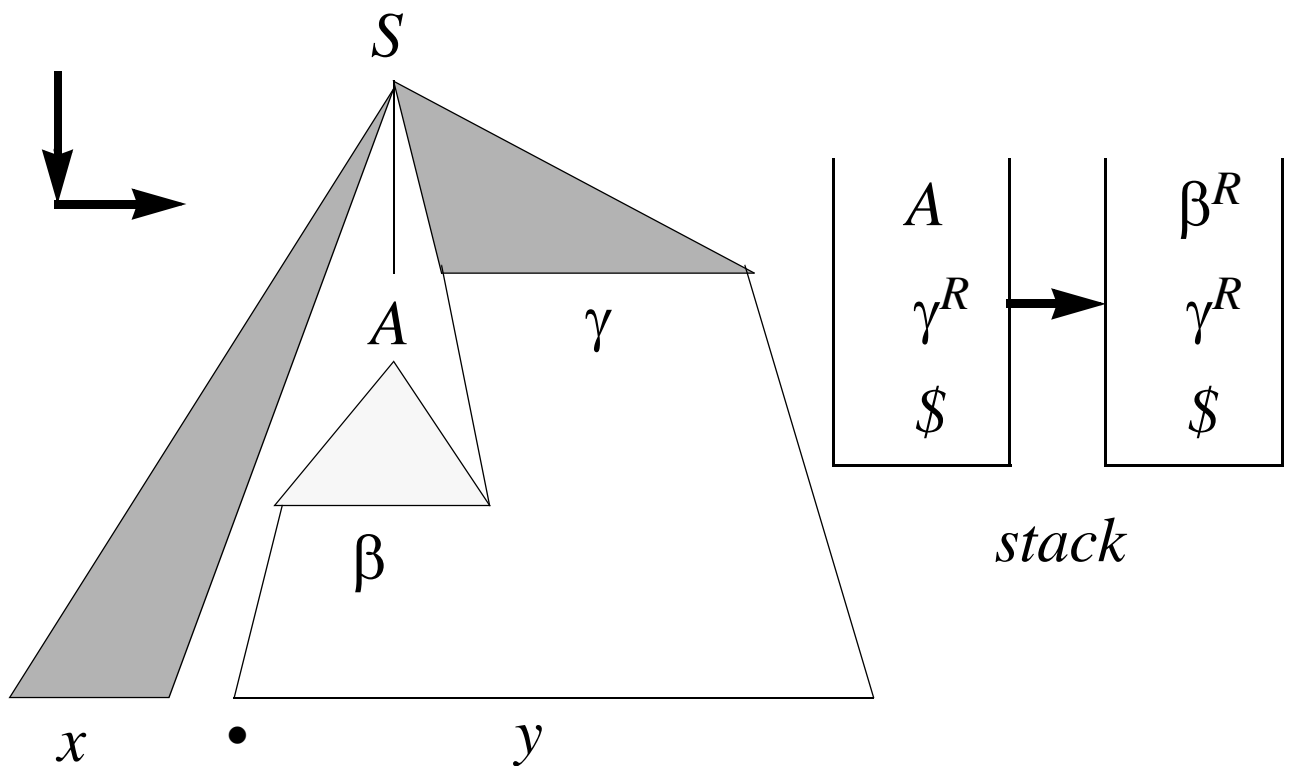
$\tau(A | \rightarrow \omega^R |) = A \rightarrow \omega$ .

(sa)  $a | a \rightarrow | \in \Gamma_P$  "**shift**  $a \in \Sigma$ ",

$\tau(a | a \rightarrow |) = \varepsilon$ .

$$S \xrightarrow{*}_{lm} xA\beta \xrightarrow{lm} x\omega\beta \xrightarrow{*}_{lm} xy$$

$$\$S | xy\$ \xrightarrow{*} \$\beta^R A | y\$ \Rightarrow_p \$\beta^R \omega^R | y\$ \xrightarrow{*} \$ | \$$$



The **shift-reduce**(or **bottom-up**) machine for  $G$  is a pda  $M_S(V, \Sigma, \Gamma_S, P, \tau, \varepsilon, \{S\}, \$, |)$ , where  $\Gamma_S$  and  $\tau$  are of form:

$$(sa) \quad |a \rightarrow a| \in \Gamma_S \quad \text{"shift } a \in \Sigma",$$

$$\tau(|a \rightarrow a|) = \varepsilon.$$

$$(ra) \quad \omega | \rightarrow A | \in \Gamma_S \quad \text{"reduce by } A \rightarrow \omega \in P",$$

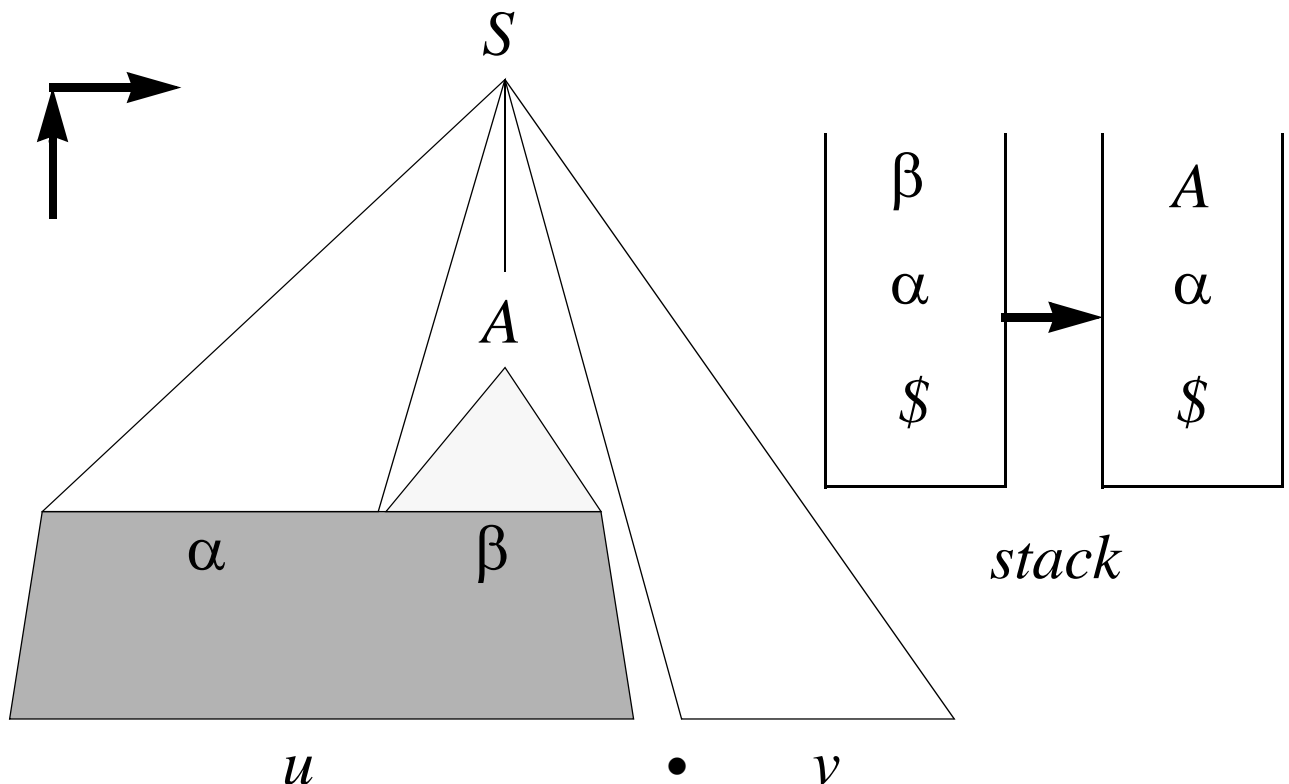
$$\tau(\omega | \rightarrow A |) = A \rightarrow \omega.$$

$$S \xRightarrow{*}_{rm} \alpha A v \Rightarrow_{rm} \alpha \omega v \xRightarrow{*}_{rm} uv.$$

