6th lecture overview

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Lecture overview

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G = (\{V, B\}, \{s, b\}, P, V)$$

$$V \rightarrow sB$$

$$B \rightarrow sB \qquad B \rightarrow bB \qquad B \rightarrow \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

$$V \to sB$$

$$B \to sB \qquad B \to bB \qquad B \to \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G = (\{V, B\}, \{s, b\}, P, V)$$

V

B

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G = (\{V, B\}, \{s, b\}, P, V)$$

V

B

$V \rightarrow sB$

$B \rightarrow sB \qquad B \rightarrow bB \qquad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b
V		
B		

$V \rightarrow sB$

$B \rightarrow sB \qquad B \rightarrow bB \qquad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$G=(\{V, B\}, \{s, b\}, P, V)$

	s	b
V		
B		

$V \rightarrow sB$

$B \rightarrow sB$ $\quad B \rightarrow bB$ $\quad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$G=(\{V, B\}, \{s, b\}, P, V)$

	s	b
V	B	
B		

$V \rightarrow sB$

$B \rightarrow sB \qquad B \rightarrow bB \qquad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b
V	B	$\emptyset$
B		

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\varnothing$	0
B			

$V \rightarrow sB$

$B \rightarrow sB$ $\quad B \rightarrow bB$ $\quad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G = (\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\emptyset$	0
B			

$V \to sB$

$B \to sB$ $\quad$ $B \to bB$ $\quad$ $B \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G = (\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\varnothing$	0
B	B		

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$G = (\{V, B\}, \{s, b\}, P, V)$

	s	b	
V	B	$\varnothing$	0
B	B		

$V \rightarrow sB$

$B \rightarrow sB$ $\quad B \rightarrow bB$ $\quad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\varnothing$	0
B	B	B	

$V \to sB$

$B \to sB \quad B \to bB \quad B \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\emptyset$	0
B	B	B	

$V \rightarrow sB$

$B \rightarrow sB$ $\quad B \rightarrow bB$ $\quad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$G=(\{V, B\}, \{s, b\}, P, V)$

	s	b	
V	B	$\emptyset$	0
B	B	B	1

$V \rightarrow sB$

$B \rightarrow sB \qquad B \rightarrow bB \qquad B \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\emptyset$	0
B	B	B	1

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

If the grammar is extended by production

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	∅	0
B	B	B	1

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

If the grammar is extended by production

$$B \to bV$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

$$G=(\{V, B\}, \{s, b\}, P, V)$$

	s	b	
V	B	$\emptyset$	0
B	B	B,V	1

$V \to sB$

$B \to sB \qquad B \to bB \qquad B \to \varepsilon$

If the grammar is extended by production

$$B \to bV$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$$A \to aB$$

$$A \to \varepsilon$$

1) $\Sigma = T$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

---

1) $\Sigma = T$

2) $Q = V$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

---

1) $\Sigma = T$

2) $Q = V$

3) $q_0 = S$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$$A \to aB$$
$$A \to \varepsilon$$

---

1) $\Sigma = T$

2) $Q = V$

3) $q_0 = S$

4) $\delta(A, a) = \delta(A, a) \cup \{B\}$ $\qquad A \to aB$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

$$A \rightarrow aB$$
$$A \rightarrow aC$$

$$\delta(A, a) = \{B, C\}$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA for a regular language given by a simple grammar

NFA $M=(Q, \Sigma, \delta, q_0, F)$

$G=(V, T, P, S)$

$A \to aB$

$A \to \varepsilon$

---

1) $\Sigma = T$

2) $Q = V$

3) $q_0 = S$

4) $\delta(A, a) = \delta(A, a) \cup \{B\}$ $\quad A \to aB$

5) $A \in F$ $\quad A \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \rightarrow wB$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \rightarrow wB$$
$$A \rightarrow w$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \rightarrow wB$$
$$A \rightarrow w$$

Left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \rightarrow wB$$
$$A \rightarrow w$$

Left-linear grammar

Productions have at most one variable at the **leftmost** place on the right-hand side

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \to wB$$
$$A \to w$$

Left-linear grammar

Productions have at most one variable at the **leftmost** place on the right-hand side

$$A \to Bw$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Right-linear grammar

Productions have at most one variable at the **rightmost** place on the right-hand side

$$A \to wB$$
$$A \to w$$

Left-linear grammar

Productions have at most one variable at the **leftmost** place on the right-hand side

$$A \to Bw$$
$$A \to w$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

0 (10)*

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

0 (10)*

Right-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

$0\ (10)^*$

Right-linear grammar

$S \rightarrow 0\ A$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

0 (10)*

Right-linear grammar

$$S \rightarrow 0\,A$$
$$A \rightarrow 10\,A$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

$0\,(10)^*$

Right-linear grammar

$$S \rightarrow 0\,A$$
$$A \rightarrow 10\,A$$
$$A \rightarrow \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

0 (10)*

Right-linear grammar

$$S \rightarrow 0\,A$$
$$A \rightarrow 10\,A$$
$$A \rightarrow \varepsilon$$

Left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

0 (10)*

Right-linear grammar

$$S \rightarrow 0\,A$$
$$A \rightarrow 10\,A$$
$$A \rightarrow \varepsilon$$

Left-linear grammar

$$S \rightarrow S\,10$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Right-linear and left-linear grammar

$$0\,(10)^*$$

Right-linear grammar

$$S \rightarrow 0\,A$$
$$A \rightarrow 10\,A$$
$$A \rightarrow \varepsilon$$

Left-linear grammar

$$S \rightarrow S\,10$$
$$S \rightarrow 0$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

$A \rightarrow a[bbS]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

$A \rightarrow a[bbS]$

$[bbS] \rightarrow b[bS]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

$$A \rightarrow a[bbS]$$
$$[bbS] \rightarrow b[bS]$$
$$[bS] \rightarrow bS$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

$$
\begin{aligned}
A &\rightarrow a[bbS] \\
[bbS] &\rightarrow b[bS] \\
[bS] &\rightarrow bS
\end{aligned}
$$

$A \Rightarrow abbS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

4) $A \rightarrow abbS$

$$
\begin{aligned}
A &\rightarrow a[bbS] \\
[bbS] &\rightarrow b[bS] \\
[bS] &\rightarrow bS
\end{aligned}
$$

$A \Rightarrow abbS$

$A \Rightarrow a[bbS] \Rightarrow ab[bS] \Rightarrow abbS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

2) $S \to bc$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

2) $S \to bc$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

2) $S \to bc$

$S \to bc[\varepsilon]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

2) $S \rightarrow bc$

$$S \rightarrow bc[\varepsilon]$$
$$[\varepsilon] \rightarrow \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

2) $S \rightarrow bc$

$S \rightarrow bc[\varepsilon]$

$[\varepsilon] \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

2) $S \rightarrow bc$

$S \rightarrow bc[\varepsilon]$
$[\varepsilon] \rightarrow \varepsilon$

$S \rightarrow b[c\varepsilon]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

2) $S \rightarrow bc$

$S \rightarrow bc[\varepsilon]$
$[\varepsilon] \rightarrow \varepsilon$

$S \rightarrow b[c\varepsilon]$
$[c\varepsilon] \rightarrow c[\varepsilon]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

2) $S \to bc$

$[\varepsilon] \to \varepsilon$

$S \to b[c\varepsilon]$
$[c\varepsilon] \to c[\varepsilon]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

2) $S \rightarrow bc$

$[\varepsilon] \rightarrow \varepsilon$

$S \rightarrow b[c\varepsilon]$
$[c\varepsilon] \rightarrow c[\varepsilon]$

$S \Rightarrow bc$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

2) $S \to bc$

$[\varepsilon] \to \varepsilon$

$S \to b[c\varepsilon]$
$[c\varepsilon] \to c[\varepsilon]$

$S \Rightarrow bc$

$S \Rightarrow b[c\varepsilon] \Rightarrow bc[\varepsilon] \Rightarrow bc$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	4) $A \to abbS$
2) $S \to bc$	5) $A \to cA$
3) $S \to A$	6) $A \to \varepsilon$

$A \to aB$

$A \to \varepsilon$

3) $S \to A$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow \boxed{A}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow A$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow A$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow A$

$S \rightarrow cA$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow A$

$S \rightarrow cA$

$S \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	4) $A \rightarrow abbS$
2) $S \rightarrow bc$	5) $A \rightarrow cA$
3) $S \rightarrow A$	6) $A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow \varepsilon$

3) $S \rightarrow A$

$S \rightarrow cA$
$S \rightarrow \varepsilon$
$S \rightarrow a[bbS]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$
5) $A \rightarrow cA$
6) $A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

2) $S \rightarrow bc$

1) $S \rightarrow aA$

5) $A \rightarrow cA$

6) $A \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$\boxed{2)\ S \rightarrow bc}$

1) $S \rightarrow aA$	2*a*) $S \rightarrow b[c\varepsilon]$
5) $A \rightarrow cA$	2*b*) $[c\varepsilon] \rightarrow c[\varepsilon]$
6) $A \rightarrow \varepsilon$	2*c*) $[\varepsilon] \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	2*a*) $S \to b[c\varepsilon]$
5) $A \to cA$	2*b*) $[c\varepsilon] \to c[\varepsilon]$
6) $A \to \varepsilon$	2*c*) $[\varepsilon] \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

3) $S \to A$

1) $S \to aA$	2a) $S \to b[c\varepsilon]$
5) $A \to cA$	2b) $[c\varepsilon] \to c[\varepsilon]$
6) $A \to \varepsilon$	2c) $[\varepsilon] \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

3) $S \to A$

1) $S \to aA$	2*a*) $S \to b[c\varepsilon]$	3*a*) $S \to a[bbS]$
5) $A \to cA$	2*b*) $[c\varepsilon] \to c[\varepsilon]$	3*b*) $S \to cA$
6) $A \to \varepsilon$	2*c*) $[\varepsilon] \to \varepsilon$	3*c*) $S \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	2*a*) $S \rightarrow b[c\varepsilon]$	3*a*) $S \rightarrow a[bbS]$
5) $A \rightarrow cA$	2*b*) $[c\varepsilon] \rightarrow c[\varepsilon]$	3*b*) $S \rightarrow cA$
6) $A \rightarrow \varepsilon$	2*c*) $[\varepsilon] \rightarrow \varepsilon$	3*c*) $S \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

4) $A \rightarrow abbS$

1) $S \rightarrow aA$	2*a*) $S \rightarrow b[c\varepsilon]$	3*a*) $S \rightarrow a[bbS]$
5) $A \rightarrow cA$	2*b*) $[c\varepsilon] \rightarrow c[\varepsilon]$	3*b*) $S \rightarrow cA$
6) $A \rightarrow \varepsilon$	2*c*) $[\varepsilon] \rightarrow \varepsilon$	3*c*) $S \rightarrow \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

4) $A \to abbS$

1) $S \to aA$	2*a*) $S \to b[c\varepsilon]$	3*a*) $S \to a[bbS]$	4*a*) $A \to a[bbS]$
5) $A \to cA$	2*b*) $[c\varepsilon] \to c[\varepsilon]$	3*b*) $S \to cA$	4*b*) $[bbS] \to b[bS]$
6) $A \to \varepsilon$	2*c*) $[\varepsilon] \to \varepsilon$	3*c*) $S \to \varepsilon$	4*c*) $[bS] \to bS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \rightarrow aA$	2*a*) $S \rightarrow b[c\varepsilon]$	3*a*) $S \rightarrow a[bbS]$	4*a*) $A \rightarrow a[bbS]$
5) $A \rightarrow cA$	2*b*) $[c\varepsilon] \rightarrow c[\varepsilon]$	3*b*) $S \rightarrow cA$	4*b*) $[bbS] \rightarrow b[bS]$
6) $A \rightarrow \varepsilon$	2*c*) $[\varepsilon] \rightarrow \varepsilon$	3*c*) $S \rightarrow \varepsilon$	4*c*) $[bS] \rightarrow bS$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

1) $S \to aA$	2a) $S \to b[c\varepsilon]$	3a) $S \to a[bbS]$	4a) $A \to a[bbS]$
5) $A \to cA$	2b) $[c\varepsilon] \to c[\varepsilon]$	3b) $S \to cA$	4b) $[bbS] \to b[bS]$
6) $A \to \varepsilon$	2c) $[\varepsilon] \to \varepsilon$	3c) $S \to \varepsilon$	4c) $[bS] \to bS$

	a	*b*	*c*	
S	*A*, [*bbS*]	[*c*ε]	*A*	1
[*c*ε]			[ε]	0
[ε]				1
A	[*bbS*]		*A*	1
[*bbS*]		[*bS*]		0
[*bS*]		*S*		0

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

---

1) $[\varepsilon] \rightarrow \varepsilon$ $\qquad\qquad$ $A \rightarrow w$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$ $A \to \varepsilon$	$A \to wB$ $A \to w$

1) $[\varepsilon] \to \varepsilon$ $\quad$ $A \to w$

$A \to w[\varepsilon]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

1) $[\varepsilon] \rightarrow \varepsilon$
$A \rightarrow w[\varepsilon]$ $\qquad$ $A \rightarrow w$

2) $A \rightarrow a_1\ [a_2 \ldots\ a_n\ B]$ $\qquad$ $A \rightarrow a_1 \ldots\ a_n\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

1) $[\varepsilon] \rightarrow \varepsilon$ $\quad$ $A \rightarrow w$

$A \rightarrow w[\varepsilon]$

2) $A \rightarrow a_1\ [a_2 \ldots\ a_n\ B]$ $\quad$ $A \rightarrow a_1 \ldots a_n\ B$

$[a_2 \ldots\ a_n\ B] \rightarrow a_2\ [a_3 \ldots\ a_n\ B]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$ $A \to \varepsilon$	$A \to wB$ $A \to w$

1) $[\varepsilon] \to \varepsilon$
$A \to w[\varepsilon]$ $\qquad$ $A \to w$

2) $A \to a_1\ [a_2 \ldots\ a_n\ B]$ $\qquad$ $A \to a_1\ \ldots\ a_n\ B$
$[a_2 \ldots\ a_n\ B] \to a_2\ [a_3 \ldots\ a_n\ B]$
$[a_3 \ldots\ a_n\ B] \to a_3\ [a_4 \ldots\ a_n\ B]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$ $A \to \varepsilon$	$A \to wB$ $A \to w$

1) $[\varepsilon] \to \varepsilon$
$A \to w[\varepsilon]$ — $A \to w$

2) $A \to a_1\ [a_2 \ldots\ a_n\ B]$
$[a_2 \ldots\ a_n\ B] \to a_2\ [a_3 \ldots\ a_n\ B]$
$[a_3 \ldots\ a_n\ B] \to a_3\ [a_4 \ldots\ a_n\ B]$
\- - - — $A \to a_1\ \ldots\ a_n\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$ $A \to \varepsilon$	$A \to wB$ $A \to w$

1) $[\varepsilon] \to \varepsilon$
$A \to w[\varepsilon]$ — $A \to w$

2) $A \to a_1\, [a_2 \ldots a_n\, B]$ — $A \to a_1 \ldots a_n\, B$
$[a_2 \ldots a_n\, B] \to a_2\, [a_3 \ldots a_n\, B]$
$[a_3 \ldots a_n\, B] \to a_3\, [a_4 \ldots a_n\, B]$
- - -
$[a_i \ldots a_n\, B] \to a_i\, [a_{i+1} \ldots a_n\, B]$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$
$A \to \varepsilon$

$A \to wB$
$A \to w$

---

1) $[\varepsilon] \to \varepsilon$
$A \to w[\varepsilon]$

$A \to w$

2) $A \to a_1\ [a_2 \ldots\ a_n\ B]$
$[a_2 \ldots\ a_n\ B] \to a_2\ [a_3 \ldots\ a_n\ B]$
$[a_3 \ldots\ a_n\ B] \to a_3\ [a_4 \ldots\ a_n\ B]$
\- - -
$[a_i \ldots\ a_n\ B] \to a_i\ [a_{i+1} \ldots\ a_n\ B]$
\- - -

$A \to a_1\ \ldots\ a_n\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \to aB$ $A \to \varepsilon$	$A \to wB$ $A \to w$

1) $[\varepsilon] \to \varepsilon$
$A \to w[\varepsilon]$

$A \to w$

2) $A \to a_1\ [a_2 \ldots\ a_n\ B]$
$[a_2 \ldots\ a_n\ B] \to a_2\ [a_3 \ldots\ a_n\ B]$
$[a_3 \ldots\ a_n\ B] \to a_3\ [a_4 \ldots\ a_n\ B]$
\- - -
$[a_i \ldots\ a_n\ B] \to a_i\ [a_{i+1} \ldots\ a_n\ B]$
\- - -
$[a_{n-1} a_n\ B] \to a_{n-1}\ [a_n\ B]$

$A \to a_1 \ldots\ a_n\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

---

1) $[\varepsilon] \rightarrow \varepsilon$
$A \rightarrow w[\varepsilon]$ — $A \rightarrow w$

2) $A \rightarrow a_1\ [a_2 \ldots\ a_n\ B]$ — $A \rightarrow a_1 \ldots a_n\ B$

$[a_2 \ldots\ a_n\ B] \rightarrow a_2\ [a_3 \ldots\ a_n\ B]$

$[a_3 \ldots\ a_n\ B] \rightarrow a_3\ [a_4 \ldots\ a_n\ B]$

\- - -

$[a_i \ldots\ a_n\ B] \rightarrow a_i\ [a_{i+1} \ldots\ a_n\ B]$

\- - -

$[a_{n-1} a_n\ B] \rightarrow a_{n-1}\ [a_n\ B]$

$[a_n\ B] \rightarrow a_n\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an NFA from a right-linear grammar

$A \rightarrow aB$ $A \rightarrow \varepsilon$	$A \rightarrow wB$ $A \rightarrow w$

---

1)	$[\varepsilon] \rightarrow \varepsilon$ $A \rightarrow w[\varepsilon]$	$A \rightarrow w$
2)	$A \rightarrow a_1\ [a_2 \ldots\ a_n\ B]$ $[a_2 \ldots\ a_n\ B] \rightarrow a_2\ [a_3 \ldots\ a_n\ B]$ $[a_3 \ldots\ a_n\ B] \rightarrow a_3\ [a_4 \ldots\ a_n\ B]$ - - - $[a_i \ldots\ a_n\ B] \rightarrow a_i\ [a_{i+1} \ldots\ a_n\ B]$ - - - $[a_{n-1} a_n\ B] \rightarrow a_{n-1}\ [a_n\ B]$ $[a_n\ B] \rightarrow a_n\ B$	$A \rightarrow a_1 \ldots\ a_n\ B$
3)	$A \rightarrow y$	$A \rightarrow B,\ B \rightarrow y$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \rightarrow Bw$ $A \rightarrow w$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \to Bw$ $A \to w$

1) We construct a right-linear grammar $G'=(V, T, P', S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \rightarrow Bw$ $A \rightarrow w$

1) We construct a right-linear grammar $G' = (V, T, P', S)$

$$P' = \{ A \rightarrow \alpha^R \mid A \rightarrow \alpha \text{ belongs to } P \}$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \rightarrow Bw$ $A \rightarrow w$

1) We construct a right-linear grammar $G' = (V, T, P', S)$

$$P' = \{ A \rightarrow \alpha^R \mid A \rightarrow \alpha \text{ belongs to } P \}$$
$$L(G') = L(G)^R$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \rightarrow Bw$ $A \rightarrow w$

1) We construct a right-linear grammar $G' = (V, T, P', S)$

$$P' = \{ A \rightarrow \alpha^R \mid A \rightarrow \alpha \text{ belongs to } P \}$$
$$L(G') = L(G)^R$$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA

$A \to Bw$
$A \to w$

---

1) We construct a right-linear grammar $G' = (V, T, P', S)$

$P' = \{ A \to \alpha^R \mid A \to \alpha$ belongs to $P \}$
$L(G') = L(G)^R$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

3) We construct an ε-NFA M' which accepts $L(M') = L(M)^R = L(G')^R = L(G)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \to Bw$ $A \to w$

1) We construct a right-linear grammar $G'=(V, T, P', S)$

$P' = \{ A \to \alpha^R \mid A \to \alpha$ belongs to $P \}$
$L(G') = L(G)^R$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

3) We construct an ε-NFA M' which accepts $L(M') = L(M)^R = L(G')^R = L(G)$

- NFA M is rearranged to have a single accepting state

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \to Bw$ $A \to w$

1) We construct a right-linear grammar $G'=(V, T, P', S)$

$P' = \{ A \to \alpha^R \mid A \to \alpha$ belongs to $P \}$
$L(G') = L(G)^R$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

3) We construct an ε-NFA M' which accepts $L(M') = L(M)^R = L(G')^R = L(G)$

- NFA M is rearranged to have a single accepting state
- initial state ε-NFA M' = accepting state NFA M

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA	$A \rightarrow Bw$ $A \rightarrow w$

1) We construct a right-linear grammar $G' = (V, T, P', S)$

$P' = \{ A \rightarrow \alpha^R \mid A \rightarrow \alpha$ belongs to $P \}$
$L(G') = L(G)^R$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

3) We construct an ε-NFA M' which accepts $L(M') = L(M)^R = L(G')^R = L(G)$

- NFA M is rearranged to have a single accepting state
- initial state ε-NFA M' = accepting state NFA M
- accepting state ε-NFA M' = initial state NFA M

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

ε-NFA

$A \rightarrow Bw$
$A \rightarrow w$

---

1) We construct a right-linear grammar $G'=(V, T, P', S)$

$P' = \{ A \rightarrow \alpha^R \mid A \rightarrow \alpha \text{ belongs to } P \}$
$L(G') = L(G)^R$

2) We construct an NFA M which accepts $L(M) = L(G') = L(G)^R$

3) We construct an ε-NFA M' which accepts $L(M') = L(M)^R = L(G')^R = L(G)$

- NFA M is rearranged to have a single accepting state
- initial state ε-NFA M' = accepting state NFA M
- accepting state ε-NFA M' = initial state NFA M
- transitions NFA M' = reversed transitions of NFA M

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L = (\{S\}, \{0, 1\}, \{S \rightarrow S\ 10 \mid 0\}, S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$

Generates the language $0(10)^*$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S\to S\ 10 \mid 0\}, S)$

Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S\to 01\ S \mid 0\}, S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\ 10 \mid 0\}, S)$

Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\ S \mid 0\}, S)$

Generates the language $(01)^*0$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \to S\ 10 \mid 0\}, S)$

Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \to 01\ S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \to aB, \quad A \to \varepsilon$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S\to S\ 10 \mid 0\}, S)$

Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S\to 01\ S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \to aB, \quad A \to \varepsilon$

$S \to 0\ [1S] \mid 0\ [\varepsilon], \qquad [1S] \to 1S, \qquad [\varepsilon] \to \varepsilon$

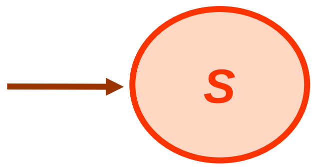
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar G_L=({S}, {0, 1}, {$S \rightarrow S$ 10 | 0}, S)
Generates the language 0(10)*

We construct a right-linear grammar G_D=({S}, {0, 1}, {$S \rightarrow$ 01 S | 0}, S)
Generates the language (01)*0

We rearrange the grammar to the form $A \rightarrow aB$, $A \rightarrow \varepsilon$
$S \rightarrow 0\ [1S]\ |\ 0\ [\varepsilon]$, $[1S] \rightarrow 1S$, $[\varepsilon] \rightarrow \varepsilon$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$

