

Regular Expressions

A **regular expression** denotes a **set** of strings (a language).

Given a finite alphabet Σ , the following constants are defined as regular expressions:

- \emptyset denoting the **empty set**,
- ϵ denoting the set $\{\epsilon\}$,
- a in Σ denoting the set containing only the character $\{a\}$

If r and s are regular expression (denoting the sets R and S , respectively) then $(r+s)$, (rs) and r^* denotes the set $R \cup S$, RS and R^* , respectively.

$L(r)$ indicates the language denoted by r

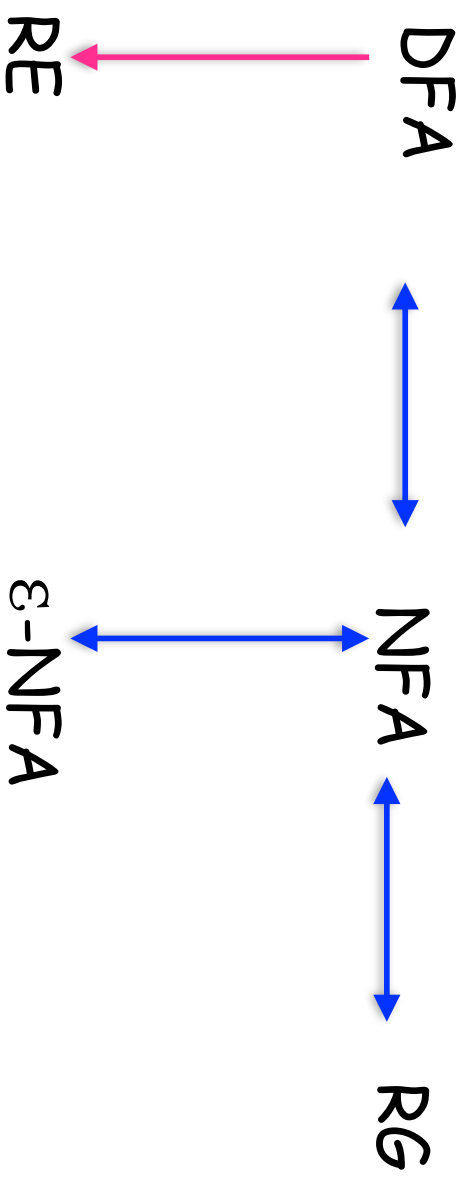
Examples

- $(0^* + 1^* + (01)^*)$ denotes the language

$$L = \{ x \mid \exists n \in \mathbb{N}. x = 0^n \vee x = 1^n \vee x = (01)^n \}$$

- $a|b^*$ denotes $\{\epsilon, "a", "b", "bb", "bbb", \dots\}$
- $(a+b)^*$ denotes all the strings formed with "a" and "b"
- $ab^*(c+\epsilon)$ denotes the set of strings starting with "a", then zero or more "b"s and finally optionally a "c"
- $(0_+(1(01^*0)^*1))^*$ denotes the set of binary numbers that are multiples of 3

Roadmap



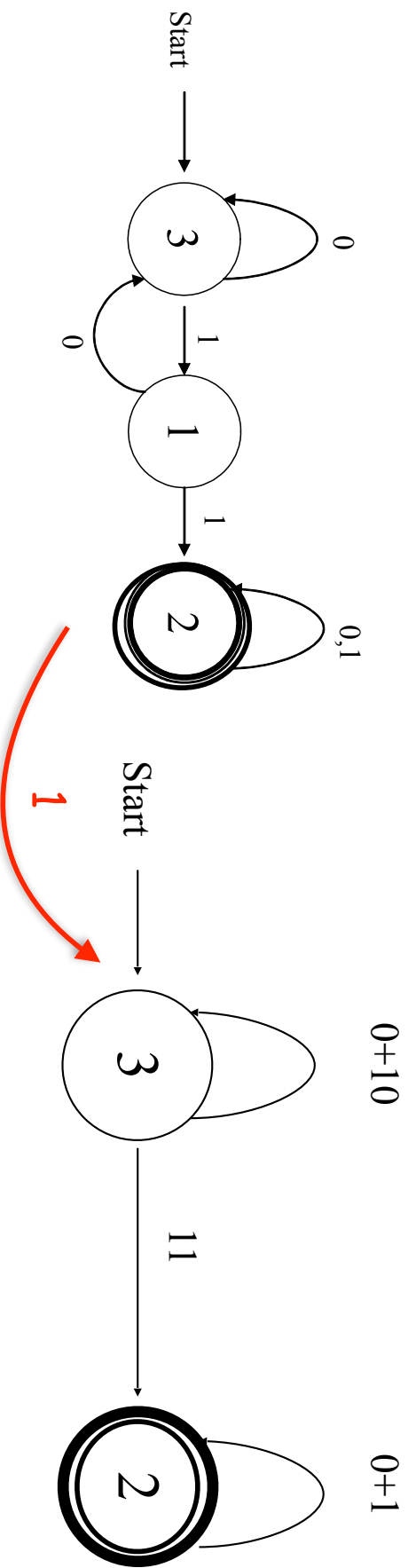
Encoding the language of a DFA into a RE

Theorem 3

For each DFA D , there is a regular expression r such that $L(D)=L(r)$.

Construction:

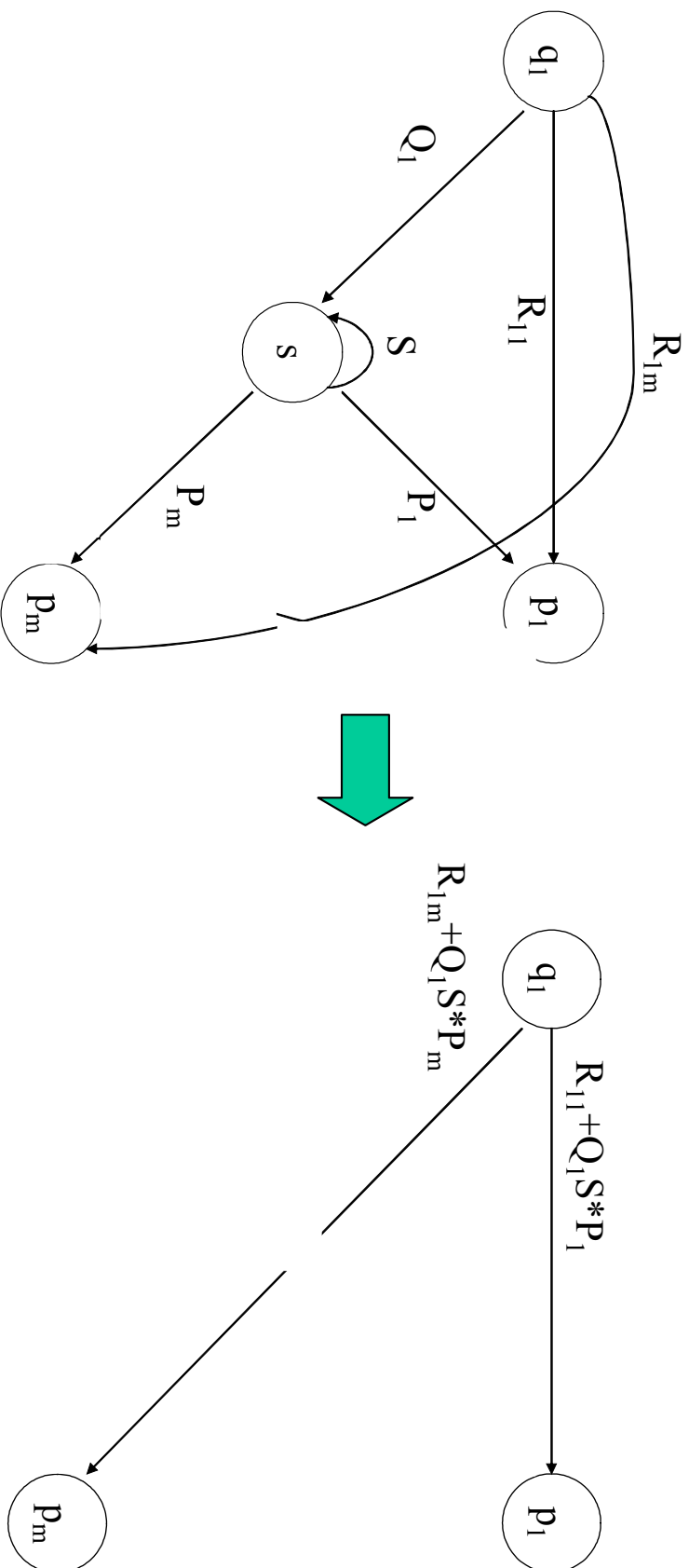
- 1) Eliminates states of the automaton replacing the edges with regular expressions that includes the behavior of the eliminated paths
- 2) When the automaton has just one starting and all final states, we synthesize the corresponding RE



State Elimination

Note: q_i and p_j may be the same state!

- Figure below shows the elimination of a state s . The labels on all edges are regular expressions.
- To remove s , we must make labels for the paths between q_1 and p_1, \dots, p_m we had in the original DFA through s .



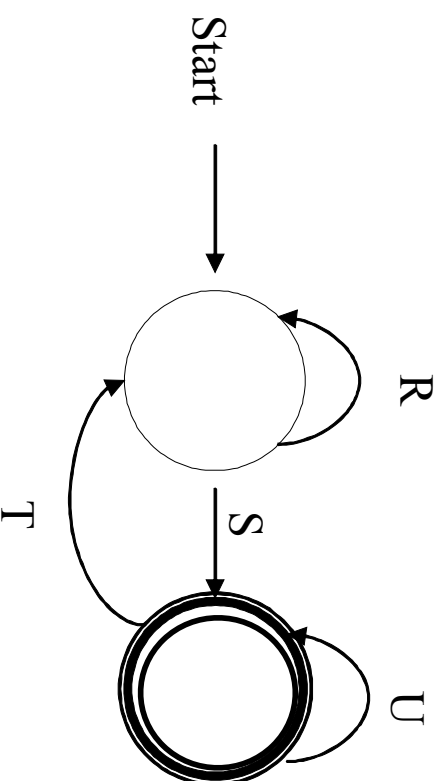
From a DFA to RE State Elimination Point (1)

Apply the state elimination process to produce an equivalent automaton with regular expression labels on the edges:

- Start with intermediate states and then moving to accepting states,
- The result will be some state automaton with one start state and (one or more than one) accepting states.

From a DFA to RE State Elimination Point (2)

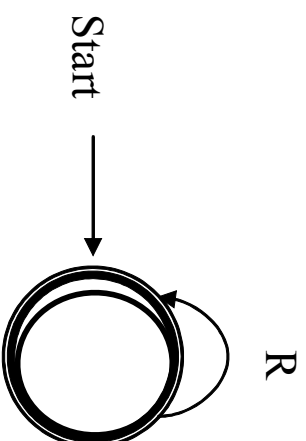
Just one final state and a different starting state



We can describe this automaton as: $(R+SU^*T)^*SU^*$

From a DFA to RE State Elimination Point (2)

Just one final state that coincides with the starting state



We can describe this automaton as simply R^* .

From a DFA to RE State Elimination Point (2)

Several final states s_1, s_2, \dots, s_n

Repeat the previous steps for each s_i turning any other accepting state in non accepting.

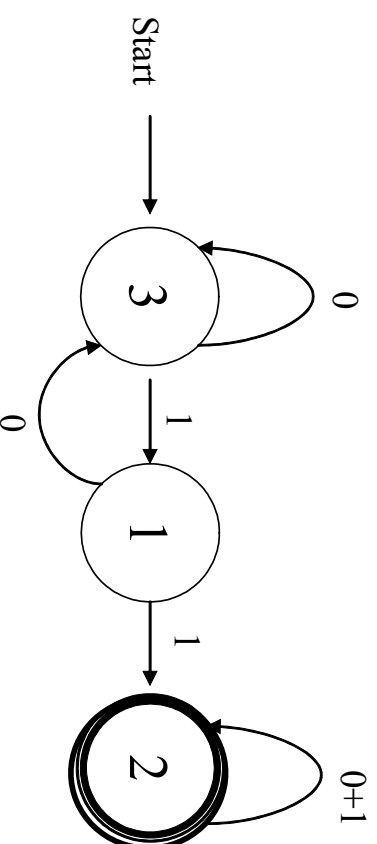
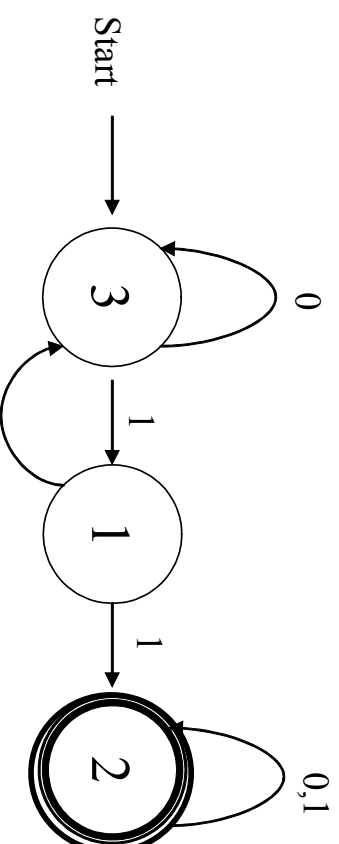
In this way we get n different regular expressions, R_1, R_2, \dots, R_n .

