

1906002132015
Programlama Dilleri Temelleri

BAİBÜ Bilgisayar Müh.

Ders 12

Dr. Öğr. Üyesi İsmail Hakkı Parlak

Parsing (Ayrıştırma)

Top-Down Parsing (Tepeden Alta Ayrıştırma)

Start sembolü ile başlanır ve hedef dize (string) elde edilene kadar üretim kuralları (production rules) uygulanır.

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$

w , G içinde geçerli midir?

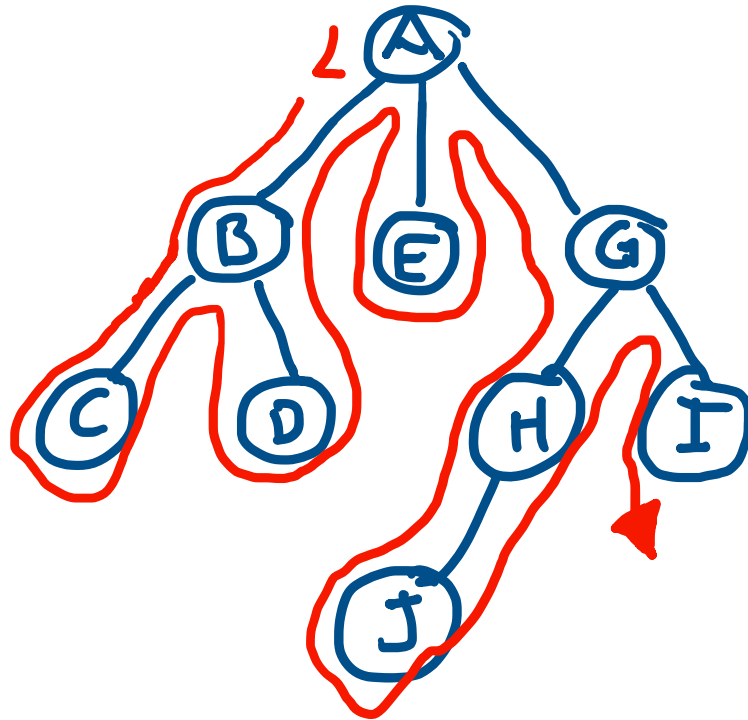
Tepeden Alta Ayırıştırma

- Tepeden alta ayırıştırmada soldan türetim (left-most derivation) uygulanır.
- Ayırıştırma ağacı (parse tree) pre-order (önden sıralı) şekilde oluşturulur.

```
def pre_order(node):  
    if node == None:  
        return  
  
    print(node.data)  
    pre_order(node.left)  
    pre_order(node.right)
```

- Başarısızlık durumunda geri izleme (backtracking) uygulanır.