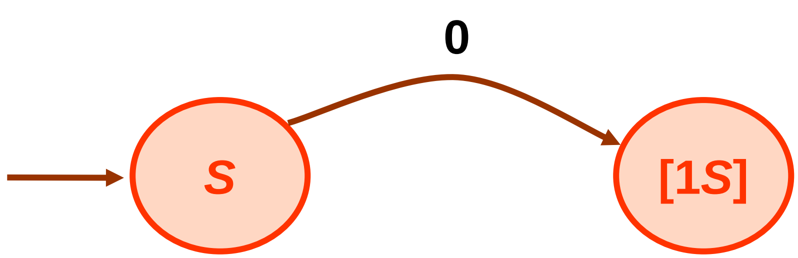
Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB, \quad A \rightarrow \varepsilon$

$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon], \qquad [1S] \rightarrow 1S, \qquad [\varepsilon] \rightarrow \varepsilon$



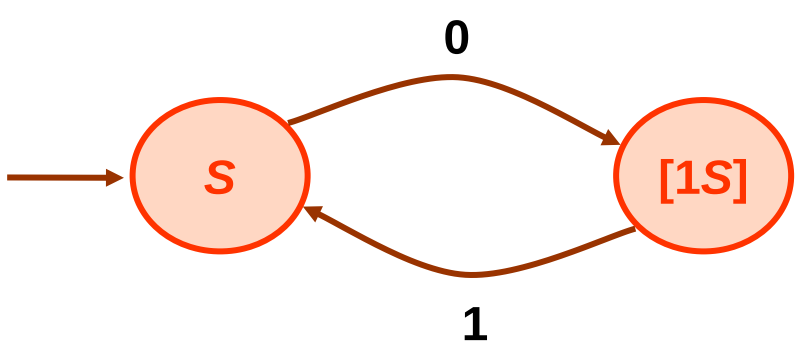
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$
Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$
Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB$, $A \rightarrow \varepsilon$
$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon]$, $[1S] \rightarrow 1S$, $[\varepsilon] \rightarrow \varepsilon$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \to S\,10 \mid 0\}, S)$

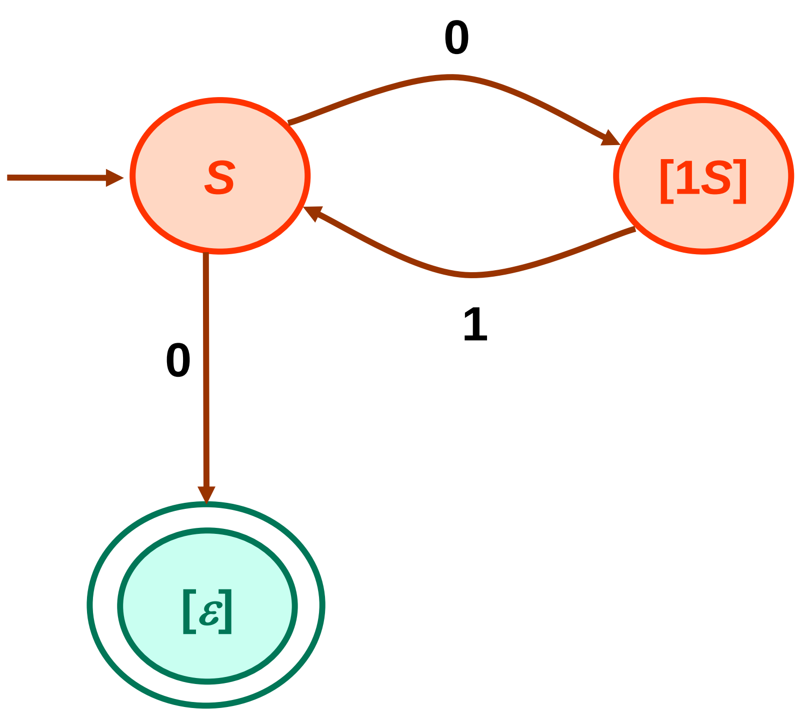
Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \to 01\,S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \to aB, \quad A \to \varepsilon$

$S \to 0\,[1S] \mid 0\,[\varepsilon], \qquad [1S] \to 1S, \qquad [\varepsilon] \to \varepsilon$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$

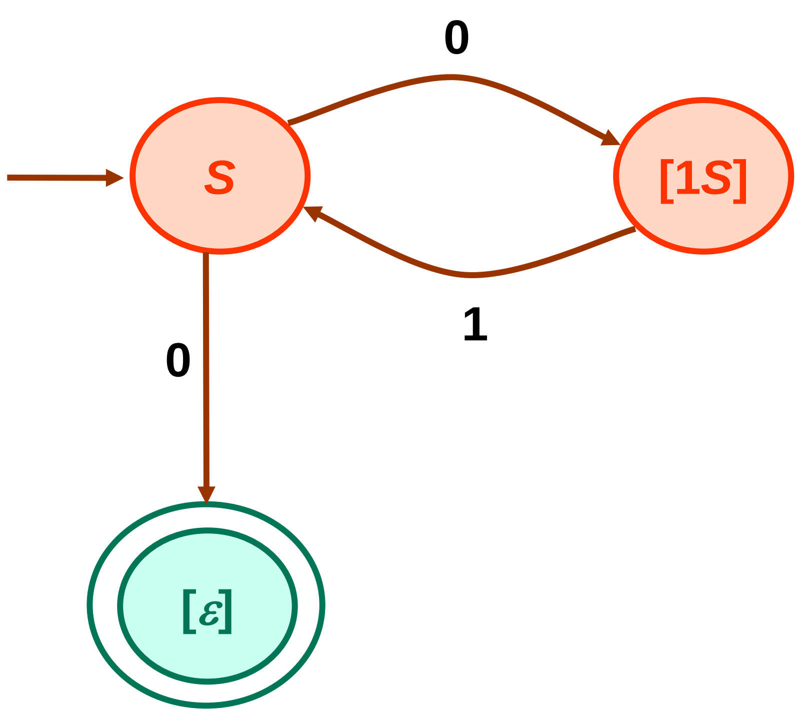
Generates the language $0(10)^*$

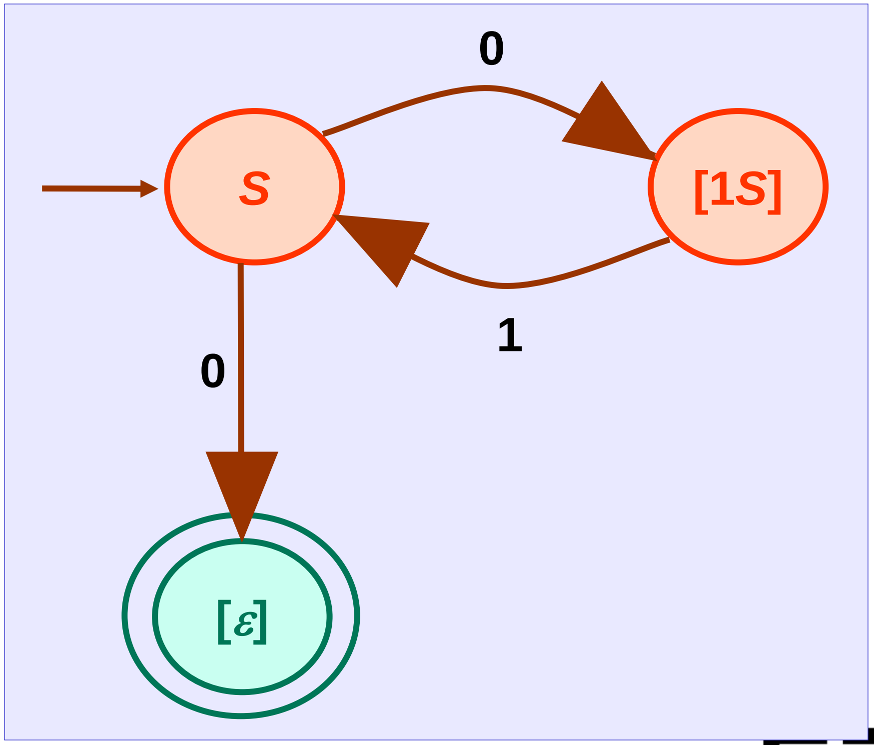
We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB, \quad A \rightarrow \varepsilon$

$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon], \qquad [1S] \rightarrow 1S, \qquad [\varepsilon] \rightarrow \varepsilon$





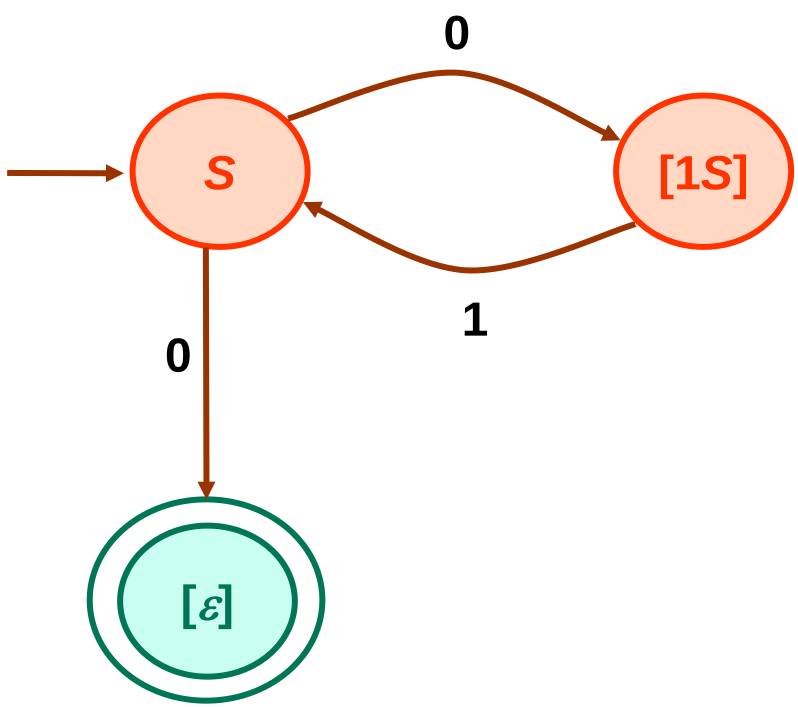
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

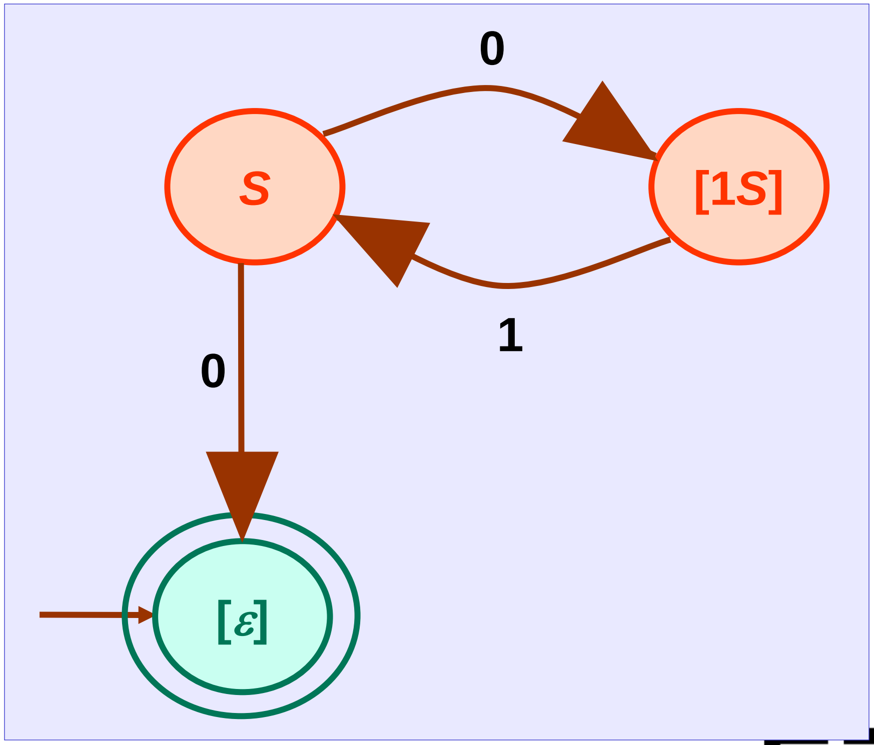
Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \to S\,10 \mid 0\}, S)$
Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \to 01\,S \mid 0\}, S)$
Generates the language $(01)^*0$

We rearrange the grammar to the form $A \to aB$, $A \to \varepsilon$
$S \to 0\,[1S] \mid 0\,[\varepsilon]$, $[1S] \to 1S$, $[\varepsilon] \to \varepsilon$





Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$

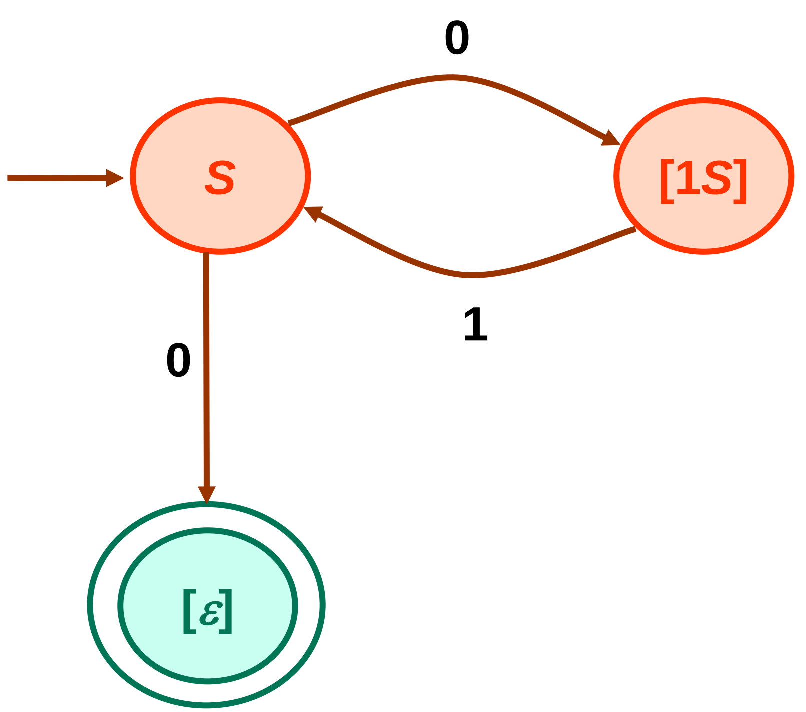
Generates the language $0(10)^*$

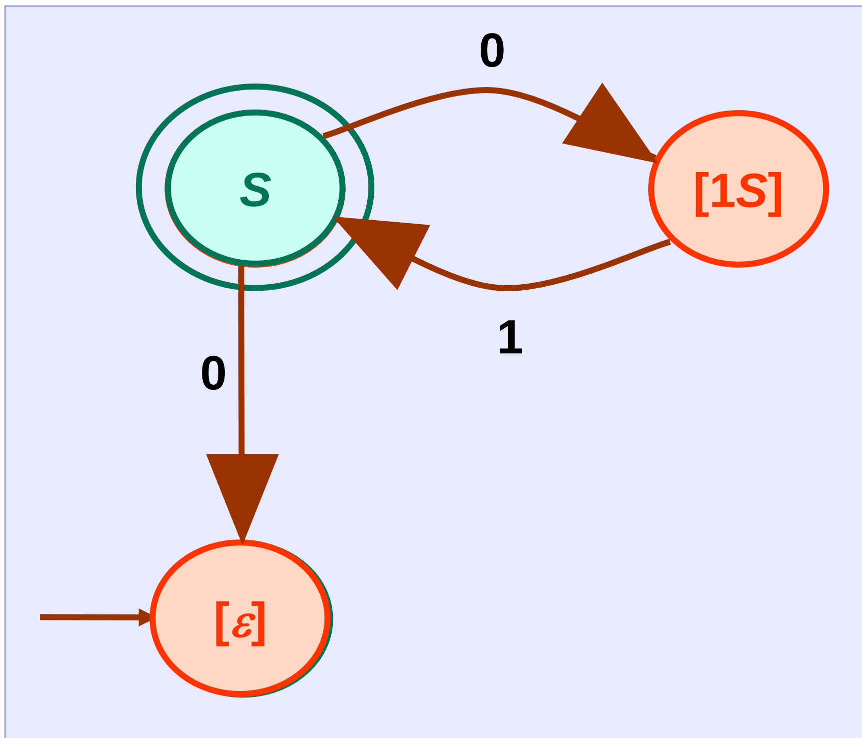
We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB$, $A \rightarrow \varepsilon$

$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon]$, $[1S] \rightarrow 1S$, $[\varepsilon] \rightarrow \varepsilon$





Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$

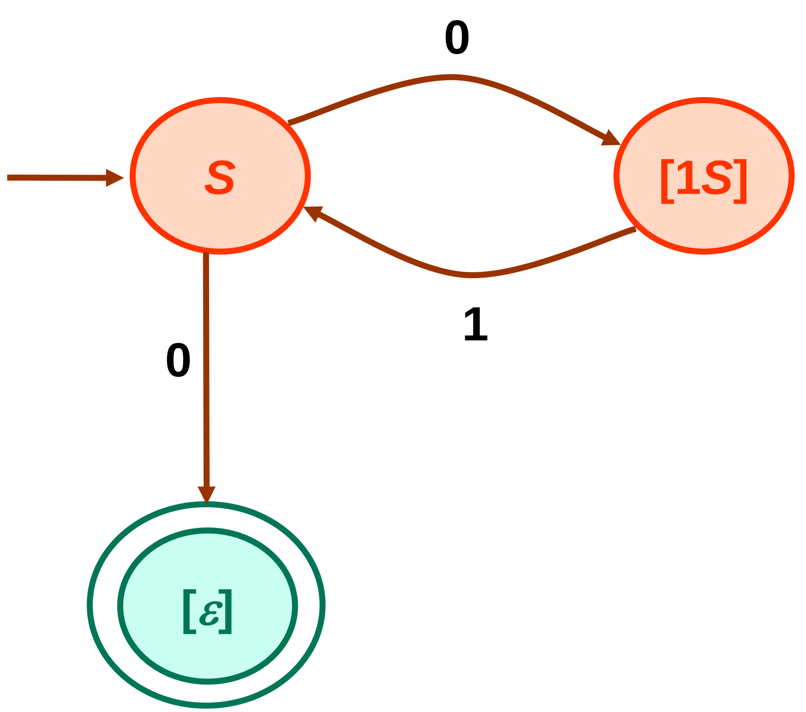
Generates the language $0(10)^*$

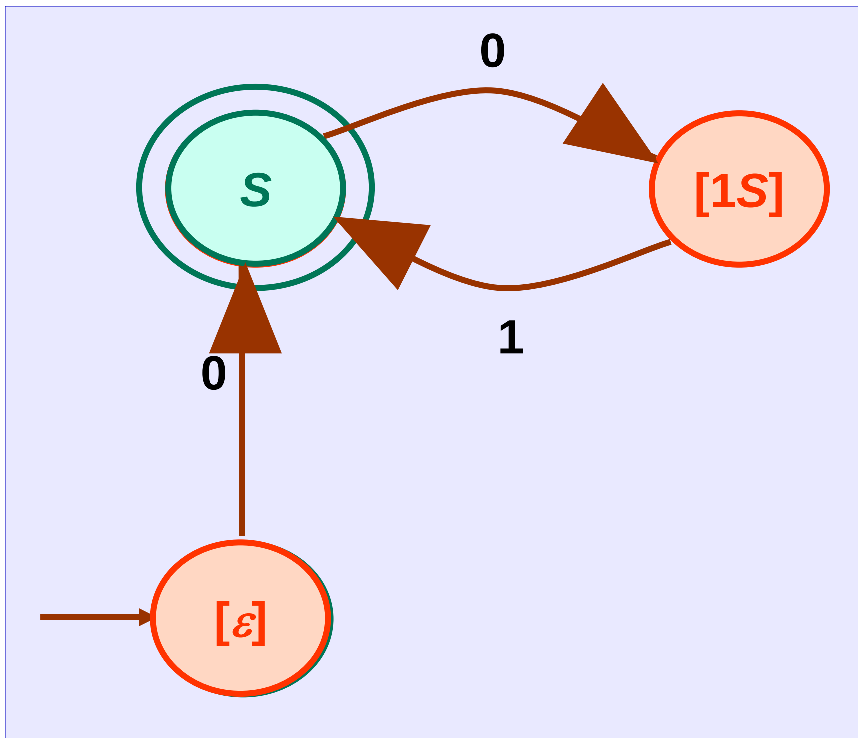
We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$

Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB$, $A \rightarrow \varepsilon$

$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon]$, $[1S] \rightarrow 1S$, $[\varepsilon] \rightarrow \varepsilon$





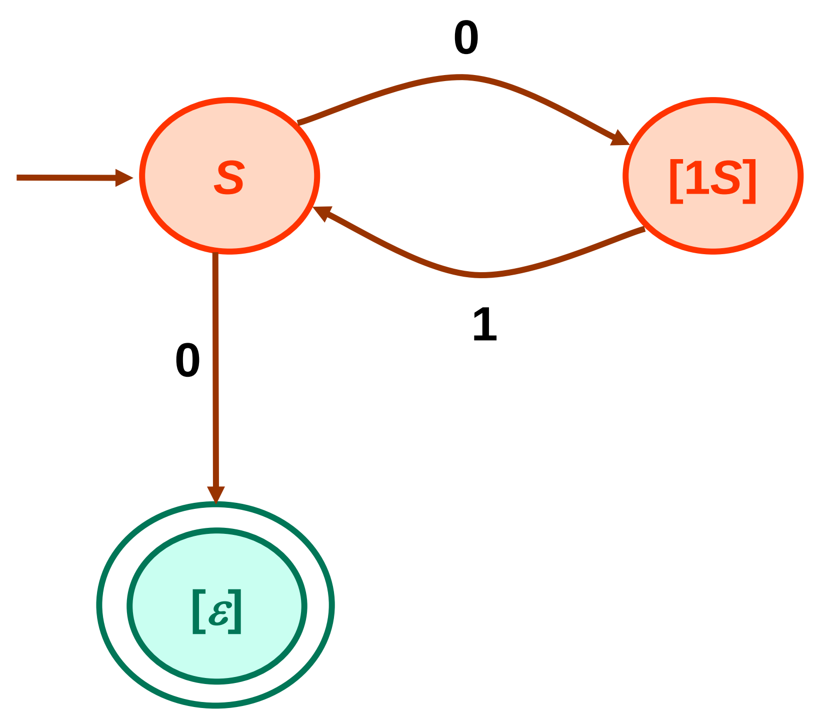
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