The desired regular expression for the automaton is then the union of each of the n regular expressions: $R_1 \cup R_2 \dots \cup R_N$

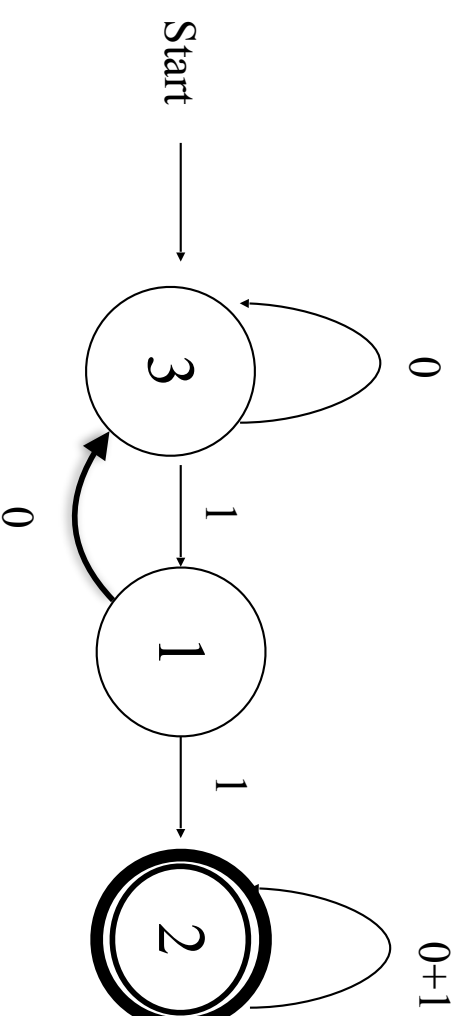
DFA \rightarrow RE Example

- Convert the following to a RE

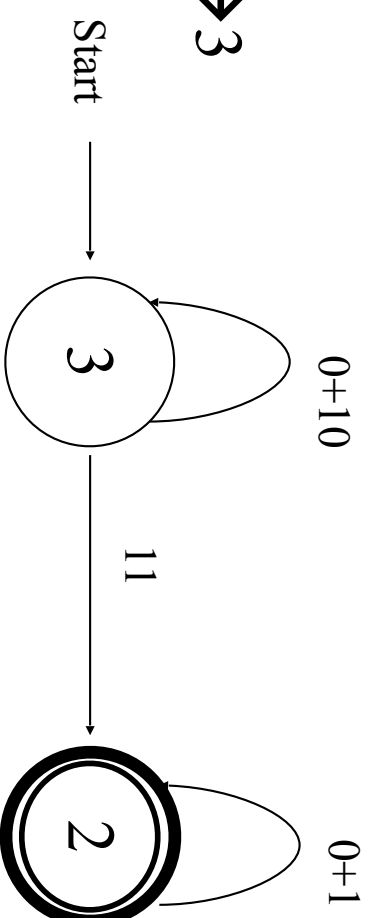
- First convert the edges to RE's:



DFA → RE Example (2)



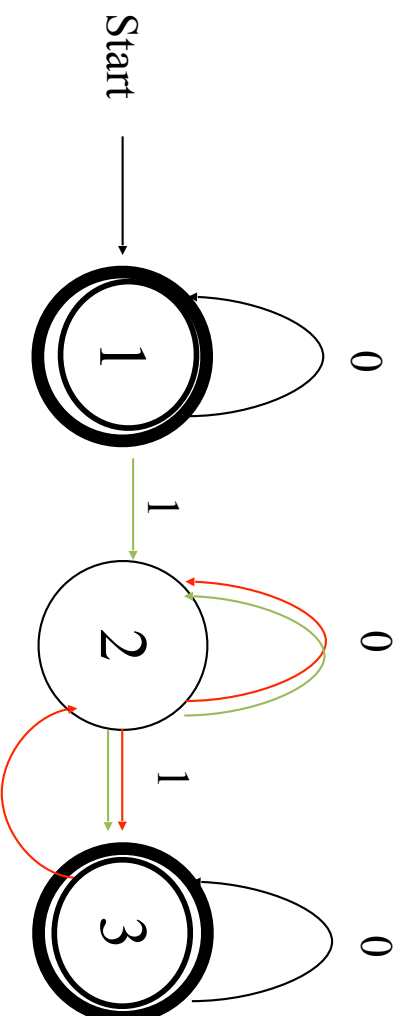
Note edge from 3 → 3



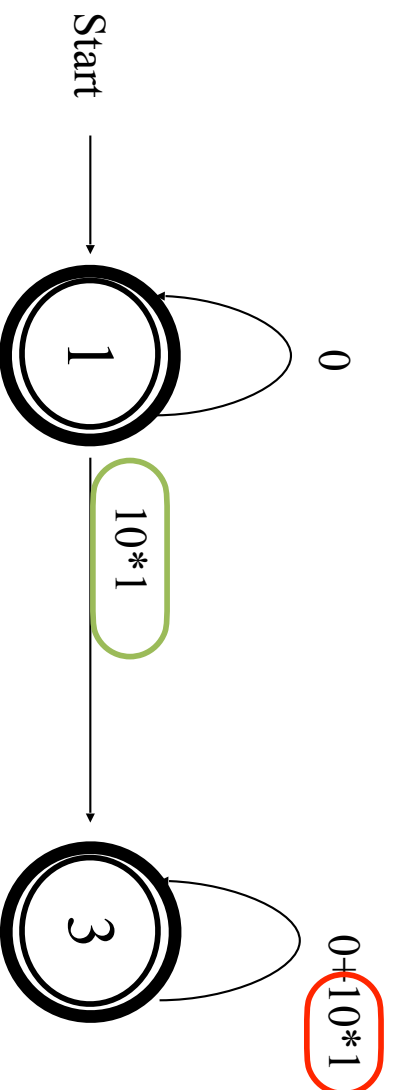
Answer: $(0+10)^*11(0+1)^*$

Third Example

- Automata that accepts even number of 1's

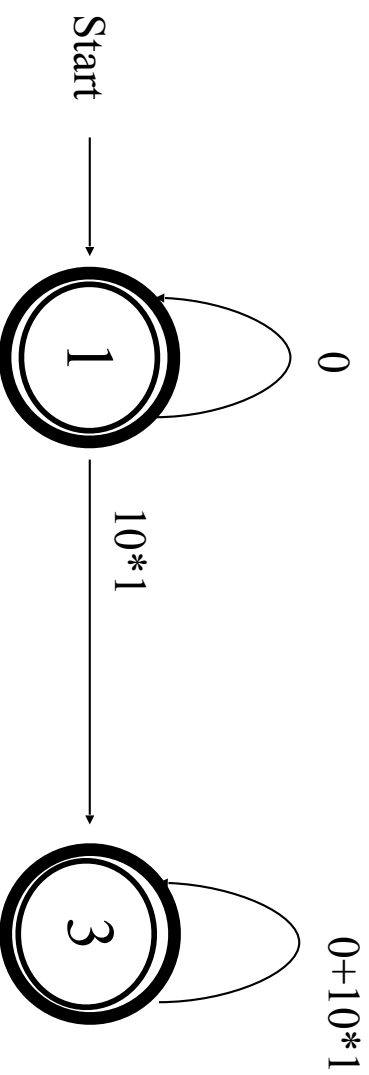


- Eliminate state 2:

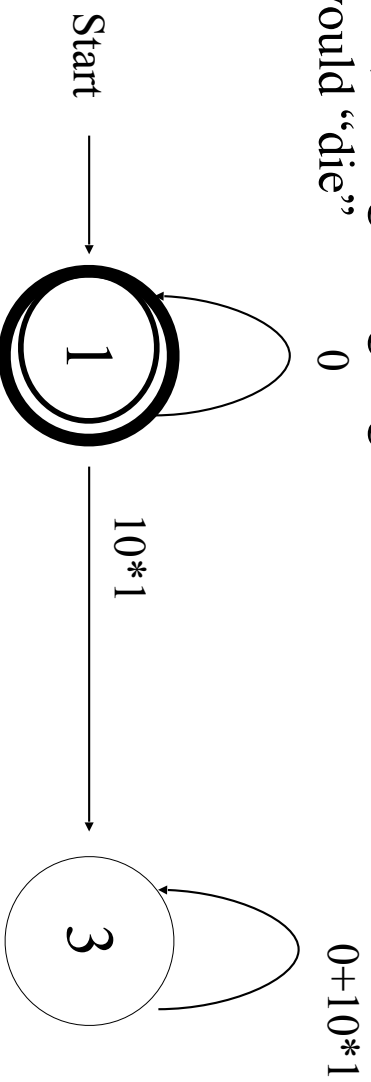


Third Example (2)

- Two accepting states, turn off state 3 first

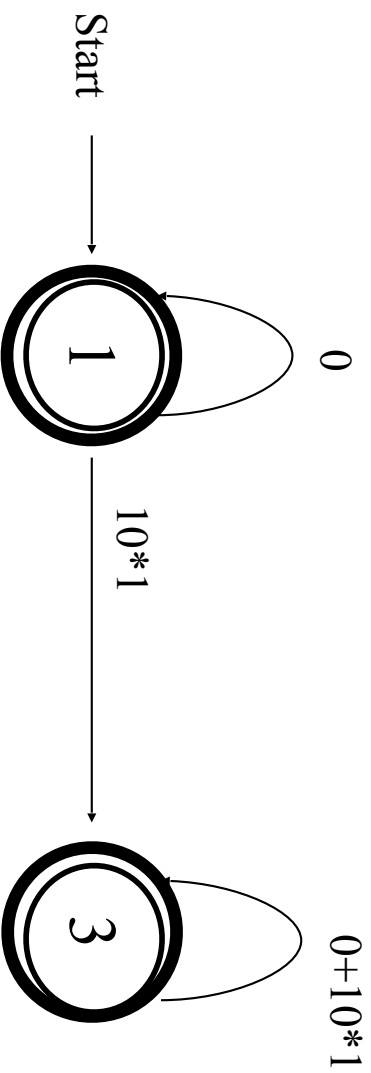


This is just 0^* , can ignore going to state 3 since we would "die"

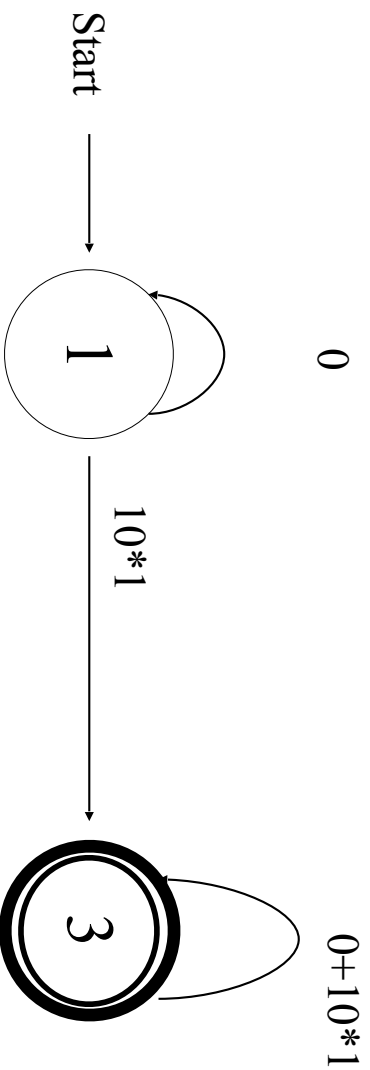


Second Example (3)

- Turn off state 1 second:



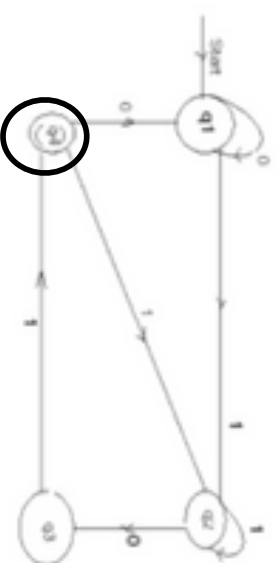
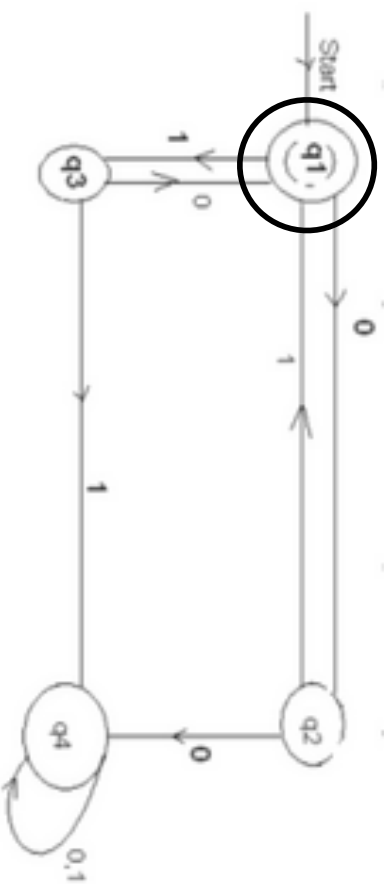
This is just $0^*10^*1(0+10^*1)^*$



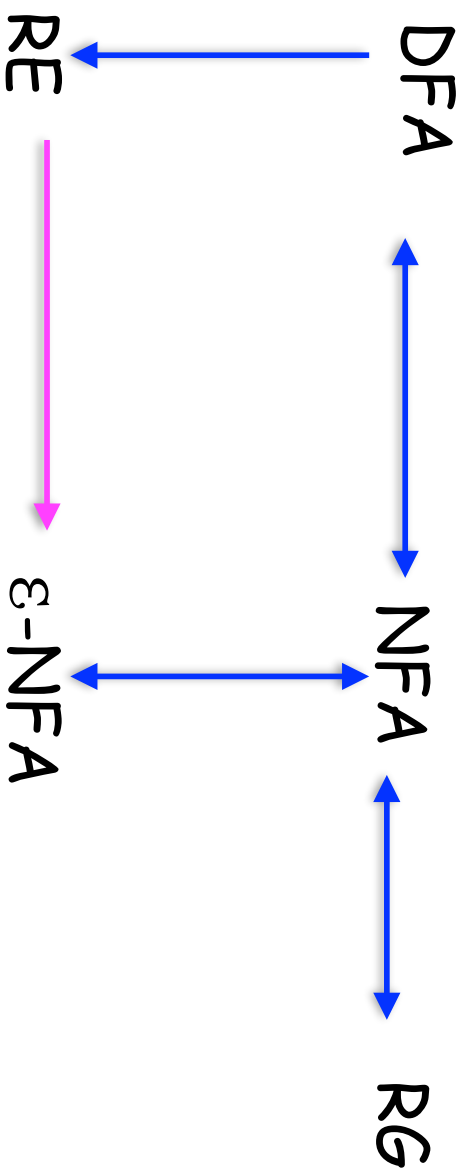
Combine from previous slide to get $0^* + 0^*10^*1(0+10^*1)^*$

Exercises

Convert the following DFA into a RE



Roadmap

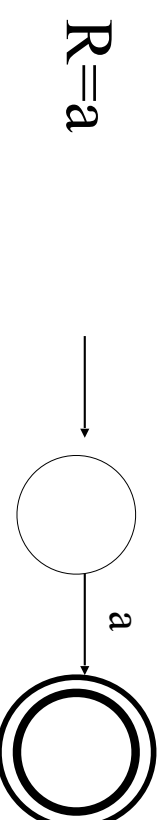


Converting a RE to an Automata

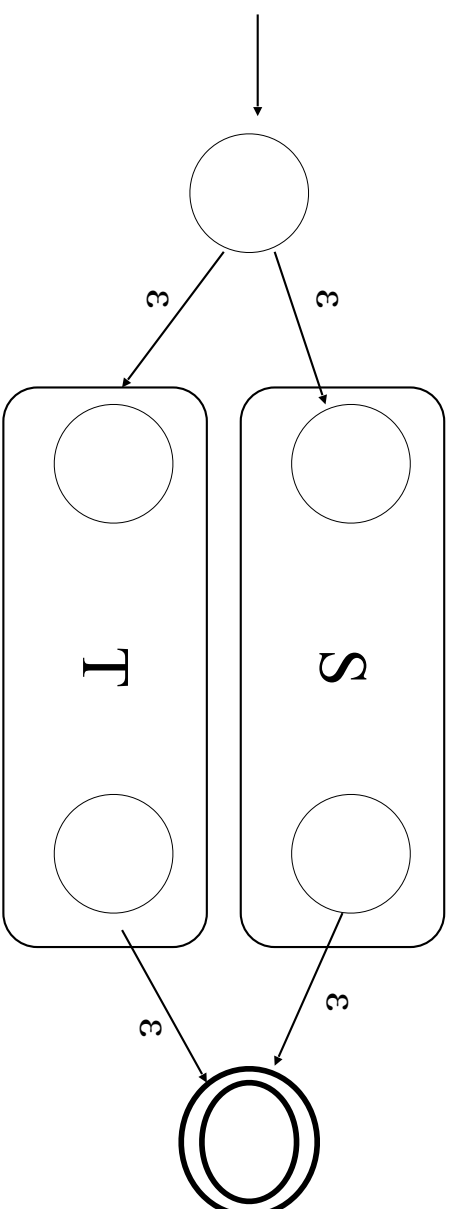
- We can convert a RE to an ϵ -NFA
 - Inductive construction
 - Start with a simple basis, use that to build more complex parts of the NFA

RE to ϵ -NFA

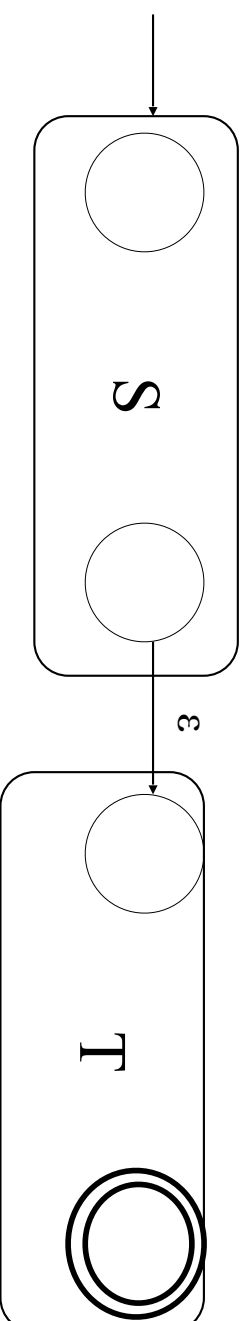
- Basis:



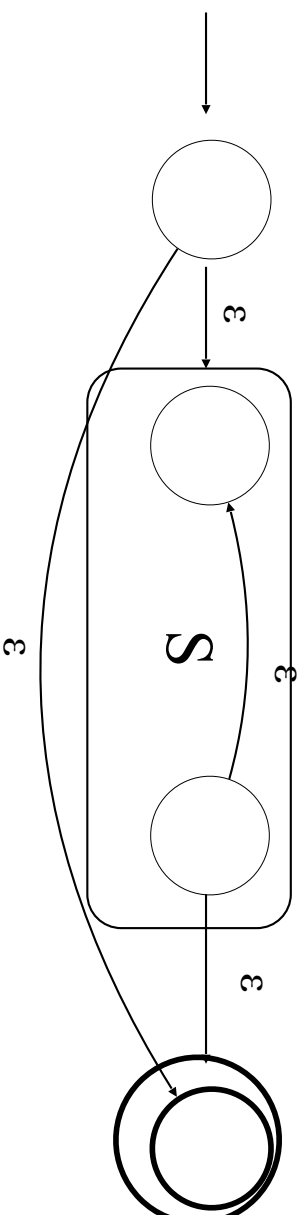
$$R=S+T$$



$$R=ST$$

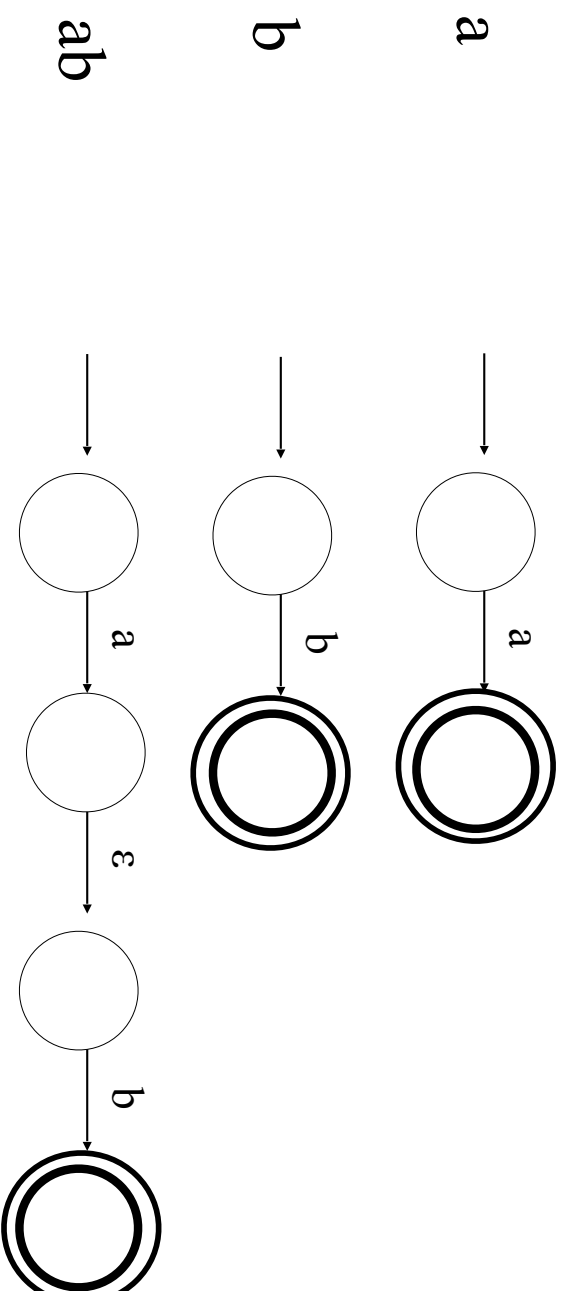


$$R=S^*$$



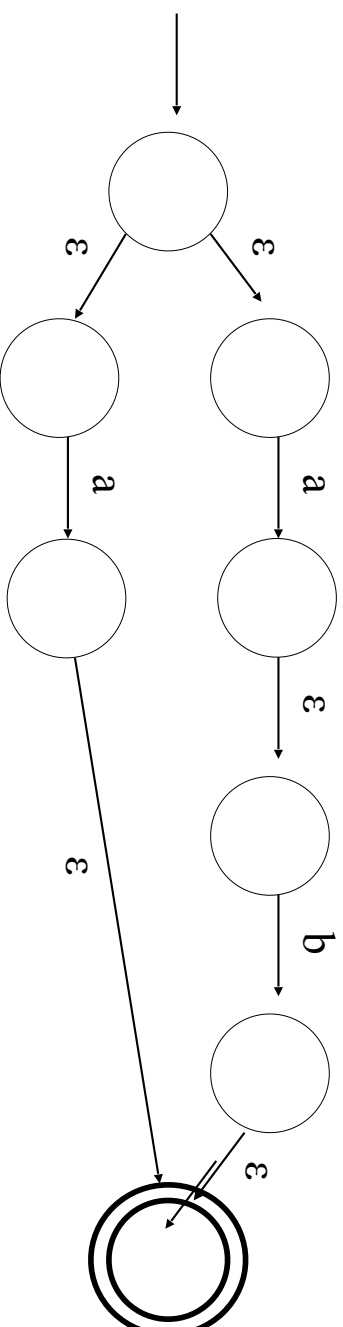
RE to ϵ -NFA Example

- Convert $R = (ab+a)^*$ to an NFA
 - We proceed by steps, starting from simple elements and working our way up