**Reversed** order of rightmost derivation

$$uv \xleftarrow{*}_{rm} \alpha \omega v \xleftarrow{rm} \alpha A v \xleftarrow{*}_{rm} S.$$

$$\$ | uv \$ \xRightarrow{*} \$ \alpha \omega | v \$ \Rightarrow_r \$ \alpha A | v \$ \xRightarrow{*} \$ S | \$$$



$$\begin{aligned}
\$S \mid a_1 \dots a_l b_1 \dots b_m \$ &\Rightarrow_p \$\sigma^R \mid a_1 \dots a_l b_1 \dots b_m \$ \\
&\stackrel{*}{\Rightarrow} \$\beta^R A \mid b_1 \dots b_m \$ \Rightarrow_p \$\beta^R \omega^R \mid b_1 \dots b_m \$ \stackrel{*}{\Rightarrow} \$ \mid \$ \\
&\stackrel{*}{\Rightarrow} \$\beta^R a_l \mid a_l b_1 \dots b_m \$ \Rightarrow_s \$\beta^R \mid b_1 \dots b_m \$ \stackrel{*}{\Rightarrow} \$ \mid \$.
\end{aligned}$$

$$\begin{aligned}
\$ \mid a_1 \dots a_l b_1 \dots b_m \$ &\Rightarrow_s \$a_1 \mid a_2 \dots a_l b_1 \dots b_m \$ \\
&\stackrel{*}{\Rightarrow} \$\alpha \omega \mid b_1 \dots b_m \$ \Rightarrow_r \$\alpha A \mid b_1 \dots b_m \$ \stackrel{*}{\Rightarrow} \$S \mid \$ \\
&\stackrel{*}{\Rightarrow} \$\alpha \omega \mid a_l b_1 \dots b_m \$ \Rightarrow_s \$\alpha \omega a_l \mid b_1 \dots b_m \$ \stackrel{*}{\Rightarrow} \$S \mid \$.
\end{aligned}$$

Let  $G = (N, \Sigma, S, P)$  be a cfg,  $w$  be a sentence of  $G$ , and  $\pi_l$  and  $\pi_r$  in  $P^*$  be a left- and right-parses of  $w$ , respectively. And let  $M_L = (V, \Sigma, \Gamma_L, P, \tau_L, S, \{\varepsilon\})$  and  $M_R = (V, \Sigma, \Gamma_R, P, \tau_R, \varepsilon, \{S\})$  be a produce-shift and shift-reduce parsers for  $G$ , respectively; and  $\theta_L$  in  $\Gamma_L^*$  and  $\theta_R$  in  $\Gamma_R^*$  are sequences of action strings for  $w$  in  $M_R$  and  $M_L$ , respectively.

$$\begin{aligned}
S &\xrightarrow[lm]{\pi_l} x \text{ in } G, & S &\xrightarrow[rm]{\pi_r^R} x \text{ in } G \\
S \mid x &\xRightarrow{\theta_L} \mid \text{ in } M_L, & \mid x &\xRightarrow{\theta_R} S \mid \text{ in } M_R
\end{aligned}$$

Then

- (1)  $|\theta_L| = |\theta_R| = |\pi_l| + |x| = |\pi_r| + |x|$ , and
- (2)  $\tau_L(\theta_L) = \pi_l$ ,  $\tau_R(\theta_R) = \pi_r$ .

Let  $G = (N, \Sigma, S, P)$  be a cfg,  $x$  be a sentence of  $G$ , and  $\Xi$  be a parse tree of  $x$ . And let  $M_L = (V, \Sigma, \Gamma_L, P, \tau_L, S, \{\varepsilon\})$  and  $M_R = (V, \Sigma, \Gamma_R, P, \tau_R, \varepsilon, \{S\})$  be a produce-shift and shift-reduce parsers for  $G$ , respectively. Then following actions in  $M_L$  and  $M_R$  generates the parse tree  $\Xi$  of  $x$ .

(pa)  $A \mid \rightarrow \omega^R \mid \in \Gamma_L$   
 make nodes  $X_1, \dots, X_n$  and  $A$ , where  $\omega = X_1 \dots X_n$   
 make a subtree( $A, \omega$ )

(sa)  $a \mid a \rightarrow \mid \in \Gamma_L$   
 no action

(sa)  $\mid a \rightarrow a \mid \in \Gamma_R$   
 make a leaf node  $a$

(ra)  $\omega \mid \rightarrow A \mid \in \Gamma_R$   
 make a interior node  $A$ ;  
 make a subtree ( $A, \omega$ )

### 5.3 Strong LL(k) Parsing

**Fact 5.23** Let  $M = (V, \Sigma, \Gamma, \Delta, \tau, \gamma_s, F)$  and  $M' = (V, \Sigma, \Gamma', \Delta, \tau, \gamma_s, F)$  be pdt's with for some  $\delta \in V^*$ ,  $z \in \Sigma^*$ ,  
 $\underline{\delta}\alpha \mid xy\underline{z} \rightarrow \underline{\delta}\beta \mid y\underline{z} \in \Gamma'$ , if  $\alpha \mid xy \rightarrow \beta \mid y \in \Gamma$ . Then  
 $L(M') \subseteq L(M)$ .

*adding lookback and lookaheads  
 reduces language set*

$First_{G,k}(\gamma) = k:L_G(\gamma)$ .

$Follow_{G,k}(\gamma) = \{y \in \Sigma^* \mid S \xRightarrow{*} \alpha\gamma\beta \text{ in } G, y \in First_{G,k}(\beta)\}$ .

$First_{G,k} \equiv First_k$ ,  $Follow_{G,k} \equiv Follow_k$ .

$First_k(W) = \bigcup_{\gamma \in W} First_k(\gamma)$  where  $W \subseteq V^*$ .

### Fact 5.24

(a)  $First_k(\alpha\beta\gamma) = k:First_{k+n}(\alpha\beta\gamma) = First_k(\alpha First_{k+n}(\beta)\gamma)$

(b)  $Follow_k(\gamma) = k:Follow_{k+n}(\gamma) = First_k(Follow_{k+n}(\gamma))$

$\$^k$ -augmented grammar  $G'$  for  $G = (N, \Sigma, P, S)$ .

$G' = \{N \cup \{S'\}, \Sigma \cup \{\$\}, P \cup \{S' \rightarrow \$^k S \$^k\}, S'\}$

where  $S', \$ \notin V$ .

$First'_k, Follow'_k$  denotes  $First_{G',k}$  and  $Follow_{G',k}$

**Fact 5.25**  $Follow'_k(\gamma) = k:Follow_k(\gamma)\$^k$ .

Let  $G = (N, \Sigma, P, S)$  be a grammar and  $k$  be a natural number. The **strong LL( $k$ ) parser** (or **SLL( $k$ ) parser**) for  $G$  is a pdt  $M = (V, \Sigma, \Gamma, P, \tau, S, \{\varepsilon\})$ ,

$$(pa) \quad A \mid y \rightarrow \omega^R \mid y \in \Gamma$$

$$y \in \text{First}_k(\omega \text{Follow}_k(A) \$^k)$$

"produce by  $A \rightarrow \omega$  on lookahead  $y$ ",

$$\tau(A \mid y \rightarrow \omega^R \mid y) = A \rightarrow \omega.$$

$$(sa) \quad a \mid a \rightarrow \mid \in \Gamma \quad \text{"shift } a\text{"}$$

$$a \in \Sigma,$$

$$\tau(a \mid a \rightarrow \mid) = \varepsilon.$$

**Fact 5.26** For any reduced grammar, SLL(0) parser for  $G$  is just the produce-shift parser.

**Theorem 5.31** The SLL( $k$ ) parser  $M$  for a grammar  $G$  is a left parser for  $G$  for all  $k \geq 0$ . Moreover for any sentence  $w$  in  $L(G)$ ,  $M$  produces all left parses of  $w$  in  $G$  and  $\text{Time}_M(w) = \text{Time}_G(w) + |w|$ .