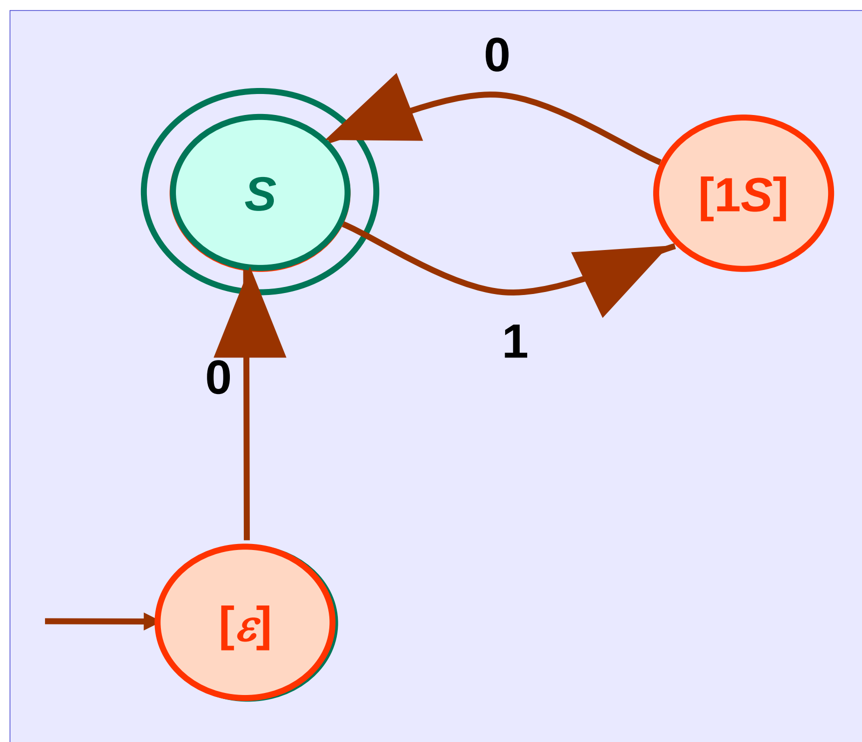
Constructing an ε-NFA from a left-linear grammar

Left-linear grammar $G_L=(\{S\}, \{0, 1\}, \{S \rightarrow S\,10 \mid 0\}, S)$
Generates the language $0(10)^*$

We construct a right-linear grammar $G_D=(\{S\}, \{0, 1\}, \{S \rightarrow 01\,S \mid 0\}, S)$
Generates the language $(01)^*0$

We rearrange the grammar to the form $A \rightarrow aB$, $A \rightarrow \varepsilon$
$S \rightarrow 0\,[1S] \mid 0\,[\varepsilon]$, $[1S] \rightarrow 1S$, $[\varepsilon] \rightarrow \varepsilon$





Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$$A \rightarrow Bw$$
$$A \rightarrow w$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$$A \rightarrow Bw$$
$$A \rightarrow w$$

NFA

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \rightarrow Bw$
$A \rightarrow w$

NFA

1) We construct an ε-NFA M which accepts $L(M) = L^R$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \rightarrow Bw$
$A \rightarrow w$

NFA

---

1) We construct an ε-NFA M which accepts $L(M) = L^R$

2) We construct a right-linear grammar G

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \to Bw$
$A \to w$

NFA

1) We construct an ε-NFA M which accepts $L(M) = L^R$

2) We construct a right-linear grammar G generating the language $L(G) = L(M) = L^R$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \rightarrow Bw$ $A \rightarrow w$	NFA

1) We construct an ε-NFA M which accepts $L(M) = L^R$

2) We construct a right-linear grammar G generating the language $L(G) = L(M) = L^R$

3) Right-hand sides of productions are reversed

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \rightarrow Bw$ $A \rightarrow w$	NFA

1) We construct an ε-NFA M which accepts $L(M) = L^R$

2) We construct a right-linear grammar G generating the language $L(G) = L(M) = L^R$

3) Right-hand sides of productions are reversed
The constructed grammar G' is a left-linear grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Constructing a left-linear grammar from an NFA

$A \rightarrow Bw$
$A \rightarrow w$

NFA

1) We construct an ε-NFA M which accepts $L(M) = L^R$

2) We construct a right-linear grammar G generating the language $L(G) = L(M) = L^R$

3) Right-hand sides of productions are reversed
The constructed grammar G' is a left-linear grammar
generating the language $L(G') = L(G)^R = L(M)^R = L$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Lecture overview

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$	$E \rightarrow a$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

Derivation (1)

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

Derivation (1)

$\underline{E}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$	$E \rightarrow a$

E

$a \odot a \odot a$

Derivation (1)

$\underline{E}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$

$E \rightarrow a$

E

$a \odot a \odot a$

Derivation (1)

$\underline{E}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

E

$a \odot a \odot a$

Derivation (1)

$\underline{E}$

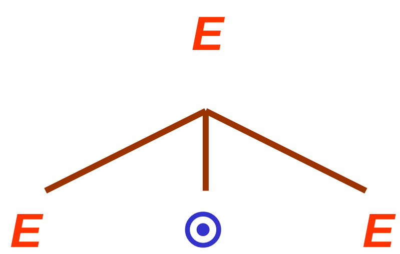
$\underline{E} \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

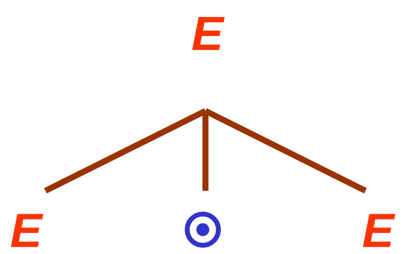
$\underline{E} \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

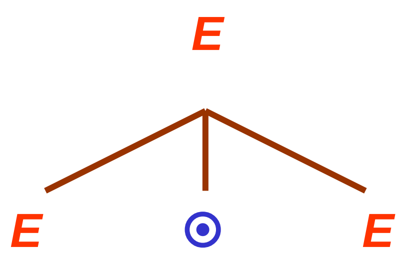
$\underline{E} \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

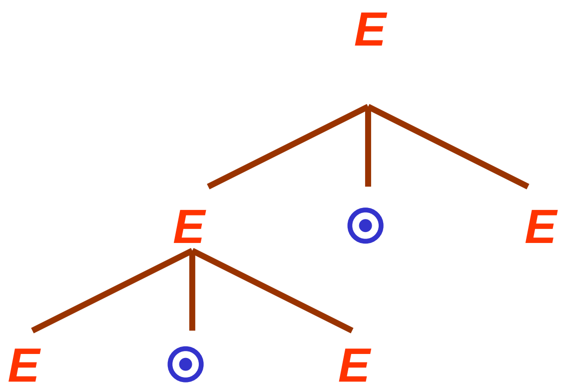
$\underline{E} \odot E \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

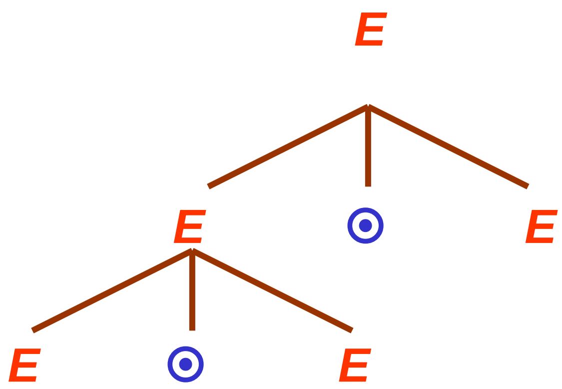
$\underline{E} \odot E \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

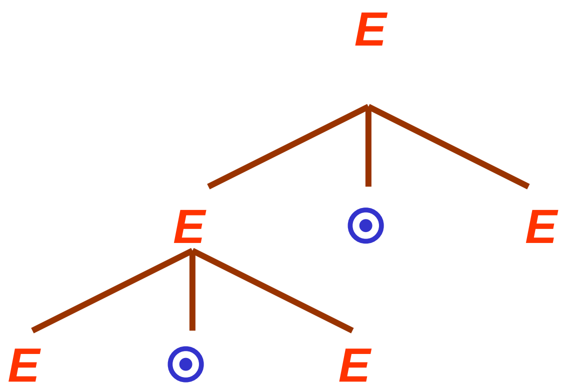
$\underline{E} \odot E \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

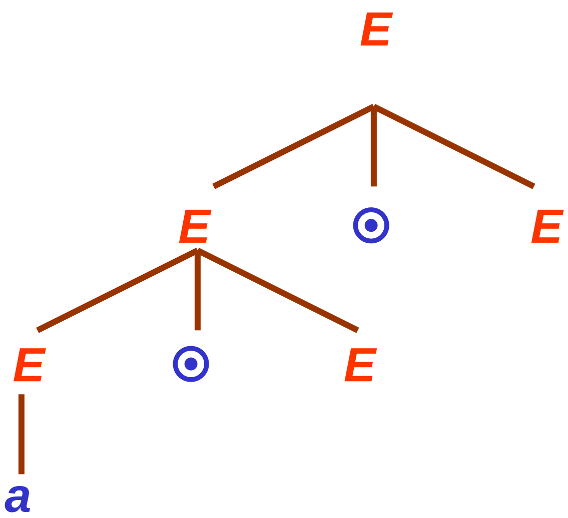
$a \odot \underline{E} \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

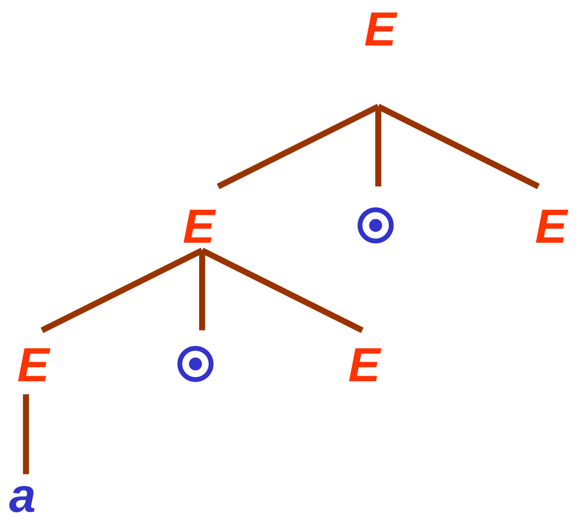
$a \odot \underline{E} \odot E$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$	$E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

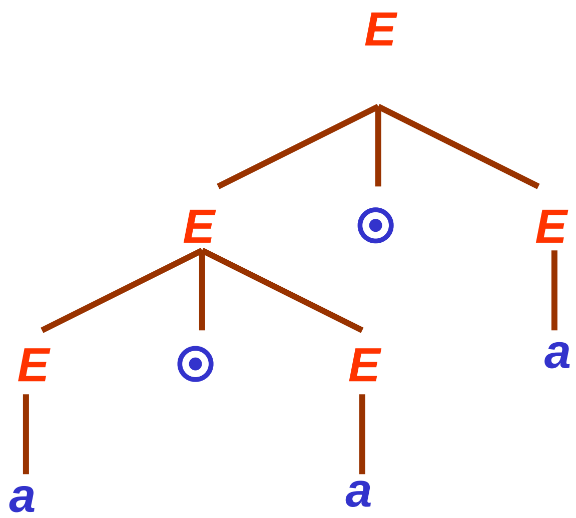
$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

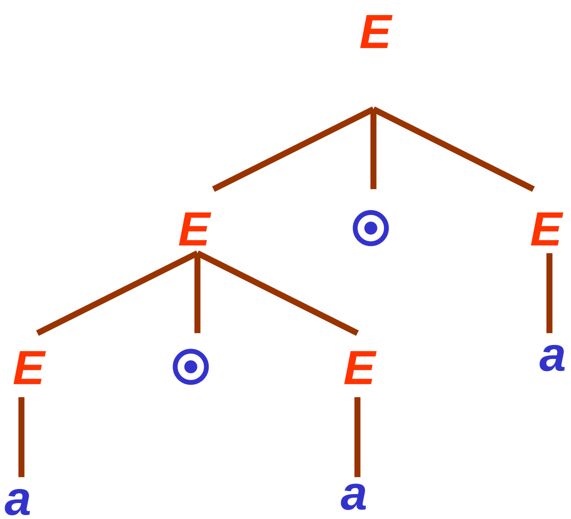
$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

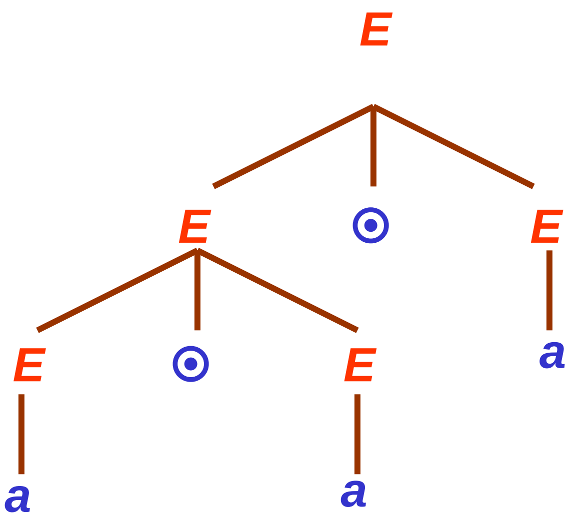
Derivation (2)

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

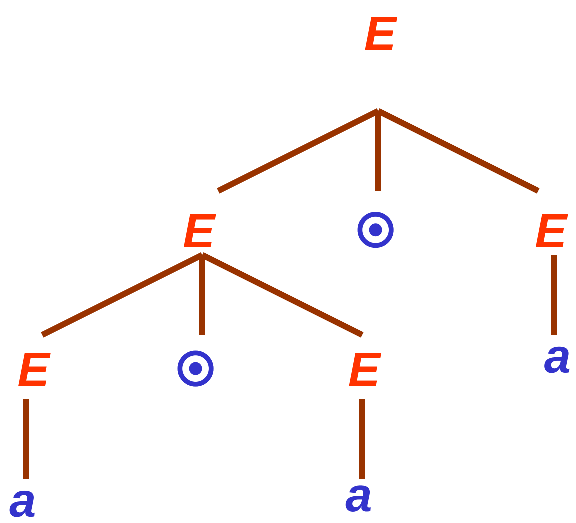
$\underline{E}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



E

Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

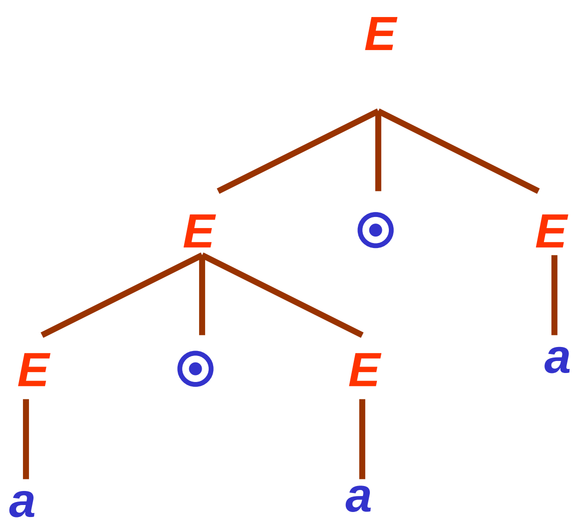
$\underline{E}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$



E

Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

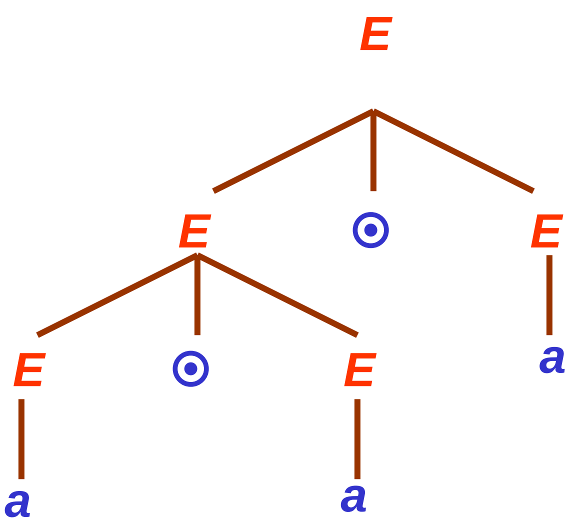
$\underline{E} \odot E$

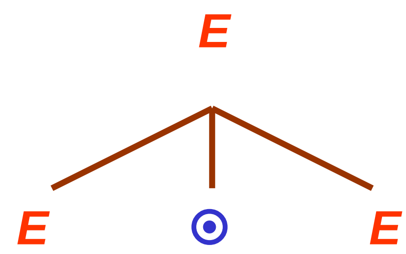
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

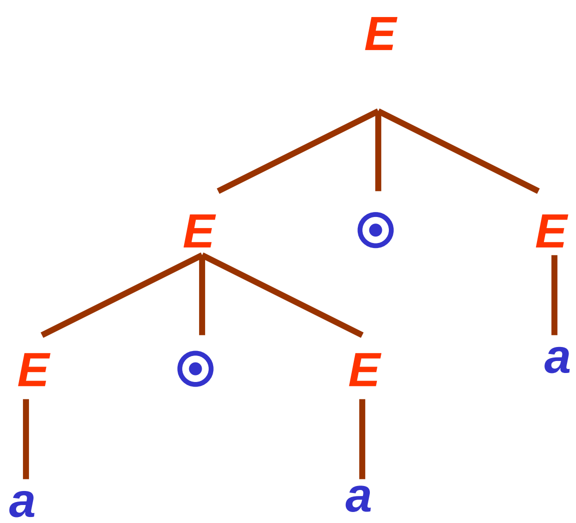
$\underline{E} \odot E$

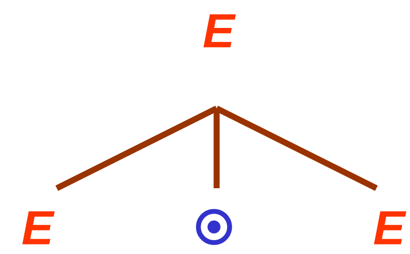
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

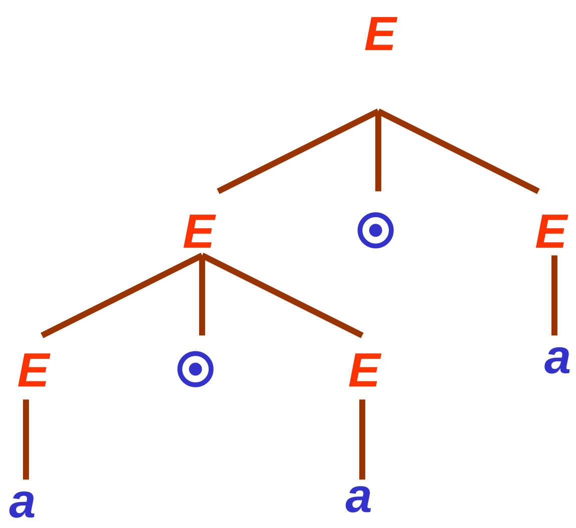
$a \odot \underline{E}$

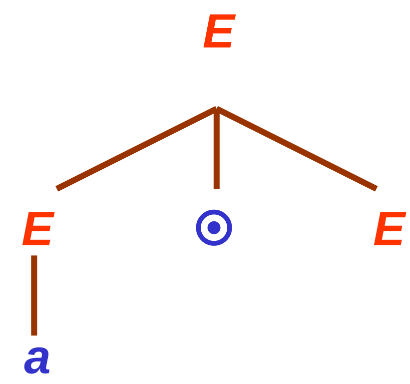
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

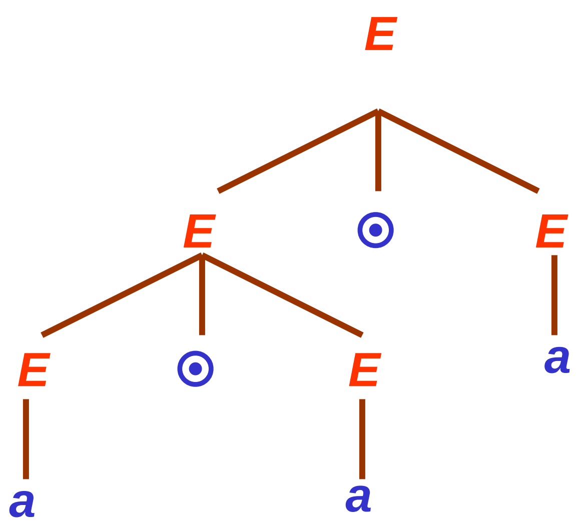
$a \odot \underline{E}$

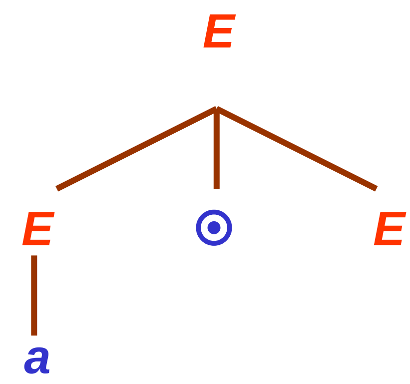
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

$a \odot \underline{E}$

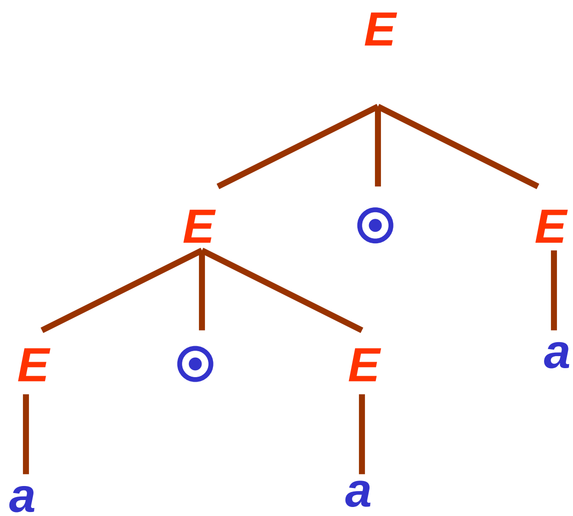
$a \odot \underline{E} \odot E$

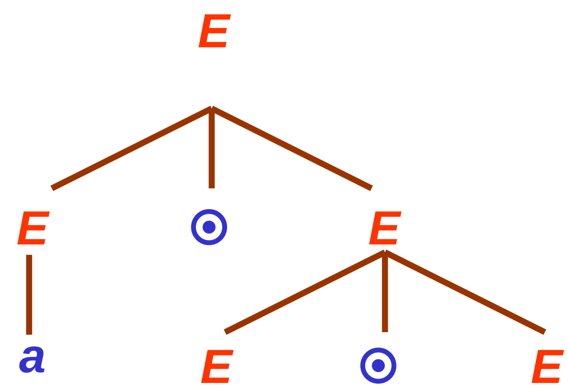
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Derivation (2)

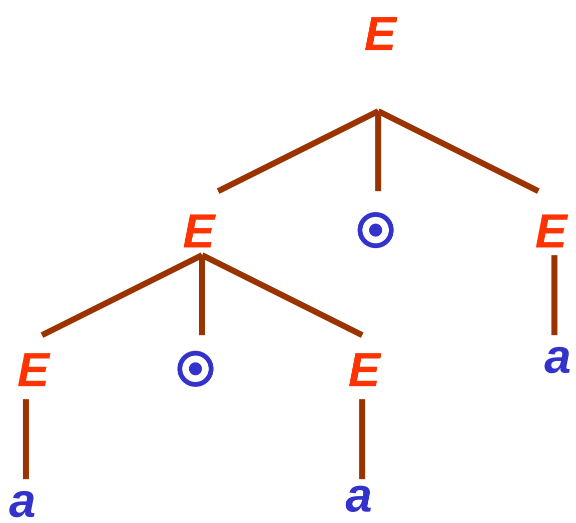
$$\underline{E}$$
$$\underline{E} \odot E$$
$$a \odot \underline{E}$$
$$a \odot \underline{E} \odot E$$