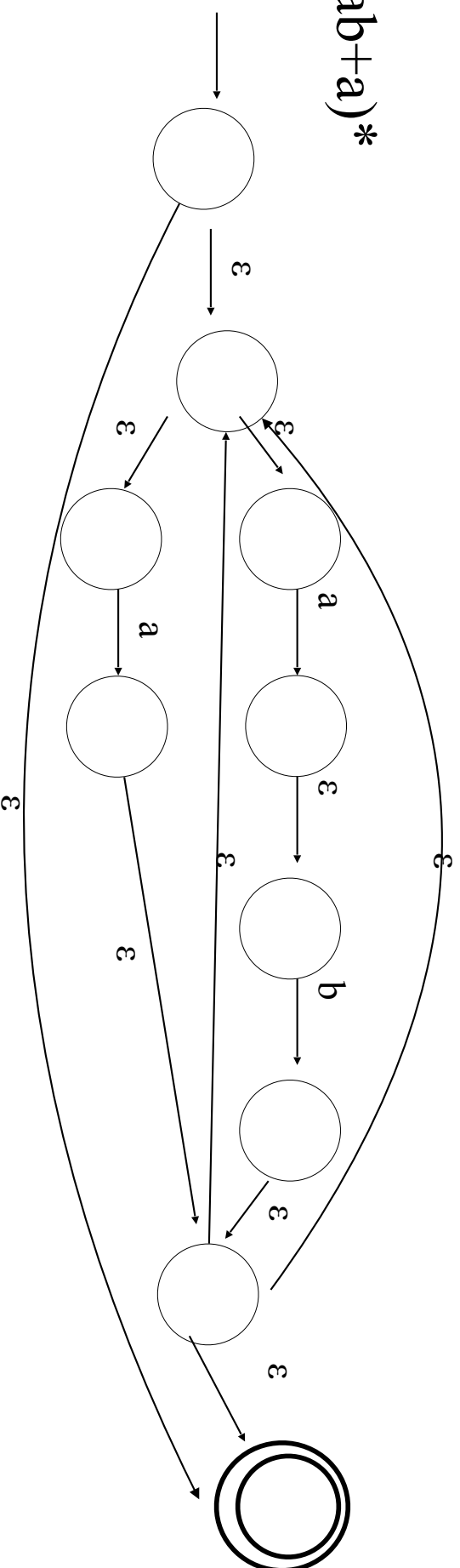


RE to ϵ -NFA Example (2)

$ab+a$



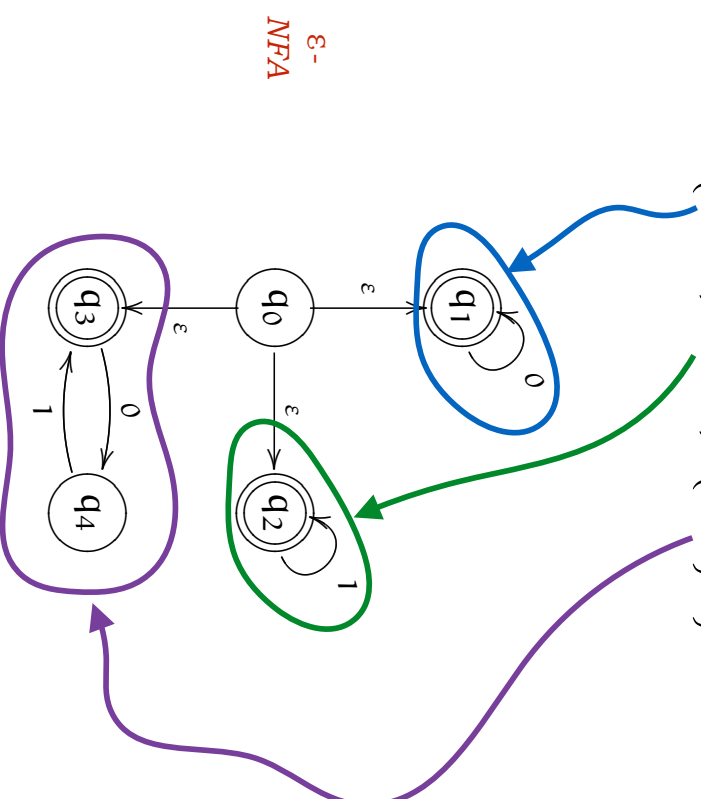
$(ab+a)^*$



Esempio: from RE to ϵ -NFA

$$L = \{ x \mid \exists n \in \mathbb{N}. x = 0^n \vee x = 1^n \vee x = (01)^n \}$$

$$(0^* + 1^* + (01)^*).$$



What have we shown?

- Regular expressions, finite state automata and regular grammars are different ways of expressing the same languages
- In some cases you may find it easier to start with one and move to the other
 - e.g., the language of an even number of 1's is typically easier to design as a NFA or DFA and then convert it to a RE

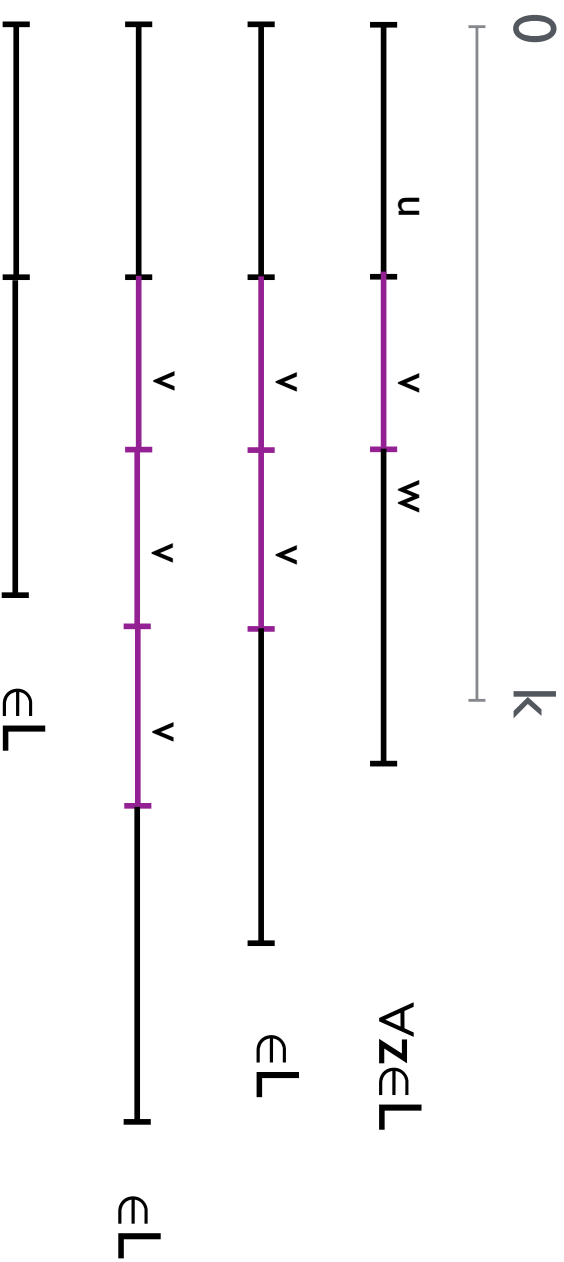
Not all languages are regular!

- $L = \{ a^n b^n \mid n \in \text{Nat} \}$

Pumping Lemma

Given L an infinite regular language then there exists an integer k such that for any string $z \in L$, $|z| \geq k$ it is possible to split z into 3 substrings

$z = uvw$ with $|uv| \leq k$, $|v| > 0$ such that $\forall i \in \mathbf{N}, uv^iw \in L$



Negating the PL

The PL gives a necessary condition, that can be used to prove that a language **is not a regular language!**

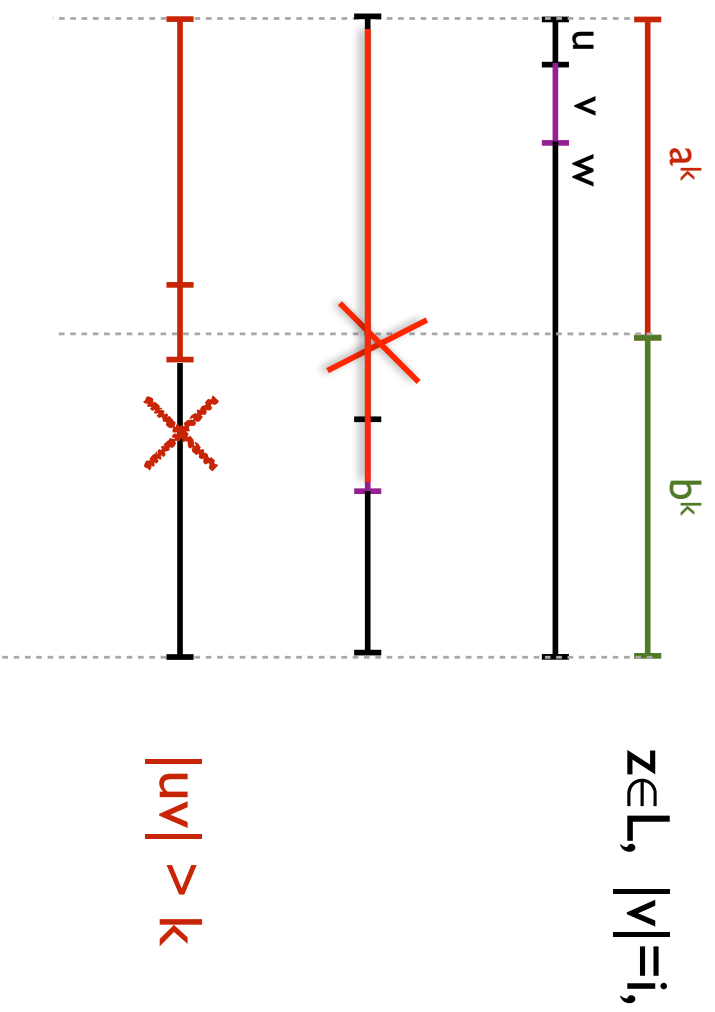
If $\forall k \in \mathbf{N} \exists z \in L. |z| \geq k$ for all possible splitting

$z = uvw$ with $|uv| \leq k, |v| > 0 \exists i \in \mathbf{N}$ such that $uv^i w \notin L$

then **L is not a regular language!**

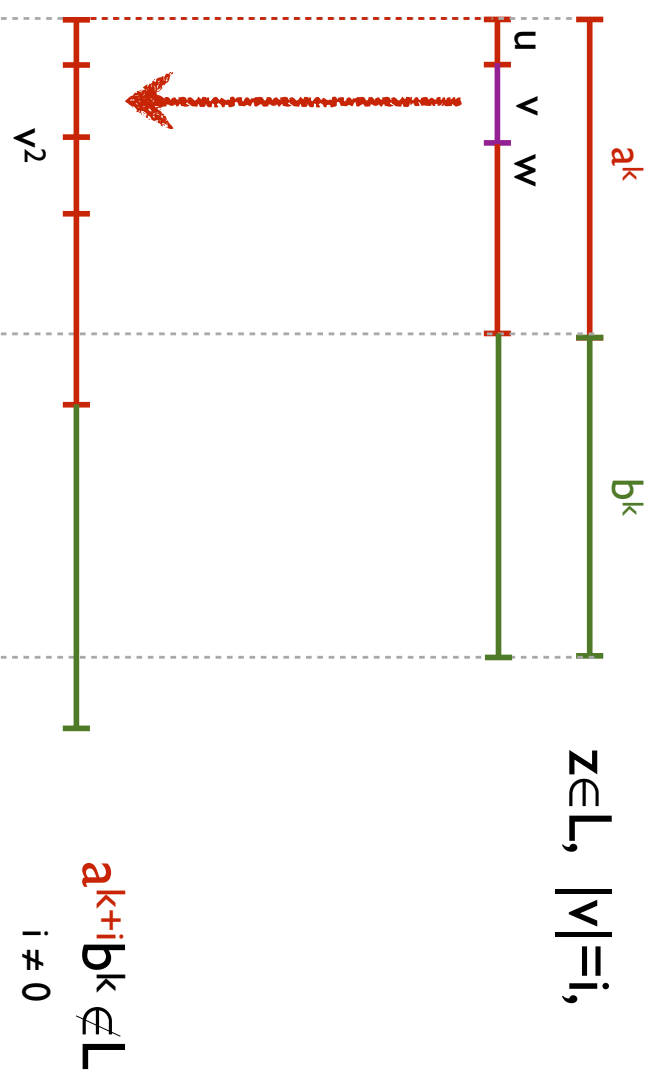
Example

- $L = \{ a^n b^n \mid n \in \mathbb{N} \}$, take any $k \in \mathbb{N}$
- Consider the string $z = a^k b^k$



Esempio

- $L = \{ a^n b^n \mid n \in \mathbb{N} \}$, take any $k \in \mathbb{N}$
- Consider the string $z = a^k b^k$



Exercises

Prove that the following are not regular languages

$$L_{pal} = \{w \in \{0,1\}^* \mid w = w^R\} = \{\epsilon, 0, 1, 00, 11, 00100, 01110, \dots\}$$

$$L_0 = \{0^n 1 0^n \mid n \geq 1\}$$

$$L_1 = \{0^n 1^m \mid n \leq m\}$$

Property of Regular languages

The regular languages are closed with respect to the union, concatenation and Kleene closure.

The complement of a regular language is always regular.