**Proof**

SLL( $k$ ) parser  $\Rightarrow$  linear left parser

**Lemma 5.27, 5.28**

It is trivial, since SLL( $k$ ) is

a restriction of predictive parser

SLL( $k$ ) parser  $\Leftarrow$  linear left parser

in **Lemma 5.29 and 5.30**.

**Lemma 5.29** Let  $G = (N, \Sigma, P, S)$  be a grammar and  $(M, \tau)$  be an  $SLL(k)$  parser.

If  $\gamma^R \xrightarrow[lm]{\pi} x\delta^R$  in  $G$ ,  $k:y \in First_k(\delta^R Follow_k(\gamma^R))$ ,

and  $\delta^R = \varepsilon$  or  $1:\delta^R \in N$ , then

$\$ \gamma \mid xy \$ \xrightarrow{\theta} \$ \delta \mid y \$$ ,  $\theta \in \Gamma^*$  and

$\tau(\theta) = \pi$ ,  $|\theta| = |\pi| + |x|$ .

**Proof** Induction on the length of rule string  $\pi$ .

i)  $\pi = \varepsilon$ .  $\gamma^R = x\delta^R$ .

$\$ \gamma \mid xy \$ = \$ \delta x^R \mid xy \$ \xrightarrow{\theta} \$ \delta \mid y \$$ ;  $|\theta| = |x|$ ,  $\tau(\theta) = \pi = \varepsilon$ .

ii)  $\pi = \pi' \cdot A \rightarrow \omega$ ,  $\pi' \neq \varepsilon$ ,  $\exists \theta'$ ,  $\tau(\theta') = \pi'$ .

$\gamma^R \xrightarrow[lm]{\pi'} x' \delta'^R = x' A \delta'' \xrightarrow[lm]{A \rightarrow \omega} x' \omega \delta'' = x' z \delta^R = x \delta^R$ ,

where  $|\theta'| = |\pi'| + |x'|$ ,  $\delta'^R = A \delta''$ ,  $\omega \delta'' = z \delta^R$ ,  $x' z = x$ .

Let  $y' = zy$ .

$\$ \gamma \mid xy \$ = \$ \gamma \mid x' zy \$ = \$ \gamma \mid x' y' \$ \xrightarrow{\theta'} \$ \delta' \mid y' \$ = \$ \delta''^R A \mid y' \$$

$\xrightarrow{r} \$ \delta''^R \omega^R \mid y' \$ = \$ (\omega \delta'')^R \mid zy \$ = \$ \delta z^R \mid zy \$$

$\xrightarrow{\theta''} \$ \delta \mid y \$$ , where  $|\theta''| = |z|$ ,  $\tau(\theta'') = \varepsilon$ .

$\therefore |\theta| = |\theta'| + 1 + |\theta''| = |\pi'| + |x'| + 1 + |z| = |\pi| + |x|$ .

$k:y' \in First_k(\delta'^R Follow_k(\gamma^R)) = First_k(A \delta'' Follow_k(\gamma^R))$

$k:y' \in First_k(\omega \delta'' Follow_k(\gamma^R)) = First_k(z \delta^R Follow_k(\gamma^R))$

$\therefore k:y \in First_k(\delta^R Follow_k(\gamma^R))$ . Q.E.D.

**Lemma 5.30** Let  $(M, \tau)$  is a  $SLL(k)$  parser for  $G$ .

$L(G) \subseteq L(M)$ ,  $Time_M(w) \leq Time_G(w) + |w|$ .



### 5.4 Strong LL(k) Grammars

$G$  is strong LL(k) (or SLL(k)),

if its SLL(k) parser is deterministic.

$L$  is strong LL(k) (or SLL(k)),

if  $L$  is generated by SLL(k) grammar.

**Lemma 5.32** Let  $G = (N, \Sigma, P, S)$  be a grammar.

If  $Y \xrightarrow{n} \alpha X \beta$ ,  $\alpha \xrightarrow{*} x$ , and  $\beta \xrightarrow{*} y$  in  $G$ . Then

$Y \xrightarrow[lm]{*} x X \gamma$ ,  $\gamma \xrightarrow{*} y$  in  $G$ .

**Proof** Induction on  $n$

i)  $n=0$ ,  $\alpha=\beta=\varepsilon$ ,  $X=Y$ ,  $x=y=\varepsilon$ , choose  $\gamma=\varepsilon$ .

ii)  $n>0$ , three cases

(a)  $Y \xrightarrow{n-1} \alpha' A \beta' \Rightarrow \alpha' \omega \beta' = \alpha' \omega \beta'' X \beta' = \alpha X \beta$ .

$\alpha' \xrightarrow{*} x'$ ,  $\beta' = \gamma \xrightarrow{*} y$

$\therefore Y \xrightarrow[lm]{*} x' A \beta' \xrightarrow[lm]{*} x' \omega \beta'' X \beta' \xrightarrow[lm]{*} x' x'' X \beta' = x X \gamma$ .

$\gamma = \beta' \xrightarrow{*} y$

(b)  $Y \xrightarrow{n-1} \alpha' A \beta' \Rightarrow \alpha' \omega \beta' = \alpha' \omega_1 X \omega_2 \beta' = \alpha X \beta$ .

$\alpha' \xrightarrow{*} x'$ ,  $\beta' \xrightarrow{*} y'$ ,  $\gamma' \xrightarrow{*} y'$ , and

$Y \xrightarrow[lm]{*} x' A \gamma' \xrightarrow[lm]{*} x' \omega \gamma' = x' \omega_1 X \omega_2 \gamma' \xrightarrow[lm]{*} x X \omega_2 \gamma' = x X \gamma$ ,

$\gamma = \beta = \omega_2 \gamma' \xrightarrow{*} y$ .

(c)  $Y \xrightarrow{n-1} \alpha' A \beta' \Rightarrow \alpha' \omega \beta' = \alpha X \alpha'' \omega \beta' = \alpha X \beta$ .

$\alpha \xrightarrow{*} x$ ,  $\alpha'' \omega \beta' \xrightarrow{*} y$

$Y \xrightarrow[lm]{*} x X \alpha'' A \beta' = x X \gamma$ ,  $\gamma = \alpha'' \omega \beta' \xrightarrow{*} y$ .

**Lemma 5.33**

$Follow_k(X) = \{y \in \Sigma^* \cup \Sigma^* \$^k \mid S \xrightarrow{*}_{lm} xX\beta, y \in First_k(\beta \$^k)\}$

Two actions

$$A_1 \mid y_1 \rightarrow \omega_1^R \mid y_1, A_2 \mid y_2 \rightarrow \omega_2^R \mid y_2,$$

**exhibits a produce-produce conflict, if**

$$A_1 = A_2, y_1 = y_2, \text{ and } \omega_1 \neq \omega_2.$$

A nonterminal  $A$  has the **SLL(k) property, if**

$$First_k(\omega_1 Follow_k(A)) \cap First_k(\omega_2 Follow_k(A)) = \emptyset,$$

$$\forall A \rightarrow \omega_1 \mid \omega_2 \in P.$$

**Theorem 5.34**

**(Characterization of SLL(k) Grammars)**

(a) The SLL(k) parser for  $G$  is **deterministic**.

(b) **No pair of produce actions in SLL(k) parser for  $G$  exhibits a produce-produce conflicts.**

(c) All nonterminals of  $G$  has the **SLL(k) property**.

