CS420

Introduction to the Theory of Computation

Lecture 8: Context-free grammars

Tiago Cogumbreiro

Attendance code: 5094

estalee.com

Last day to submit a grade of NA

Today we will learn...

- Accepting a string with a CFG
- Chomsky Normal Form
- Exercises
- Study closure under $\cap$
- Study closure under $\cdot$
- Study closure under *
- Converting a DFA into a CFG

Section 2.1

- Lecture notes by Prof. Sariel Har-Peled and Prof. Madhusudan Parthasarathy: courses.engr.illinois.edu/cs373/sp2009/lectures/lect_12.pdf
- Slides by Prof. Laura Kallmeyer: user.phil-fak.uni-duesseldorf.de/~kallmeyer/Parsing/cyk.pdf

Accepting a string with CFGs

Accepting a string with CFGs

CYK Algorithm

- Independently proposed by Cocke, Kasami and Younger in the 60s.
- Expects a **normalized** grammar
- Today we learn how to normalize a grammar so that a CYK can test acceptance
- The algorithm CYK is outside the scope of the course

Cocke and Schwartz (1970); Grune and Jacobs (2008); Hopcro and Ullman (1979, 1994); Kasami (1965); Younger (1967)

Chomsky Normal Form

Chomsky Normal Form (CNF)

Definition

Every production is in the form of $C \to XY$ or in the form of $C \to c$ where c is a single letter and X and Y are variables. Note the absence of productions yielding ϵ!

Why?

- Used in proofs and algorithms (such as in CYK)
- We will learn a series of steps that take a CFG and yields a CFG that is CNF

CNF algorithm

Let A, B, C, D be variables, c be terminals, and w, u, v are strings of variables/terminals.

1. **Add start:** Add $A \rightarrow B$, where A is new and A is the initial rule of the given grammar.
2. **Remove ϵ:** For all $A \rightarrow \epsilon$: for every occurrence of A in $B \rightarrow wAu$, add $B \rightarrow wu$
3. **Remove unit:** Remove all rules $A \rightarrow B$. For every rule removed $A \rightarrow B$ add a rule $A \rightarrow u$ if $B \rightarrow u$ and $A \rightarrow u$ was not removed.
4. **Make binary:** While there are rules $A \rightarrow BCDw$ add $A \rightarrow BA'$ and $A' \rightarrow CDw$ where A' is new.

Step 1: add start rule

Step 1: add start rule

Add $A \to B$, where A is new and A is the initial rule of the given grammar.

Before

$$S \to ASA \mid aB \mid a$$
$$A \to B \mid S \mid \epsilon$$
$$B \to b \mid \epsilon$$

Step 1: add start rule

Add $A \rightarrow B$, where A is new and A is the initial rule of the given grammar.

Before

$$S \rightarrow ASA \mid aB \mid a$$
$$A \rightarrow B \mid S \mid \epsilon$$
$$B \rightarrow b \mid \epsilon$$

After

$$S_0 \rightarrow S$$
$$A \rightarrow B \mid S \mid \epsilon$$
$$B \rightarrow b \mid \epsilon$$
$$S \rightarrow ASA \mid aB \mid a$$

Step 2: Remove ϵ

For all $A \to \epsilon$: for every occurrence of A in $B \to wAu$, add $B \to wu$

Before

$$S_0 \to S$$
$$S \to ASA \mid aB \mid a$$
$$A \to B \mid S \mid \epsilon$$
$$B \to b \mid \epsilon$$

Step 2: Remove ϵ

For all $A \to \epsilon$: for every occurrence of A in $B \to wAu$, add $B \to wu$

Before

$$S_0 \to S$$
$$S \to ASA \mid aB \mid a$$
$$A \to B \mid S \mid \epsilon$$
$$B \to b \mid \epsilon$$

After

$$S_0 \to S$$
$$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$$
$$A \to B \mid S$$
$$B \to b$$

> We must remove update every right-hand occurrence of A and B
>
> - from $S \to \mathbf{A}SA$, add $S \to SA$; then, from $S \to SA$, add $S \to S$
> - from $S \to AS\mathbf{A}$, add $S \to AS$; then, from $S \to AS$, add $S \to S$ (already there)
> - from $S \to aB$, add $S \to a$ (already there)
> - from $A \to \mathbf{B}$ we do not add $A \to \epsilon$ because ϵ

Step 3: Remove unit transitions

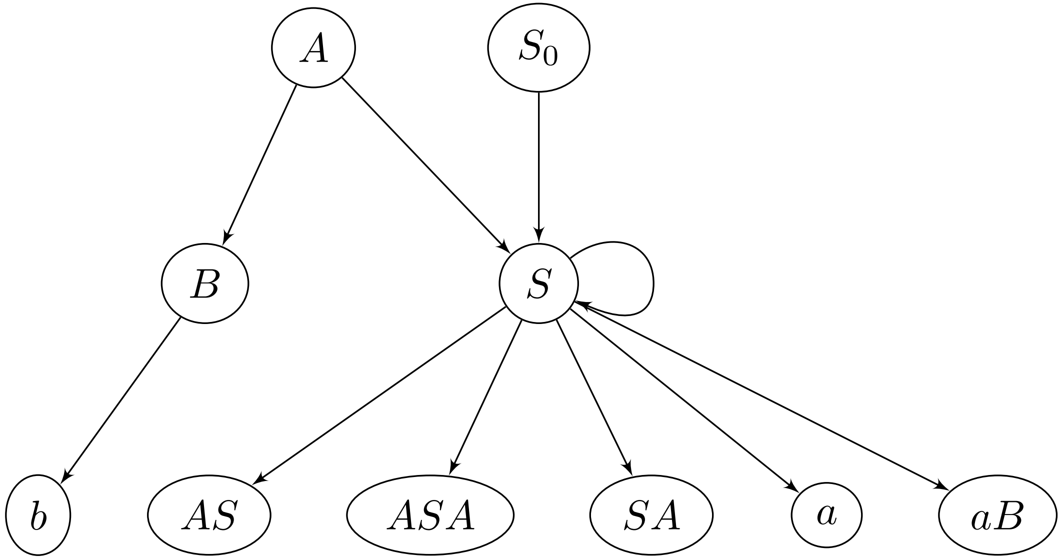
Step 3: Remove unit transitions

Visually

Draw a directed graph

- **the nodes** are variables, and the body of each rule
- **the arcs** go from each variable to each body of the rule

$$S_0 \to S$$
$$S \to AS \mid ASA \mid S \mid SA \mid a \mid aB$$
$$A \to B \mid S$$
$$B \to b$$



Step 3: Remove unit transitions

Visually

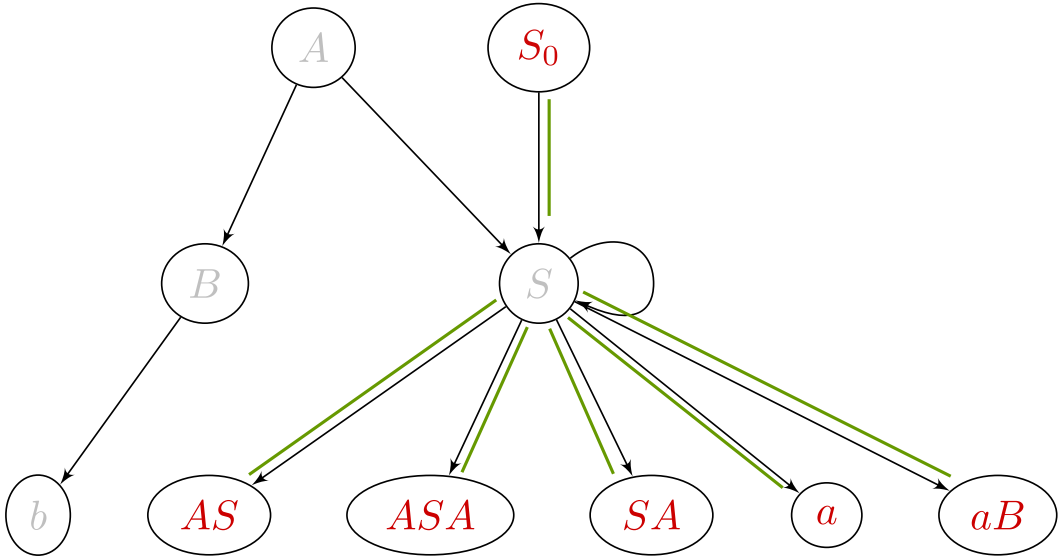
> The resulting grammar: for each variable create a production to each reachable leaf.

$$S_0 \to \underbrace{AS \mid ASA \mid SA \mid a \mid aB}_{S}$$

$$S \to$$

$$A \to$$

$$B \to$$



Step 3: Remove unit transitions

Visually

> The resulting grammar: for each variable create a production to each reachable leaf.