The regular language are closed under intersection

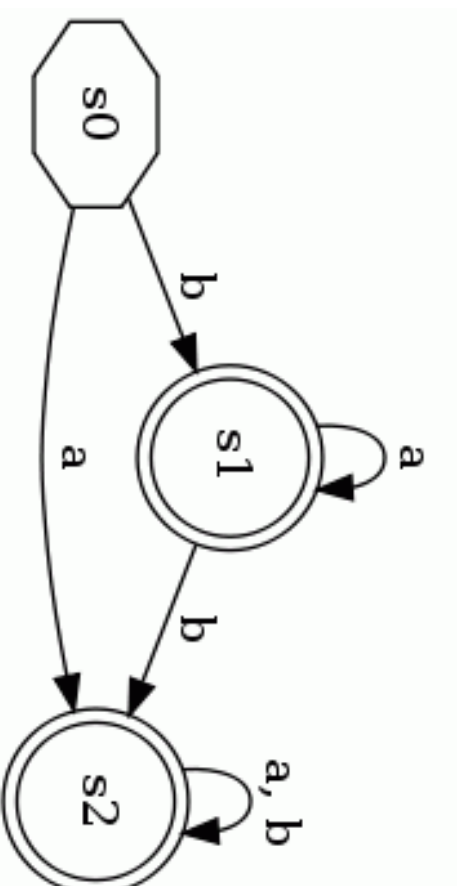
Decision Properties:

Approximately all the properties are **decidable** in case of finite automaton.

- (i) Emptiness
- (ii) Non-emptiness
- (iii) Finiteness
- (iv) Infiniteness
- (v) Membership

DFA Minimization

- Some states can be redundant:
 - The following DFA accepts $(a|b)^+$
 - State s_1 is not necessary



DFA Minimization

- The task of the DFA minimization is to automatically transform a given DFA into a state-minimized DFA
 - Several algorithms and variants are known

A DFA Minimization Algorithm

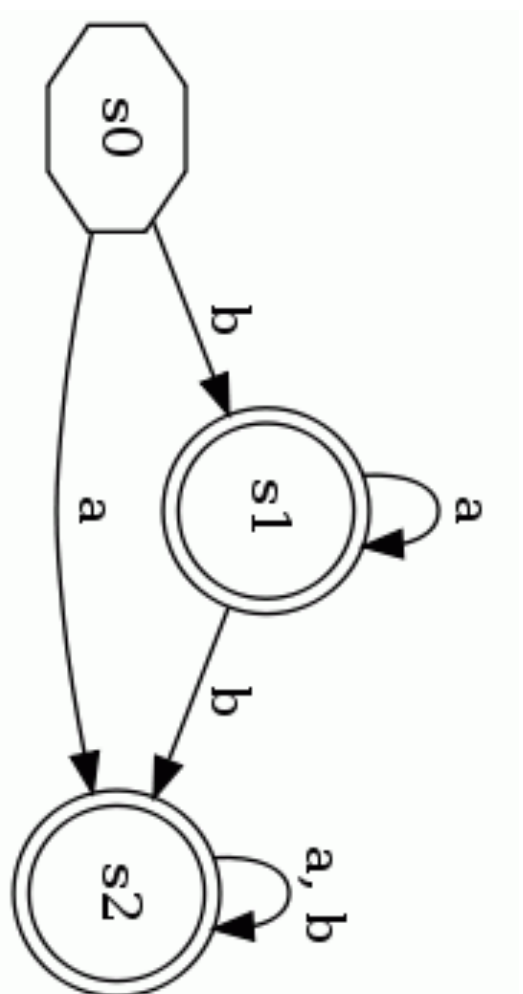
- Recall that a DFA $M=(Q, \Sigma, \delta, q_0, F)$
- Two states p and q are distinct if
 - $p \in F$ and $q \notin F$ or vice versa, or
 - $\delta(p, a)$ and $\delta(q, a)$, for some a in Σ , are distinct
- Using this inductive definition, we can calculate which states are distinct

DFA Minimization Algorithm

- Create lower-triangular table **DISTINCT**, initially blank
- For every pair of states (p, q) :
 - If p is final and q is not, or vice versa
 - $\text{DISTINCT}(p, q) = \epsilon$
- Loop until no change for an iteration:
 - For every pair of states (p, q) and each symbol a
 - If $\text{DISTINCT}(p, q)$ is blank and $\text{DISTINCT}(\delta(p, a), \delta(q, a))$ is not blank
 - $\text{DISTINCT}(p, q) = a$
- Combine all states that are not distinct

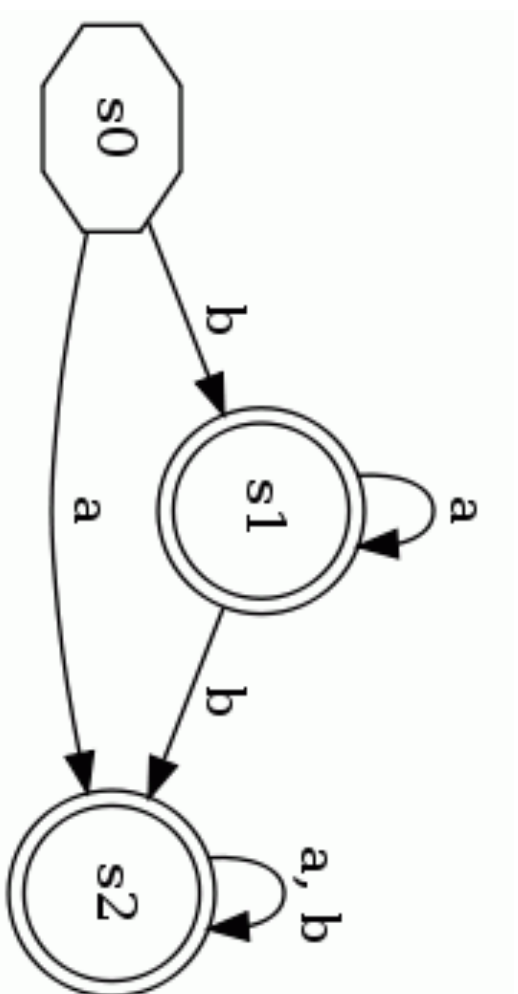
Very Simple Example

s0			
s1			
s2			
	s0	s1	s2



Very Simple Example

s0			
s1	ϵ		
s2	ϵ		
	s0	s1	s2



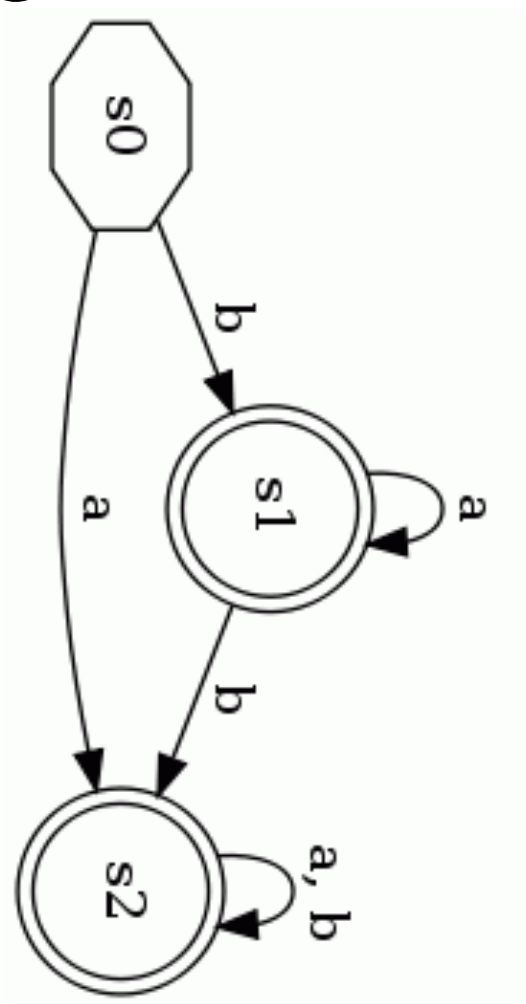
Label pairs with ϵ where one is a final state and the other is not

Very Simple Example

s0			
s1	ϵ		
s2	ϵ		
	s0	s1	s2

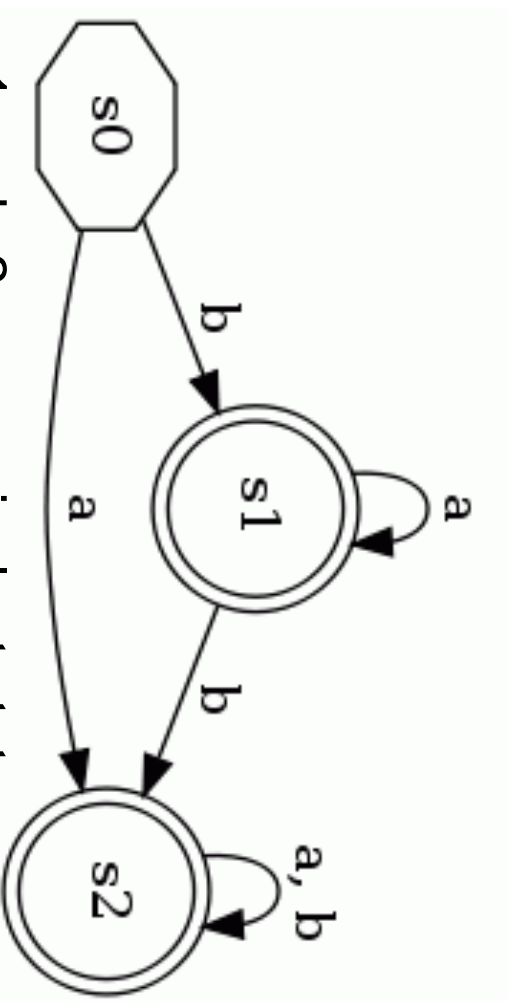
- $\text{DISTINCT}(p, q)$ is blank and
 $\text{DISTINCT}(\delta(p, a), \delta(q, a))$ is not blank
- $\text{DISTINCT}(p, q) = a$

Main loop (no changes occur)



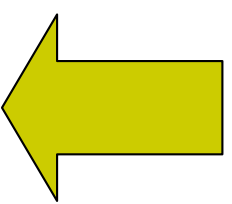
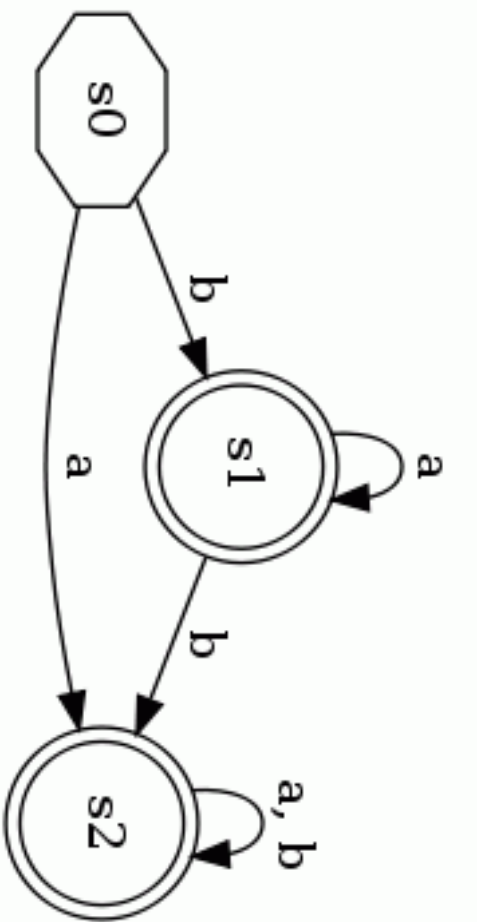
Very Simple Example

s0			
s1	ϵ		
s2	ϵ		
	s0	s1	s2

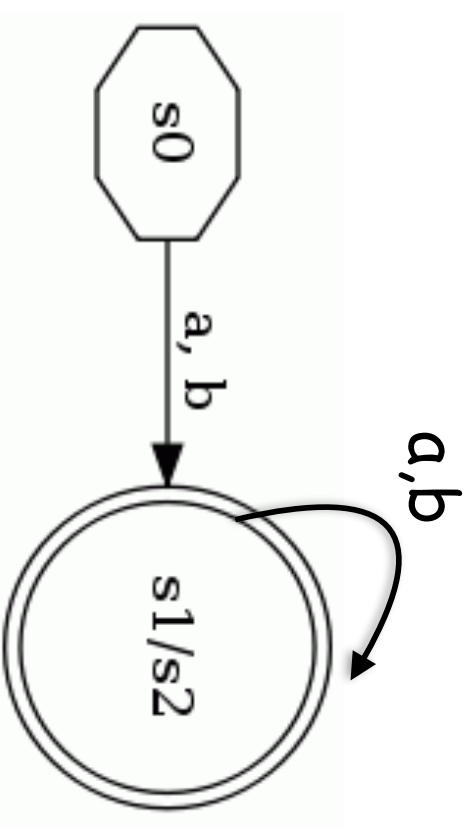
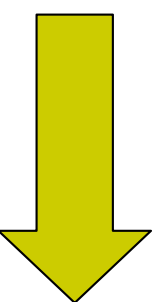