Pre-order (Önden Sıralı) Gezme



A B C D E G H J I

Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

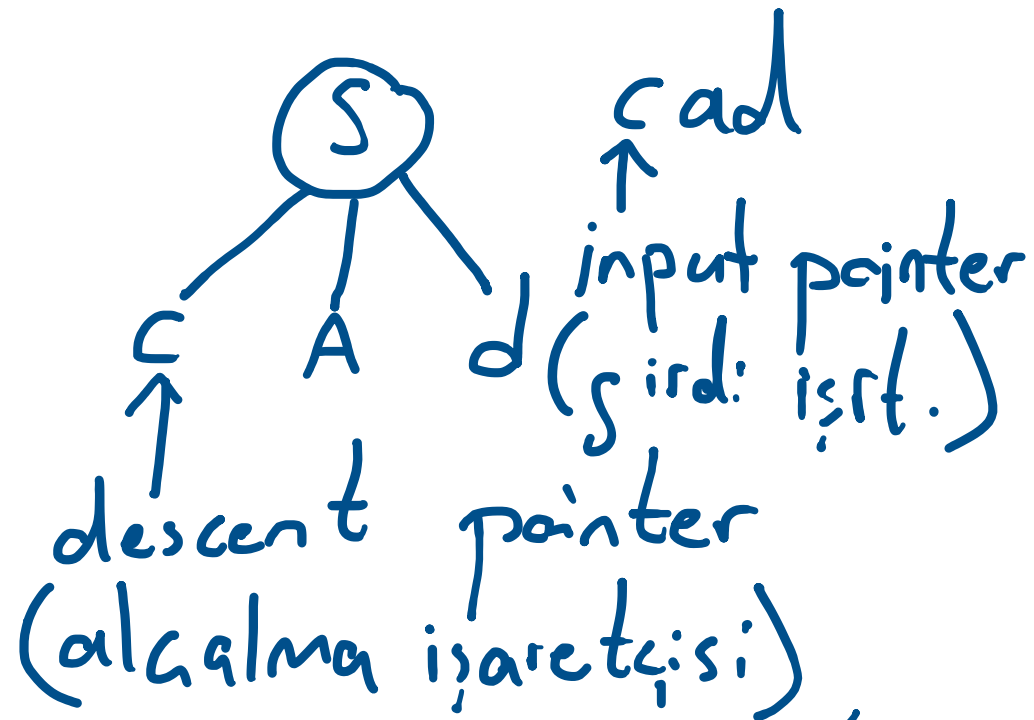
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$



input pointer == descent pointer ✓
match (eşleşme)

Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

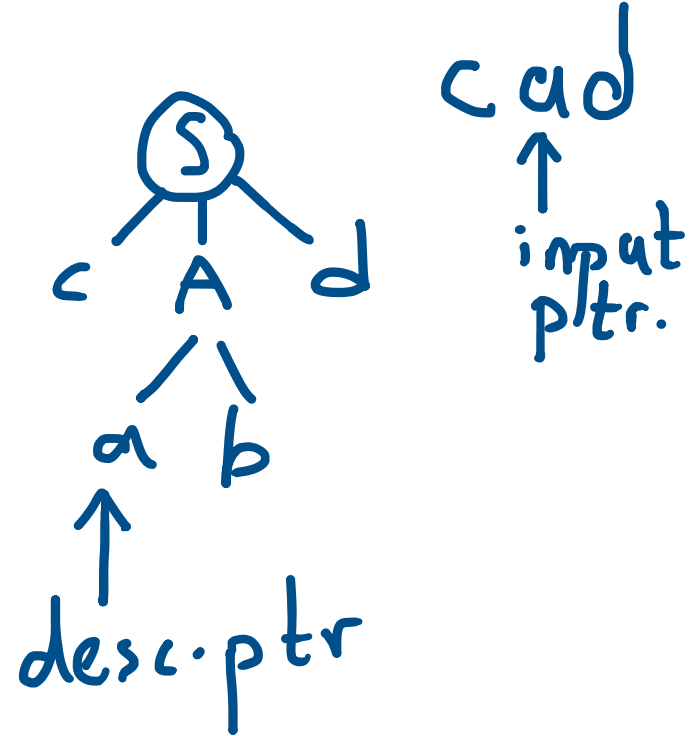
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$



$inp. ptr == desc. ptr \checkmark$
(eşleşme)

Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

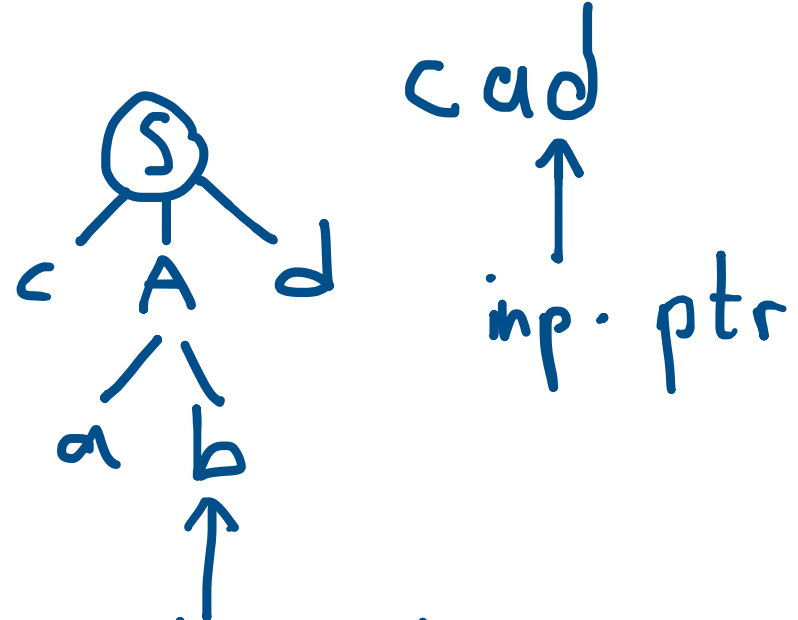
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$