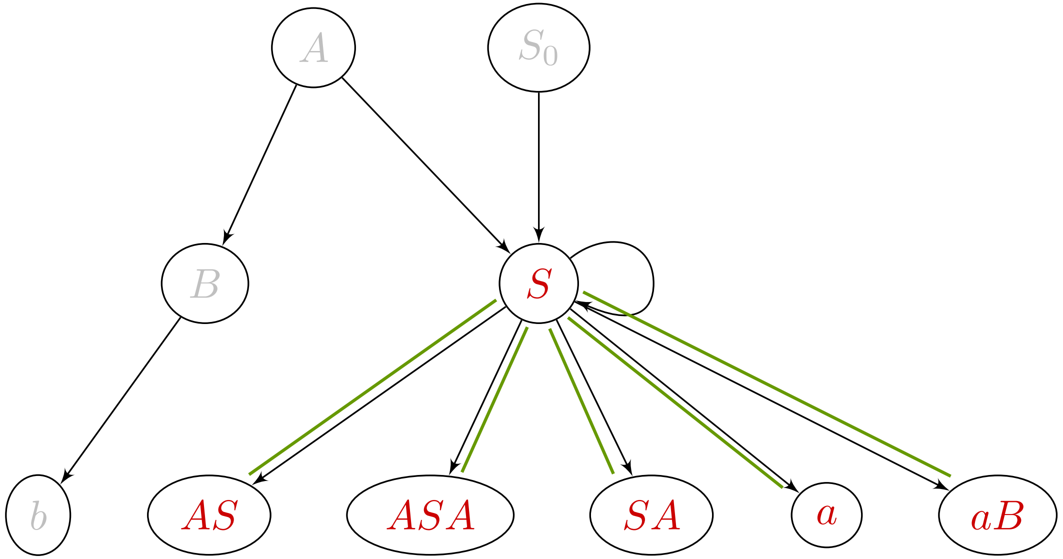
$S_0 \rightarrow AS \mid ASA \mid SA \mid a \mid aB$

$S \rightarrow AS \mid ASA \mid SA \mid a \mid aB$

$A \rightarrow$

$B \rightarrow$

(Note that we omit the self loop S.)



Step 3: Remove unit transitions

Visually

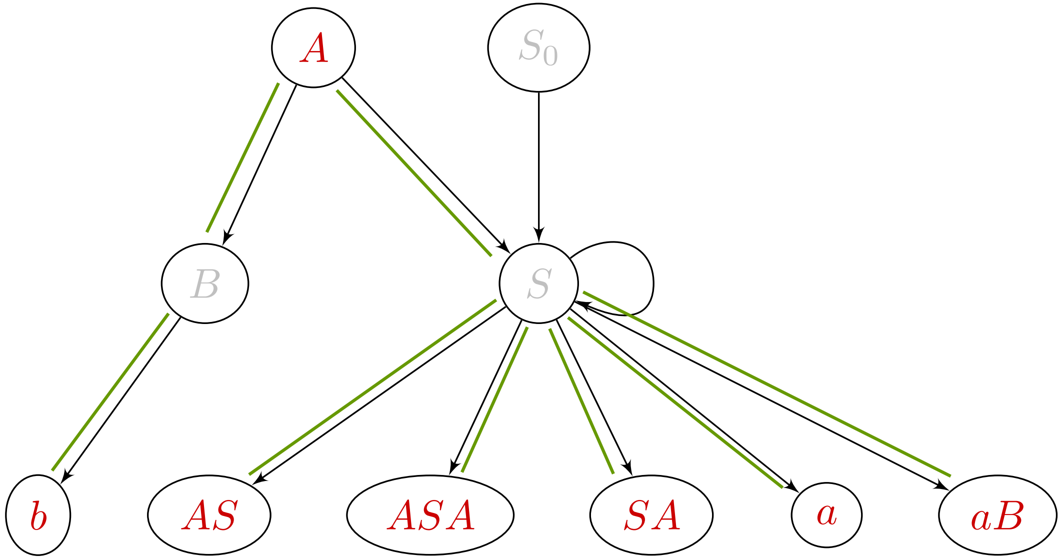
> The resulting grammar: for each variable create a production to each reachable leaf.

$$S_0 \to AS \mid ASA \mid SA \mid a \mid aB$$

$$S \to AS \mid ASA \mid SA \mid a \mid aB$$

$$A \to \underbrace{b}_{B} \mid \underbrace{AS \mid ASA \mid SA \mid a \mid aB}_{S}$$

$$B \to$$

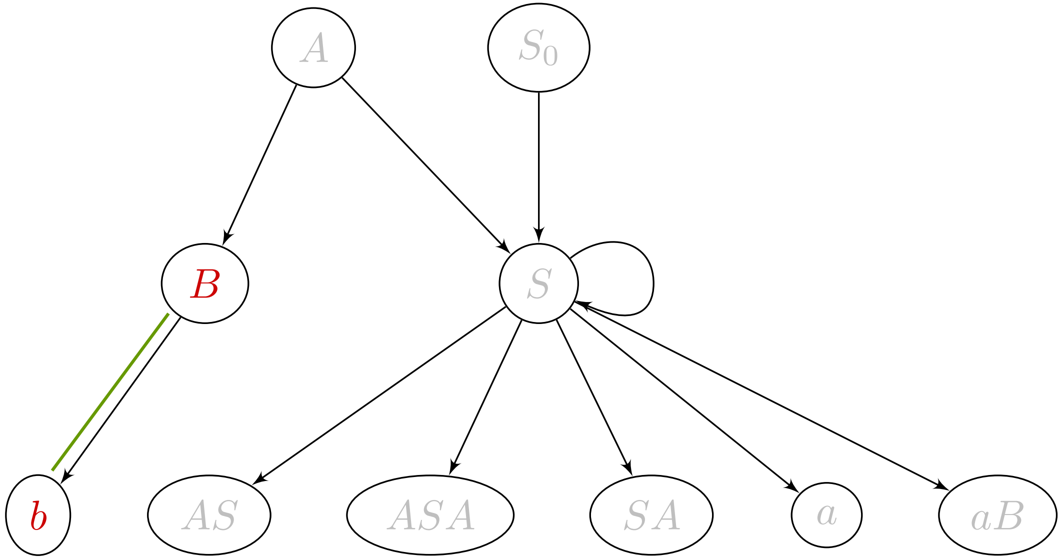


Step 3: Remove unit transitions

Visually

> The resulting grammar: for each variable create a production to each reachable leaf.

$$S_0 \to AS \mid ASA \mid SA \mid a \mid aB$$
$$S \to AS \mid ASA \mid SA \mid a \mid aB$$
$$A \to b \mid AS \mid ASA \mid SA \mid a \mid aB$$
$$B \to b$$