DISTINCT(s_1, s_2) is empty, so s_1 and s_2 are equivalent states

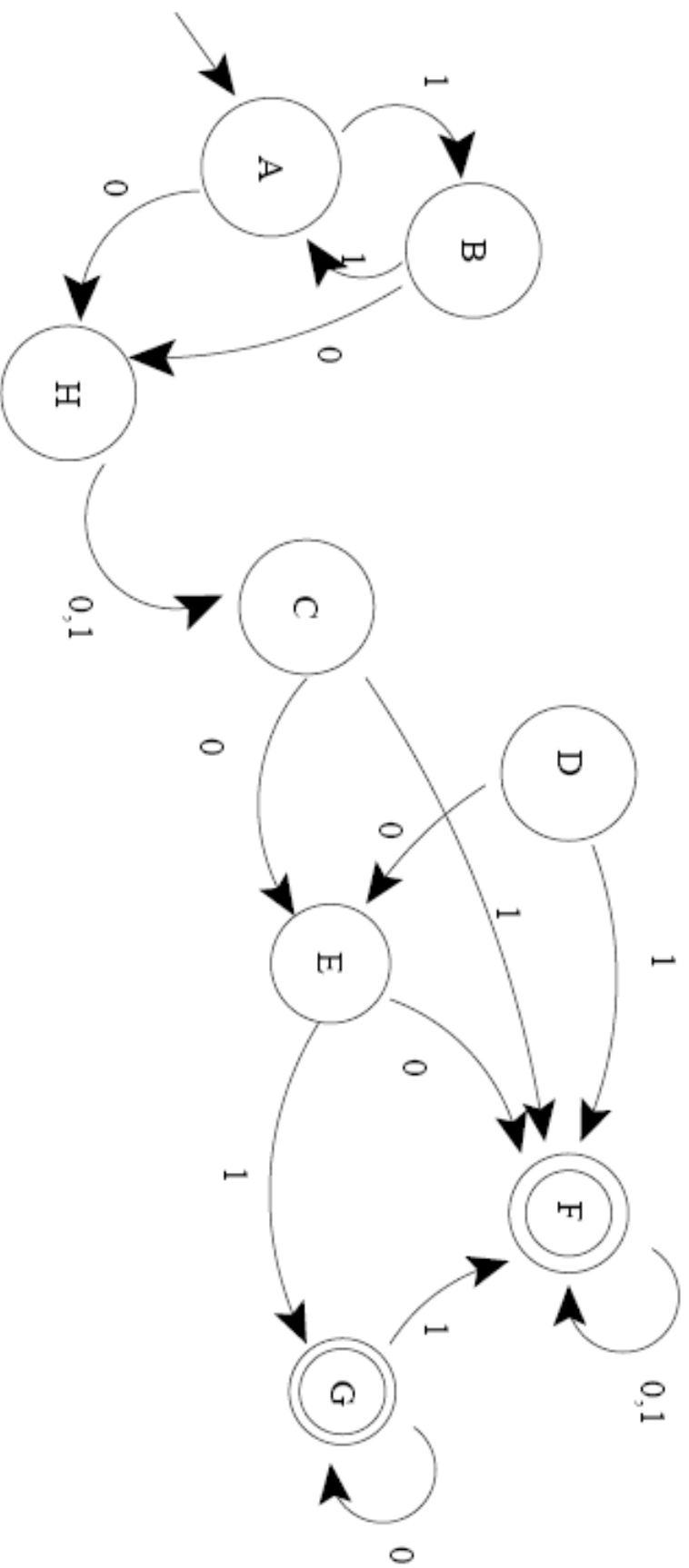
Very Simple Example



Merge s1 and s2

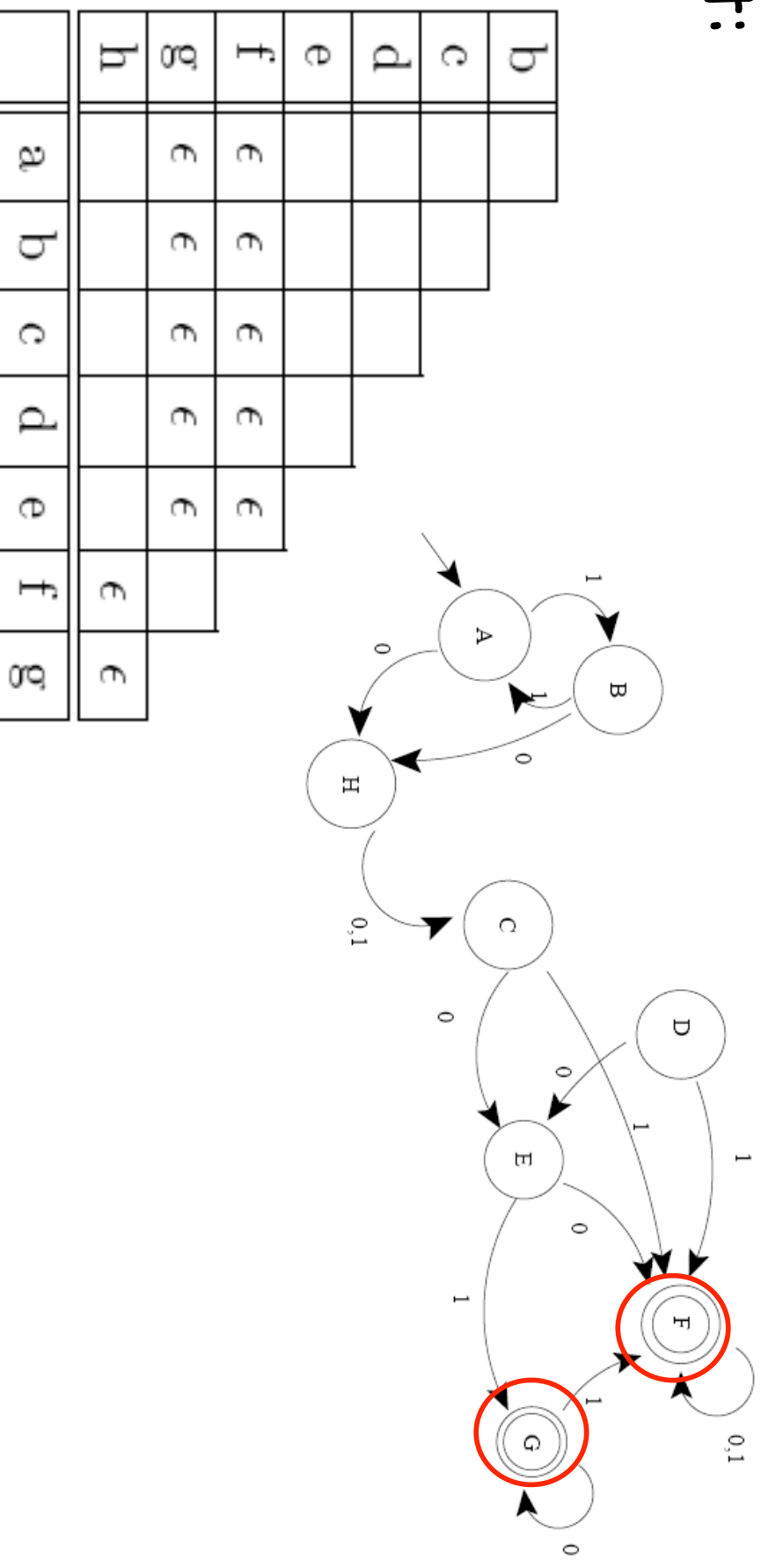


More Complex Example



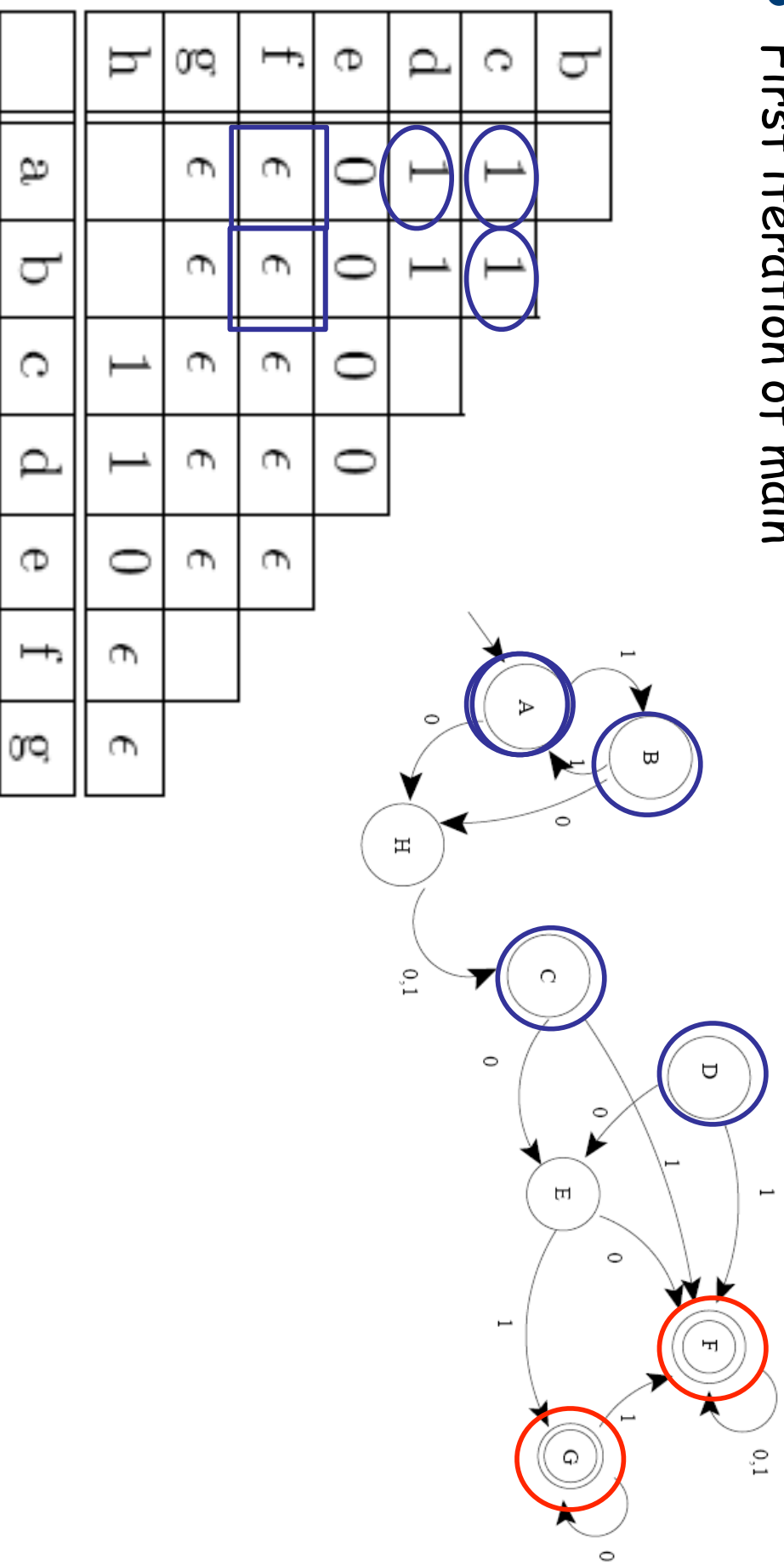
More Complex Example

- Check for pairs with one state final and one not:



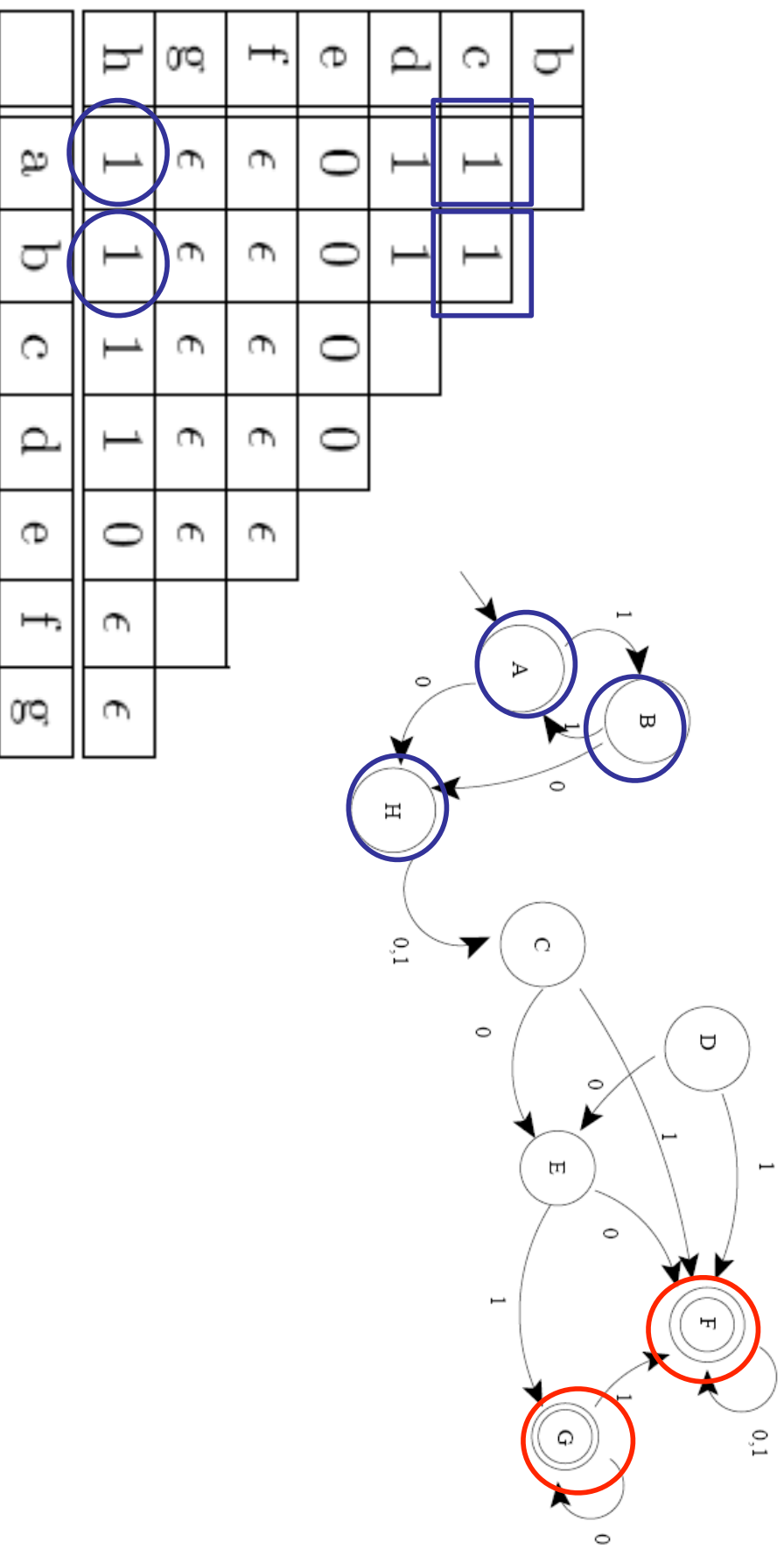
More Complex Example

- First iteration of main



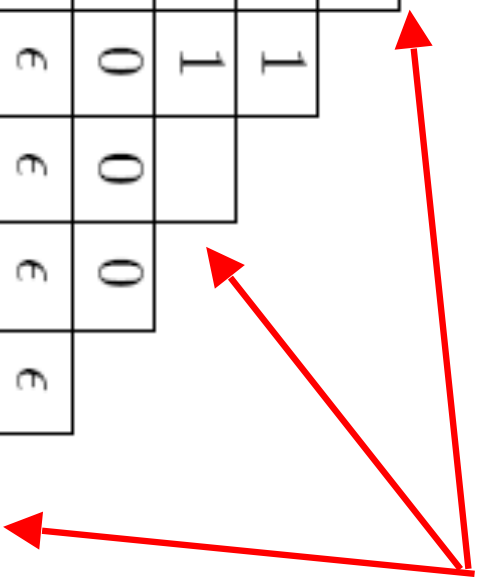
More Complex Example

- Second iteration of main loop:



More Complex Example

- Third iteration makes no changes
 - Blank cells are equivalent pairs of states

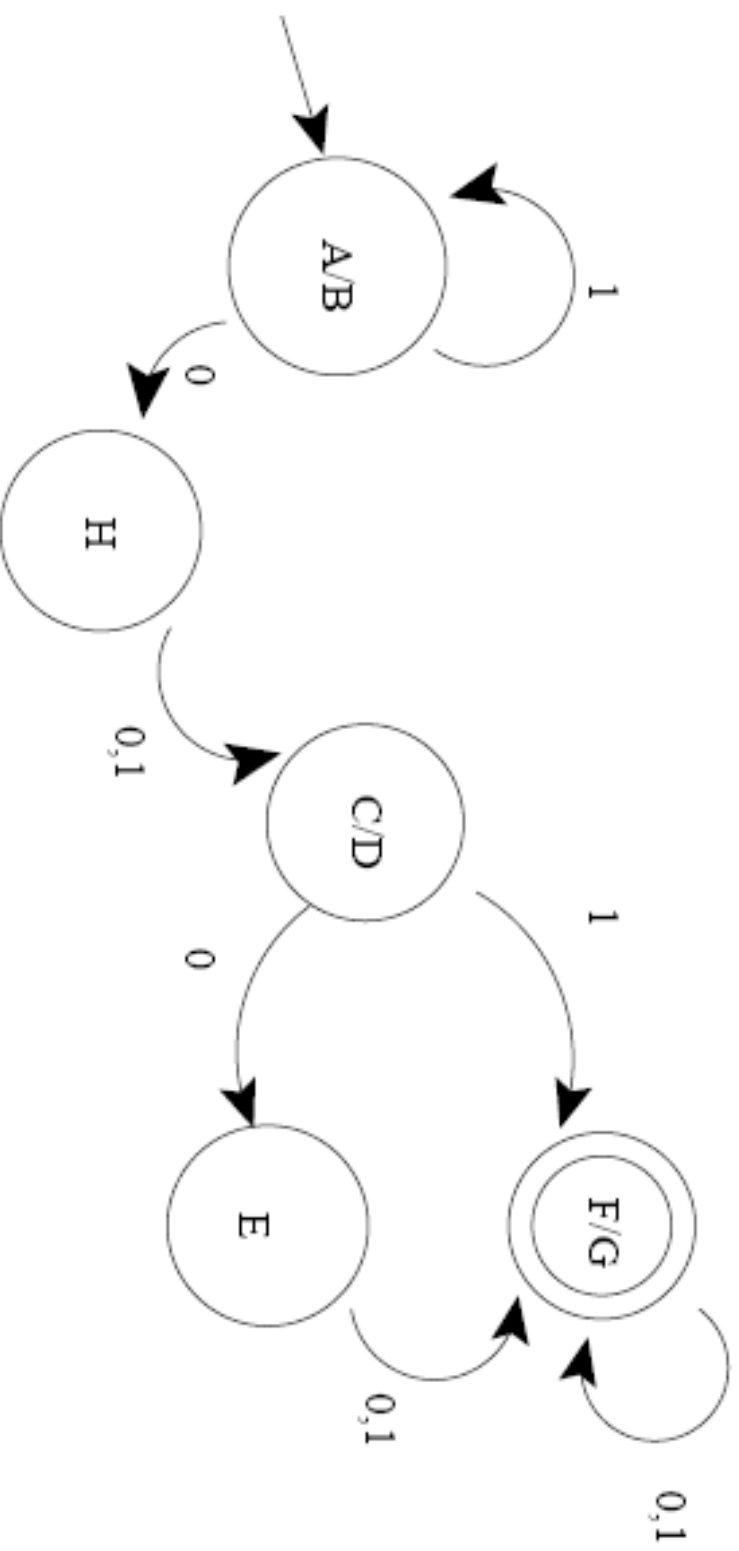


The diagram shows a grid of cells with red arrows pointing to blank cells, indicating equivalent pairs of states. The grid is as follows:

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ε	ε	ε	ε	ε		
g	ε	ε	ε	ε	ε		
h	1	1	1	1	0	ε	ε
	a	b	c	d	e	f	g

More Complex Example

- Combine equivalent states for minimized DFA:



Conclusion

- The algorithm described is $O(kn^2)$
 - John Hopcraft describes another more complex algorithm that is $O(k(n \log n))$

Exercises

Minimize the following automata

	0	1
A	B	A
B	A	C
C	D	B
D	D	A
E	D	F
F	G	E
G	F	G
H	G	H

→ (pointing to row A)
* (pointing to row D)

	0	1
A	B	E
B	C	F
C	D	H
D	E	H
E	F	I
F	G	B
G	H	B
H	I	C
I	A	E

→ (pointing to row A)
* (pointing to row C)
* (pointing to row F)
* (pointing to row I)

Linguaggi Context Free

Context free Grammars

A **Context free Grammar** (Σ, N, S, P) is a grammar, where

- every production has the form $U \rightarrow V$

$$U \in N \text{ and } V \in (\Sigma \cup N)^+$$

- only for the starting symbol S , we can have $S \rightarrow \epsilon$

Example

$G = \{E\}, \{or, and, not, (,), 0, 1\}, [E, P]$

$E \mapsto 0$

$E \mapsto 1$

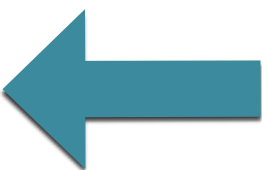
$E \mapsto (E \text{ or } E)$

$E \mapsto (E \text{ and } E)$

$E \mapsto (\text{not } E)$

Esempio

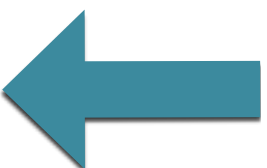
$$S \rightarrow 0S1 \mid \varepsilon$$



$$\{0^n 1^n : n \geq 0\}$$

Example

$S \rightarrow \varepsilon | 0 | 1 | 0S | 1S | 1$



$Z \equiv \{x \in \{0, 1\}^* \mid x \equiv x^R\}$

Parse tree

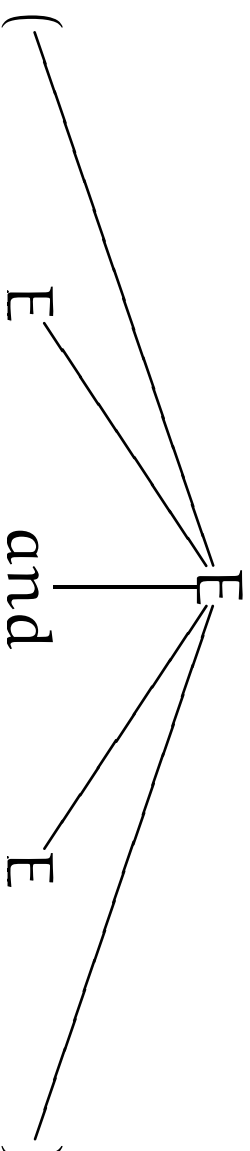
Given a grammar (Σ, N, S, P) .

The parse tree is the graph representation of a derivation, which can be defined in the following way:

- every vertex has a label in $\Sigma \cup N \cup \{\epsilon\}$,
- the label of the root and of every internal vertex belongs to N ,
- if a vertex is labeled with A and has m children labeled with X_1, \dots, X_k
- then the production $A \rightarrow X_1 \dots X_k$ belongs to P ,
- if a vertex is labeled with ϵ then it is a leaf and is an only child.