inp ptr \neq desc ptr. X
mismatch!
 \rightarrow Backtrack

Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

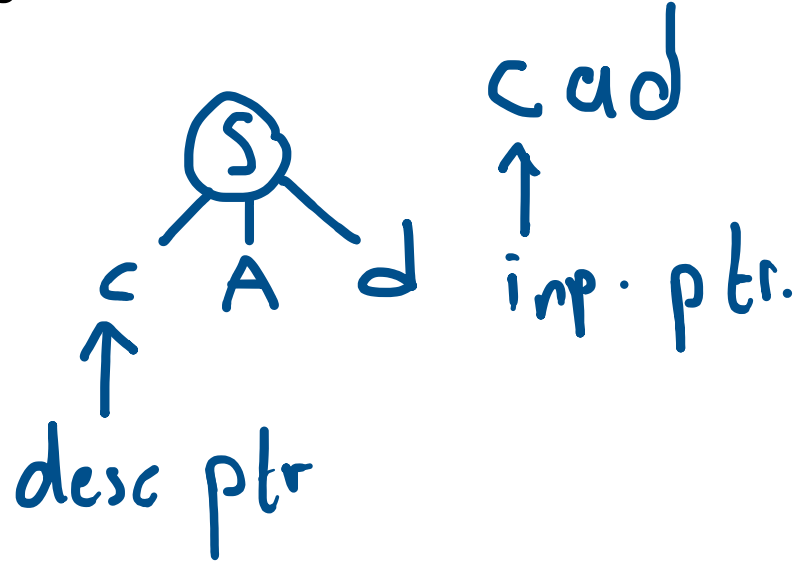
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$



Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

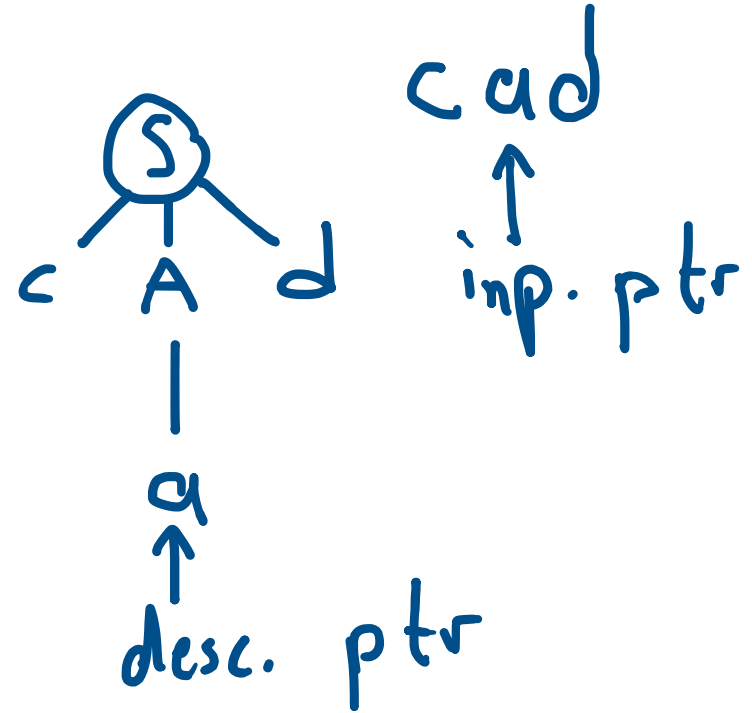
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$