(d) The conditions

$$S \xrightarrow{*}_{lm} x_1 A \beta_1 \xrightarrow{*}_{lm} x_1 \omega_1 \beta_1 \xrightarrow{*}_{lm} x_1 y_1,$$

$$S \xrightarrow{*}_{lm} x_2 A \beta_2 \xrightarrow{*}_{lm} x_2 \omega_2 \beta_2 \xrightarrow{*}_{lm} x_2 y_2,$$

$$k:y_1 = k:y_2,$$

**always imply that  $\omega_1 = \omega_2$ .**

**Proof.**

$M$  is nondeterministic (a')

$M$  has distinct produce actions (b')

$$A \mid y' \rightarrow \omega_1^R \mid y', A \mid y'z \rightarrow \omega_2^R \mid y'z.$$

$$A \rightarrow \omega_1 / \omega_2 \in P,$$

$$y' \in \text{First}_k'(\omega_1 \text{Follow}_k'(A)),$$

$$y'z \in \text{First}_k'(\omega_2 \text{Follow}_k'(A)),$$

$$\omega_1 \neq \omega_2 \text{ or } z \neq \varepsilon.$$

$$y' = k:y\$^k, y \in \text{First}_k(\omega_1 \text{Follow}_k(A)),$$

$$y'z = k:v\$^k, v \in \text{First}_k(\omega_2 \text{Follow}_k(A)),$$

$$z = \varepsilon \text{ and } y = v.$$

$$A \rightarrow \omega_1 / \omega_2 \in P, \omega_1 \neq \omega_2, \quad (c')$$

$$y \in \text{First}_k(\omega_1 \text{Follow}_k(A)) \cap \text{First}_k(\omega_2 \text{Follow}_k(A)).$$

$$S \xrightarrow[\text{lm}]{*} x_1 A \beta_1, y \in \text{First}_k(\omega_1 \beta_1), \quad (d')$$

$$S \xrightarrow[\text{lm}]{*} x_2 A \beta_2, y \in \text{First}_k(\omega_2 \beta_2).$$

$$A \rightarrow \omega_1 / \omega_2 \in P, \omega_1 \neq \omega_2,$$

**Theorem 5.35** For all natural number  $k$ , the class of  $SLL(k)$  grammars is properly contained in the class of  $SLL(k+1)$  grammars.

**Proof**

$$\begin{aligned}
 & First_k(\omega Follow_k(A)) \\
 = & First_k(\omega First_k(Follow_{k+1}(A))) \\
 = & First_k(\omega Follow_{k+1}(A)) \\
 = & k:First_{k+1}(\omega Follow_{k+1}(A))
 \end{aligned}$$

$$\begin{aligned}
 G_k &= (\{S\}, \{a\}, \{S \rightarrow a^k \mid a^{k+1}\}, S) \\
 & G_k \text{ is } SLL(k+1) \text{ but not } SLL(k).
 \end{aligned}$$

**Proposition 5.36**

$$\begin{aligned}
 L_k &= \{a^n w \mid n \geq 1, w \in \{b, c, b^k d\}^n\} \\
 L_k &\text{ is } SLL(k) \text{ but not } SLL(k-1) \text{ for } k \geq 1.
 \end{aligned}$$

**Theorem 5.37** Any  $SLL(k)$  grammar is **unambiguous**.

A configuration  $\phi$  is **looping**, if

$$\forall n \geq 0, \exists \phi_n, \phi \xrightarrow{n} \phi_n.$$

**Fact 5.38** If  $\phi$  is a looping configuration,

$$\exists y, \gamma_0, \gamma_1, \dots :$$

$$\phi \xrightarrow{*} \$\gamma_0 \mid y$, \forall i \geq 0: \$\gamma_i \mid y$ \Rightarrow \$\gamma_{i+1} \mid y$$$

**Fact 5.39** A pda  $M$  **loops forever** on input string  $w$ , iff the initial configuration for  $w$  is looping.

A nonterminal  $A$  is **left-recursive** if  $A \xrightarrow{+} A\beta$ .

A grammar is **left-recursive**, if it has a left-recursive nonterminal.

**Theorem 5.40** Let  $G$  be a reduced left-recursive grammar. Then the SLL( $k$ ) parser  $M$  for  $G$  **loops forever** on some sentence  $w \in L(G)$ .

**Proof** Since  $G$  is left-recursive and reduced.

$$S \xrightarrow[lm]{*} xA\beta, A \xrightarrow[lm]{\pi} A\delta, A \xrightarrow{*} u, \delta \xrightarrow{*} v, \beta \xrightarrow{*} z.$$

Let  $\psi_n = (A\delta^n\beta)^R$ ,  $y_n = uv^n z$ . Then

$$\forall n \geq 0, S \xrightarrow[lm]{*} xA\beta \xrightarrow[lm]{\pi^n} xA\delta^n\beta = x\psi_n^R.$$

$$\$S \mid xy_n$ \xrightarrow{*} \$\beta A \mid y_n$ \xrightarrow{\theta_n} \$\beta(\delta^R)^n A \mid y_n = \psi_n \mid y_n$,$$

$$\tau(\theta_n) = \pi^n.$$

**Corollary 5.41** *A reduced left-recursive grammar is not SLL(k) for any  $k \geq 0$ .*

**Lemma 5.42** *Let  $G = (N, \Sigma, P, S)$  be a grammar and,  $r_i \in P$  such that*

$$\forall i \geq 0: A_i \gamma_i \xrightarrow[lm]{r_i} A_{i+1} \gamma_{i+1}.$$

*Then  $\exists i \in \mathbb{N} \ni A_i$  is left-recursive.,*

**Proof** *Let  $i_0; \forall i \geq 0: |\gamma_{i_0}| \leq |\gamma_i|$*

*Let  $i_k; \forall i \geq i_k: i_k > i_{k-1}, |\gamma_{i_k}| \leq |\gamma_i|$ . Then*

$$\forall k \geq 0: A_{i_k} \gamma_{i_k} \xrightarrow[lm]{\pi_k} A_{i_{k+1}} \gamma_{i_{k+1}}, \pi_k = r_{i_k} \cdots r_{i_{k+1}-1}.$$

$$|\pi_k| = |i_{k+1} - i_k| > 0, \pi_k = \pi'_k \pi''_k$$

$$A_{i_k} \xrightarrow[lm]{\pi'_k} \alpha_k, \gamma_{i_k} \xrightarrow[lm]{\pi''_k} \beta_k, \alpha_k \beta_k = A_{i_{k+1}} \gamma_{i_{k+1}}, |\pi'_k| > 0.$$

*Whenever  $|\pi''_k| > 0$ , then  $\alpha_k \in \Sigma^*$ .*

$$A_{i_k} \gamma_{i_k} \xrightarrow[lm]{\pi'_k} \alpha_k \gamma_{i_k} \xrightarrow[lm]{\pi''_k} \alpha_k \beta_k = A_{i_{k+1}} \gamma_{i_{k+1}}, \pi'_k \pi''_k = \pi_k$$

$$\alpha_k \gamma_{i_k} = A_j \gamma_j, \text{ where } j = i_k + |\pi'_k|.$$

$$\exists \alpha'_k: \alpha_k = A_j \alpha'_k, \text{ since } |\gamma_{i_k}| \leq |\gamma_j|.$$

$$\therefore \alpha_k \notin \Sigma^*, |\pi''_k| = 0; \text{ hence } j = i_{k+1},$$

$$\forall k \geq 0: A_{i_k} \xrightarrow[lm]{\pi'_k} \alpha_k = A_{i_{k+1}} \alpha'_k.$$

**Theorem 5.43** Let  $M$  be the  $SLL(k)$  parser for a grammar  $G$ . If some configuration is looping in  $M$ , then  $G$  is left-recursive.

**Proof** If  $\phi$  is a looping configuration,

$$\exists y, \gamma_0, \gamma_1, \dots, r_0, r_1, \dots :$$

$$\phi \xRightarrow{*} \$\gamma_0 \mid y$, \forall i \geq 0: \$\gamma_i \mid y$ \xRightarrow{r_i} \$\gamma_{i+1} \mid y$$$

$$\gamma_i = \alpha_i^R A_i,$$

$$\begin{aligned} \forall i \geq 0: A_i \alpha_i &= (\alpha_i^R A_i)^R = \gamma_i \xRightarrow{R, \tau(r_i)} \gamma_{i+1}^R \\ &= (\alpha_{i+1}^R A_{i+1})^R = A_{i+1} \alpha_{i+1} \end{aligned}$$

Since  $|\tau(r_i)| = 1$ ,  $G$  is left-recursive.