Step 3: remove unit transitions (formally)

Remove all rules $A \to B$. For every rule removed $A \to B$ add a rule $A \to u$ if $B \to u$ and $A \to u$ was not removed. Let $G = (V, \Gamma, R, S)$. Formally, the set of rules can be defined as

$$R_{new} = \left(R \cup \{X \to w \mid X \to^* A \in \mathcal{U}^* \wedge A \to w \in R\}\right) - \mathcal{U}^*$$

Unit pairs

We define the unit pair relation as:

$$\mathcal{U} = \{A \to B \mid A \to B \in G\}$$

Reachable unit pairs

The **transitive closure** is defined as usual

$$\frac{X \to^* Y \in \mathcal{U}^* \qquad Y \to^* Z \in \mathcal{U}^*}{X \to^* Z \in \mathcal{U}^*}$$

$$\frac{X \to Y \in \mathcal{U}}{X \to^* \in \mathcal{U}^*}$$

Step 3: Remove unit transitions

Remove all rules $A \to B$. For every rule removed $A \to B$ add a rule $A \to u$ if $B \to u$ and $A \to u$ was not removed.

Before

$$S_0 \to \underline{S}$$
$$S \to AS \mid ASA \mid \underline{S} \mid SA \mid a \mid aB$$
$$A \to \underline{B} \mid \underline{S}$$
$$B \to b$$

After

$$S_0 \to \underbrace{AS \mid ASA \mid SA \mid a \mid aB}_{S}$$
$$S \to AS \mid ASA \mid SA \mid a \mid aB$$
$$A \to \underbrace{b}_{B} \mid \underbrace{AS \mid ASA \mid SA \mid a \mid aB}_{S}$$
$$B \to b$$

> Let $A \to B$, $A \to S$, $S \to S$, and $S_0 \to S$ be the removed set $\mathcal{U}$. Here, $\mathcal{U}^* = \mathcal{U}$. Add the productions of S to A and to S_0. Add the productions of B to A.

Step 4: Make binary

Step 4: Make binary

We "break down" every rule with more than two variables/terminals. Next, we replace every terminal in rules that have two variables.

Before

$$A \to AS \mid ASA \mid SA \mid a \mid aB \mid b$$

$$B \to b$$

$$S \to AS \mid ASA \mid SA \mid a \mid aB$$

$$S_0 \to AS \mid ASA \mid SA \mid a \mid aB$$

Step 4: Make binary

We "break down" every rule with more than two variables/terminals. Next, we replace every terminal in rules that have two variables.

Before

$$A \to AS \mid ASA \mid SA \mid a \mid aB \mid b$$
$$B \to b$$
$$S \to AS \mid ASA \mid SA \mid a \mid aB$$
$$S_0 \to AS \mid ASA \mid SA \mid a \mid aB$$

After

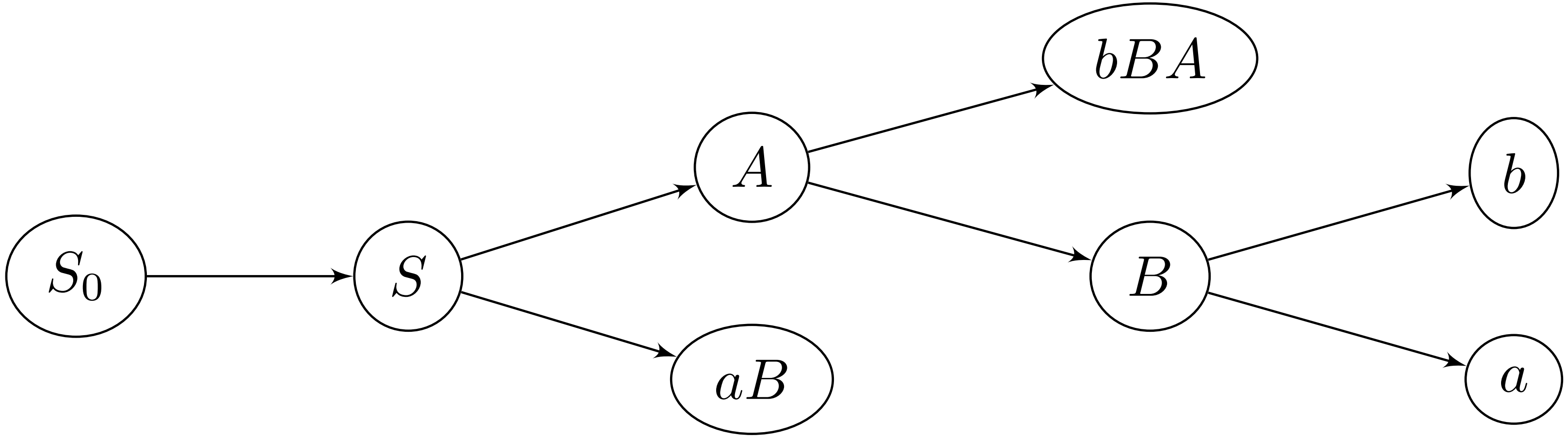
$$A \to AS \mid AX_1 \mid SA \mid a \mid X_aB \mid b$$
$$B \to b$$
$$S \to AS \mid AX_2 \mid SA \mid a \mid X_aB$$
$$S_0 \to AS \mid AX_3 \mid SA \mid a \mid X_aB$$
$$X_1 \to SA$$
$$X_2 \to SA$$
$$X_3 \to SA$$
$$X_a \to a$$
$$X_b \to b$$

More on removing unit transitions

On unit transitions and transitivity

Example 2

$$S_0 \to S$$
$$S \to A \mid aB$$
$$A \to bBA \mid B$$
$$B \to b \mid a$$

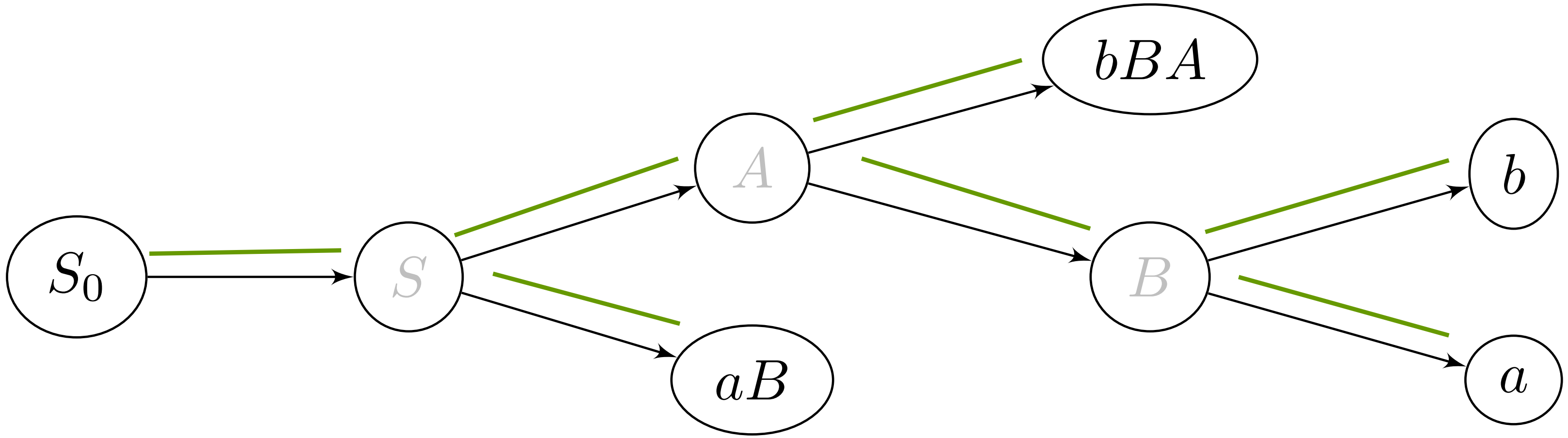


What do you think is the resulting grammar?