inp ptr == desc ptr ✓
eşleşme

Tepeden Alta Ayırıştırma

$G = (\{S, A\}, \{a, b, c, d\}, R, S)$

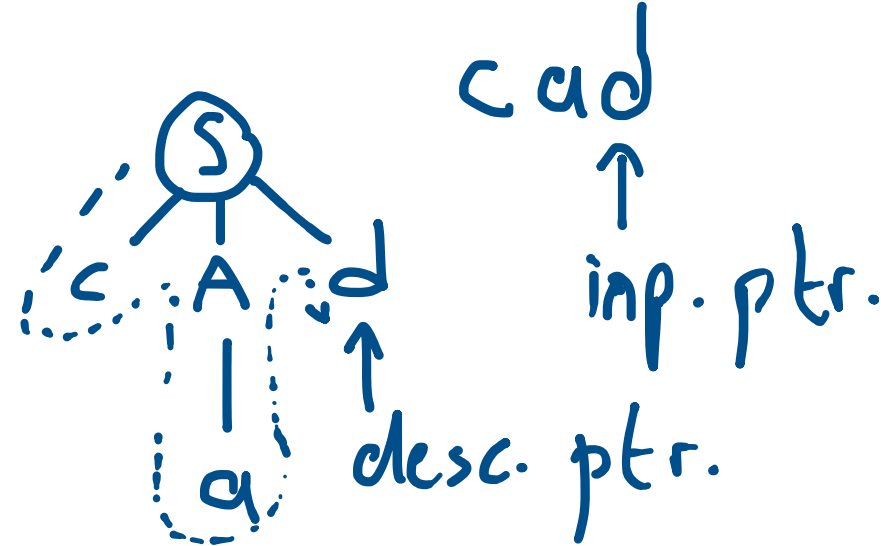
$R = \{$

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

$\}$

$w = cad$

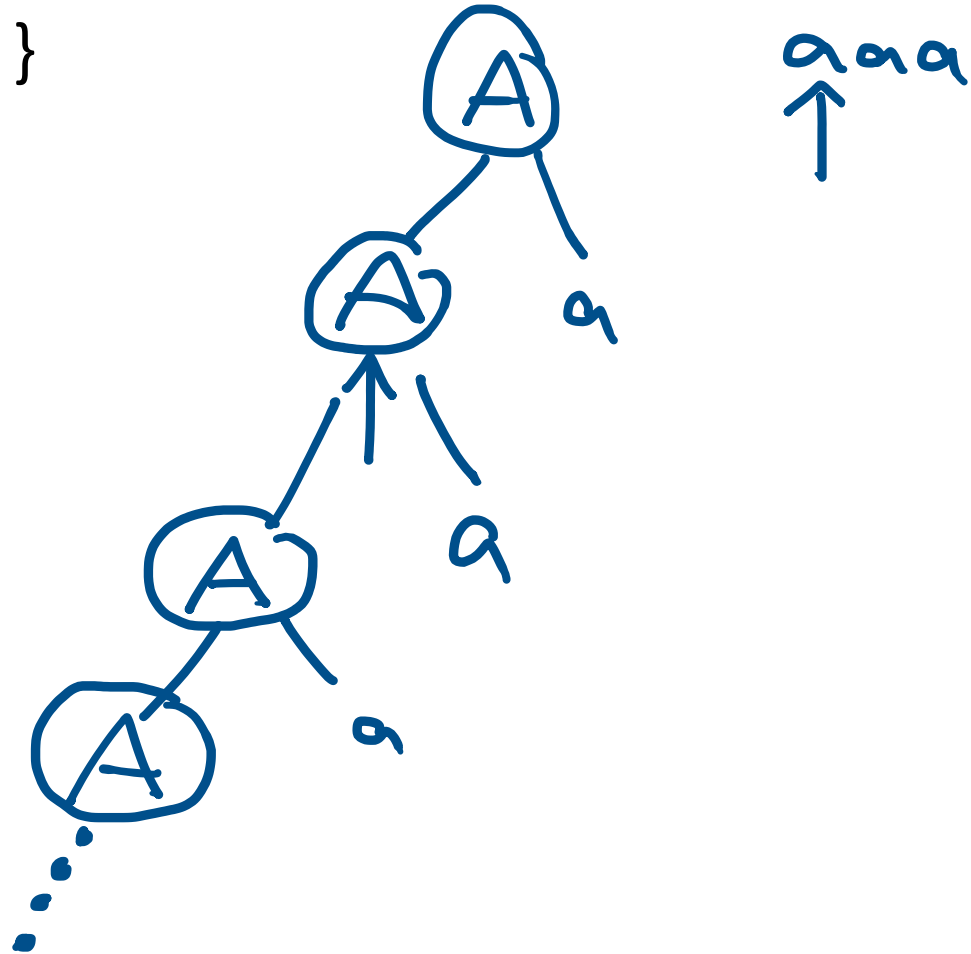


$inp\ ptr == desc\ ptr \quad \checkmark$
eşleşme

Recursive Descent Parser (Öz yineli alçalmalı ayırıştırma)

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

- $R = \{ A \rightarrow Aa \mid \varepsilon \}$
- Hedef aaa



Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

Kurtulacağımız kural:

$A \rightarrow A\alpha \mid \beta$ (β , A ile başlamayan herhangi bir terim.)

Kurtaran kurallar:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$A \rightarrow A\alpha \mid \beta$$

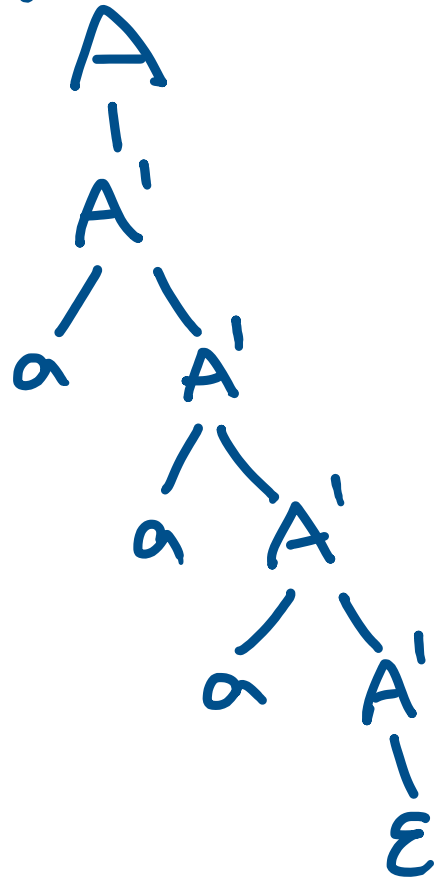
$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

$$\begin{array}{c} A \rightarrow A\alpha \mid \varepsilon \\ \quad \downarrow \quad \downarrow \\ \quad \alpha \quad \beta \\ A \rightarrow \varepsilon A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

hedef "aaaa"



$$A \rightarrow Aa \mid \epsilon$$

\downarrow \downarrow
 a β

$$A \rightarrow \epsilon A'$$

$$A' \rightarrow aA' \mid \epsilon$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$\begin{array}{l} A \rightarrow A\alpha \mid \beta \quad \times \\ A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \quad \checkmark$$

Örn:

$$\begin{array}{l} S \rightarrow ABC \quad \checkmark \\ A \rightarrow Aa \mid Ad \mid b \quad \times \\ B \rightarrow Bb \mid e \quad \times \\ C \rightarrow Cc \mid g \quad \times \end{array}$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$\begin{array}{l} A \rightarrow A\alpha \mid \beta \quad \times \\ A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \quad \checkmark$$