**Theorem 5.44** Let  $G$  be a reduced grammar and  $k$  a natural number. Then  $G$  is left-recursive iff the  $SLL(k)$  parser for  $G$  loops forever on some sentence in  $L(G)$ .

## 5.5 Construction of Strong LL(1) Parser

**Fact 5.45** The size of SLL( $k$ ) parser for  $G = (N, \Sigma, P, S)$  is  $O(k \cdot |\Sigma|^k \cdot |G|)$ .

**Proof**  $A \mid y \rightarrow \omega^R \mid y \in \Gamma$ , size  $|A\omega| + 2k + 2$  is  $O(k)$   
 $\forall A \mid y \rightarrow \omega^R \mid y \in \Gamma$ ,  $|\Sigma|^k \cdot |P|$  produce actions.

$X$  begins  $A$  if  $A \rightarrow \alpha X \beta \in P$ ,  $\alpha \xrightarrow{*} \varepsilon$   
 $A \mid X$ .

$X$  ends  $A$  if  $A \rightarrow \alpha X \beta \in P$ ,  $\beta \xrightarrow{*} \varepsilon$   
 $X \mid r A$ .

$X$  adjoins  $Y$  if  $A \rightarrow \alpha X \gamma Y \beta \in P$ ,  $\gamma \xrightarrow{*} \varepsilon$   
 $X \mid f Y$

$X$  terminals  $Y$  if  $Y \in \Sigma$ ,  $X = Y$ .  
 $a = b$ .

**Lemma 5.47** Let  $A \in N$ ,  $X \in V$ ,  $\alpha \in V^*$  and  $m \leq n$ .

(1)  $A \mid^m X$  implies  $A \xrightarrow[lm]{n} X\alpha$ .

(2)  $A \xrightarrow{n} X\alpha$  implies  $A \mid^m X$ .

(3)  $X \mid^m A$  implies  $A \xrightarrow[rm]{n} \alpha X$ .

(4)  $A \xrightarrow[rm]{n} \alpha X$  implies  $X \mid^m A$ .

$a \in \text{First}_1(A)$ , iff  $A \mid^* a$ .  $\exists$ .  $A \mid^* X$  and  $X = a$ .

$a \in \text{Follow}_1(A)$ , iff  $A \mid^* Y$ ,  $Y \mid X$ ,  $X \mid^* a$ .



**Proof** Induction on the length of the rule string  $\pi$ .

i)  $|\pi| = 1$ , choose  $X'=X$ ,  $Y'=Y$ ,  $r'=\pi=A \rightarrow \gamma X \psi Y \delta$ ,  
 $\pi'=\varepsilon$ ,  $\gamma'=\delta'=\alpha'=\beta'=\varepsilon$ ,  $\psi'=\psi$ .

ii)  $\pi=\pi_1 r$ ,  $r=A \rightarrow \omega_1$ ,  $|\pi_1| \geq 1$ .

$$A \xRightarrow{\pi_1} \gamma_1 A_1 \delta_1 \xRightarrow{r} \gamma_1 \omega_1 \delta_1 = \gamma X \psi Y \delta.$$

$$(a) \gamma_1 \omega_1 \delta_1 = \gamma_1 \omega_1 \delta'_1 \delta''_1 = \gamma \delta''_1 = \gamma X \psi Y \delta.$$

$$g_1 A_1 \delta_1 = \gamma_1 A_1 \delta'_1 \delta''_1 = \gamma_1 A_1 \delta'_1 X \psi Y \delta.$$

$$(b) \gamma_1 \omega_1 \delta_1 = \gamma_1 \underline{\alpha' X \psi'} \delta_1 = \gamma X \psi' \delta_1 = \gamma X \psi' \psi'' Y \delta \\ = \gamma X \psi Y \delta.$$

$$g_1 A_1 \delta_1 = \gamma_1 A_1 \psi'' Y \delta,$$

$$\psi'' \xRightarrow{*} \varepsilon, A_1 \xRightarrow{rm} \omega_1 = \alpha' X \psi' \xRightarrow{rm} \alpha' X.$$

$$(c) \gamma_1 \omega_1 \delta_1 = \gamma X \psi' \omega_1 \psi'' Y \delta = \gamma X \psi Y \delta.$$

$$g_1 A_1 \delta_1 = \gamma X \psi' A_1 \psi'' Y \delta, \psi' A_1 \psi'' \xRightarrow{*} \varepsilon.$$

$$(d) \gamma_1 \omega_1 \delta_1 = \gamma_1 \underline{\psi'' Y \beta'} \delta_1 = \gamma_1 \psi'' Y \delta = \gamma X \psi' \psi'' Y \delta \\ = \gamma X \psi Y \delta.$$

$$g_1 A_1 \delta_1 = \gamma X \psi' A_1 \delta_1,$$

$$\psi' \xRightarrow{*} \varepsilon, A_1 \xRightarrow{lm} \omega_1 = \psi'' Y \beta' \xRightarrow{lm} Y \beta'.$$

$$(e) \gamma_1 \omega_1 \delta_1 = \gamma'_1 \gamma''_1 \omega_1 \delta_1 = \gamma'_1 \underline{\delta} = \gamma X \psi Y \delta.$$

$$g_1 A_1 \delta_1 = \gamma'_1 \gamma''_1 A_1 \delta_1 = \gamma X \psi Y \gamma''_1 A_1 \delta_1.$$

$$(f) \gamma_1 \omega_1 \delta_1 = \gamma_1 \underline{\alpha X \psi Y \beta} \delta_1 = \gamma X \psi Y \beta \delta_1 = \gamma X \psi Y \delta.$$

Choose  $X'=X$ ,  $Y'=Y$ ,  $r'=r$ ,  $\gamma'=\gamma_1$ ,  $\delta'=\delta_1$ ,  $\alpha'=\beta'=\varepsilon$ ,  
 $\pi'=\pi_1$ .

**Lemma 5.51** Let  $G = (N, \Sigma, P, S)$  be a grammar. Then  $a \in \text{Follow}_1(X)$  implies  $X r^* X' f Y' l^* a$ .

Moreover, if  $G$  is reduced then the converse also holds.

**Proof**  $a \in \text{Follow}_1(X)$  iff  $S \xRightarrow{*} \gamma X a \gamma$ .

$B \rightarrow \alpha X' \psi' Y' \beta \in P, X' \xRightarrow{*} \alpha' X, \psi' \xRightarrow{*} \varepsilon, Y' \xRightarrow{*} a \beta'$ .

If  $G$  is reduced,  $S \xRightarrow{*} \gamma' B \delta \Rightarrow \gamma' \alpha X' \psi' Y' \beta \delta$

$\xRightarrow{*} \gamma' \alpha \alpha' X a \beta' \beta \delta \xRightarrow{*} \gamma' \alpha \alpha' X a \gamma$ .

*a position (or item core) of  $G$  is of the form;*

$A \rightarrow \alpha \bullet \beta$ , where  $A \rightarrow \alpha \beta \in P, \bullet \notin (N \cup \Sigma)$

(a)  $A \rightarrow \alpha \bullet X \beta$  *points* ( $\uparrow$ )  $X$ .

(b)  $A \rightarrow \alpha \bullet X \beta$  *passes-any* ( $\downarrow$ )  $A \rightarrow \alpha X \bullet \beta$ .

(c)  $A \rightarrow \alpha \bullet X \beta$  *passes-null* ( $\downarrow$ )  $A \rightarrow \alpha X \bullet \beta$ , if  $X \xRightarrow{*} \varepsilon$ .

$\exists a \in \Sigma, A \rightarrow \omega \in P: a \in \text{First}_1(\omega \text{Follow}_1(A))$  iff

$\omega = \alpha X \beta$ , where  $\alpha \xRightarrow{*} \varepsilon$  and  $a \in \text{First}_1(X)$ , or

$\omega \xRightarrow{*} \varepsilon$  and  $a \in \text{Follow}_1(A)$ .