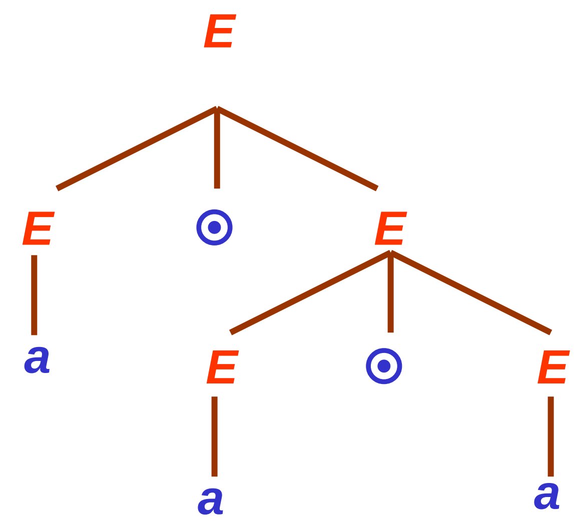
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$





Derivation (1)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Derivation (2)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$a \odot \underline{E}$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

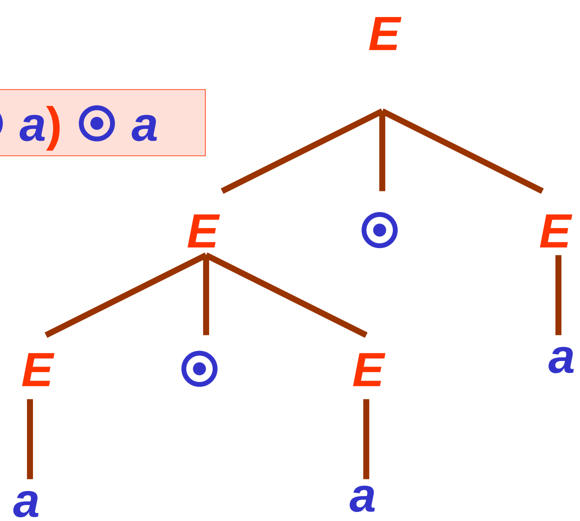
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

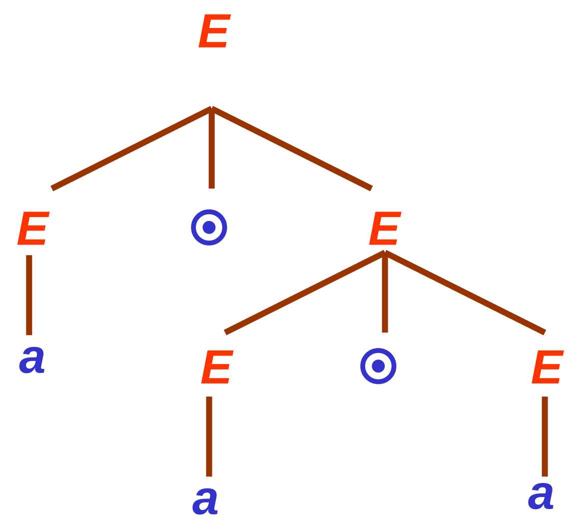
Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$





Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

$a \odot \underline{E}$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

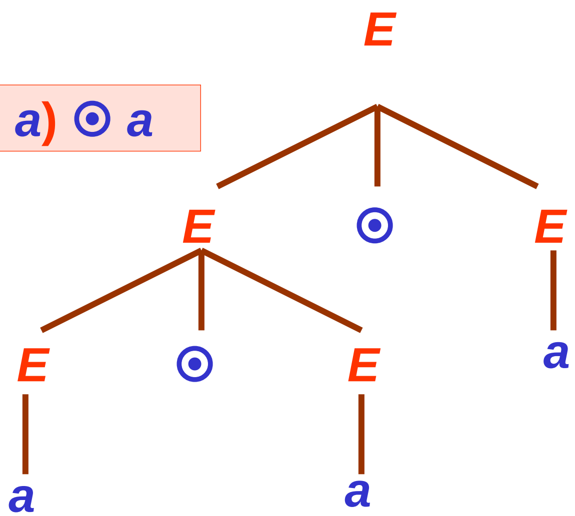
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

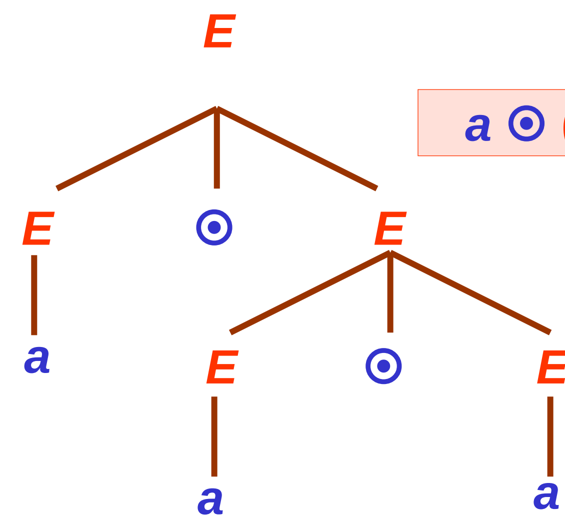
$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Derivation (2)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$a \odot \underline{E}$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

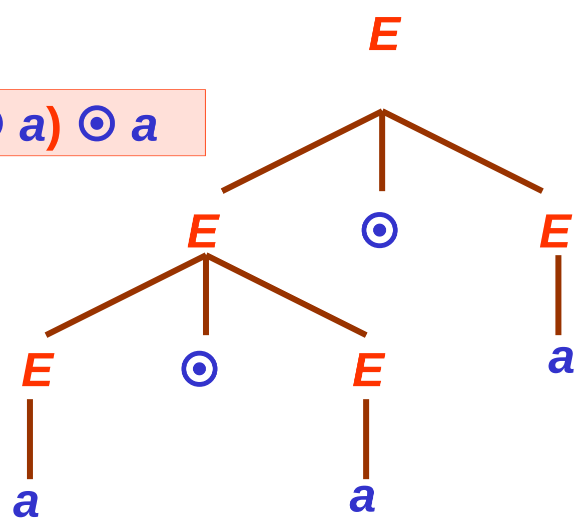
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

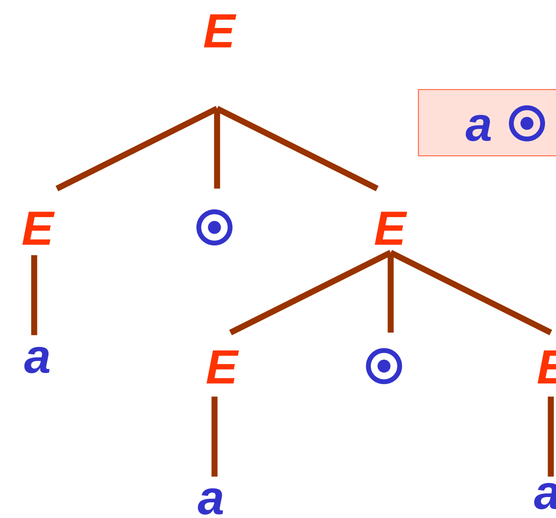
$E \rightarrow E \odot E$ $\quad E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)

$\underline{E}$
$\underline{E} \odot E$
$\underline{E} \odot E \odot E$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (2)

$\underline{E}$
$\underline{E} \odot E$
$a \odot \underline{E}$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (3)

$\underline{E}$
$E \odot \underline{E}$
$E \odot E \odot \underline{E}$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

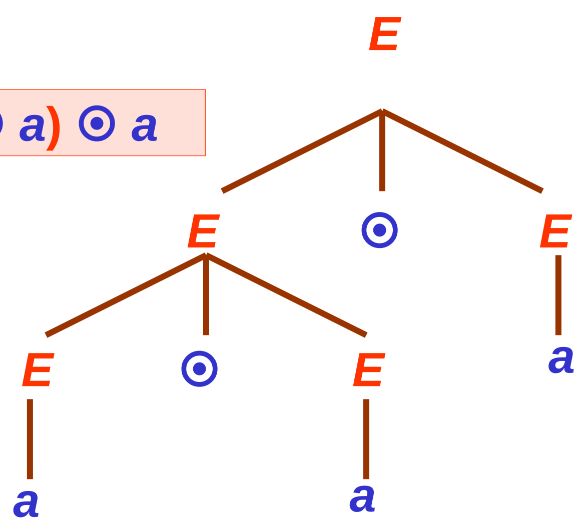
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

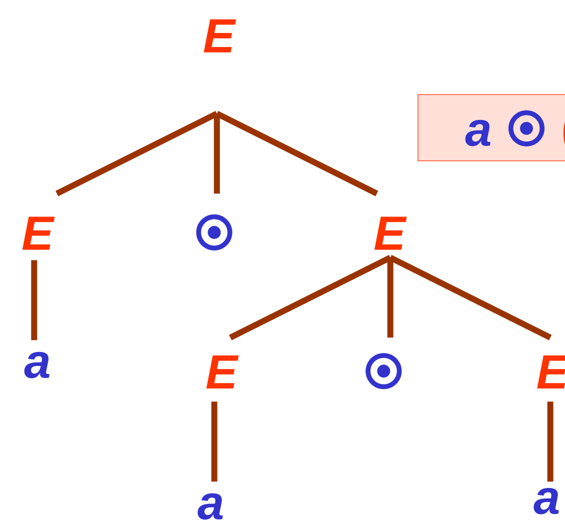
$E \rightarrow E \odot E$ $\quad$ $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Derivation (4)

$$\underline{E}$$
$$E \odot \underline{E}$$
$$\underline{E} \odot a$$
$$E \odot \underline{E} \odot a$$
$$\underline{E} \odot a \odot a$$
$$a \odot a \odot a$$

Derivation (2)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$a \odot \underline{E}$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Derivation (3)

$$\underline{E}$$
$$E \odot \underline{E}$$
$$E \odot E \odot \underline{E}$$
$$E \odot \underline{E} \odot a$$
$$\underline{E} \odot a \odot a$$
$$a \odot a \odot a$$

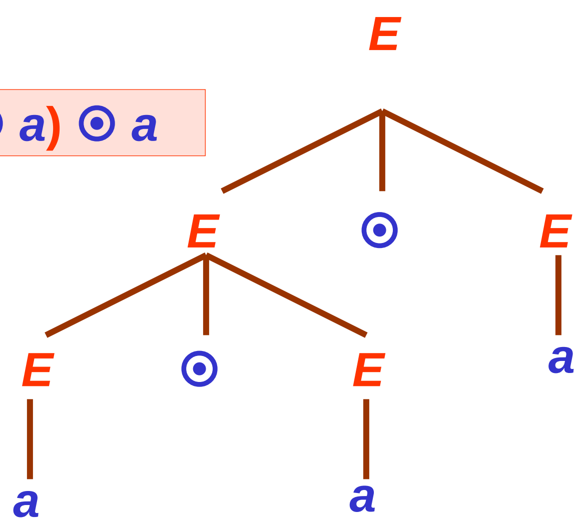
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

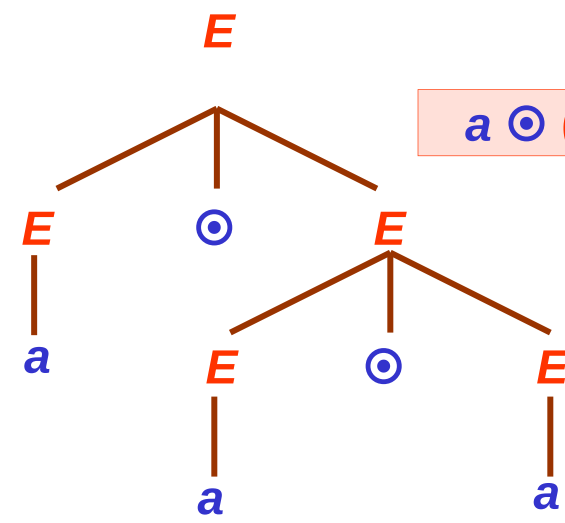
$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)	Derivation (4)	Derivation (2)	Derivation (3)
$\underline{E}$	$\underline{E}$	$\underline{E}$	$\underline{E}$
$\underline{E} \odot E$	$E \odot \underline{E}$	$\underline{E} \odot E$	$E \odot \underline{E}$
$\underline{E} \odot E \odot E$	$\underline{E} \odot a$	$a \odot \underline{E}$	$E \odot E \odot \underline{E}$
$a \odot \underline{E} \odot E$	$E \odot \underline{E} \odot a$	$a \odot \underline{E} \odot E$	$E \odot \underline{E} \odot a$
$a \odot a \odot \underline{E}$	$\underline{E} \odot a \odot a$	$a \odot a \odot \underline{E}$	$\underline{E} \odot a \odot a$
$a \odot a \odot a$	$a \odot a \odot a$	$a \odot a \odot a$	$a \odot a \odot a$

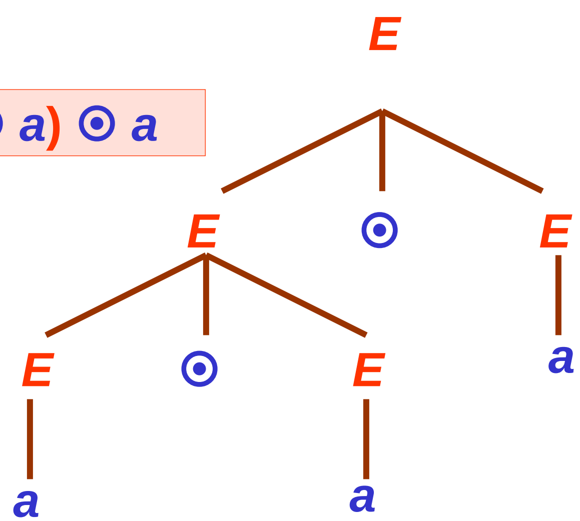
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

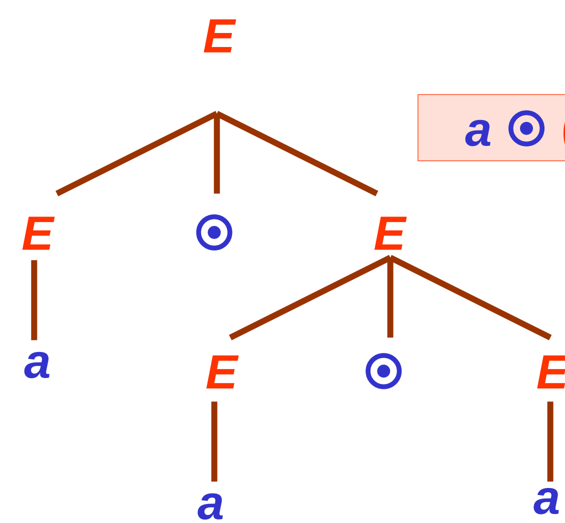
Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$





$a \odot (a \odot a)$

Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (4)

$\underline{E}$

$E \odot \underline{E}$

$\underline{E} \odot a$

$E \odot \underline{E} \odot a$

$\underline{E} \odot a \odot a$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

$a \odot \underline{E}$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (3)

$\underline{E}$

$E \odot \underline{E}$

$E \odot E \odot \underline{E}$

$E \odot \underline{E} \odot a$

$\underline{E} \odot a \odot a$

$a \odot a \odot a$

generating a string by leftmost derivations

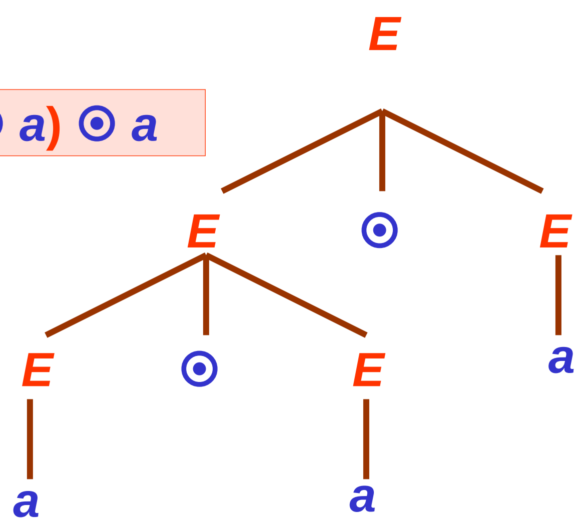
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

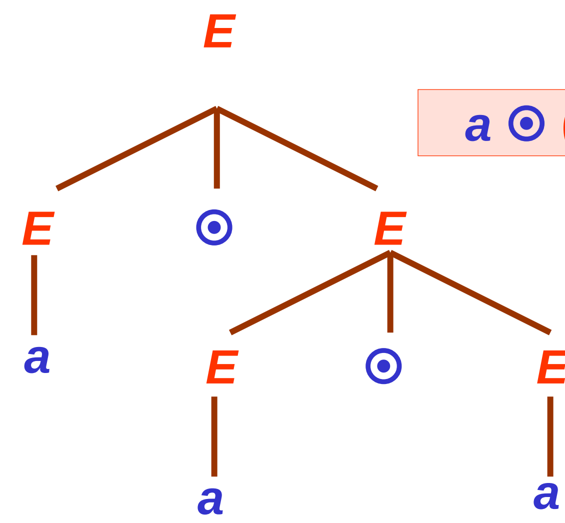
Ambiguity in grammars and languages

$E \rightarrow E \odot E$ $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$





$a \odot (a \odot a)$

Derivation (1)

$\underline{E}$

$\underline{E} \odot E$

$\underline{E} \odot E \odot E$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (4)

$\underline{E}$

$E \odot \underline{E}$

$\underline{E} \odot a$

$E \odot \underline{E} \odot a$

$\underline{E} \odot a \odot a$

$a \odot a \odot a$

Derivation (2)

$\underline{E}$

$\underline{E} \odot E$

$a \odot \underline{E}$

$a \odot \underline{E} \odot E$

$a \odot a \odot \underline{E}$

$a \odot a \odot a$

Derivation (3)

$\underline{E}$

$E \odot \underline{E}$

$E \odot E \odot \underline{E}$

$E \odot \underline{E} \odot a$

$\underline{E} \odot a \odot a$

$a \odot a \odot a$

generating a string by leftmost derivations

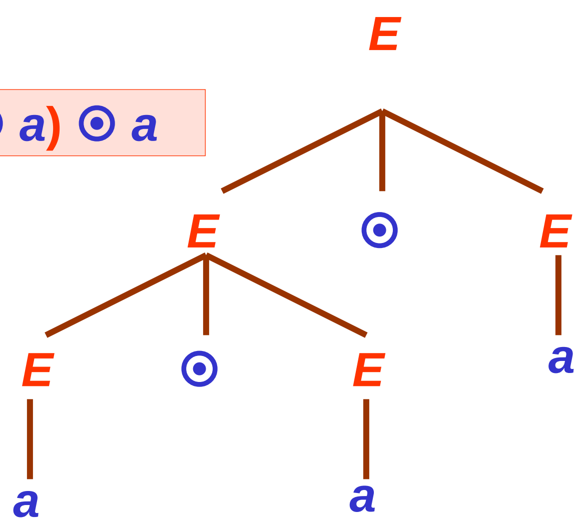
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

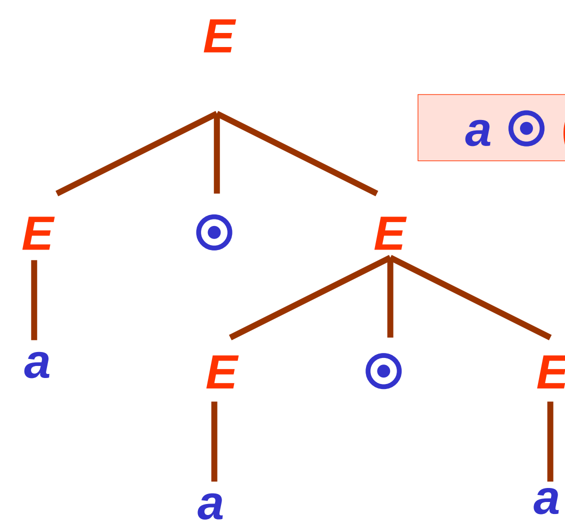
$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)

$\underline{E}$
$\underline{E} \odot E$
$\underline{E} \odot E \odot E$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (4)

$\underline{E}$
$E \odot \underline{E}$
$\underline{E} \odot a$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

Derivation (2)

$\underline{E}$
$\underline{E} \odot E$
$a \odot \underline{E}$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (3)

$\underline{E}$
$E \odot \underline{E}$
$E \odot E \odot \underline{E}$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

generating a string by leftmost derivations

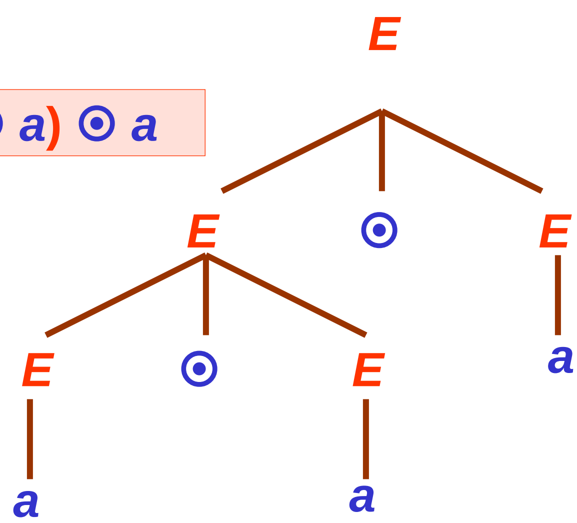
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

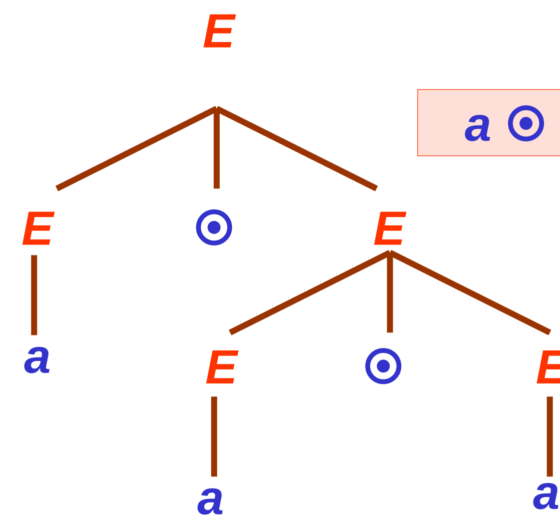
$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$



$a \odot (a \odot a)$



Derivation (1)

$\underline{E}$
$\underline{E} \odot E$
$\underline{E} \odot E \odot E$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (4)

$\underline{E}$
$E \odot \underline{E}$
$\underline{E} \odot a$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

Derivation (2)

$\underline{E}$
$\underline{E} \odot E$
$a \odot \underline{E}$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (3)

$\underline{E}$
$E \odot \underline{E}$
$E \odot E \odot \underline{E}$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

generating a string by leftmost derivations

generating a string by rightmost derivations

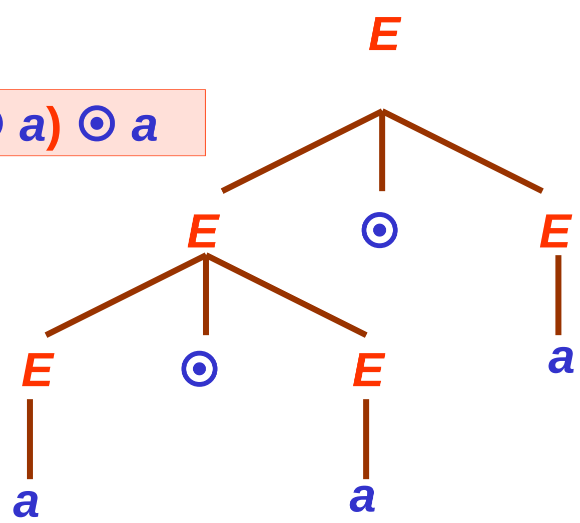
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