More on removing unit transitions

On unit transitions and transitivity

Example 2 (S_0)

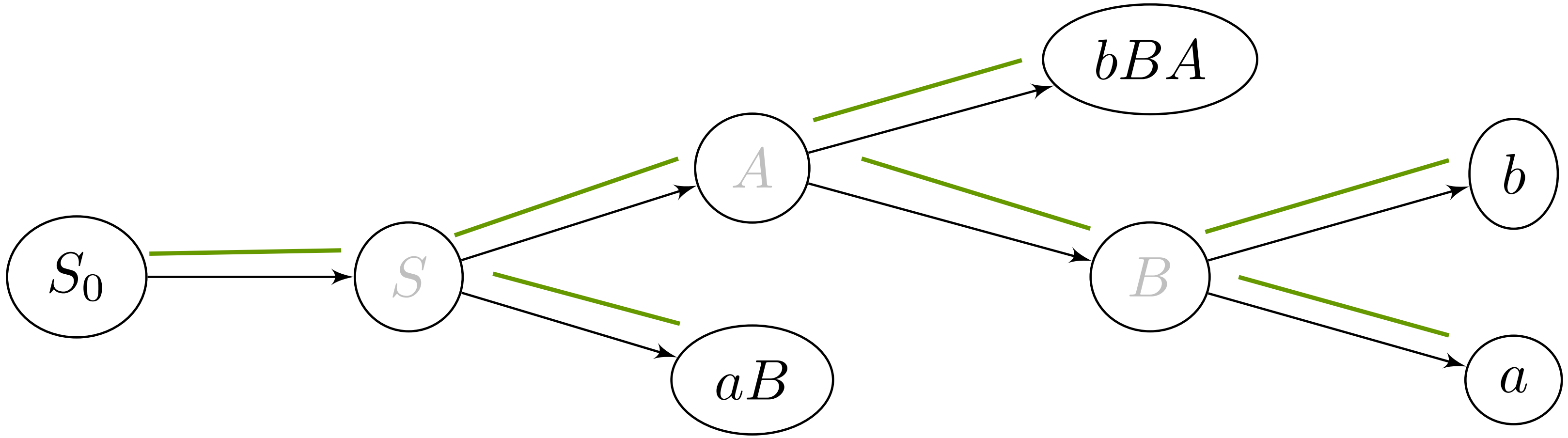


$S_0 \rightarrow$

More on removing unit transitions

On unit transitions and transitivity

Example 2 (S_0)



$$S_0 \to \underbrace{bBA}_{A} \mid \underbrace{b \mid a}_{B} \mid \underbrace{aB}_{S}$$

$$S \to$$

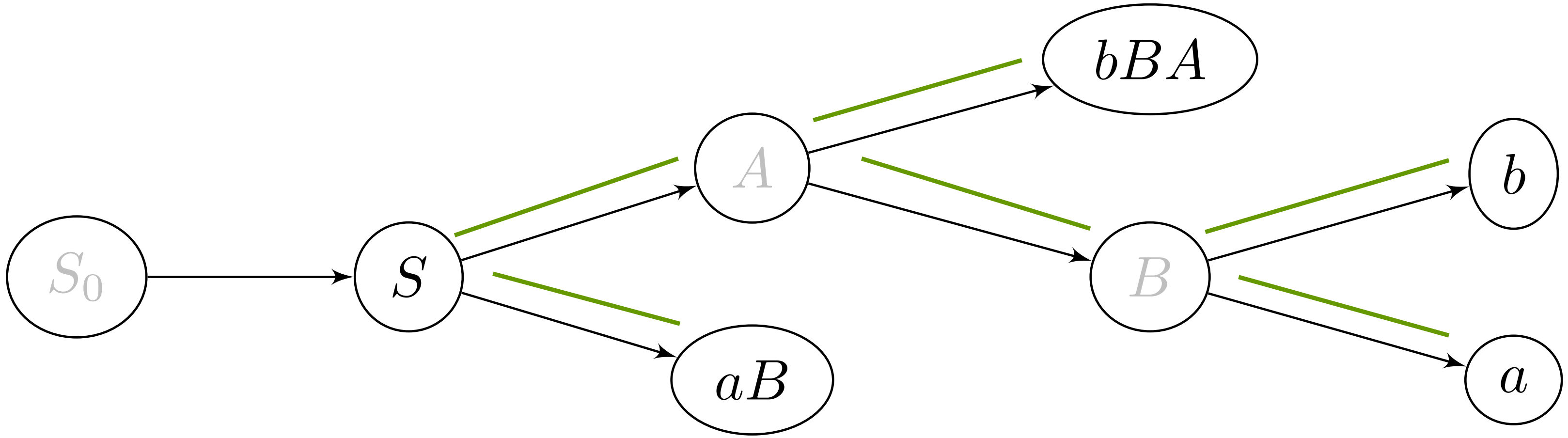
$$A \to$$

$$B \to$$

More on removing unit transitions

On unit transitions and transitivity

Example 2 ($\boldsymbol{S}$)



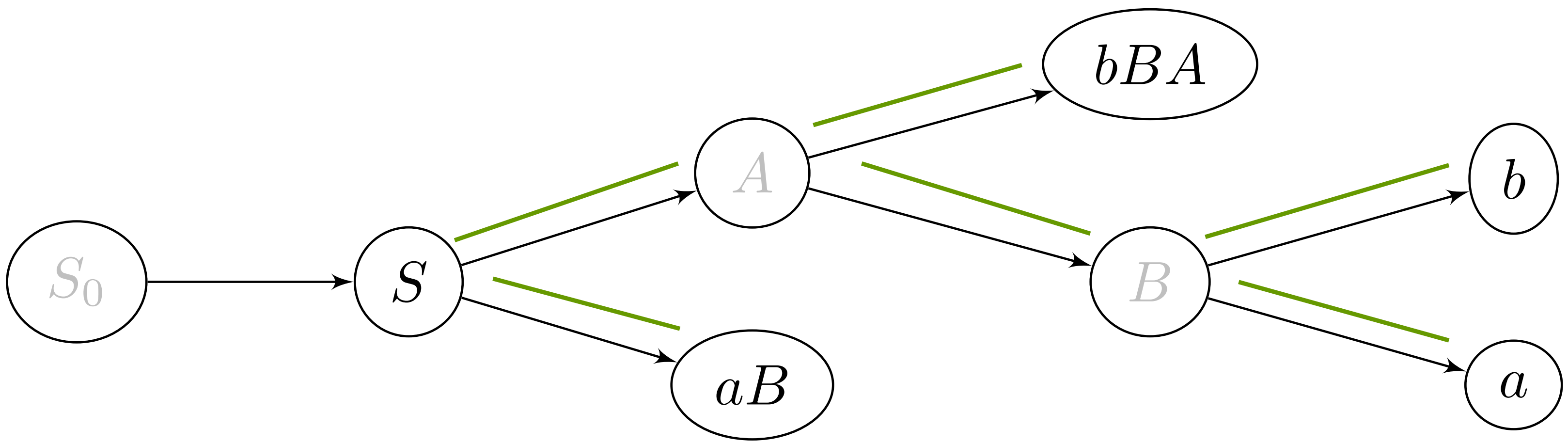
$S_0 \to A \mid b \mid a \mid aB$

$\boldsymbol{S} \to$

More on removing unit transitions

On unit transitions and transitivity

Example 2 (S)



$S_0 \to A \mid b \mid a \mid aB$

$S \to \underbrace{bBA}_{A} \mid \underbrace{b \mid a}_{B} \mid \underbrace{aB}_{S}$