$A \rightarrow \bullet \omega \Downarrow^* A \rightarrow \alpha \bullet X \beta \uparrow X l^* a$ , or

$A \rightarrow \bullet \omega \Downarrow^* A \rightarrow \omega \bullet \wedge A r^* f l^* a$ .

$$First_1: V^* \rightarrow 2^{\Sigma^{\leq 1}} = 2^{\Sigma \cup \{\epsilon\}}.$$

$$Follow_1: N \rightarrow 2^{\Sigma \cup \{\$\}^1}.$$

$$\therefore First_1(\alpha Follow_1(A)): P \rightarrow 2^{\Sigma \cup \{\$\}^1}.$$

$$First_1(\alpha\beta) = First_1(\alpha), \text{ if } \alpha \not\Rightarrow^* \epsilon,$$

$$First_1(\alpha) \cup First_1(\beta), \text{ if } \alpha \Rightarrow^* \epsilon, \beta \Rightarrow^* \epsilon,$$

$$First_1(\alpha) \cup First_1(\beta) \setminus \{\epsilon\}, \text{ if } \alpha \Rightarrow^* \epsilon, \beta \not\Rightarrow^* \epsilon.$$

$$First: V^* \rightarrow 2^{\Sigma}$$

$$First(\alpha) = \{a \in \Sigma \mid \alpha \Rightarrow^* ax, x \in \Sigma^*\}$$

$$First(\alpha\beta) = First(\alpha), \text{ if } \alpha \not\Rightarrow^* \epsilon,$$

$$First(\alpha) \cup First(\beta), \text{ if } \alpha \Rightarrow^* \epsilon.$$

$$\text{But } \alpha\beta \Rightarrow^* \epsilon, \text{ iff } \alpha \Rightarrow^* \epsilon, \beta \Rightarrow^* \epsilon.$$

$$First_1(\alpha Follow_1(A)) = First_1(\alpha Follow_1(A\$))$$

$$= First(\alpha), \text{ if } \alpha \not\Rightarrow^* \epsilon,$$

$$First(\alpha) \cup Follow(A), \text{ if } \alpha \Rightarrow^* \epsilon.$$

$$First(\alpha) = \{a \mid a \in First(X), \alpha = \beta X \gamma, \beta \Rightarrow^* \epsilon\}$$

$$= \{a \mid \alpha = \beta a \gamma, \beta \Rightarrow^* \epsilon\}$$

$$\cup \{a \mid a \in First(A), \alpha = \beta A \gamma, \beta \Rightarrow^* \epsilon\}$$

*First*:  $N \rightarrow 2^\Sigma$ .

$$\begin{aligned}
 \text{First}(A) &= \{a \mid a \in \text{First}(\alpha), A \rightarrow \alpha \in P\} \\
 &= \{a \mid A \rightarrow \alpha a \beta \in P, \alpha \Rightarrow^* \varepsilon\} \\
 &\quad \cup \{a \mid a \in \text{First}(B), A \rightarrow \alpha B \beta, \alpha \Rightarrow^* \varepsilon\} \\
 &= \{a \mid A \mid a\} \cup \{a \mid a \in \text{First}(B), A \mid B\} \\
 &= \{a \mid a \in \text{First}(X), A \mid X\} \\
 &= \{a \mid a \in \text{Init\_First}(A)\} \cup \{a \mid a \in \text{First}(B), A \mid B\} \\
 &\quad \text{where } \text{Init\_First}(A) = \{a \mid A \mid a\} \\
 &= \text{Init\_First}(A) \cup \cup_{A \mid B} \text{First}(B)
 \end{aligned}$$

$$\begin{aligned}
 F(x) &= G(x) \cup \cup_{x R y} F(y) && \text{recursion} \\
 &= \cup_{x R^* y} G(y) && \text{iteration}
 \end{aligned}$$

$$\text{First}(A) = \cup_{A \mid^* B} \text{Init\_First}(B)$$

**for**  $A \rightarrow X_1 \dots X_n \in P$  **do**

$i := 1$ ;  $\text{more} := \mathbf{true}$ ;

**while**  $i \leq n \wedge \text{more}$  **do**

**if**  $i < n \wedge X_i \Rightarrow^* \varepsilon \wedge X_{i+1} \in \Sigma$

$\rightarrow \text{Init\_First}(A) := \text{Init\_First}(A) \cup \{X_{i+1}\}$

|  $i < n \wedge X_i \Rightarrow^* \varepsilon \wedge X_{i+1} \in N$

$\rightarrow \text{make\_l\_rel}(A, X_{i+1})$

|  $X_i \not\Rightarrow^* \varepsilon \rightarrow \text{more} := \mathbf{false}$

**od od fi**

## 5.7 Simple Precedence Parsing

one lookahead in SLL(1) parser

one **lookback** into "stack"

**precedence** relation

simple precedence parser

$\varepsilon$ -free

**simple precedence parser**

(1)  $X\omega \mid a \rightarrow XA \mid a$  "reduce by  $A \rightarrow \omega$ "

on lookback  $X$  and lookahead  $a$

$X \leq l:\omega, \omega:l \succ a,$

$X \in V \cup \{\$, \}, a \in \Sigma \cup \{\$ \}.$

$\tau(X\omega \mid a \rightarrow XA \mid a) = A \rightarrow \omega.$

(2)  $Y \mid b \rightarrow Yb \mid$  "shift  $b$ "

on lookback  $Y$  and lookahead  $b$

$Y \leq b,$

$Y \in V \cup \{\$, \}, b \in \Sigma.$

$\tau(Y \mid b \rightarrow Yb \mid) = \varepsilon.$

### Precedence relations

Let  $G = (N, \Sigma, P, S)$ .

$\dot{=} = \text{adjoins } (f)$ ,

$X \dot{=} Y$ , if  $A \rightarrow \alpha X \gamma Y \beta \in P$ ,  $\gamma \Rightarrow^* \varepsilon$ .

$\triangleleft = \text{adjoins } (\text{begins}^+)^{-1} (f(l^+)^{-1})$ ,

$X \triangleleft Y$ , if  $A \rightarrow \alpha X \gamma B \beta \in P$ ,  $\gamma \Rightarrow^* \varepsilon$ ,  $B \xrightarrow[lm]{+} Y\delta$ .

$\triangleright = \text{ends}^+ \text{ adjoins } (\text{begins}^*)^{-1} \text{ terminal}$   
 $(r^+ f(l^*)^{-1})$ ,

$X \triangleright a$ , if  $B \xrightarrow[rm]{+} \gamma X$ ,  $A \rightarrow \alpha B \xi Y' \beta \in P$ ,  $\xi \Rightarrow^* \varepsilon$ ,  
 $Y' \xrightarrow[lm]{*} a\delta$ .

where  $\dot{=} \cup \triangleleft = \leq$

**Lemma 5.58** Let  $G$  be a  $\varepsilon$ -free grammar. Then

(1)  $X \dot{=} Y$ , iff  $A \rightarrow \alpha XY \beta \in P$ .

(2)  $X \triangleleft Y$ , iff  $A \rightarrow \alpha XB \beta \in P$ ,  $B \xrightarrow[lm]{+} Y\delta$

or  $X \dot{=} Y$ ,  $B \dot{l}^* Y$ .

(3)  $X \leq Y$ , iff  $A \rightarrow \alpha XY' \beta \in P$ ,  $Y' \xrightarrow[lm]{*} Y\delta$ .

(4)  $X \triangleright a$ , iff  $A \rightarrow \alpha BY' \beta \in P$ ,  $B \xrightarrow[rm]{+} \gamma X$ ,  $Y' \xrightarrow[lm]{*} a\delta$ .

or  $B \dot{=} Y'$ ,  $B r^+ X$ ,  $Y' \dot{l}^* a$ .