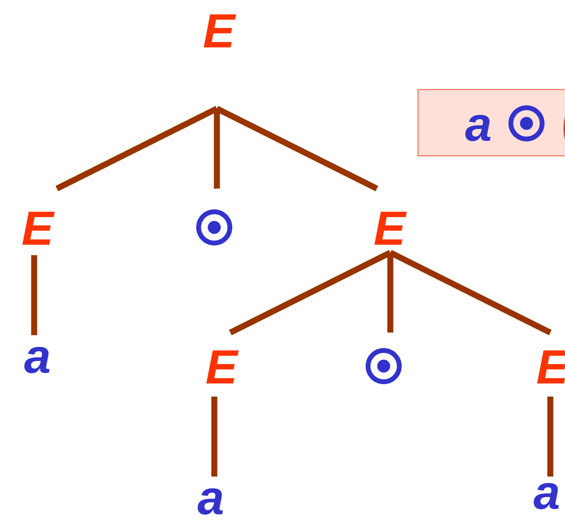
Ambiguity in grammars and languages

$E \rightarrow E \odot E$ | $E \rightarrow a$

$a \odot a \odot a$

$(a \odot a) \odot a$





$a \odot (a \odot a)$

Derivation (1)

$\underline{E}$
$\underline{E} \odot E$
$\underline{E} \odot E \odot E$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (4)

$\underline{E}$
$E \odot \underline{E}$
$\underline{E} \odot a$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

Derivation (2)

$\underline{E}$
$\underline{E} \odot E$
$a \odot \underline{E}$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (3)

$\underline{E}$
$E \odot \underline{E}$
$E \odot E \odot \underline{E}$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

generating a string by leftmost derivations

generating a string by rightmost derivations

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

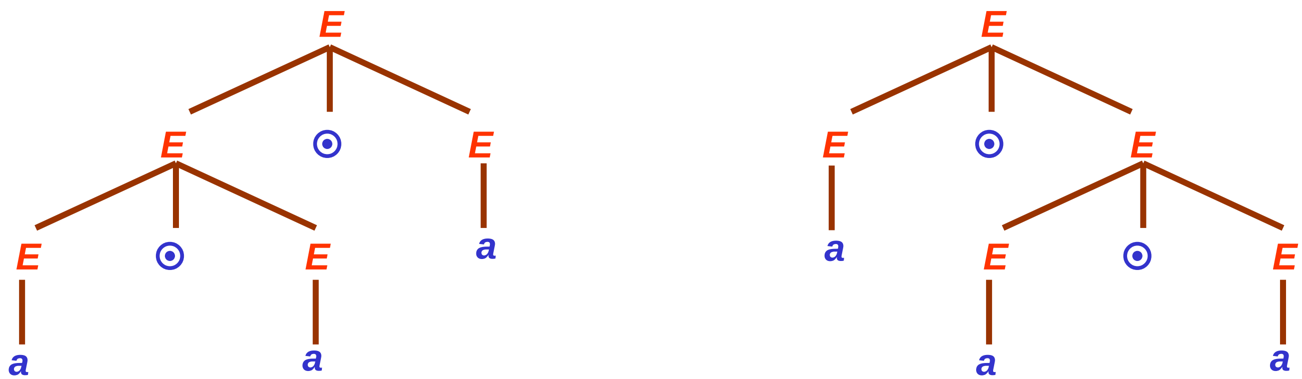
Ambiguity in grammars and languages

$a \odot a \odot a$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

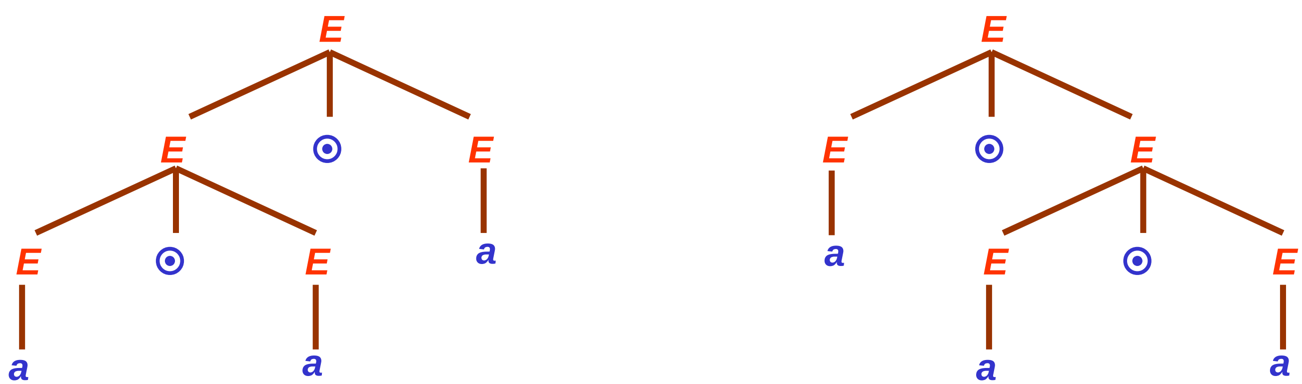
$a \odot a \odot a$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$a \odot a \odot a$



Postupak (1)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$\underline{E} \odot E \odot E$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

Postupak (4)

$$\underline{E}$$
$$E \odot \underline{E}$$
$$\underline{E} \odot a$$
$$E \odot \underline{E} \odot a$$
$$\underline{E} \odot a \odot a$$
$$a \odot a \odot a$$

Postupak (2)

$$\underline{E}$$
$$\underline{E} \odot E$$
$$a \odot \underline{E}$$
$$a \odot \underline{E} \odot E$$
$$a \odot a \odot \underline{E}$$
$$a \odot a \odot a$$

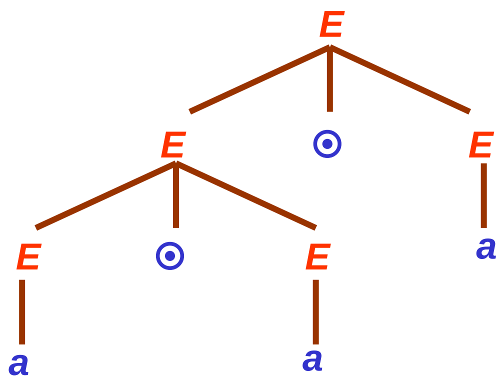
Postupak (3)

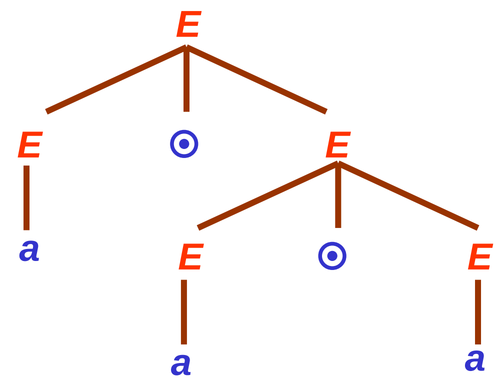
$$\underline{E}$$
$$E \odot \underline{E}$$
$$E \odot E \odot \underline{E}$$
$$E \odot \underline{E} \odot a$$
$$\underline{E} \odot a \odot a$$
$$a \odot a \odot a$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

$a \odot a \odot a$





Derivation (1)

$\underline{E}$
$\underline{E} \odot E$
$\underline{E} \odot E \odot E$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (4)

$\underline{E}$
$E \odot \underline{E}$
$\underline{E} \odot a$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

Derivation (2)

$\underline{E}$
$\underline{E} \odot E$
$a \odot \underline{E}$
$a \odot \underline{E} \odot E$
$a \odot a \odot \underline{E}$
$a \odot a \odot a$

Derivation (3)

$\underline{E}$
$E \odot \underline{E}$
$E \odot E \odot \underline{E}$
$E \odot \underline{E} \odot a$
$\underline{E} \odot a \odot a$
$a \odot a \odot a$

generating a string by leftmost derivations

generating a string by rightmost derivations

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar *G* is ambiguous

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar G is ambiguous

if a string $w \in L(G)$ can be generated by more than one parse tree

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar G is ambiguous

if a string $w \in L(G)$ can be generated by more than one parse tree

or

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar *G* is ambiguous

if a string $w \in L(G)$ can be generated by more than one parse tree

or

if a string $w \in L(G)$ can be generated by more than one leftmost derivation

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar *G* is ambiguous

if a string $w \in L(G)$ can be generated by more than one parse tree

or

if a string $w \in L(G)$ can be generated by more than one leftmost derivation

or

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A context-free grammar *G* is ambiguous

if a string $w \in L(G)$ can be generated by more than one parse tree

or
if a string $w \in L(G)$ can be generated by more than one leftmost derivation

or
if a string $w \in L(G)$ can be generated by more than one rightmost derivation

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A string *w* is ambiguous for grammar *G*

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A string *w* is ambiguous for grammar *G*

if a string $w \in L(G)$ can be generated by more than one parse tree

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A string *w* is ambiguous for grammar *G*

if a string $w \in L(G)$ can be generated by more than one parse tree

A language *L* is inherently ambiguous

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A string w is ambiguous for grammar G

if a string $w \in L(G)$ can be generated by more than one parse tree

A language L is inherently ambiguous

if L cannot be generated by an unambiguous grammar G

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

A string *w* is ambiguous for grammar *G*

if a string $w \in L(G)$ can be generated by more than one parse tree

A language *L* is inherently ambiguous

if *L* cannot be generated by an unambiguous grammar *G* (all its grammars are ambiguous)

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

$$L_n = L_1 \cup L_2 = \{\, a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1 \} \cup \{\, a^n b^m c^m d^n \mid n \geq 1,\ m \geq 1 \}$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

$$L_n = L_1 \cup L_2 = \{\, a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1 \} \cup \{\, a^n b^m c^m d^n \mid n \geq 1,\ m \geq 1 \}$$

$$L(G_n) = L(G_1) \cup L(G_2)$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

$$L_n = L_1 \cup L_2 = \{ a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{ a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

$$L(G_n) = L(G_1) \cup L(G_2)$$

$$a^n b^n c^n d^n \in L$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

$$L_n = L_1 \cup L_2 = \{\, a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1 \,\} \cup \{\, a^n b^m c^m d^n \mid n \geq 1,\ m \geq 1 \,\}$$

$$L(G_n) = L(G_1) \cup L(G_2)$$

$a^n b^n c^n d^n \in L$

$a^n b^n c^n d^n \in L(G_1)$ and $a^n b^n c^n d^n \in L(G_2)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Ambiguity in grammars and languages

Inherently ambiguous language

$L_n = L_1 \cup L_2 = \{\ a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1\} \cup \{\ a^n b^m c^m d^n \mid n \geq 1,\ m \geq 1\}$

$L(G_n) = L(G_1) \cup L(G_2)$

$a^n b^n c^n d^n \ \in L$

$a^n b^n c^n d^n \ \in L(G_1)$ and $a^n b^n c^n d^n \ \in L(G_2)$

$a^n b^n c^n d^n$ - ambiguous string

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Resolving ambiguity

Changing the language

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Resolving ambiguity

Changing the grammar

Changing the language

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = \quad (\{E, T\}, \{a, \odot\},$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\},$
$\{E \to E \odot T \mid T,\ T \to a\},$
$E)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

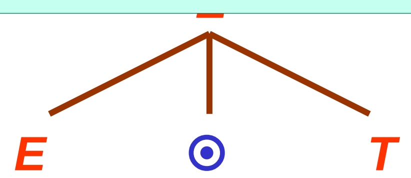
Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\},$
$\{E \to E \odot T \mid T, T \to a\},$
$E)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

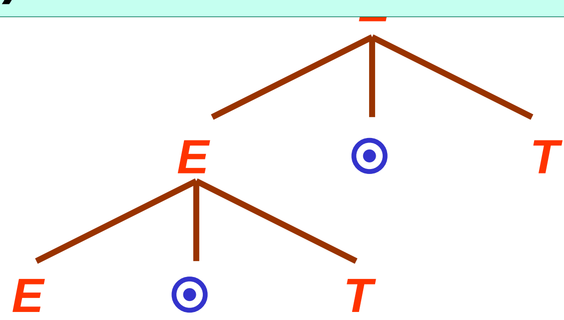
$G_1 = (\{E, T\}, \{a, \odot\},$
$\{E \rightarrow E \odot T \mid T,\ T \rightarrow a\},$
$E)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

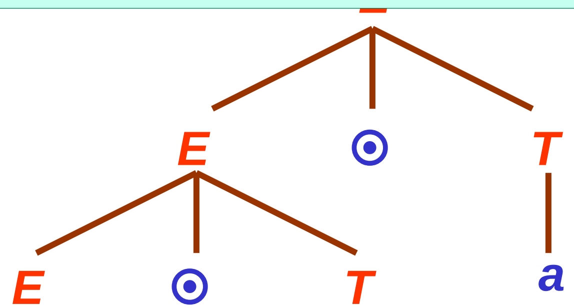
$G_1 = (\{E, T\}, \{a, \odot\}, \{E \rightarrow E \odot T \mid T,\ T \rightarrow a\}, E)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

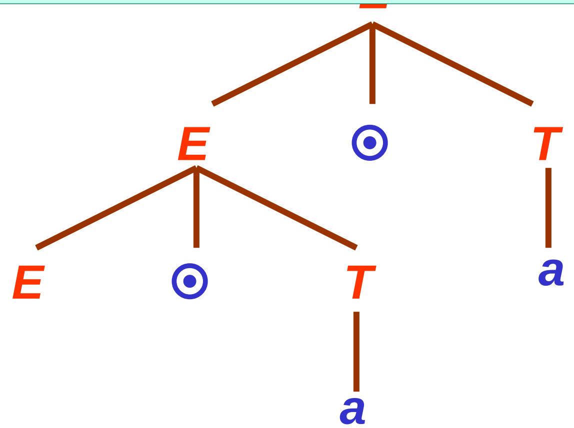
$G_1 = (\{E, T\}, \{a, \odot\},$
$\{E \to E \odot T \mid T, T \to a\},$
$E)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

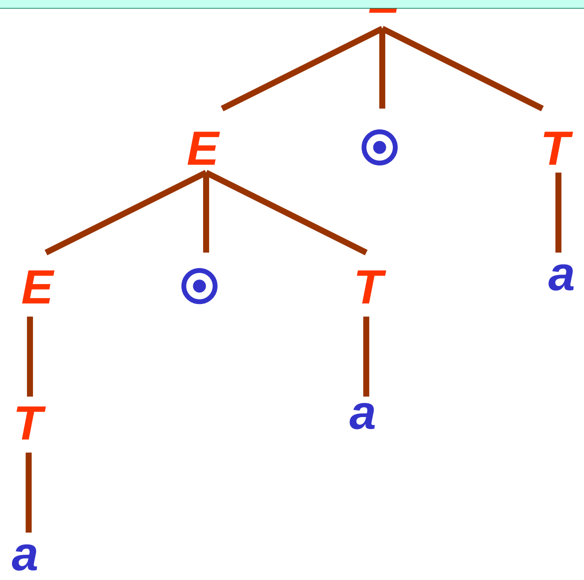
$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\},$
$\{E \to E \odot T \mid T,\ T \to a\},$
$E)$

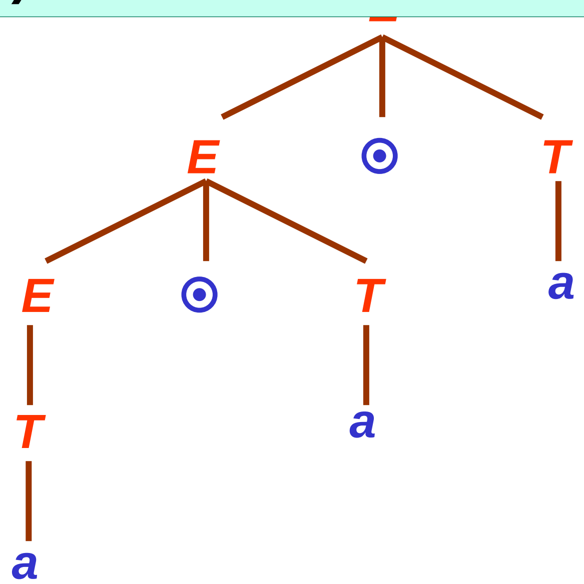


Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$

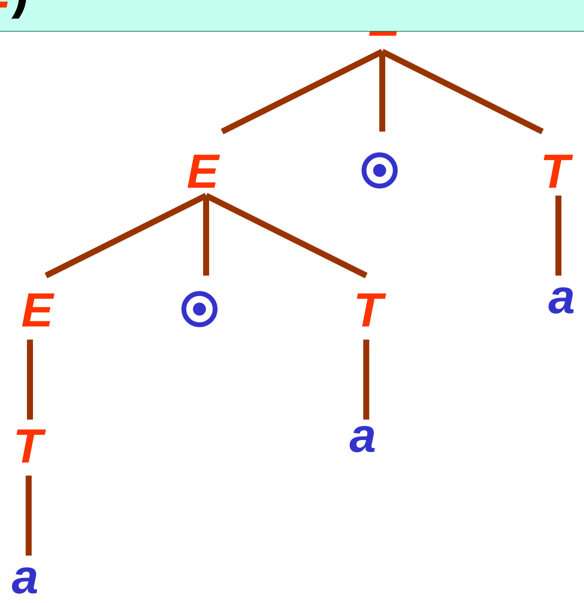
$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \rightarrow E \odot T \mid T,\ T \rightarrow a\}, E)$



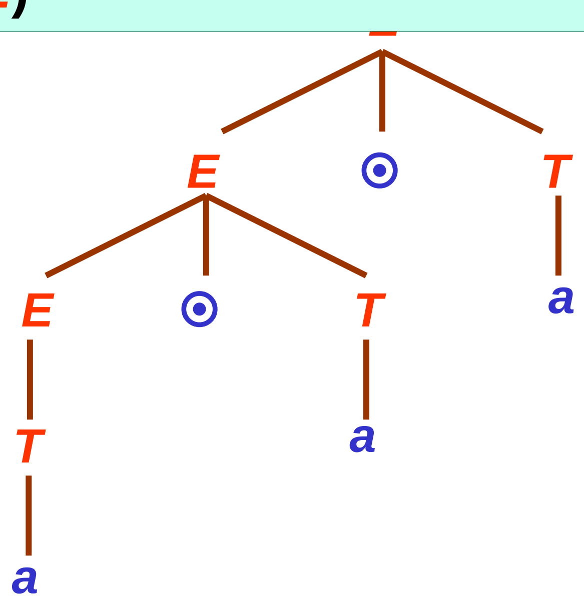
$G_2 = (\{E, T\}, \{a, \odot\}, \{E \rightarrow T \odot E \mid T,\ T \rightarrow a\}, E)$

E

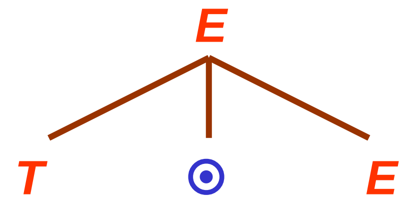
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$



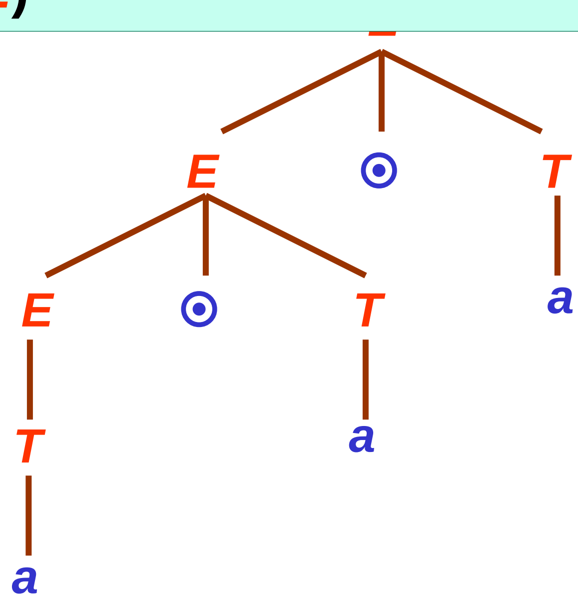
$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$



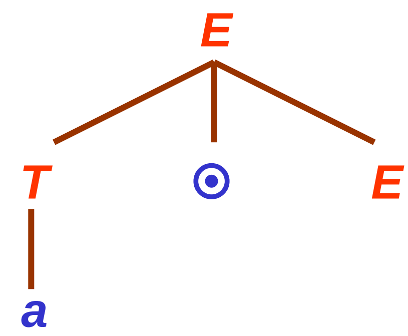
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T, T \to a\}, E)$



$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T, T \to a\}, E)$

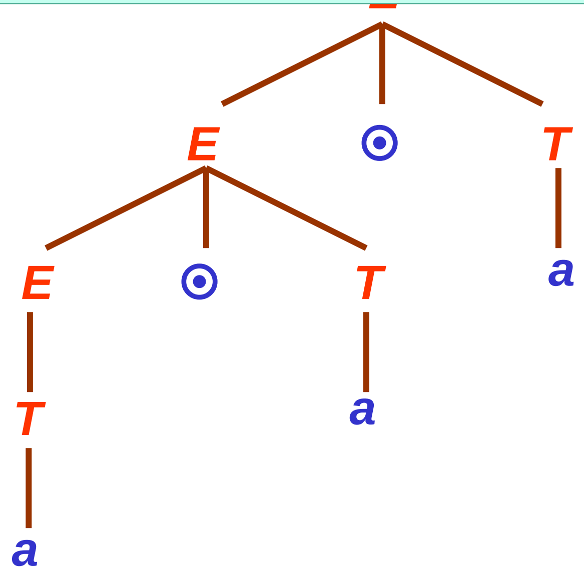


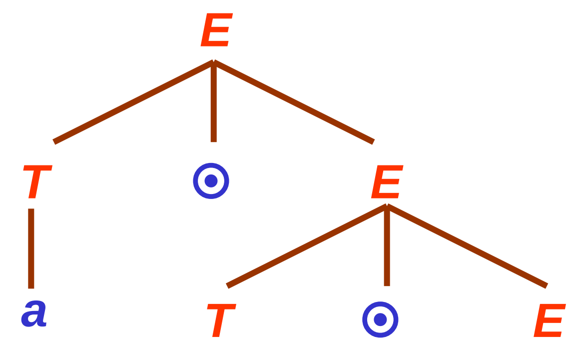
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$

$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$

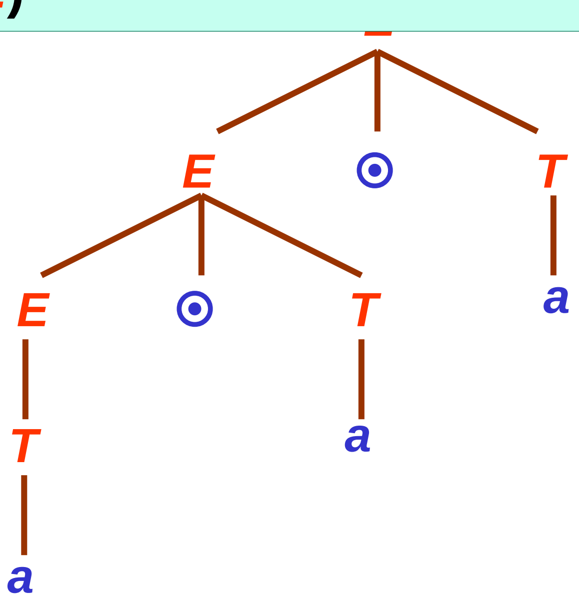




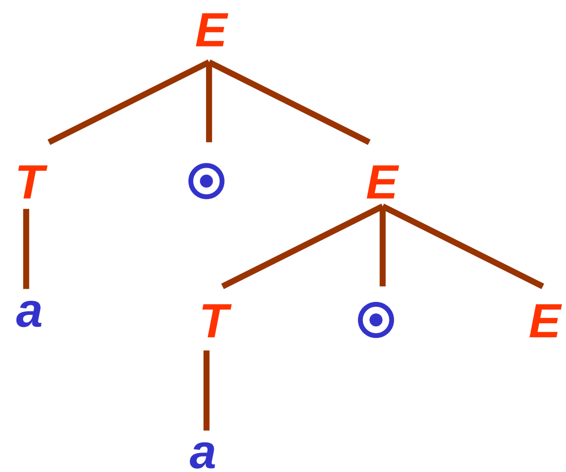
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \rightarrow E \odot T \mid T, T \rightarrow a\}, E)$