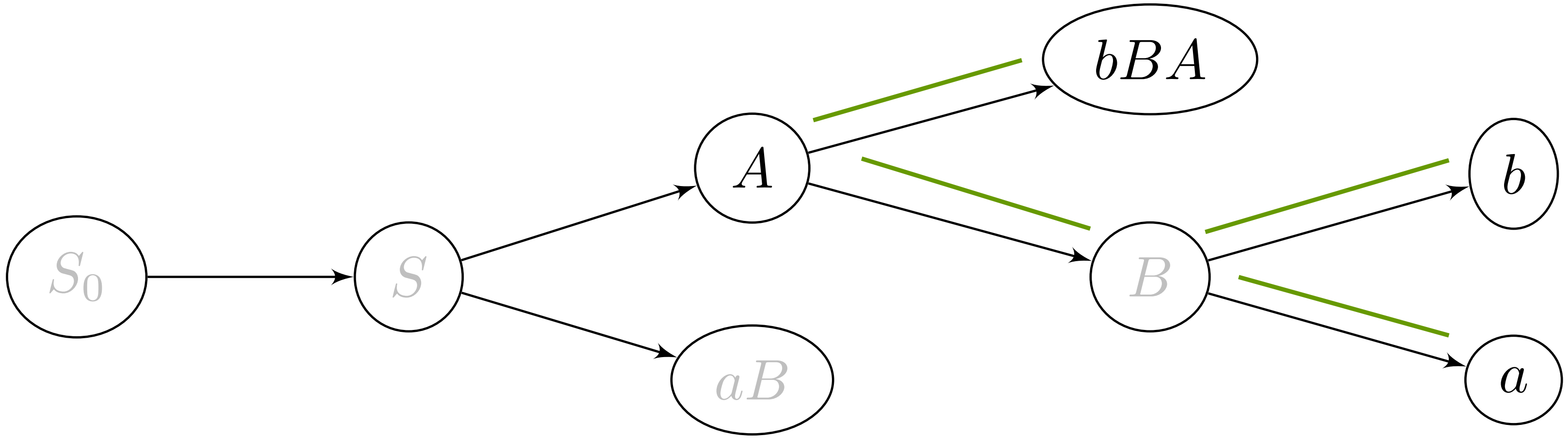
$A \to$

$B \to$

More on removing unit transitions

On unit transitions and transitivity

Example 2 (A)

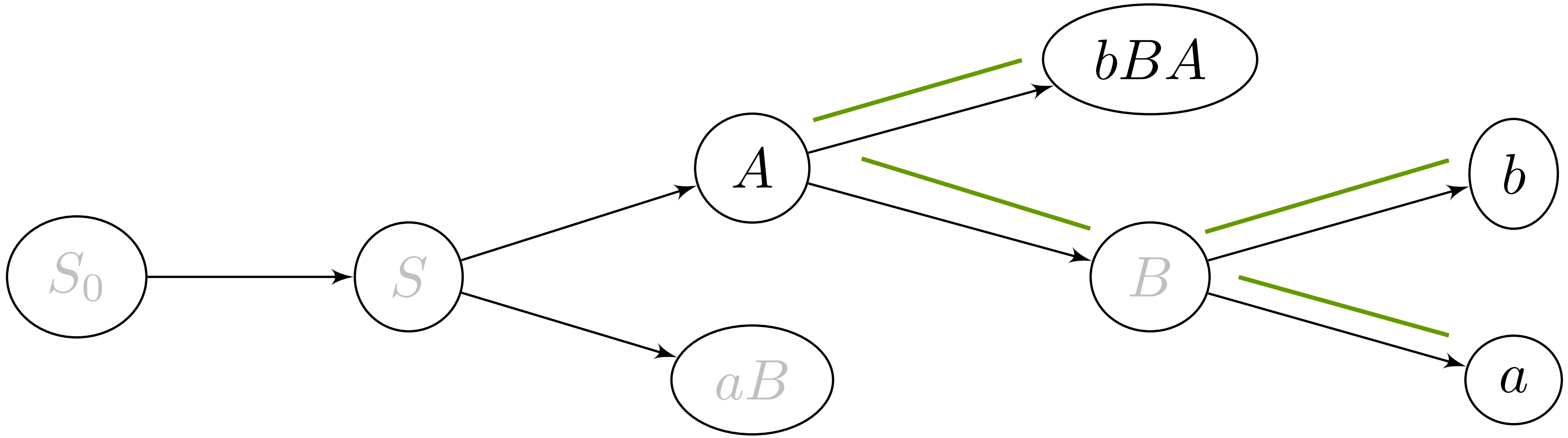


$S_0 \to A \mid b \mid a \mid aB$
$S \to A \mid b \mid a \mid aB$
$A \to$

More on removing unit transitions

On unit transitions and transitivity

Example 2 ($\boldsymbol{A}$)



$S_0 \to A \mid b \mid a \mid aB$

$S \to A \mid b \mid a \mid aB$

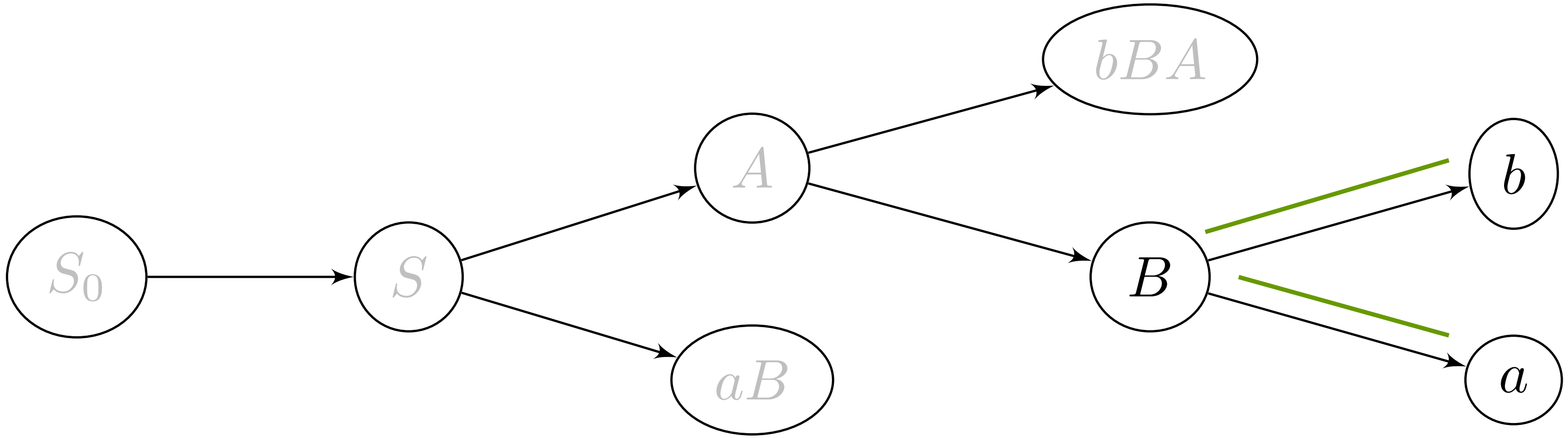
$A \to \underbrace{bBA}_{A} \mid \underbrace{b \mid a}_{B}$

$B \to$

More on removing unit transitions

On unit transitions and transitivity

Example 2 (B)