**Example**

$S' \rightarrow \$ S \$$   
 $S \rightarrow a \mid b D L e$   
 $D \rightarrow d$   
 $L \rightarrow ; \mid S \mid L ; S$

$X \dot{=} A \quad X \triangleleft A l^+ Y$

$\$ \dot{=} S \quad \$ \triangleleft \{a, b\}$

$b \dot{=} D \quad b \triangleleft \{d\}$

$D \dot{=} L \quad D \triangleleft \{L, ;\}$

$;\dot{=} S \quad ; \triangleleft \{a, b\}$

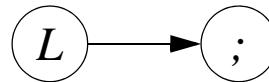
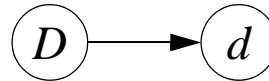
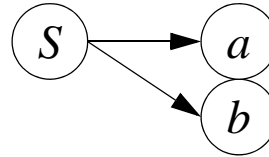
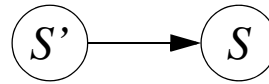
$A \dot{=} X \quad A r^+ Y \triangleright X l^* a$

$S \dot{=} \$ \quad \{a, e\} \triangleright \{\$\}$

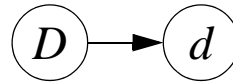
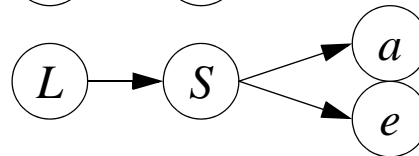
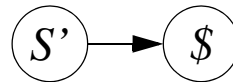
$D \dot{=} L \quad \{d\} \triangleright \{;\}$

$L \dot{=} e \quad \{S, a, e\} \triangleright \{e\}$

$L \dot{=} ; \quad \{S, a, e\} \triangleright \{;\}$



*l-graph*



*r-graph*

	$S$	$D$	$L$	$a$	$d$	$b$	$;$	$e$	$\$$
$S$							$\triangleright$	$\triangleright$	$\dot{=}$
$D$			$\leq$				$\triangleleft$		
$L$							$\dot{=}$	$\dot{=}$	
$a$							$\triangleright$	$\triangleright$	$\triangleright$
$d$							$\triangleright$		
$b$		$\dot{=}$			$\triangleleft$				
$;$	$\dot{=}$			$\triangleleft$	$\triangleleft$				
$e$							$\triangleright$	$\triangleright$	$\triangleright$
$\$$	$\dot{=}$			$\triangleleft$	$\triangleleft$				

**reduce actions** $S \rightarrow a$  $S \rightarrow bDL e$  $D \rightarrow d,$  $L \rightarrow ; S$  $L \rightarrow L ; S$ **lookback  $X$**  $\{\$, ;\} \triangleleft a=1:\omega,$  $\{\$, ;\} \triangleleft b=1:\omega,$  $b \triangleleft d=1:\omega,$  $D \triangleleft ;=1:\omega,$  $D \triangleleft L=1:\omega,$ **lookahead  $a$**  $a=\omega:1 \triangleright \{e, ;, \$\}$  $e=\omega:1 \triangleright \{e, ;, \$\}$  $d=\omega:1 \triangleright ;$  $S=\omega:1 \triangleright \{e, ;\}$  $S=\omega:1 \triangleright \{e, ;\}$ **shift actions** $\{;, \$\} \leq a$  $b \leq d$  $\{;, \$\} \leq b$  $\{D, L\} \leq ;$  $L \leq e, S \leq \$.$



**Theorem 5.65** *The simple precedence parser  $M$  for any  $\varepsilon$ -free grammar  $G$  is a right parser for  $G$ . Moreover, for each sentence  $w \in L(G)$ ,  $M$  produces all right parses of  $w$  in  $G$ , and  $\text{Time}_M(w) = \text{Time}_G(w) + |w|$ .*

**Proof**

*simple precedence parser  $\Rightarrow$  linear right parser in Lemma 5.60, 5.61.*

*Since simple precedence parser is a restriction of shift-reduce parser, it is trivial.*

*simple precedence parser  $\Leftarrow$  linear right parser in Lemma 5.62, 5.63, 5.64*

**Lemma 5.60** *If  $\$ \gamma \mid xy \$ \xrightarrow{\pi} \$ \delta \mid y \$$  in  $M$ . Then*

$$\delta \xrightarrow[rm]{\tau(\pi)^R} \gamma x \text{ in } G, \text{ and } |\pi| = |\tau(\pi)| + |x|.$$

**Lemma 5.61** *Let  $(M, \tau)$  is a simple precedence parser for  $G$ .*

$$L(G) \supseteq L(M), \text{ Time}_G(w) \leq \text{Time}_M(w) - |w|.$$

**Lemma 5.62** Let  $G = (N, \Sigma, P, S)$  be an  $\varepsilon$ -free grammar,  $X, Y \in V$ ,  $A \in N$ ,  $a \in \Sigma$ , and  $\alpha, \beta \in V^*$ .

(1) If  $X \leq A$  and  $G$  has  $A \rightarrow Y\beta$ , then  $X \leq Y$ .

(2) If  $A (\leq \cup \succ) a$  and  $G$  has  $A \rightarrow \alpha X$ , then  $X \succ a$ .

**Proof**

(1)  $B \rightarrow \alpha'XY'\beta'$  and  $Y' \xrightarrow[lm]{*} A\delta$ .

$Y' \xrightarrow[lm]{+} Y\beta\delta$ ,  $\therefore X \leq Y$ .

(2)  $B \rightarrow \alpha'X'Y'\beta'$ , where  $X' \xrightarrow[rm]{*} \gamma A$  and  $Y' \xrightarrow[lm]{*} a\delta$ .

$X' \xrightarrow[rm]{+} \gamma\alpha X$ ,  $\therefore X \succ a$ .

A string  $\gamma \in V^*$  is a **valid stack string**, if  $\gamma = X_1 \dots X_n$  is  $\varepsilon$  or  $\$ \leq X_1 \leq X_2 \dots \leq X_n$ .

Any two successive symbols in  $\gamma$  must be  $\leq$  related.

**Lemma 5.63** If  $\delta \xRightarrow[rm]{\pi^R} \gamma x$  in  $G$ ,  $\delta$  is valid stack string,

$\$ \delta : 1 (\leq \cup \triangleright) 1 : y \$$ , and either  $\gamma = \varepsilon$  or  $\gamma : 1 \in N$ .

Then  $\$ \gamma \mid xy \$ \xRightarrow{\theta} \$ \delta \mid y \$$  in  $M$ ,  $\gamma$  is valid stack string,  
 $\$ \gamma : 1 (\leq \cup \triangleright) 1 : xy \$$ ,  $\tau(\theta) = \pi$ ,  $|\theta| = |\pi| + |x|$ .

**Proof**

Induction on  $|\pi|$ .

i)  $\pi = \varepsilon$ ,  $\delta = \gamma x$ , and the symbols in  $x$  are  $\leq$  related.

$\$ \gamma \mid xy \$ \xRightarrow{\theta} \$ \gamma x \mid y \$ = \$ \delta \mid y \$$ ,  $\tau(\theta) = \varepsilon$ ,  $|\theta| = |x|$ .

Since  $\$ \gamma x : 1 (\leq \cup \triangleright) 1 : y \$$ , and  $\gamma x$  is valid stack string,  $\$ \gamma : 1 (\leq \cup \triangleright) 1 : xy \$$  ( $\gamma : 1 \leq 1 : x$ , if  $x \neq \varepsilon$ ).

ii) Assume  $\pi = r\pi'$ ,  $r = A \rightarrow \omega$ ,

$\delta \xRightarrow[rm]{\pi^R} \gamma' x' = \gamma'' A x' \xRightarrow[rm]{} \gamma'' \omega x' = \gamma x$  in  $G$ , and

$\$ \delta : 1 (\leq \cup \triangleright) 1 : y \$$ . Then

$\$ \gamma' \mid x' y' \$ \xRightarrow{\theta'} \$ \delta \mid y' \$$  in  $M$ ,

$\underline{\gamma'' A}$  is a valid stack string,  $\$ \gamma'' A : 1 (\leq \cup \triangleright) 1 : x' y' \$$ ,  
 $\tau(\theta') = \pi'$ , and  $|\theta'| = |\pi'| + |x'|$ .