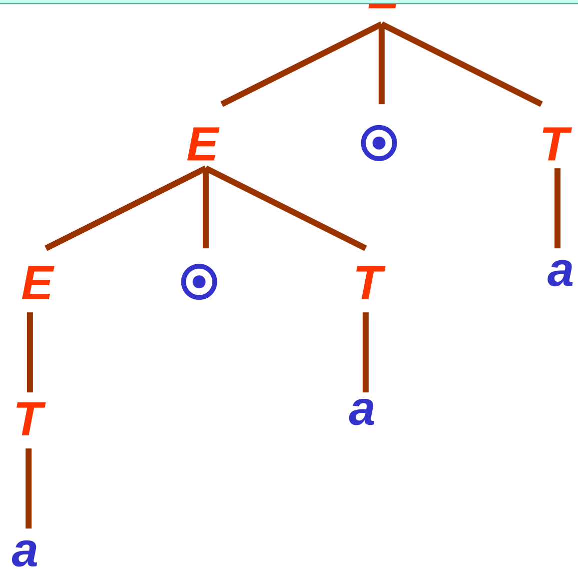
$G_2 = (\{E, T\}, \{a, \odot\}, \{E \rightarrow T \odot E \mid T, T \rightarrow a\}, E)$



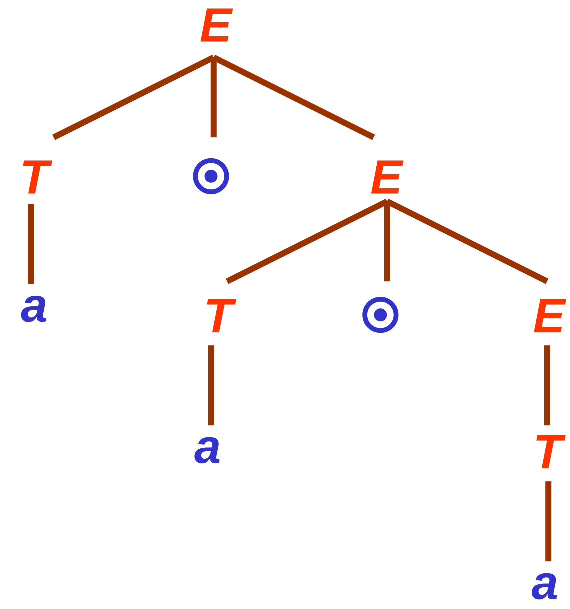
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$



$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$

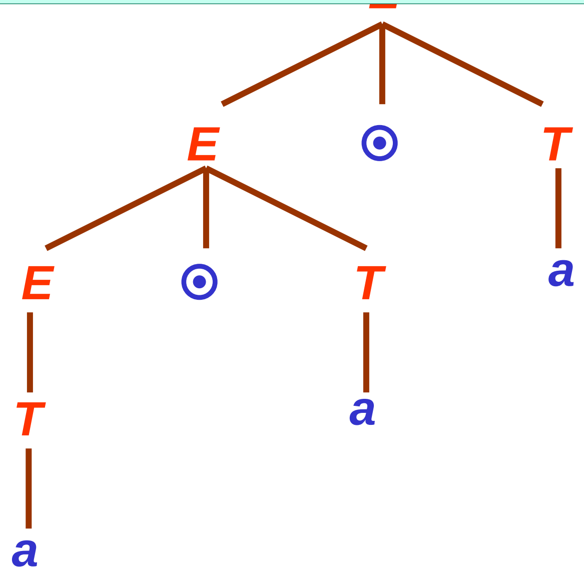


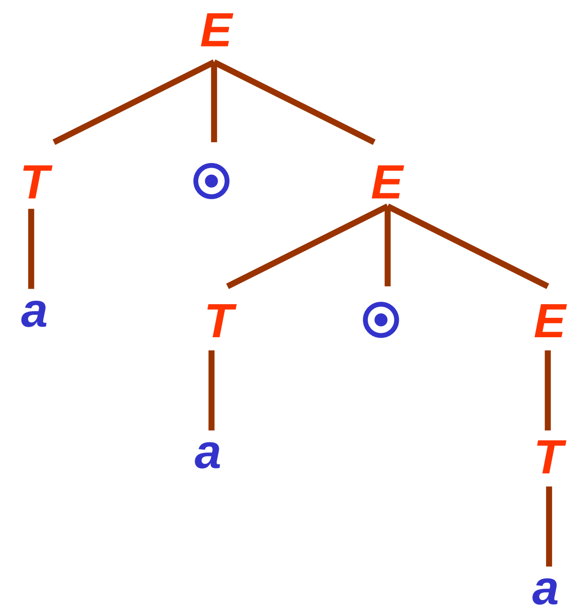
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T, T \to a\}, E)$

$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T, T \to a\}, E)$





$\underline{E}$

$\underline{E} \odot T$

$\underline{E} \odot T \odot T$

$\underline{T} \odot T \odot T$

$a \odot T \odot T$

$a \odot a \odot T$

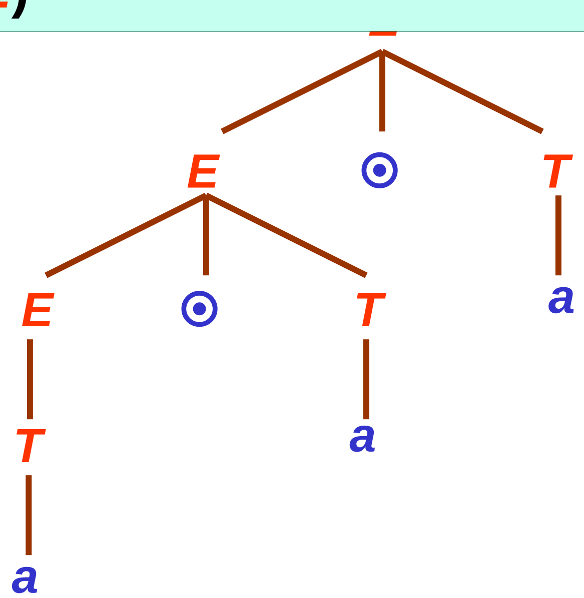
$a \odot a \odot a$

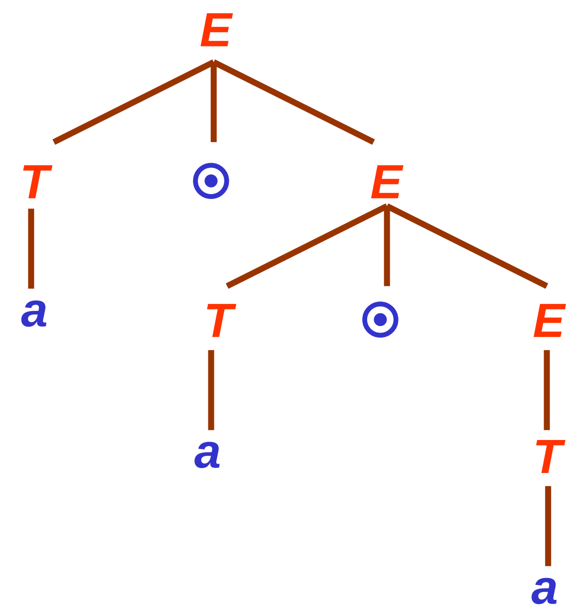
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$

$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$





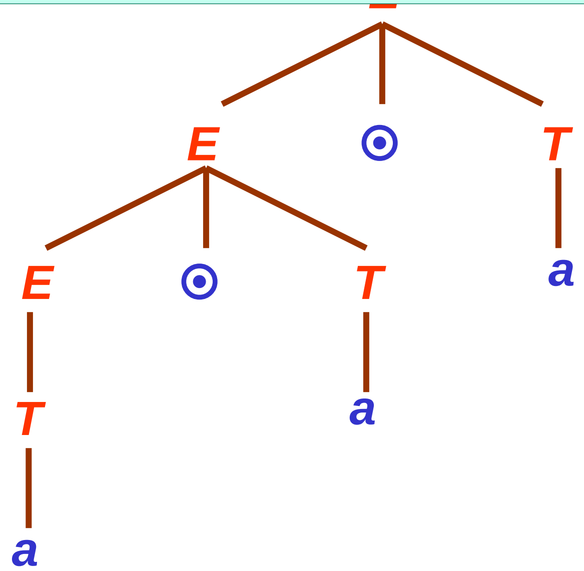
$$
\begin{array}{c}
\underline{E} \\
\underline{E} \odot T \\
\underline{E} \odot T \odot T \\
\underline{T} \odot T \odot T \\
a \odot T \odot T \\
a \odot a \odot T \\
a \odot a \odot a
\end{array}
\qquad\qquad
\begin{array}{c}
\underline{E} \\
E \odot \underline{T} \\
\underline{E} \odot a \\
E \odot \underline{T} \odot a \\
\underline{E} \odot a \odot a \\
\underline{T} \odot a \odot a \\
a \odot a \odot a
\end{array}
$$

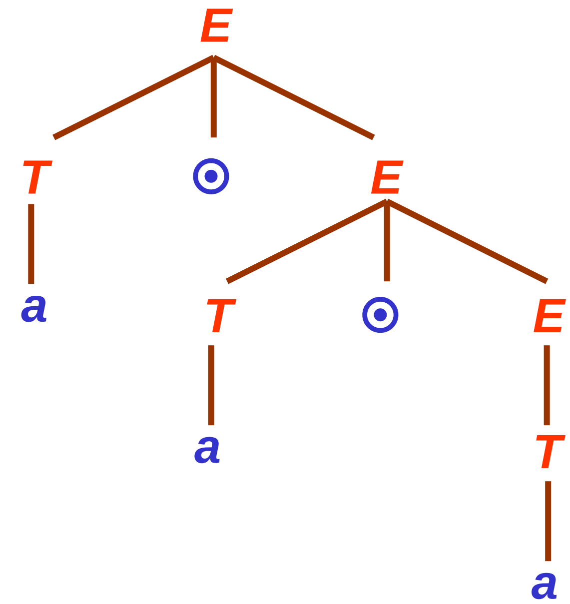
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$

$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$





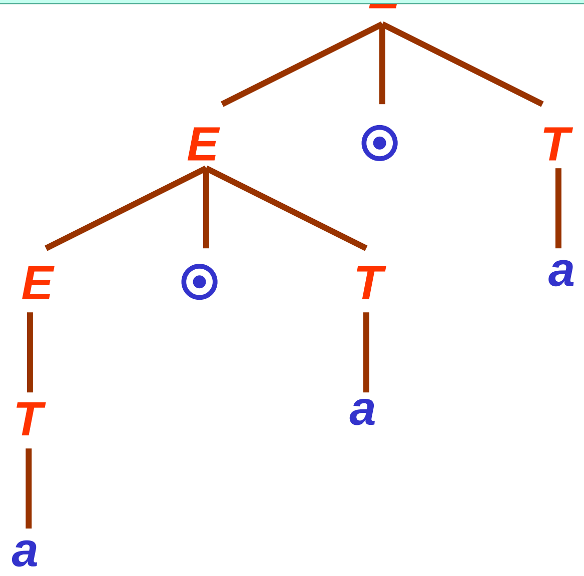
$$\begin{array}{c} \underline{E} \\ \underline{E} \odot T \\ \underline{E} \odot T \odot T \\ \underline{T} \odot T \odot T \\ a \odot T \odot T \\ a \odot a \odot T \\ a \odot a \odot a \end{array} \qquad \begin{array}{c} \underline{E} \\ E \odot \underline{T} \\ \underline{E} \odot a \\ E \odot \underline{T} \odot a \\ \underline{E} \odot a \odot a \\ \underline{T} \odot a \odot a \\ a \odot a \odot a \end{array} \qquad \begin{array}{c} \underline{E} \\ \underline{T} \odot E \\ a \odot \underline{E} \\ a \odot \underline{T} \odot E \\ a \odot a \odot \underline{E} \\ a \odot a \odot \underline{T} \\ a \odot a \odot a \end{array}$$

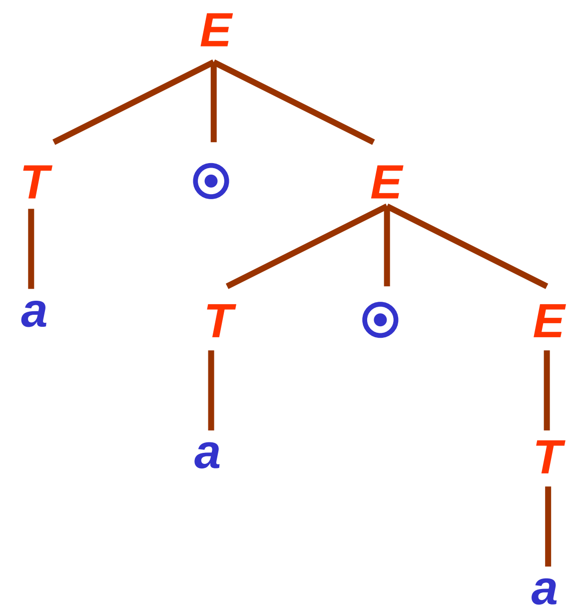
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{E, T\}, \{a, \odot\}, \{E \to E \odot T \mid T,\ T \to a\}, E)$

$G_2 = (\{E, T\}, \{a, \odot\}, \{E \to T \odot E \mid T,\ T \to a\}, E)$





$$
\begin{array}{c}
\underline{E} \\
\underline{E} \odot T \\
\underline{E} \odot T \odot T \\
\underline{T} \odot T \odot T \\
a \odot T \odot T \\
a \odot a \odot T \\
a \odot a \odot a
\end{array}
\qquad
\begin{array}{c}
\underline{E} \\
E \odot \underline{T} \\
\underline{E} \odot a \\
E \odot \underline{T} \odot a \\
\underline{E} \odot a \odot a \\
\underline{T} \odot a \odot a \\
a \odot a \odot a
\end{array}
$$

$$
\begin{array}{c}
\underline{E} \\
\underline{T} \odot E \\
a \odot \underline{E} \\
a \odot \underline{T} \odot E \\
a \odot a \odot \underline{E} \\
a \odot a \odot \underline{T} \\
a \odot a \odot a
\end{array}
\qquad
\begin{array}{c}
\underline{E} \\
T \odot \underline{E} \\
T \odot T \odot \underline{E} \\
T \odot T \odot \underline{T} \\
T \odot \underline{T} \odot a \\
\underline{T} \odot a \odot a \\
a \odot a \odot a
\end{array}
$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G =$ ({S, B}, {**if**, **then**, **else**},

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G =$ $(\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G =$ $(\{S, B\}, \{$**if**, **then**, **else**$\},$
$\{S \rightarrow$ **if** B **then** S **else** S | **if** B **then** S, $B \rightarrow$ **true** | **false**$\}, S)$

if B **then if** B **then** S **else** S

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G =$ ({S, B}, {**if**, **then**, **else**},
{$S \to$ **if** B **then** S **else** S | **if** B **then** S, $B \to$ **true** | **false**}, S)

if B **then** **if** B **then** S **else** S

$\underline{S}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

G = ({S, B}, {**if**, **then**, **else**},
{$S \to$ **if** B **then** S **else** S | **if** B **then** S, $B \to$ **true** | **false**}, S)

if B **then** **if** B **then** S **else** S

$\underline{S}$

if B **then** $\underline{S}$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G = (\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

if *B* **then if** *B* **then** *S* **else** *S*

$\underline{S}$

$\text{if } B \text{ then } \underline{S}$

$\text{if } B \text{ then if } B \text{ then } S \text{ else } S$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

G **=** ({*S*, *B*}, {**if**, **then**, **else**},
{*S* → **if** *B* **then** *S* **else** *S* | **if** *B* **then** *S*, *B* → **true** | **false**}, *S*)

if *B* **then** **if** *B* **then** *S* **else** *S*

S

if *B* **then** *S*

if *B* **then** **if** *B* **then** *S* **else** *S*

S

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

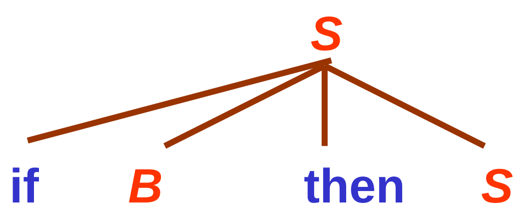
G = ({S, B}, {**if**, **then**, **else**},
{$S \rightarrow$ **if** B **then** S **else** S | **if** B **then** S, $B \rightarrow$ **true** | **false**}, S)

if B **then if** B **then** S **else** S

$\underline{S}$

if B **then** $\underline{S}$

if B **then if** B **then** S **else** S



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

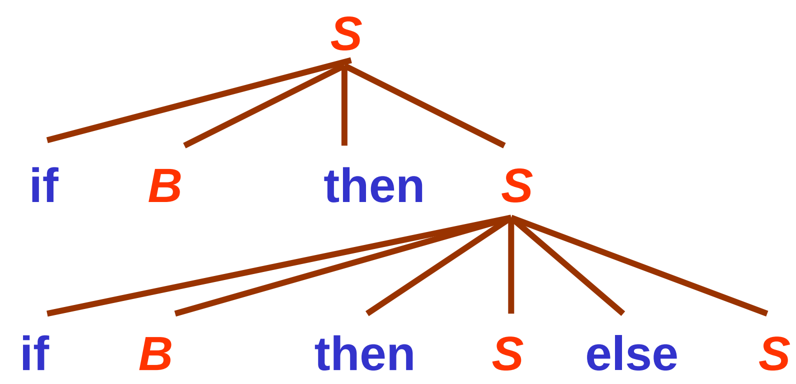
G = ({*S*, *B*}, {**if**, **then**, **else**},
{*S* → **if** *B* **then** *S* **else** *S* | **if** *B* **then** *S*, *B* → **true** | **false**}, *S*)

if *B* **then if** *B* **then** *S* **else** *S*

$\underline{S}$

if *B* **then** $\underline{S}$

if *B* **then if** *B* **then** *S* **else** *S*



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

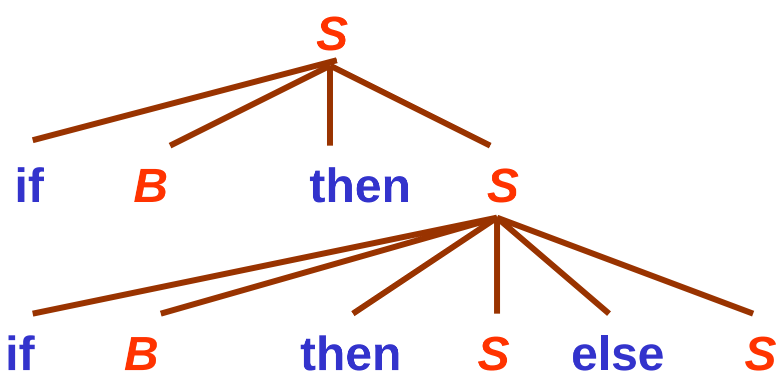
$G =$ $(\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

if *B* then if *B* then *S* else *S*

$\underline{S}$ $\qquad\qquad\qquad\qquad$ $\underline{S}$

if B then $\underline{S}$

if B then if B then S else S



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G = (\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

if *B* then if *B* then *S* else *S*

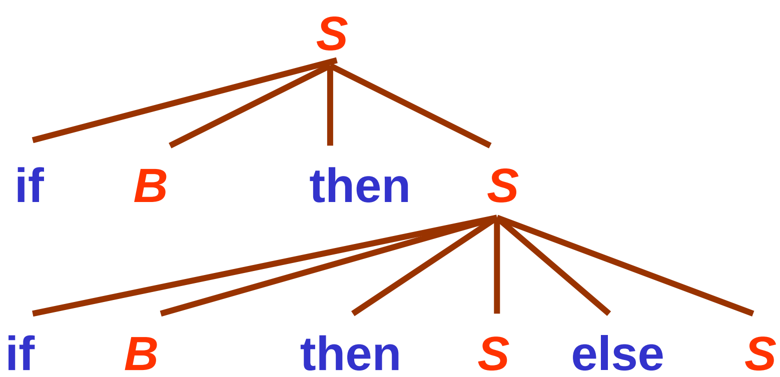
$\underline{S}$

$\text{if } B \text{ then } \underline{S}$

$\text{if } B \text{ then if } B \text{ then } S \text{ else } S$

$\underline{S}$

$\text{if } B \text{ then } \underline{S} \text{ else } S$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