Örn:

$$\begin{array}{l} ? A \rightarrow Aa \mid Ad \mid b \\ A \rightarrow A\alpha \mid b \\ \quad \uparrow \quad \uparrow \\ \quad \alpha \quad \beta \end{array}$$

1 $\left\{ \begin{array}{l} A \rightarrow bA' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \right.$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$\begin{array}{l} A \rightarrow A\alpha \mid \beta \quad \times \\ A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \quad \checkmark$$

Örn:

$$? A \rightarrow Aa \mid Ad \mid b$$

$$A \rightarrow Ad \mid b$$

$\uparrow \quad \uparrow$
 $\alpha \quad \beta$

$$2 \left\{ \begin{array}{l} A \rightarrow bA' \\ A' \rightarrow dA' \mid \varepsilon \end{array} \right.$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$A \rightarrow A\alpha \mid \beta \quad \times$$
$$\left. \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \right\} \checkmark$$

$$A \rightarrow Aa \mid Ad \mid b$$

$$1 \left\{ \begin{array}{l} A \rightarrow bA' \\ A' \rightarrow aA' \mid \varepsilon \end{array} \right.$$

$$2 \left\{ \begin{array}{l} A \rightarrow bA' \\ A' \rightarrow dA' \mid \varepsilon \end{array} \right.$$

$$1 \cup 2 =$$

$$\boxed{\begin{array}{l} A \rightarrow bA' \\ A' \rightarrow aA' \mid dA' \mid \varepsilon \end{array}}$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$\begin{array}{l} A \rightarrow A\alpha \mid \beta \quad \times \\ A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \epsilon \end{array} \quad \checkmark$$

$$\begin{array}{l} B \rightarrow Bb \mid e \quad \times \\ \quad \uparrow \quad \uparrow \\ \quad \alpha \quad \beta \\ B \rightarrow e B' \\ B' \rightarrow b B' \mid \epsilon \quad \checkmark \end{array}$$

Sol Özyinelemeden Kurtulmak (Eliminating Left Recursion)

$$\begin{array}{l} A \rightarrow A\alpha \mid \beta \quad \times \\ A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \quad \checkmark$$

$$\begin{array}{l} C \rightarrow Cc \mid g \quad \times \\ \quad \uparrow \quad \uparrow \\ \quad \alpha \quad \beta \\ C \rightarrow gC' \\ C' \rightarrow cC' \mid \varepsilon \end{array} \quad \checkmark$$

Bottom-up parsing

(Alttan tepeye ayırıştırma)

- Shift-Reduce Parsing (Kaydırma-İndirgeme Ayırıştırması): Bir dizeyi (string) gramerin başlangıç sembolüne (S) dönüştürme işlemidir.
- Gramer için bir stack (yığın), ayırıştırılan string için bir girdi (input) kullanır.
- Amaç: string'i, başlangıç sembolüne indirgeme.
- 2 işlem uygulanabilir:
 - shift (kaydırma): girdi dizesindeki mevcut sembol bir stack'e push edilir.
 - reduce (indirgeme): stack'teki sembol bir non-terminal sembole dönüştürülür.

Shift Reduce Parser

- $S \rightarrow S + S$
- $S \rightarrow S - S$
- $S \rightarrow (S)$
- $S \rightarrow a \mid b \mid c$
- input: $a - (b + c)$

Stack	Input	İşlem
\$	a - (b + c) \$	shift a
\$ a	- (b + c) \$	reduce $S \rightarrow a$
\$ S	- (b + c) \$	shift -
\$ S -	(b + c) \$	shift (
\$ S - (b + c) \$	shift b
\$ S - (b	+ c) \$	reduce $S \rightarrow b$
\$ S - (S	+ c) \$	shift +
\$ S - (S +	c) \$	shift c
\$ S - (S + c) \$	reduce $S \rightarrow c$
\$ S - (S + S) \$	shift)
\$ S - (S + S)	\$	reduce $S \rightarrow S + S$
\$ S - (S)	\$	reduce $S \rightarrow (S)$
\$ S - S	\$	reduce $S \rightarrow S - S$
\$ S	\$	accept