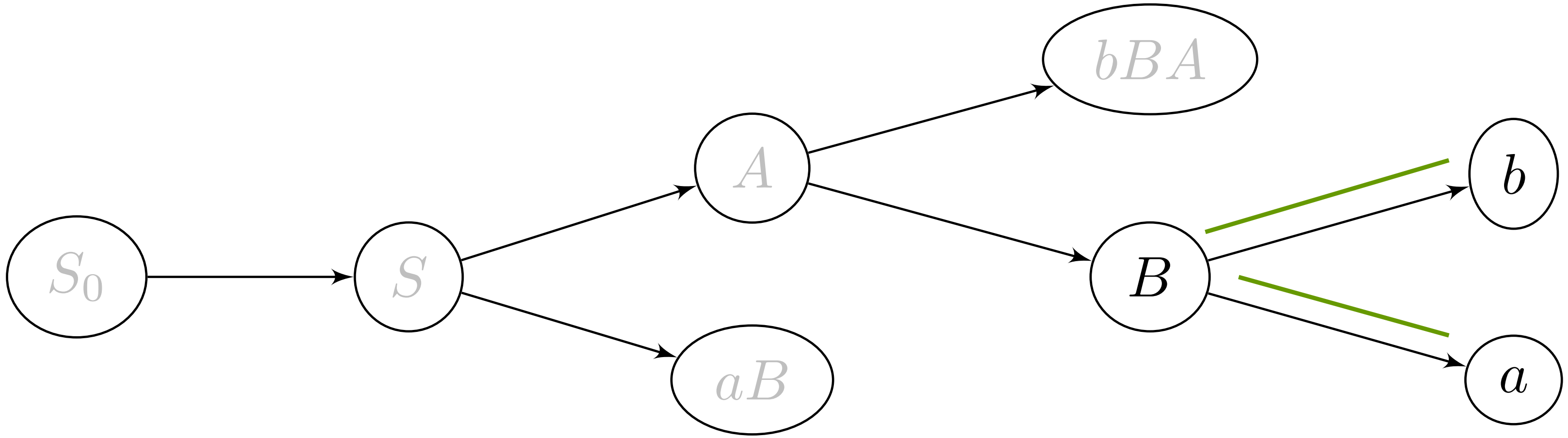


$S_0 \to A \mid b \mid a \mid aB$
$S \to A \mid b \mid a \mid aB$
$A \to bBA \mid b \mid a$
$B \to$

More on removing unit transitions

On unit transitions and transitivity

Example 2 (B)



$$S_0 \to A \mid b \mid a \mid aB$$
$$S \to A \mid b \mid a \mid aB$$
$$A \to bBA \mid b \mid a$$
$$B \to b \mid a$$

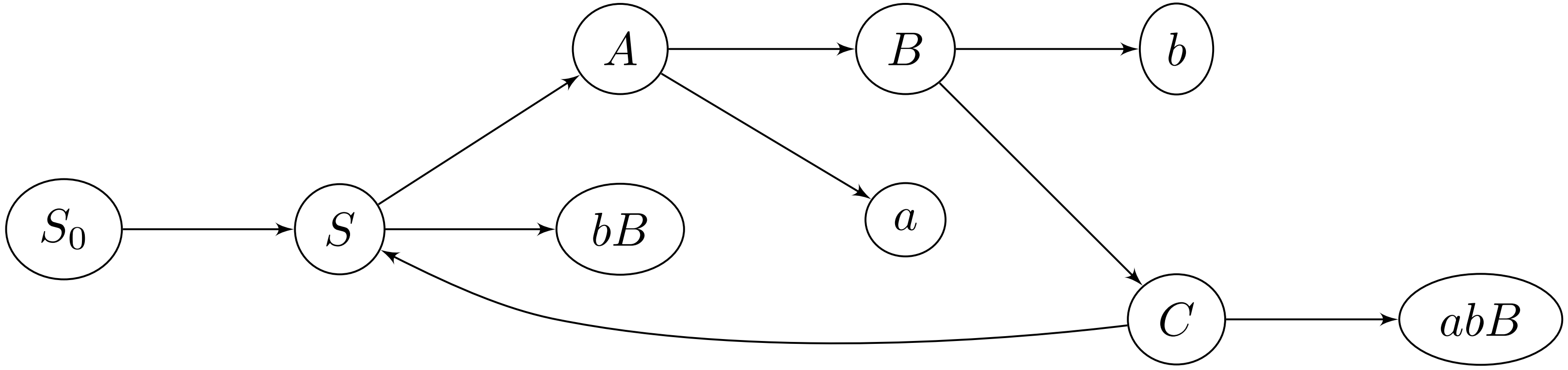
> We must take into consideration all possible paths via unit-edges.

More on removing unit transitions

On unit transitions with loops

Example 3

$$S_0 \to S$$
$$S \to A \mid bB$$
$$A \to B \mid a$$
$$B \to b \mid C$$
$$C \to abB \mid S$$

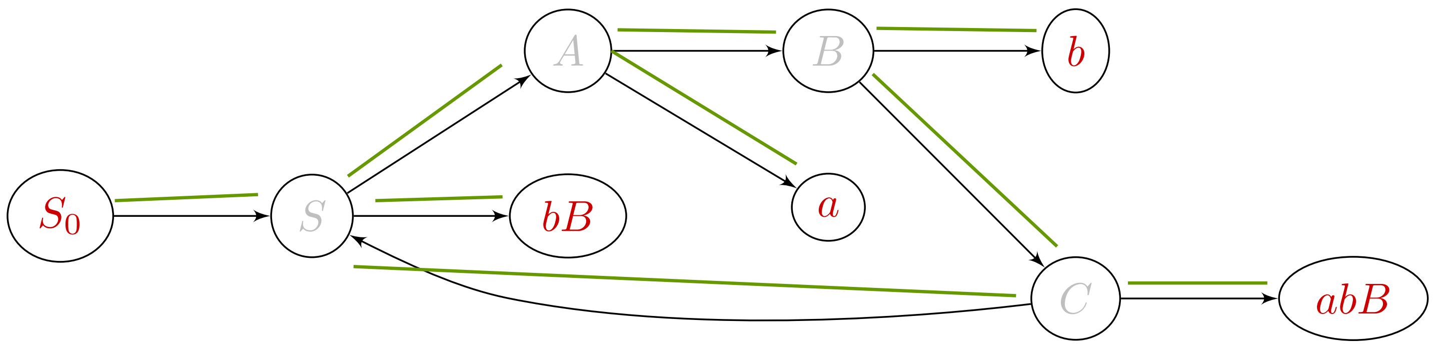


What do you think is the resulting grammar?

More on removing unit transitions

On unit transitions with loops

Example 3 (S_0)

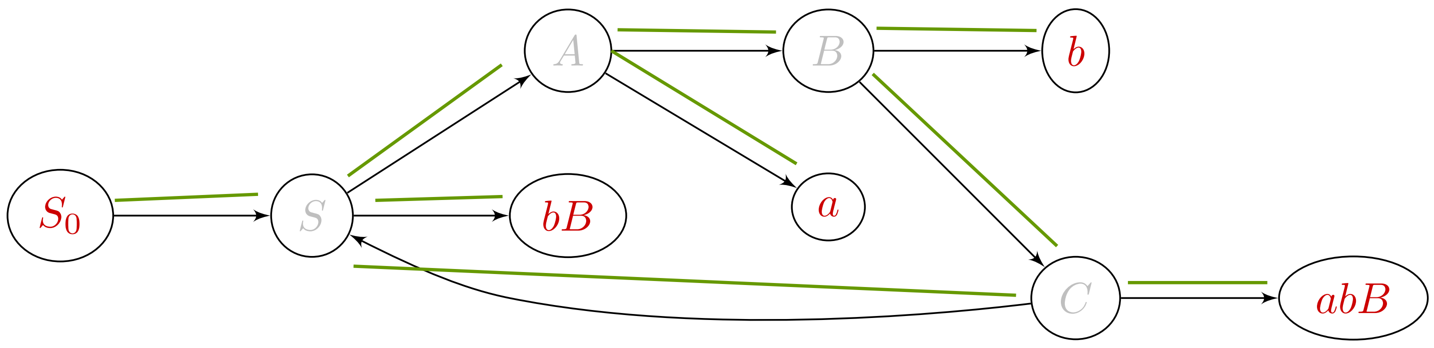


$S_0 \to$

More on removing unit transitions

On unit transitions with loops

Example 3 (S_0)



$$S_0 \to \underbrace{bB}_{S} \mid \underbrace{a}_{A} \mid \underbrace{b}_{B} \mid \underbrace{abB}_{C}$$