$\$ \gamma' \mid x' y' \$ = \$ \gamma'' A \mid x' y' \$$

Since  $\gamma'' : 1 \leq 1 : \omega$  ( $\gamma'' : 1 \leq A$ ),  $\omega : 1 \triangleright 1 : x' y'$ . (**L5.62a,b**),

$\$ \gamma'' \omega \mid x' y' \$ \Rightarrow \$ \gamma'' A \mid x' y' \$ = \$ \gamma' \mid x' y' \$$ .

$\therefore \$ \gamma \mid xy \$ = \$ \gamma'' \omega' \mid \underline{x''} x' y' \$ \xRightarrow{|\theta''|} x \$ \gamma'' \omega' x'' \mid x' y' \$ =$   
 $\$ \gamma'' \omega \mid x' \$ \Rightarrow \$ \gamma'' A \mid x' y' \$ = \$ \gamma' \mid x' y' \$ \xRightarrow{\theta'} \$ \delta \mid y' \$$

$\gamma = \gamma'' \omega$  is v.s.t,  $\$ \gamma'' \omega : 1 (\leq \cup \triangleright) 1 : x'' x' y' \$$

$\pi = \theta'' \cdot \text{red}(A \rightarrow \omega) \cdot \theta'$  where  $\theta''$  is  $x''$ -shift actions.

**Lemma 5.64** Let  $(M, \tau)$  is a simple precedence parser for  $G$ .

$$L(G) \subseteq L(M), \text{Time}_M(w) \leq \text{Time}_G(w) + |w|.$$

**Proof**

$$\delta = S, x = w, \gamma = y = \varepsilon.$$

If  $S \xrightarrow[rm]{\pi^R} w$  in  $G$  and  $\$S:1 (\leq \cup \succ) 1:\$,$  Then

$$\begin{aligned} \$ \mid w\$ \xrightarrow{\theta} \$S \mid \$ \text{ in } M, \$ (\leq \cup \succ) 1:w\$, \\ \tau(\theta) = \pi, |\theta| = |\pi| + |w| \end{aligned}$$

Since,  $\$ \doteq S \doteq \$,$   
 $\$S \mid \$ \Rightarrow \$S\$ \mid .$

An  $\varepsilon$ -free grammar is a **simple precedence grammar** if its simple precedence parser is **deterministic**, and  $S \Rightarrow^+ S$  is **impossible** in  $G$ .

**Ambiguity**  $\$S \mid \$ \Rightarrow^+ \$S \mid \$$   
 $\$S \mid \$ \Rightarrow \$S\$ \mid \text{ or}$   
 $\Rightarrow \$S \mid \$.$

**Example**

$$S \rightarrow S \mid a$$

**Theorem 5.66** Any simple precedence grammar is **unambiguous**.

**Lemma 5.67** Let  $G=(N, \Sigma, P, S)$  be a reduced  $\varepsilon$ -free grammar. Then

$\forall X \in V, \exists Y \in V \cup \{\$, \exists a \in \Sigma \cup \{\$, Y \leq X (\leq \cup \succ) a$  holds in the  $\$$ -augmented grammar  $G'$  for  $G$ .

**Proof**  $S' \xRightarrow{*} \gamma X a \delta$  in  $G'$ ,  $\gamma \in \$V^*$ ,  $\delta \in (V \cup \{\$\})^*$ .

$A \rightarrow \alpha Y X' \beta \in P'$ , where  $X' \xRightarrow{*}_{lm} X \beta'$ . (L5.50)

$\therefore Y \leq X$ .

$A \rightarrow \alpha X' Y' \beta \in P'$ ,

where  $X' \xRightarrow{*}_{rm} \alpha' X$ ,  $Y' \xRightarrow{*}_{lm} Y \beta' = a \beta'$ . (L5.50)

$\therefore X' = \alpha' X$  or  $X' \xrightarrow{\neq}_{rm} \alpha' X$ .

$\therefore X \leq a$  or  $X \succ a$ . (L5.50 (3)(4))

**Theorem 5.68** Let  $G=(N, \Sigma, P, S)$  be a  $\varepsilon$ -free grammar. The simple precedence parser  $M$  for  $G$  is deterministic whenever the following conditions are satisfied:

(a)  $(\leq \cap \succ) = \emptyset$ .

(b)  $G$  is invertible, i.e., no two rules in  $G$  have identical right-hand sides.

(c)  $\forall X, Y, A \rightarrow \alpha X Y \beta, B \rightarrow Y \beta: X \prec Y$  is impossible.

If  $M$  is deterministic then all of the above conditions hold provided that  $G$  is reduced.

**Proof**  $M$  is nondeterministic iff

$M$  has one of pairs of distinct actions:

- (1)  $X \mid a \rightarrow Xa \mid, \quad Y \alpha X \mid a \rightarrow YA \mid a. \quad s-r$
- (2)  $X \omega \mid a \rightarrow XA \mid a, \quad X \omega \mid a \rightarrow XB \mid a. \quad r-r$
- (3)  $Z \alpha XY \beta \mid a \rightarrow ZA \mid a, \quad XY \beta \mid a \rightarrow XB \mid a. \quad r-r$

By definition,  $X (\leq \cap \succ) a$  in (1),  $X \triangleleft Y$  in (3)

Conversely,

(1) If  $\exists X, a: X (\leq \cap \succ) a; \exists X \mid a \rightarrow Xa \mid$  in  $M$ .

$A' \rightarrow \alpha' X' Y' \beta'$  in  $G'$ , where  $\exists \gamma, \delta: X' \xrightarrow[rm]{+} \gamma X, Y' \xrightarrow[lm]{*} a \delta$

$\therefore \exists A \rightarrow \alpha X, \exists Y: Y \leq A$  (L5.67),  $\therefore Y \triangleleft 1: \alpha X$ , (L5.62)

Then  $\exists Y \alpha X \mid a \rightarrow YA \mid a$  in  $M$ . (1)

(2) If  $G$  is not invertible;  $\exists A, B: A \rightarrow \omega, B \rightarrow \omega, A \neq B$ .

$\exists X \in V \cup \{\$, a \in \Sigma \cup \{\$\}: X \leq A (\leq \cup \succ) a,$

$\therefore X \triangleleft 1: \omega, \omega: 1 \succ a$  (L5.62). Then both of (2) in  $M$ .

(3) Let  $\exists A \rightarrow \alpha XY \beta, B \rightarrow Y \beta$  in  $G$ .

$\exists Z \in V \cup \{\$, a \in \Sigma \cup \{\$\}: Z \leq A (\leq \cup \succ) a,$  (L5.67)

$Z \triangleleft 1: \alpha XY \beta, \alpha XY \beta: 1 \succ a$  (L5.62)

$\therefore \exists Z \alpha XY \beta \mid a \rightarrow ZA \mid a$  in  $M$ .

Since  $Y \beta: 1 = \alpha XY \beta: 1$ ,

$\exists XY \beta \mid a \rightarrow XB \mid a$  in  $M$ , whenever  $X \triangleleft Y$ .

**Corollary 5.69** *A reduced  $\varepsilon$ -free grammar  $G$  is simple precedence iff the following conditions are satisfied:*

(a)  $(\leq \cap \triangleright) = \emptyset$ .

(b)  $G$  is invertible.

(c)  $\forall X, Y, A \rightarrow \alpha XY\beta, B \rightarrow Y\beta: X \triangleleft Y$  is impossible.

(d)  $S \stackrel{+}{\Rightarrow} S$  is impossible.

**Corollary 5.70** *An  $\varepsilon$ -free grammar  $G$  is simple precedence whenever the following conditions are satisfied:*

(a)  $(\leq \cap \triangleright) = (\triangleleft \cap \doteq) = \emptyset$ .

(b)  $G$  is invertible.

(c)  $S \stackrel{+}{\Rightarrow} S$  is impossible.

**Theorem 5.71** *Given any reduced  $\varepsilon$ -free grammar  $G = (N, \Sigma, P, S)$ , it is decidable in deterministic time  $O(|V| \cdot |G|)$  whether or not  $G$  is simple precedence.*