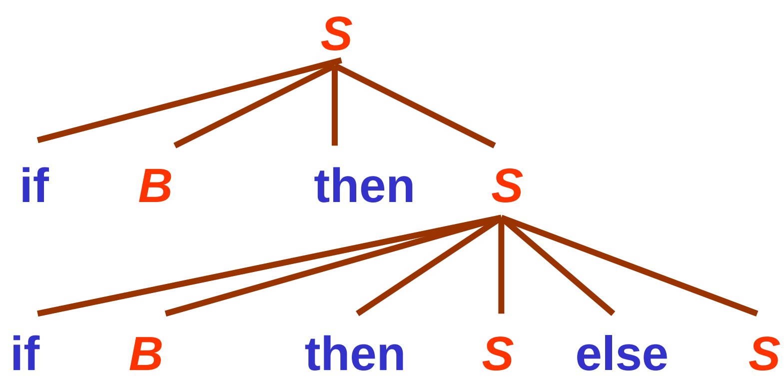
Changing the grammar

G = ({*S*, *B*}, {if, then, else},
{*S* → if *B* then *S* else *S* | if *B* then *S*, *B* → true | false}, *S*)

if *B* then if *B* then *S* else *S*

<u>*S*</u>
if *B* then <u>*S*</u>
if *B* then if *B* then *S* else *S*

<u>*S*</u>
if *B* then <u>*S*</u> else *S*
if *B* then if *B* then *S* else *S*



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

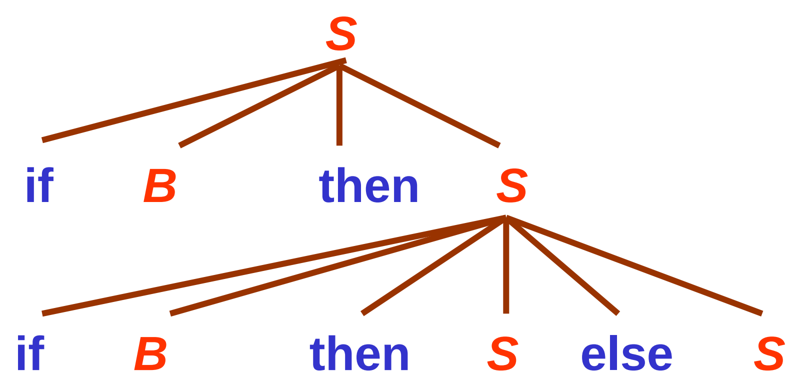
Changing the grammar

$G =$ $(\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

if *B* then if *B* then *S* else *S*

$\underline{S}$
if B then $\underline{S}$
if B then if B then S else S

$\underline{S}$
if B then $\underline{S}$ else S
if B then if B then S else S



S

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

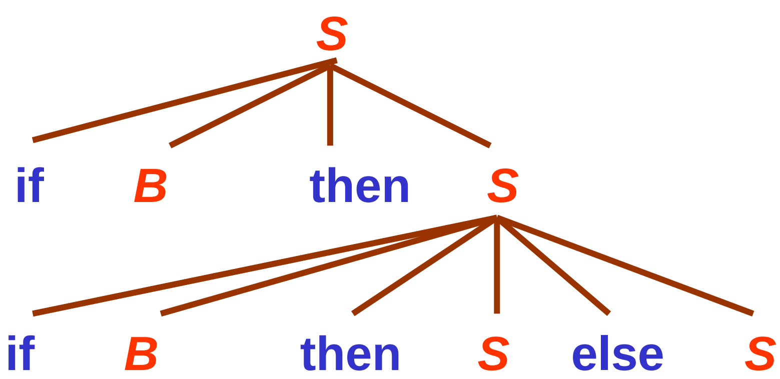
Changing the grammar

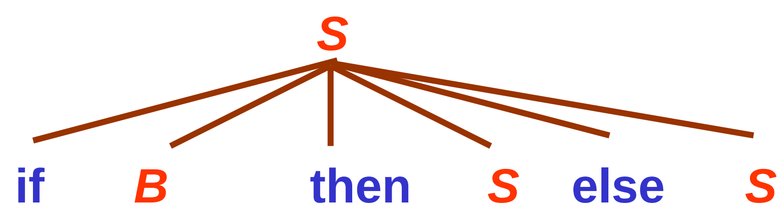
$G = (\{S, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \rightarrow \text{true} \mid \text{false}\}, S)$

if *B* then if *B* then *S* else *S*

$\underline{S}$
if B then $\underline{S}$
if B then if B then S else S

$\underline{S}$
if B then $\underline{S}$ else S
if B then if B then S else S





Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

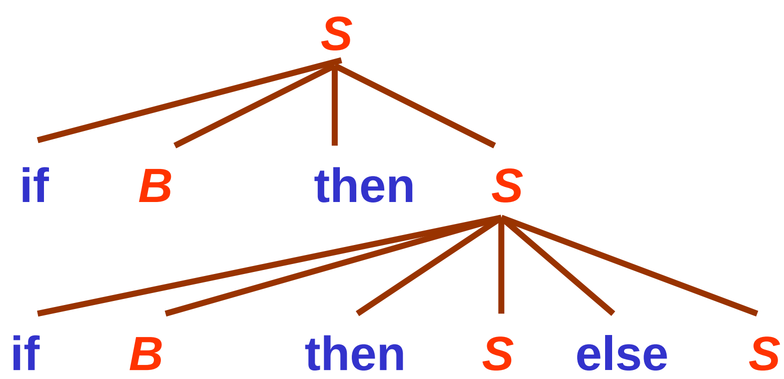
Changing the grammar

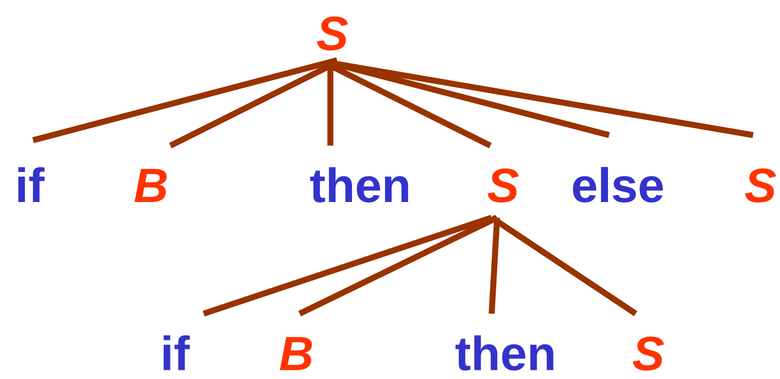
$G = (\{S, B\}, \{\text{if}, \text{then}, \text{else}\}, \{S \to \text{if } B \text{ then } S \text{ else } S \mid \text{if } B \text{ then } S,\ B \to \text{true} \mid \text{false}\}, S)$

if *B* then if *B* then *S* else *S*

<u>*S*</u>
if *B* then <u>*S*</u>
if *B* then if *B* then *S* else *S*

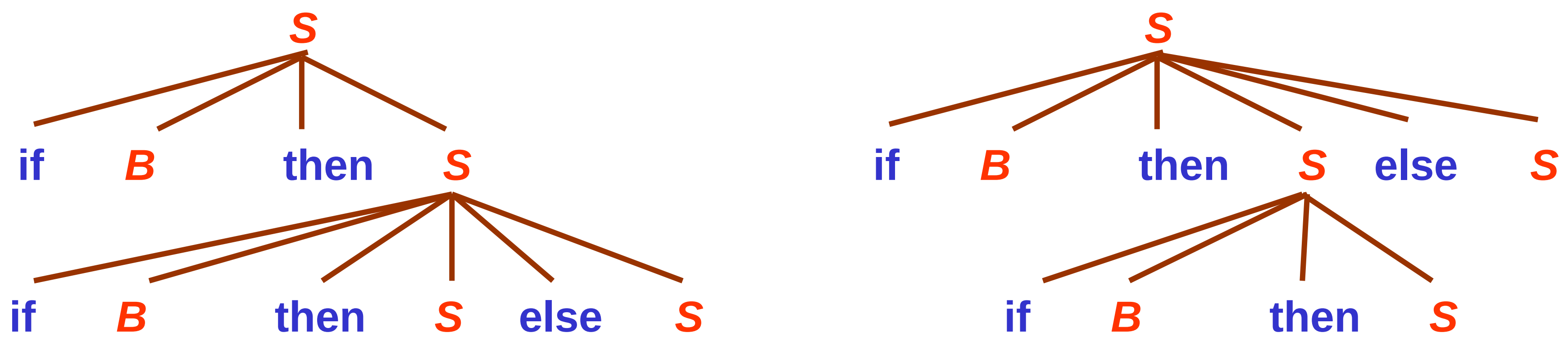
<u>*S*</u>
if *B* then <u>*S*</u> else *S*
if *B* then if *B* then *S* else *S*





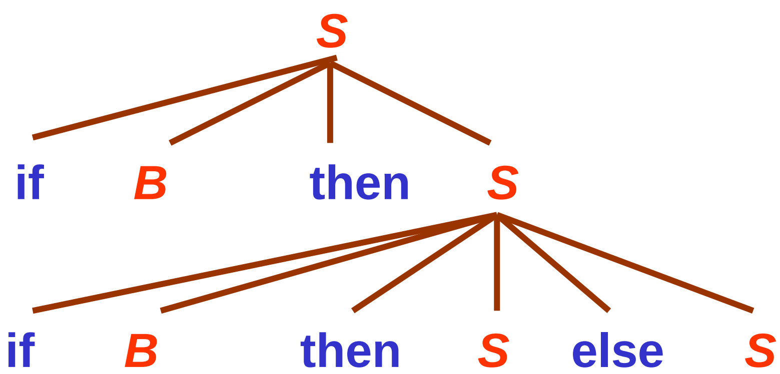
Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

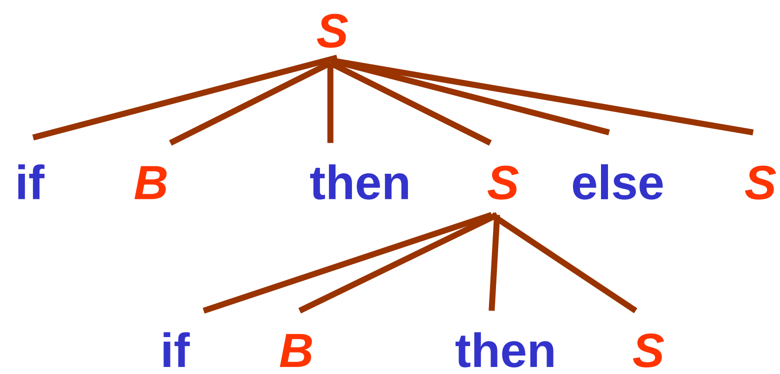
Changing the grammar



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

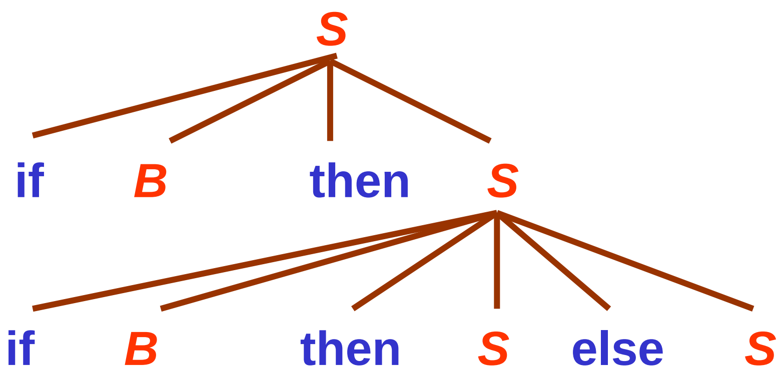


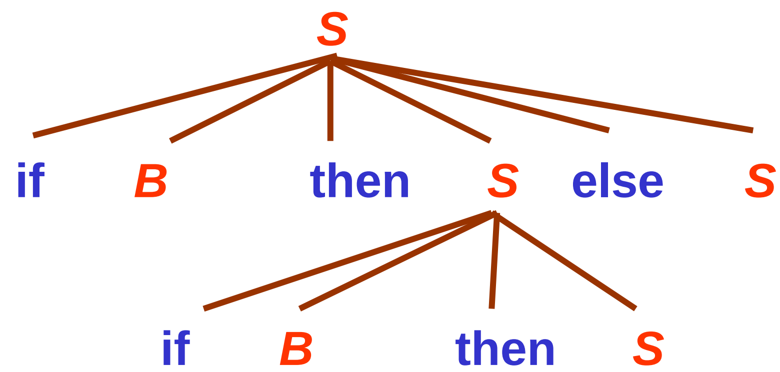


if true then if false then PRINT("*X*") else PRINT("*Y*")

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar



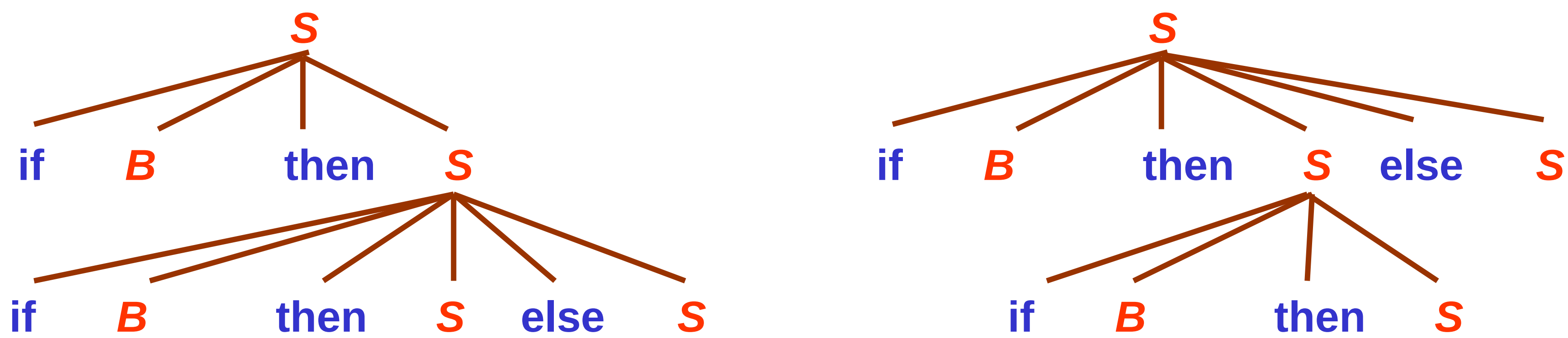


if true then if false then PRINT("*X*") else PRINT("*Y*")

if true then
if false then
PRINT("*X*")
else
PRINT("*Y*")

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar



if **true** **then** **if** **false** **then** **PRINT("*X*")** **else** **PRINT("*Y*")**

if true then
if false then
PRINT("*X*")
else
PRINT("*Y*")

if true then
if false then
PRINT("*X*")
else PRINT("*Y*")

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 =$ $(\{S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 =$ $(\{S, S_1, S_2, B\}, \{\textbf{if}, \textbf{then}, \textbf{else}\},$
$\{S \to S_1 \mid S_2$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 =$ $(\{S, S_1, S_2, B\}, \{\textbf{if}, \textbf{then}, \textbf{else}\},$
$\{S \to S_1 \mid S_2$
$S_1 \to \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{S, S_1, S_2, B\}, \{\mathbf{if}, \mathbf{then}, \mathbf{else}\},$
$\{S \to S_1 \mid S_2$
$S_1 \to \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2$
$S_2 \to \mathbf{if}\ B\ \mathbf{then}\ S \mid \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\},$
$S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{ S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \to S_1 \mid S_2$
$S_1 \to \text{if } B \text{ then } S_1 \text{ else } S_2$
$S_2 \to \text{if } B \text{ then } S \mid \text{if } B \text{ then } S_1 \text{ else } S_2 \},$
$S)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$$
\begin{aligned}
G_1 = \quad & (\{\, S, S_1, S_2, B\}, \{\mathbf{if}, \mathbf{then}, \mathbf{else}\}, \\
& \{S \to S_1 \mid S_2 \\
& S_1 \to \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2 \\
& S_2 \to \mathbf{if}\ B\ \mathbf{then}\ S \mid \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \}, \\
S)
\end{aligned}
$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \to S_1 \mid S_2$
$S_1 \to \text{if } B \text{ then } S_1 \text{ else } S_2$
$S_2 \to \text{if } B \text{ then } S \mid \text{if } B \text{ then } S_1 \text{ else } S_2\},$
$S)$

if *B* then if *B* then *S* else *S*

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 =$ $(\{S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \to S_1 \mid S_2$
$S_1 \to \text{if } B \text{ then } S_1 \text{ else } S_2$
$S_2 \to \text{if } B \text{ then } S \mid \text{if } B \text{ then } S_1 \text{ else } S_2 \},$
$S)$

S

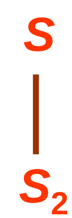
if *B* then if *B* then *S* else *S*

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$$
\begin{aligned}
G_1 = \quad & (\{ S, S_1, S_2, B\}, \{\textbf{if}, \textbf{then}, \textbf{else}\}, \\
& \{S \to S_1 \mid S_2 \\
& S_1 \to \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \\
& S_2 \to \textbf{if } B \textbf{ then } S \mid \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \}, \\
S)
\end{aligned}
$$

if B **then if** B **then** S **else** S



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$

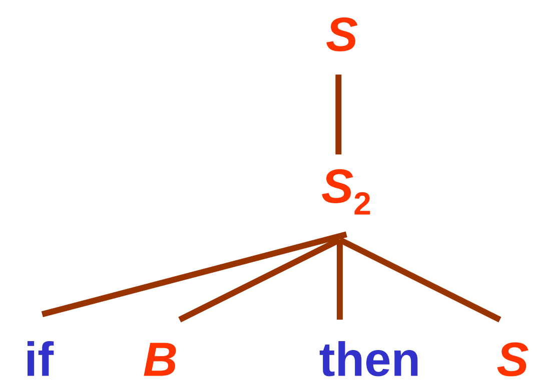
$\{S \to S_1 \mid S_2$

$S_1 \to \text{if } B \text{ then } S_1 \text{ else } S_2$

$S_2 \to \text{if } B \text{ then } S \mid \text{if } B \text{ then } S_1 \text{ else } S_2 \},$

$S)$

if *B* then if *B* then *S* else *S*

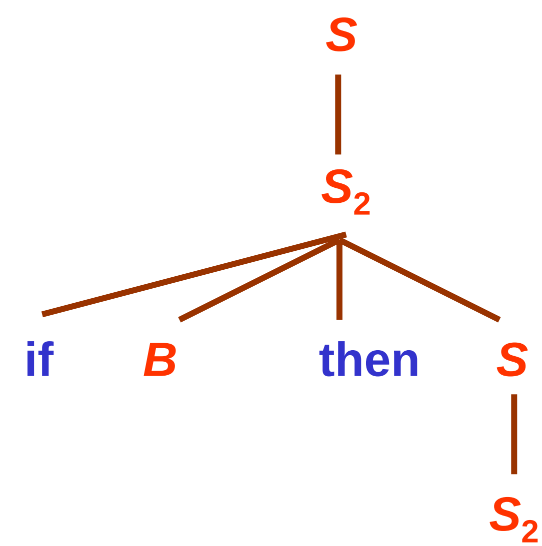


Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$$G_1 = (\{ S, S_1, S_2, B\}, \{\mathbf{if}, \mathbf{then}, \mathbf{else}\},$$
$$\{S \rightarrow S_1 \mid S_2$$
$$S_1 \rightarrow \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2$$
$$S_2 \rightarrow \mathbf{if}\ B\ \mathbf{then}\ S \mid \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \},$$
$$S)$$

if B **then if** B **then** S **else** S

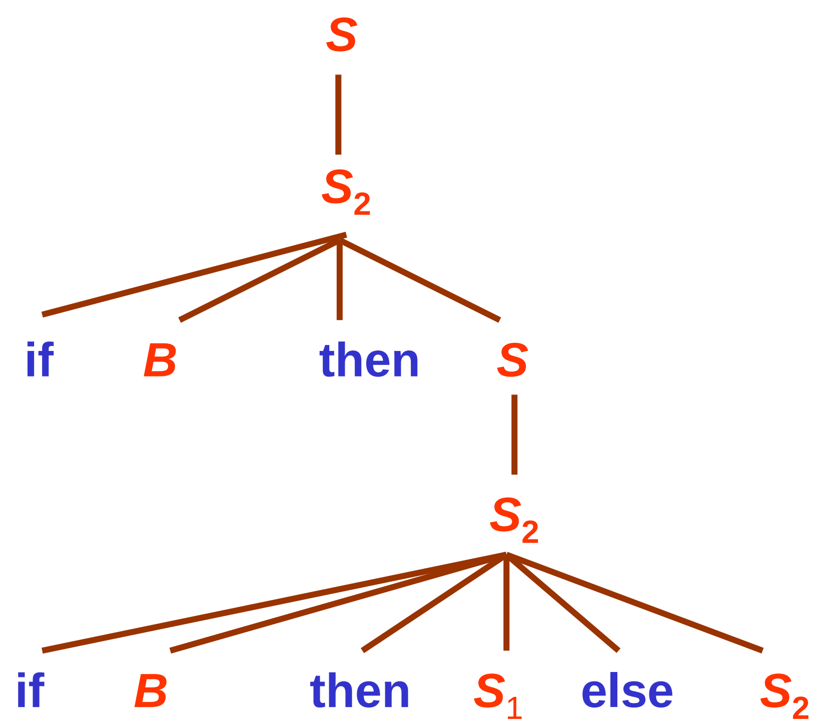


Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 =$ ({ S, S_1, S_2, B}, {if, then, else},
{$S \to S_1 \mid S_2$
$S_1 \to$ if B then S_1 else S_2
$S_2 \to$ if B then S | if B then S_1 else S_2 },
S)

if B then if B then S else S

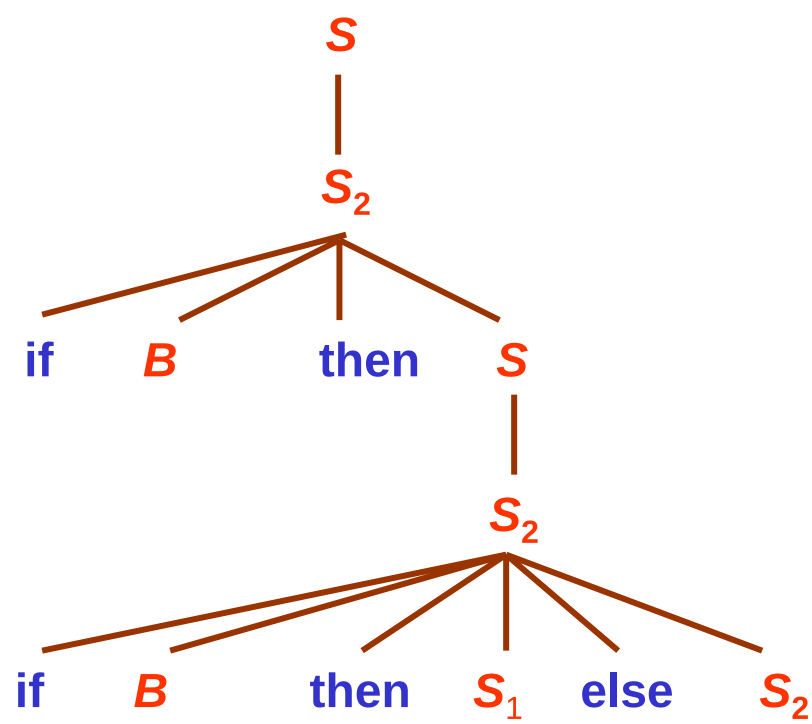


Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the grammar

$G_1 = (\{ S, S_1, S_2, B\}, \{\text{if}, \text{then}, \text{else}\},$
$\{S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow \text{if } B \text{ then } S_1 \text{ else } S_2$
$S_2 \rightarrow \text{if } B \text{ then } S \mid \text{if } B \text{ then } S_1 \text{ else } S_2 \},$
$S)$

if B then if B then S else S



$\underline{S}$
$\underline{S_2}$
if B then $\underline{S}$
if B then $\underline{S_2}$
if B then if B then S_1 else S_2

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a) \odot (a)) \odot (a)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a) \odot (a)) \odot (a)$

$(a) \odot ((a) \odot (a))$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \},$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$$G_3 = (\{\, E \,\}, \{\, a, \odot, (,) \,\}, \{\, E \rightarrow (E) \odot (E) \mid a \,\}, E)$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \},$
$\{ E \rightarrow (E) \odot (E) \mid a \}, E$
$)$

$((a)\odot(a))\odot(a)$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$$G_3 = (\{ E \}, \{ a, \odot, (,) \},$$
$$\{ E \rightarrow (E) \odot (E) \mid a \}, E$$
$$)$$

$\boxed{((a)\odot(a))\odot(a)}$

E

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

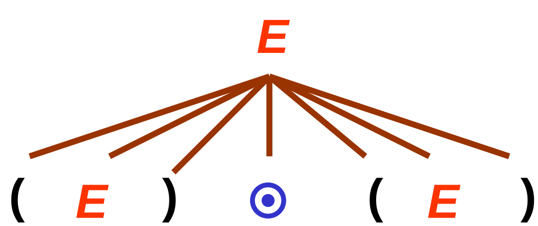
Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \}, \{ E \rightarrow (E) \odot (E) \mid a \}, E)$

$\boxed{((a)\odot(a))\odot(a)}$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