$S \to$

$A \to$

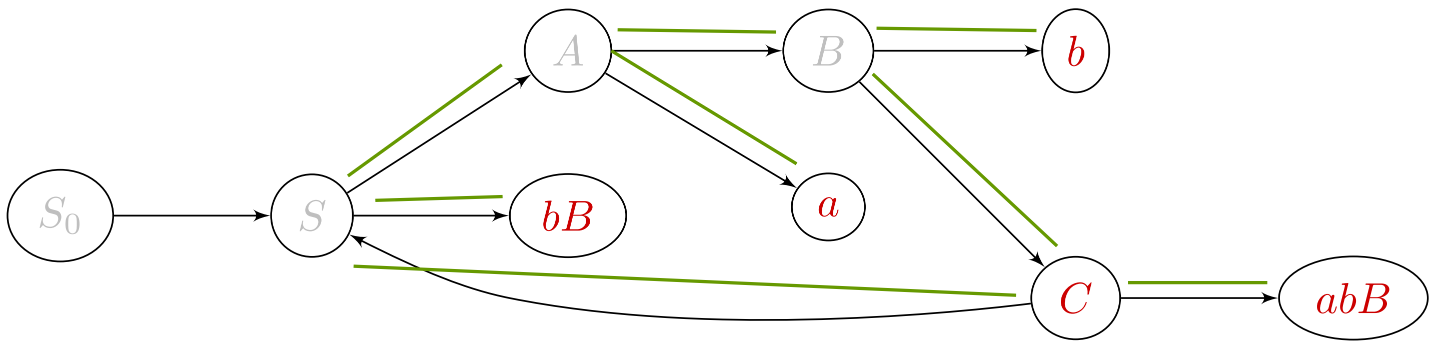
$B \to$

$C \to$

More on removing unit transitions

On unit transitions with loops

Example 3 (C)



$$S_0 \rightarrow \underbrace{bB}_{S} \mid \underbrace{a}_{A} \mid \underbrace{b}_{B} \mid \underbrace{abB}_{C}$$

$S \rightarrow$

$A \rightarrow$

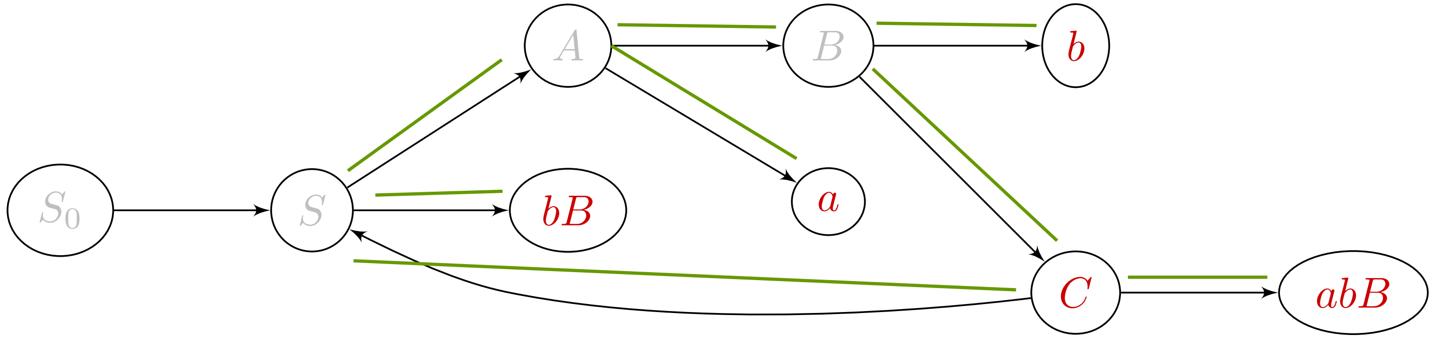
$B \rightarrow$

$C \rightarrow$

More on removing unit transitions

On unit transitions with loops

Example 3 (C)



$$S_0 \to \underbrace{bB}_{S} \mid \underbrace{a}_{A} \mid \underbrace{b}_{B} \mid \underbrace{abB}_{C}$$

$$S \to$$

$$A \to$$

$$B \to$$

$$C \to \underbrace{bB}_{S} \mid \underbrace{a}_{A} \mid \underbrace{b}_{B} \mid \underbrace{abB}_{C}$$

Note that we musth handle loops. All variables in the loop have the same productions.

CYK Deductively

The accept algorithm for CFGs

CYK Deductively

$$\frac{w(i) = c \qquad A \to c}{A \in \mathrm{CYK}(i, 1)}$$

$$\frac{A \to BC \quad B \in \mathrm{CYK}(i, l_1) \quad C \in \mathrm{CYK}(i + l_1, l_2)}{A \in \mathrm{CYK}(i, l_1 + l_2)}$$

$$\frac{|w| = N \quad \mathrm{CYK}(1, N) \neq \emptyset}{G \text{ accepts } w}$$

> Formalization based on Laura Kallmeyer's slides.

Exercises

Exercise

> Write a CFG whose language is $\{w \mid w \text{ is a palindrome}\}$. We say that string w is a palindrome if $w = \mathrm{reverse}(w)$. Let us use the alphabet $\{0, 1\}$.

Exercise

> Write a CFG whose language is $\{w \mid w \text{ is a palindrome}\}$. We say that string w is a palindrome if $w = \text{reverse}(w)$. Let us use the alphabet $\{0, 1\}$.

Solution

$$S \rightarrow 0S0 \mid 1S1 \mid 1 \mid 0 \mid \epsilon$$

Exercise

> Write a CFG whose language is $\{w \mid w \text{ is a palindrome}\}$. We say that string w is a palindrome if $w = \text{reverse}(w)$. Let us use the alphabet $\{0, 1\}$.

Solution

$S \to 0S0 \mid 1S1 \mid 1 \mid 0 \mid \epsilon$

Exercise

> What if our alphabet is $\{0, 1, 2, 3, 4, 5, 6\}$?

Exercise

> Write a CFG whose language is $\{w \mid w \text{ is a palindrome}\}$. We say that string w is a palindrome if $w = \text{reverse}(w)$. Let us use the alphabet $\{0, 1\}$.

Solution

$$S \rightarrow 0S0 \mid 1S1 \mid 1 \mid 0 \mid \epsilon$$

Exercise

> What if our alphabet is $\{0, 1, 2, 3, 4, 5, 6\}$?

$$S \rightarrow 0S0 \mid 1S1 \mid 2S2 \mid 3S3 \mid 4S4 \mid 5S5 \mid 6S6 \mid A \mid \epsilon$$
$$A \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6$$

(The rule $S \rightarrow ASA$ would not work because we would not be able to enforce both A's to be equal.)

The union operator

Example

Grammar 1

> A grammar that recognizes well-balanced brackets

$$B \to \{B\} \mid \epsilon \mid BB$$

Grammar 2

> A grammar that recognizes well-balanced parenthesis

$$P \to (P) \mid \epsilon \mid PP$$

Exercise:

> Write a grammar that recognizes either well-balanced parenthesis or well-balanced brackets

Example

Grammar 1

> A grammar that recognizes well-balanced brackets

$$B \to \{B\} \mid \epsilon \mid BB$$

Grammar 2

> A grammar that recognizes well-balanced parenthesis

$$P \to (P) \mid \epsilon \mid PP$$

Exercise:

> Write a grammar that recognizes either well-balanced parenthesis or well-balanced brackets

$$A \to B \mid P$$

The union operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$ and $B = (V_B, \Sigma_B, R_B, S_B)$, how do we define $A \cup B$?