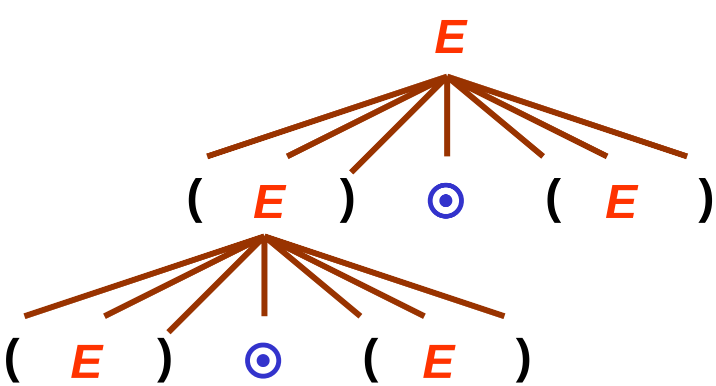
Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \}, \{ E \rightarrow (E) \odot (E) \mid a \}, E)$

$\boxed{((a)\odot(a))\odot(a)}$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

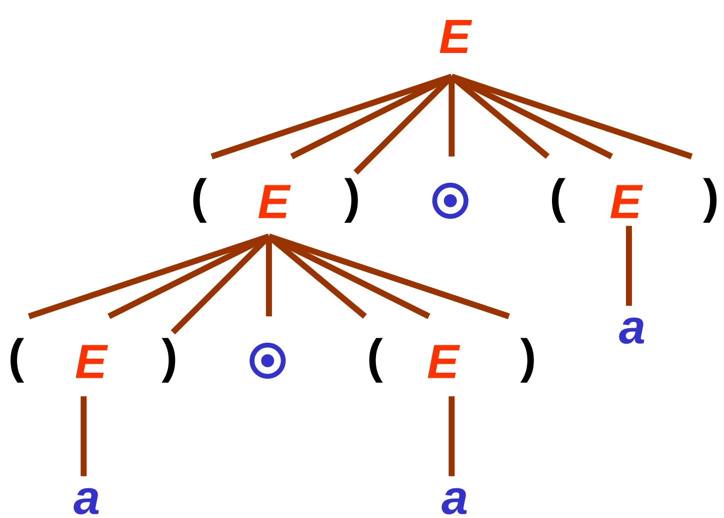
Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \},$
$\{ E \to (E) \odot (E) \mid a \}, E$
$)$

$((a)\odot(a))\odot(a)$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Changing the language

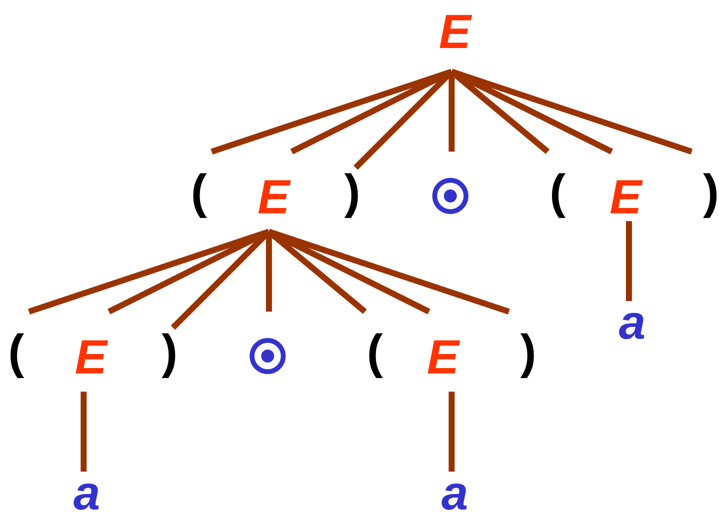
$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \}, \{ E \rightarrow (E) \odot (E) \mid a \}, E)$

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

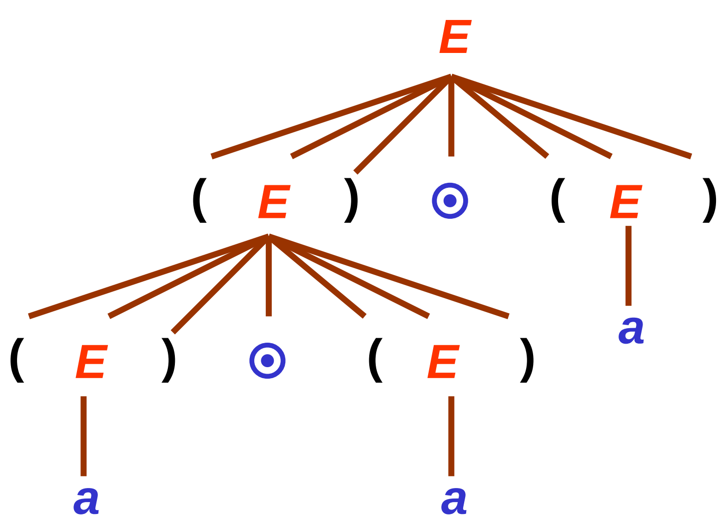
Changing the language

$((a)\odot(a))\odot(a)$

$(a)\odot((a)\odot(a))$

$G_3 = (\{\, E \,\}, \{\, a, \odot, (,) \,\},$
$\{\, E \to (E) \odot (E) \mid a \,\}, E$
$)$

$((a)\odot(a))\odot(a)$



$(a)\odot((a)\odot(a))$

E

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

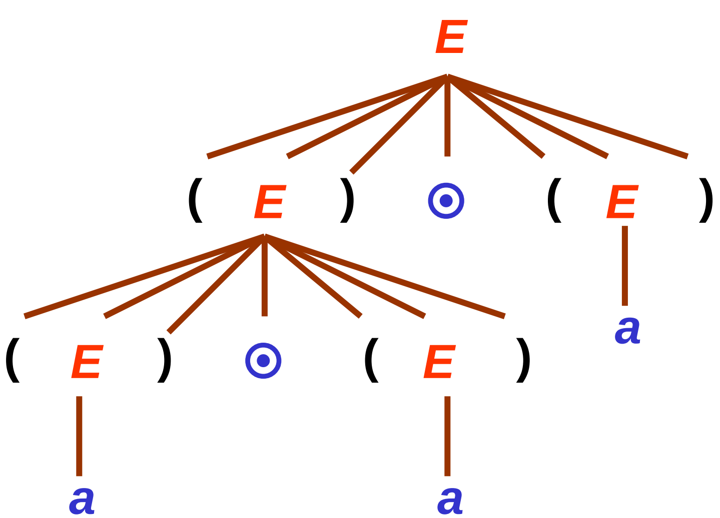
Changing the language

((a)⊙(a))⊙(a)

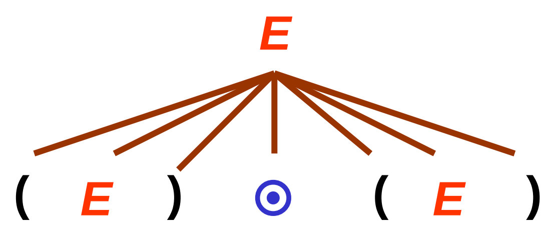
(a)⊙((a)⊙(a))

$$G_3 = (\{\, E \,\}, \{\, a, \odot, (,) \,\}, \{\, E \to (E) \odot (E) \mid a \,\}, E)$$

((a)⊙(a))⊙(a)



(a)⊙((a)⊙(a))



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

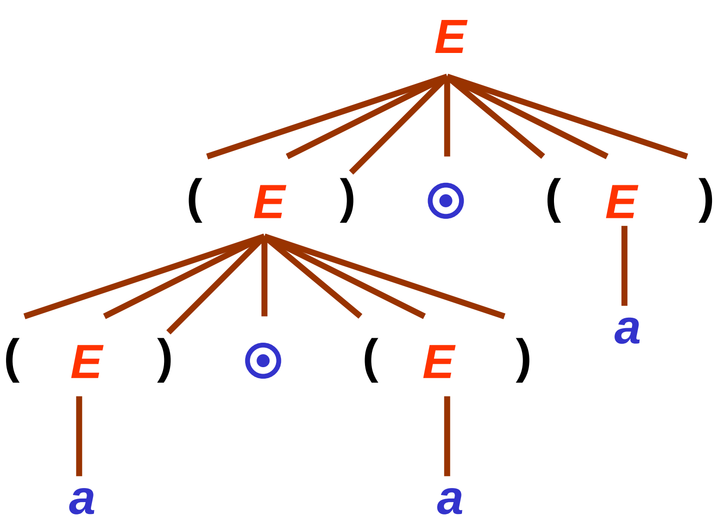
Changing the language

$((a)\odot(a))\odot(a)$

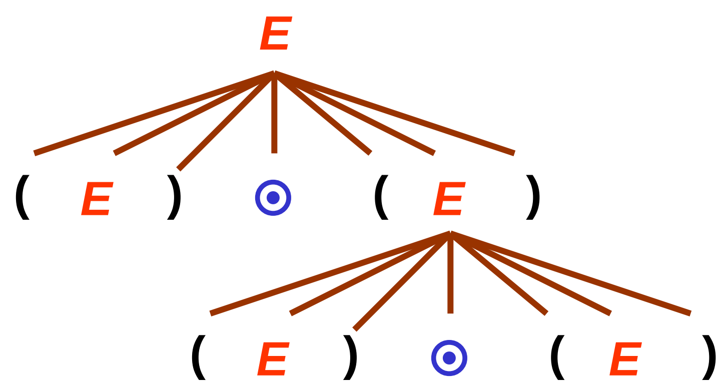
$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \},$
$\{ E \to (E) \odot (E) \mid a \}, E$
$)$

$((a)\odot(a))\odot(a)$



$(a)\odot((a)\odot(a))$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

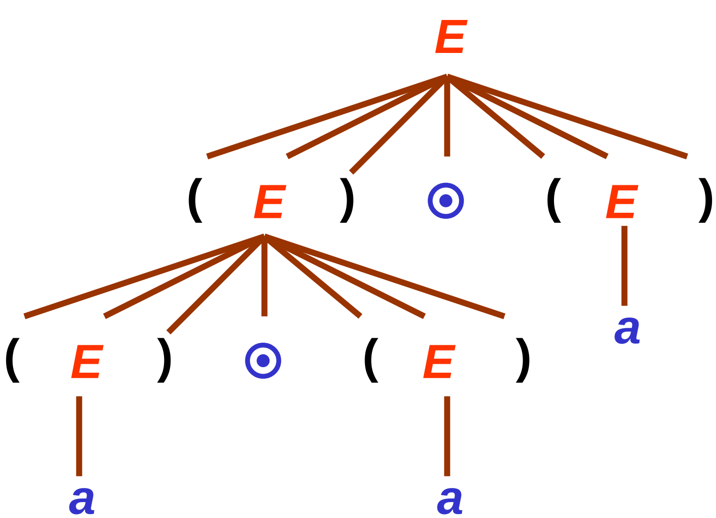
Changing the language

$((a)\odot(a))\odot(a)$

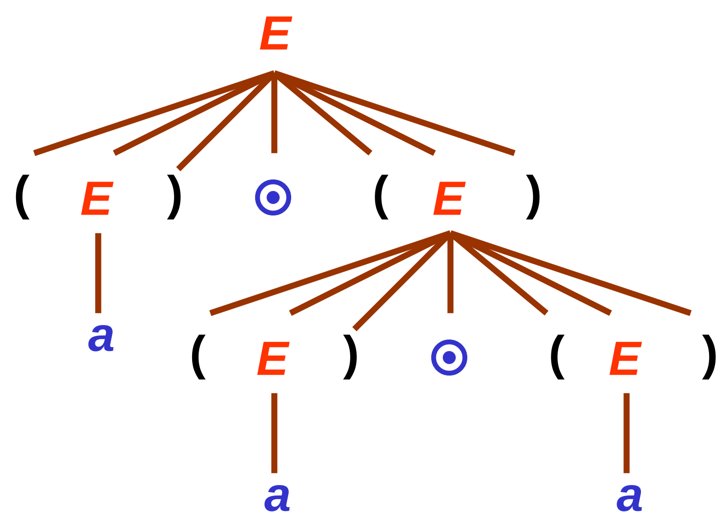
$(a)\odot((a)\odot(a))$

$G_3 = (\{ E \}, \{ a, \odot, (,) \}, \{ E \rightarrow (E) \odot (E) \mid a \}, E)$

$((a)\odot(a))\odot(a)$



$(a)\odot((a)\odot(a))$



Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of G is used in at least one derivation

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of G is used in at least one derivation

- **A symbol X of grammar $G = (V, T, P, S)$ is *useful***

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of G is used in at least one derivation

- **A symbol X of grammar $G = (V, T, P, S)$ is *useful***

$$S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of *G* is used in at least one derivation

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *useful***

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *alive***

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of G is used in at least one derivation

- **A symbol X of grammar $G = (V, T, P, S)$ is *useful***

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

- **A symbol X of grammar $G = (V, T, P, S)$ is *alive***

$$X \overset{*}{\Rightarrow} w_X$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of *G* is used in at least one derivation

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *useful***

$$S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$$

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *alive***

$$X \stackrel{*}{\Rightarrow} w_X$$

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *reachable***

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Goal**
  - any symbol of *G* is used in at least one derivation

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *useful***

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *alive***

$$X \overset{*}{\Rightarrow} w_X$$

- **A symbol *X* of grammar *G* = (*V*, *T*, *P*, *S*) is *reachable***

$$S \overset{*}{\Rightarrow} \alpha X \beta$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable ≠ useful

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable ≠ useful

$$X \overset{*}{\Rightarrow} w_X$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable ≠ useful

$$X \overset{*}{\Rightarrow} w_X$$

$$S \overset{*}{\Rightarrow} \alpha X \beta$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable ≠ useful

$$X \stackrel{*}{\Rightarrow} w_X$$

$$S \stackrel{*}{\Rightarrow} \alpha X \beta$$

At least one string α or β contains a dead symbol

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable $\neq$ useful

$$X \stackrel{*}{\Rightarrow} w_X$$

$$S \stackrel{*}{\Rightarrow} \alpha X \beta$$

At least one string α or β contains a dead symbol
There is no derivation

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

alive + reachable ≠ useful

$$X \stackrel{*}{\Rightarrow} w_X$$

$$S \stackrel{*}{\Rightarrow} \alpha X \beta$$

At least one string α or β contains a dead symbol
There is no derivation

$$S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**
  - productions of the form

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**
  - productions of the form

$$A \rightarrow B$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**
  - productions of the form

$$A \rightarrow B$$

- **ε-productions**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**
  - productions of the form

$$A \to B$$

- **ε-productions**
  - productions of the form

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Unit productions**
  - productions of the form

$$A \rightarrow B$$

- **ε-productions**
  - productions of the form

$$A \rightarrow \varepsilon$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

$$A \rightarrow BC$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

- **Greibach normal form**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

$$A \rightarrow BC$$
$$A \rightarrow a$$

- **Greibach normal form**
  - all productions have the form

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Simplifying a grammar

- **Chomsky normal form**
  - all productions have the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

- **Greibach normal form**
  - all productions have the form

$$A \rightarrow a\,\alpha$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \rightarrow a\ S\ a$

2) $S \rightarrow b\ A\ d$

3) $S \rightarrow c$

4) $A \rightarrow c\ B\ d$

5) $A \rightarrow a\ A\ d$

6) $B \rightarrow d\ A\ f$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \rightarrow a\ S\ a$
2) $S \rightarrow b\ A\ d$
3) $S \rightarrow c$
4) $A \rightarrow c\ B\ d$
5) $A \rightarrow a\ A\ d$
6) $B \rightarrow d\ A\ f$

$$A \Rightarrow \ldots\ldots\ B\ \ldots\ldots$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \rightarrow a\ S\ a$

2) $S \rightarrow b\ A\ d$

3) $S \rightarrow c$

4) $A \rightarrow c\ B\ d$

5) $A \rightarrow a\ A\ d$

6) $B \rightarrow d\ A\ f$

$$A \Rightarrow \ldots\ldots\ B\ \ldots\ldots$$

$$A \Rightarrow \ldots\ldots\ A\ \ldots\ldots$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \rightarrow a\ S\ a$
2) $S \rightarrow b\ A\ d$
3) $S \rightarrow c$
4) $A \rightarrow c\ B\ d$
5) $A \rightarrow a\ A\ d$
6) $B \rightarrow d\ A\ f$

$A \Rightarrow \ldots\ldots\ B\ \ldots\ldots$

$A \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

$B \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \rightarrow a\ S\ a$
2) $S \rightarrow b\ A\ d$
3) $S \rightarrow c$

$A \Rightarrow \ldots\ldots\ B\ \ldots\ldots$

$A \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

$B \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Example of dead symbols

1) $S \to a\ S\ a$

3) $S \to c$

$A \Rightarrow \ldots\ldots\ B\ \ldots\ldots$

$A \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

$B \Rightarrow \ldots\ldots\ A\ \ldots\ldots$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

If all right-hand symbols X_1, X_2, ..., X_n are alive:

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

If all right-hand symbols $X_1, X_2, \ldots, X_n$ are alive:

$$A \rightarrow X_1 X_2 \ldots X_n$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

If all right-hand symbols $X_1, X_2, \ldots, X_n$ are alive:

$$A \rightarrow X_1 X_2 \ldots X_n$$

then a left-hand variable A is also alive

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

If all right-hand symbols X_1, X_2, ..., X_n are alive:

$$A \rightarrow X_1 X_2 \ldots X_n$$

then a left-hand variable A is also alive

$$X_i \overset{*}{\Rightarrow} w_i,\ w_i \in T^*$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Property of alive symbols

If all right-hand symbols $X_1, X_2, \ldots, X_n$ are alive:

$$A \rightarrow X_1 X_2 \ldots X_n$$

then a left-hand variable A is also alive

$$X_i \overset{*}{\Rightarrow} w_i,\ w_i \in T^*$$

$$A \Rightarrow X_1 X_2 \ldots X_n \overset{*}{\Rightarrow} w_1 w_2 \ldots w_n$$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\, A\, B\, S$
2) $S \rightarrow b\, C\, A\, C\, d$
3) $A \rightarrow b\, A\, B$
4) $A \rightarrow c\, S\, A$
5) $A \rightarrow c\, C\, C$
6) $B \rightarrow b\, A\, B$
7) $B \rightarrow c\, S\, B$
8) $C \rightarrow c\, S$
9) $C \rightarrow c$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\,A\,B\,S$
2) $S \rightarrow b\,C\,A\,C\,d$
3) $A \rightarrow b\,A\,B$
4) $A \rightarrow c\,S\,A$
5) $A \rightarrow c\,C\,C$
6) $B \rightarrow b\,A\,B$
7) $B \rightarrow c\,S\,B$
8) $C \rightarrow c\,S$
9) $C \rightarrow c$

***AliveList* = { }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

AliveList = { C }

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C* }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \to a\ A\ B\ S$
2) $S \to b\ C\ A\ C\ d$
3) $A \to b\ A\ B$
4) $A \to c\ S\ A$
5) $A \to c\ C\ C$
6) $B \to b\ A\ B$
7) $B \to c\ S\ B$
8) $C \to c\ S$
9) $C \to c$

AliveList **= { *C*, *A* }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A* }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \to a\,A\,B\,S$
2) $S \to b\,C\,A\,C\,d$
3) $A \to b\,A\,B$
4) $A \to c\,S\,A$
5) $A \to c\,C\,C$
6) $B \to b\,A\,B$
7) $B \to c\,S\,B$
8) $C \to c\,S$
9) $C \to c$

***AliveList* = { *C*, *A*, *S* }**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\,A\,B\,S$
2) $S \rightarrow b\,C\,A\,C\,d$
3) $A \rightarrow b\,A\,B$
4) $A \rightarrow c\,S\,A$
5) $A \rightarrow c\,C\,C$
6) $B \rightarrow b\,A\,B$
7) $B \rightarrow c\,S\,B$
8) $C \rightarrow c\,S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A*, *S* }**

1) $S \rightarrow a\,A\,B\,S$
2) $S \rightarrow b\,C\,A\,C\,d$
3) $A \rightarrow b\,A\,B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

AliveList **= {** *C*, *A*, *S* **}**

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

AliveList = { C, A, S }

2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

AliveList **= { *C*, *A*, *S* }**

2) $S \rightarrow b\ C\ A\ C\ d$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A*, *S* }**

2) $S \rightarrow b\ C\ A\ C\ d$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A*, *S* }**

4) $A \rightarrow c\ S\ A$
2) $S \rightarrow b\ C\ A\ C\ d$
5) $A \rightarrow c\ C\ C$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A*, *S* }**

2) $S \rightarrow b\ C\ A\ C\ d$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

1) $S \rightarrow a\ A\ B\ S$
2) $S \rightarrow b\ C\ A\ C\ d$
3) $A \rightarrow b\ A\ B$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
6) $B \rightarrow b\ A\ B$
7) $B \rightarrow c\ S\ B$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

***AliveList* = { *C*, *A*, *S* }**

2) $S \rightarrow b\ C\ A\ C\ d$
4) $A \rightarrow c\ S\ A$
5) $A \rightarrow c\ C\ C$
8) $C \rightarrow c\ S$
9) $C \rightarrow c$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

OldAliveList **= ∅;**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = ∅;**
***NewAliveList* = { $A \mid A \rightarrow w$ and $w \in T^*$ };**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

OldAliveList **=** $\emptyset$**;**
NewAliveList **= {** $A \mid A \to w$ **and** $w \in T^*$ **};**

<u>while</u> (*OldAliveList* != *NewAliveList*)

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = $\emptyset$;**
***NewAliveList* = { $A \mid A \to w$ and $w \in T^*$ };**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = ∅;**
***NewAliveList* = { A | $A \rightarrow w$ and $w \in T^*$ };**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{
    ***OldAliveList* = *NewAliveList*;**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

OldAliveList **=** $\emptyset$**;**
NewAliveList **= {** $A \mid A \rightarrow w$ **and** $w \in T^*$ **};**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{

    OldAliveList **=** ***NewAliveList*****;**
    NewAliveList **=**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = $\varnothing$;**
***NewAliveList* = { $A \mid A \rightarrow w$ and $w \in T^*$ };**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{
    ***OldAliveList* = *NewAliveList*;**
    ***NewAliveList* =**
        OldAliveList $\cup$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = $\varnothing$;**
***NewAliveList* = { $A \mid A \rightarrow w$ and $w \in T^*$ };**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{
**   *OldAliveList* = *NewAliveList*;**
**   *NewAliveList* =**
**   *OldAliveList* $\cup$**
**   { $A \mid A \rightarrow \alpha$ and $\alpha \in (T \cup OldAliveList)^*$ };**

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

***OldAliveList* = $\emptyset$;**
***NewAliveList* = { $A \mid A \rightarrow w$ and $w \in T^*$ };**

<u>while</u> (*OldAliveList* != *NewAliveList*)
{
$\quad$ *OldAliveList* = *NewAliveList*;
$\quad$ *NewAliveList* =
$\quad\quad$ *OldAliveList* $\cup$
$\quad\quad\quad$ { $A \mid A \rightarrow \alpha$ and $\alpha \in (T \cup OldAliveList)^*$ };
}

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science

Computing the alive symbols

$OldAliveList = \emptyset;$
$NewAliveList = \{\ A \mid A \to w \text{ and } w \in T^*\ \};$

<u>**while**</u> **(*OldAliveList* != *NewAliveList*)**
{
$\quad OldAliveList = NewAliveList;$
$\quad NewAliveList =$
$\quad\quad OldAliveList \ \cup$
$\quad\quad\quad \{\ A \mid A \to \alpha \text{ and } \alpha \in (T \cup OldAliveList)^*\ \};$
}

$AliveList = NewAliveList;$

Copyright © 2022 S. Srbljić et al.: Introduction to Theoretical Computer Science