The union operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$ and $B = (V_B, \Sigma_B, R_B, S_B)$, how do we define $A \cup B$?

1. Let $V_A \cap V_B = \emptyset$
2. Let $S \notin (V_A \cup V_B \cup \Sigma_A \cup \Sigma_B)$
3. $A \cup B = (V_A \cup V_B \cup \{S\}, \Sigma_A \cup \Sigma_B, R_A \cup R_B \cup \{S \to S_A, S \to S_B\}, S)$

The union operator is closed under context-free languages!

The concat operator

Example

Grammar 1

$$L(A) = \{0^n 1^n \mid n \geq 0\}$$

Grammar 2

$$L(B) = \{1^n 0^n \mid n \geq 0\}$$

Exercise 2:

> Write a grammar that recognizes $\{0^m 1^m 1^n 0^n \mid n \geq 0 \wedge m \geq 0\}$

Example

Grammar 1

$$L(A) = \{0^n 1^n \mid n \geq 0\}$$

Grammar 2

$$L(B) = \{1^n 0^n \mid n \geq 0\}$$

Exercise 2:

> Write a grammar that recognizes $\{0^m 1^m 1^n 0^n \mid n \geq 0 \wedge m \geq 0\}$

$$S \to AB$$

The concat operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$ and $B = (V_B, \Sigma_B, R_B, S_B)$, how do we define $A \cdot B$?

The concat operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$ and $B = (V_B, \Sigma_B, R_B, S_B)$, how do we define $A \cdot B$?

1. Let $V_A \cap V_B = \emptyset$
2. Let $S \notin (V_A \cup V_B \cup \Sigma_A \cup \Sigma_B)$
3. $A \cdot B = (V_A \cup V_B \cup \{S\}, \Sigma_A \cup \Sigma_B, R_A \cup R_B \cup \{S \to S_A S_B\}, S)$

The concat operator is closed under context-free languages!

The star operator

Example

$A \to \{A\} \mid \epsilon$

How would we represent $\boldsymbol{A}^\star$?

Example

$A \rightarrow \{A\} \mid \epsilon$

How would we represent $A^\star$?

$S \rightarrow AS \mid \epsilon$

The concat operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$, how do we define $A^\star$?

The concat operator

Let $A = (V_A, \Sigma_A, R_A, S_A)$, how do we define $A^\star$?

1. Let $S \notin \left(V_A \cup \Sigma_A\right)$
2. $A^\star = (V_A \cup \{S\}, \Sigma_A, R_A \cup \{S \to S_A S, S \to \epsilon\}, S)$

The star operator is closed under context-free languages!

Convert a DFA into a CFG

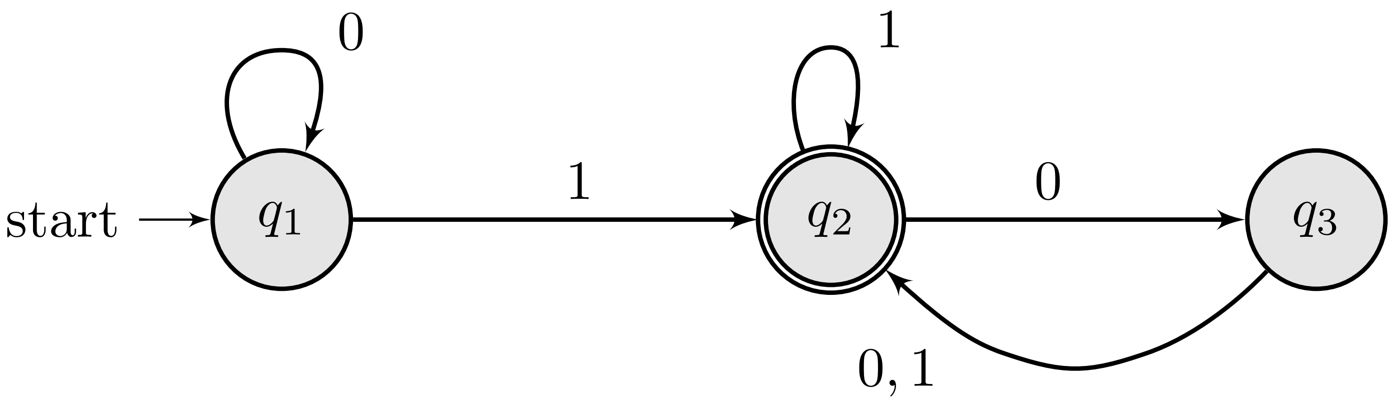
Convert a DFA into a CFG

```
def dfa_to_cfg(dfa):
  nodes, edges = dfa.as_graph()
  variables = set()
  terminals = set()
  rules = []
  start = dfa.start_state
  for q in dfa.end_states:
    rules.append(Rule(q, []))
  for ((src,dst), elems) in edges.items():
    variables.add(src)
    variables.add(dst)
    for char in elems:
      terminals.add(char)
      rules.append(Rule(src, [char, dst]))
  return CFG(variables, terminals, rules, start)
```

1. States are variables
2. Each edge $Q \xrightarrow{a} Q'$ yields a rule $Q \to aQ'$
3. Each final state Q yields a rule $Q \to \epsilon$

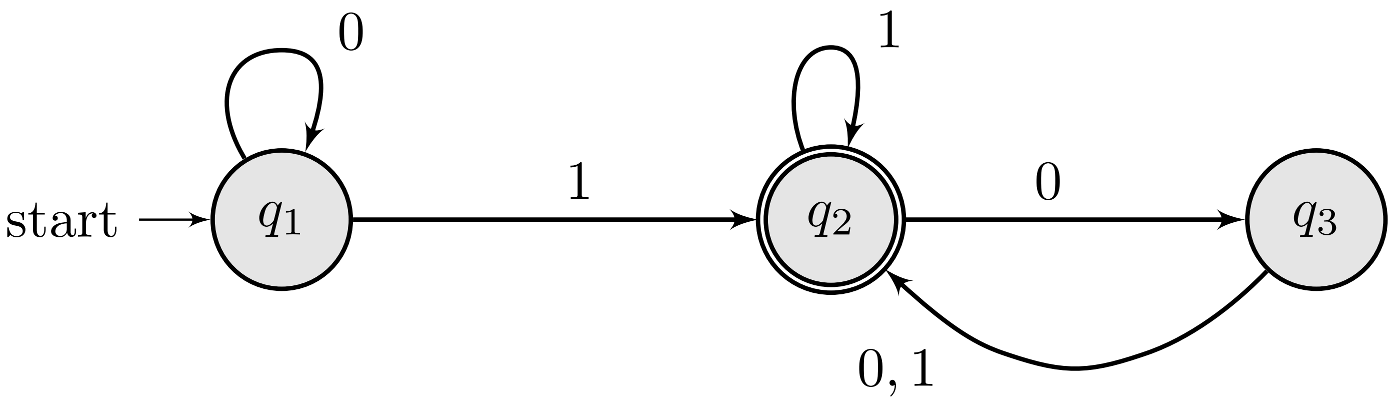
Example

- Given the following NFA, write an equivalent CFG



Example

- Given the following NFA, write an equivalent CFG



Solution

$$
\begin{aligned}
S_1 &\to 1S_2 \mid 0S_1 \\
S_2 &\to \epsilon \mid 1S_2 \mid 0S_3 \\
S_3 &\to 1S_2 \mid 0S_2
\end{aligned}
$$