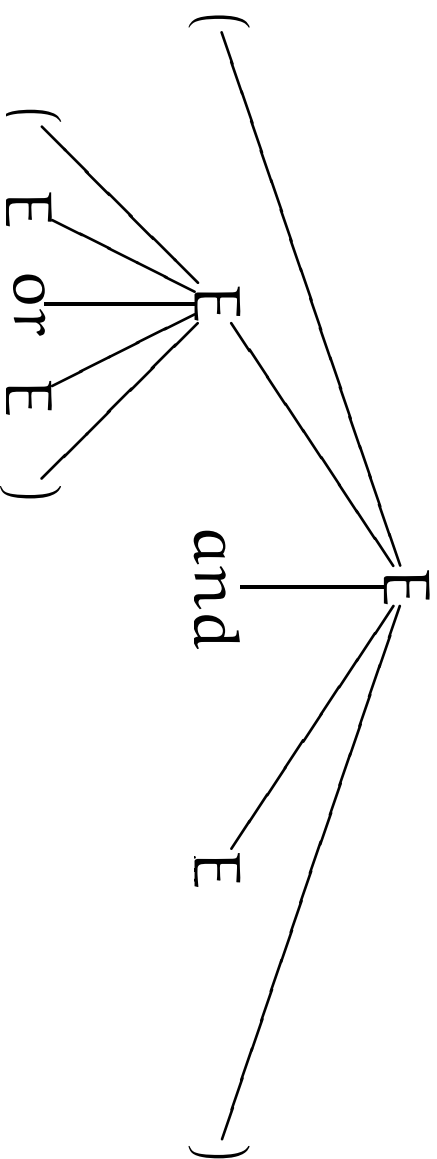
Example

$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(not E).$



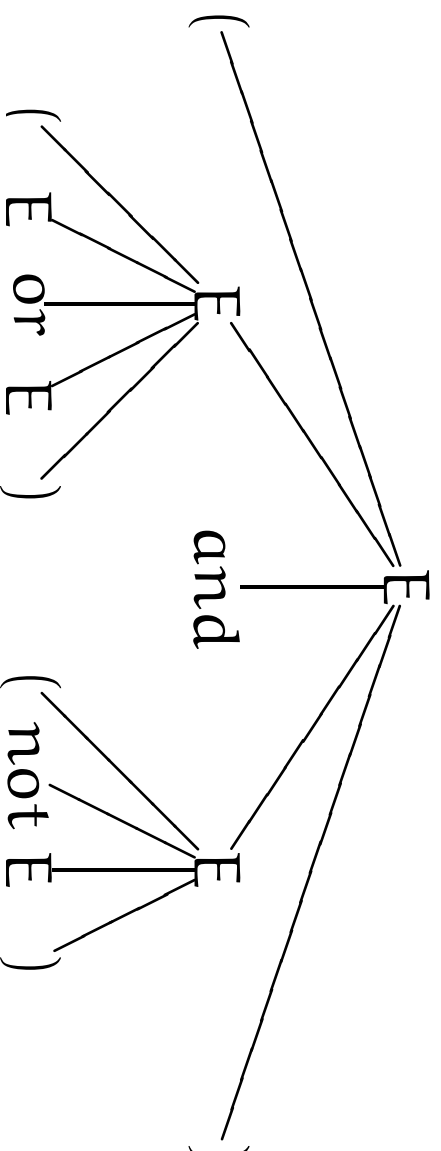
Example

$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(not E).$



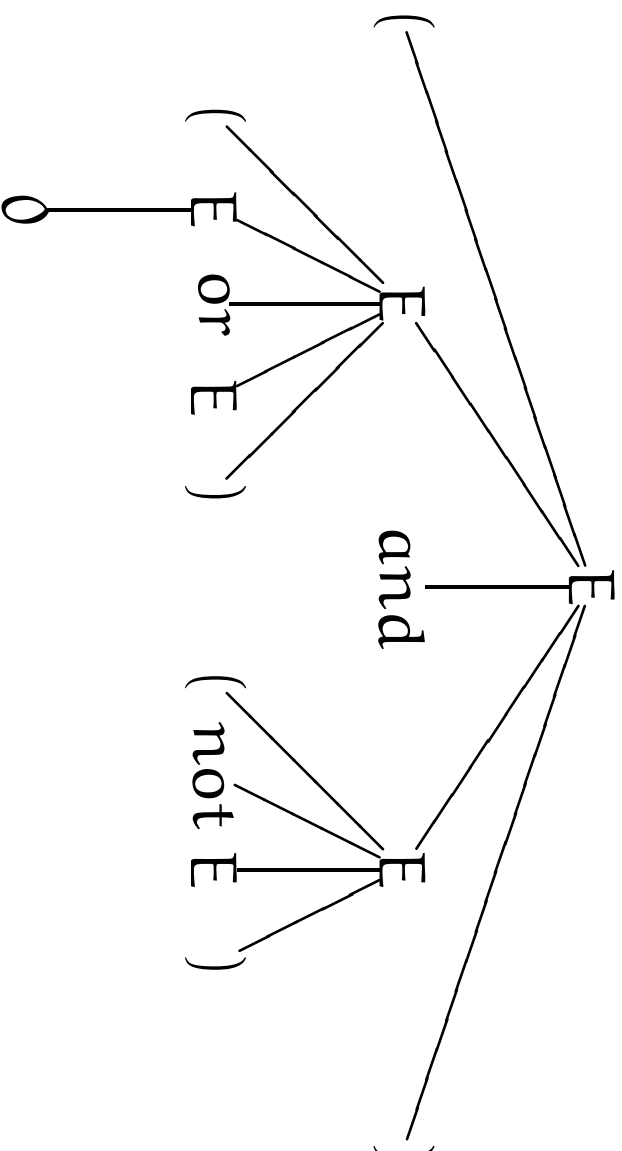
Example

$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(not E).$



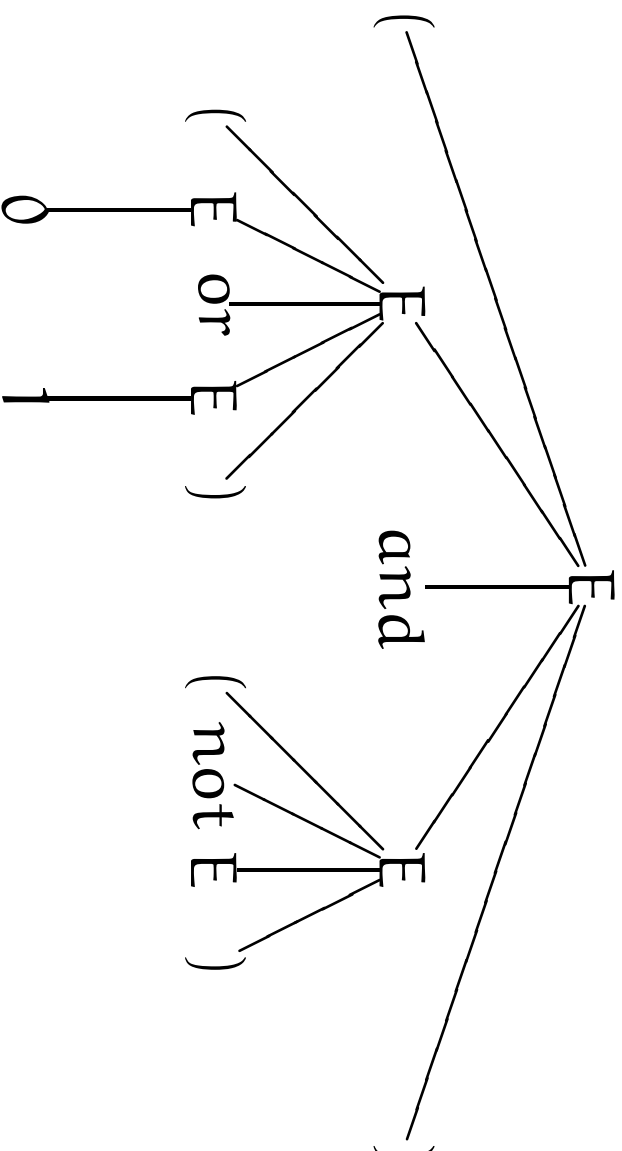
Example

$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(\text{not } E) .$



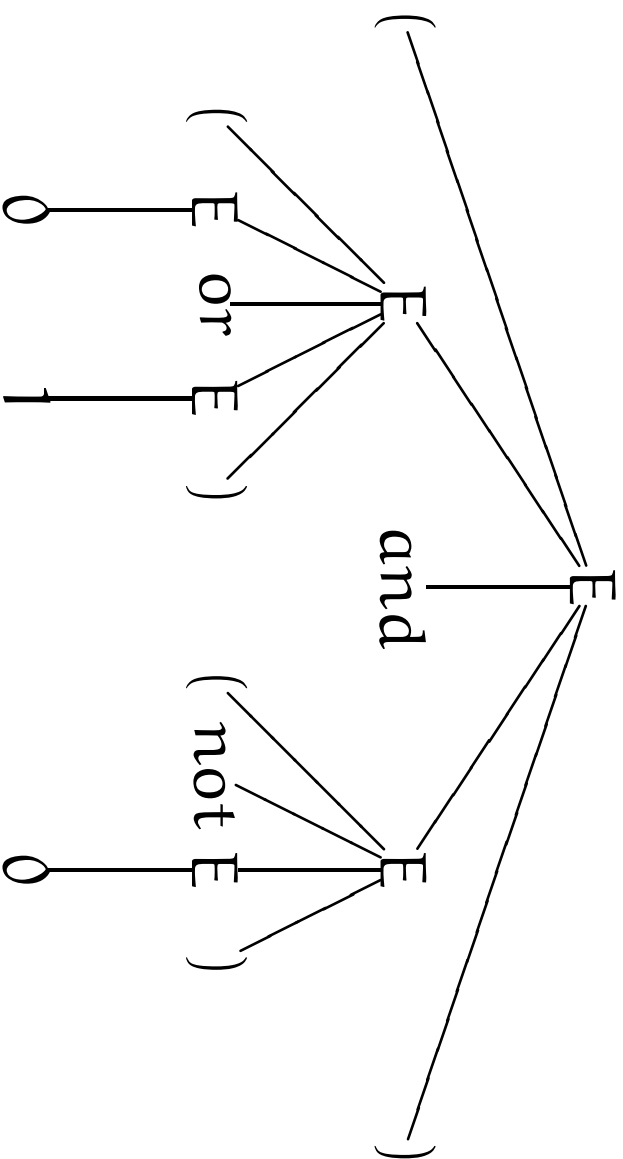
Example

$E \mapsto 01(E \text{ or } E) \mid (E \text{ and } E) \mid (\text{not } E).$



Example

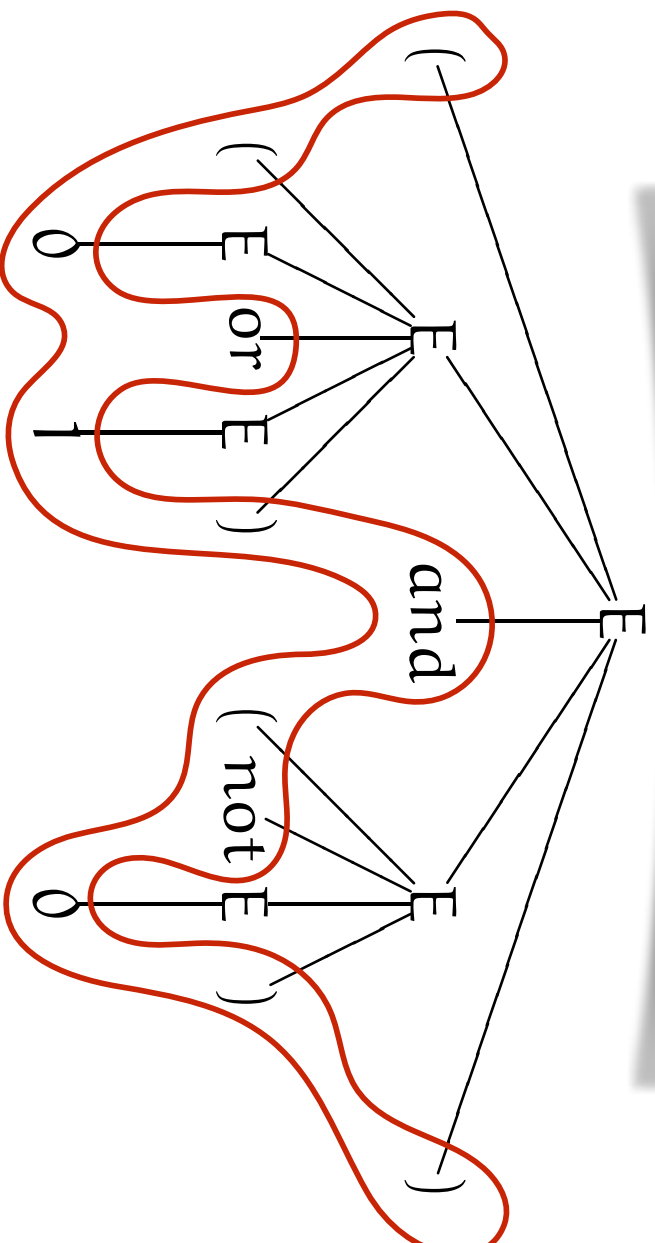
$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(\text{not } E) .$



Example

$E \mapsto 0|1|(E \text{ or } E)|(E \text{ and } E)|(\text{not } E) .$

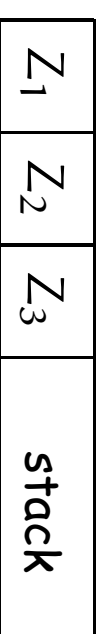
$w = ((0 \text{ or } 1) \text{ and } (\text{not } 0))$



Pushdown automata



Alphabet of stack symbols: **R**



The stack head always scans the top symbol

It performs three basic operations:

Push: add a new symbol at the top of the stack

Pop: read and remove the top symbol

Empty: verify if the stack is empty

Pushdown automata

A push down automaton is $M = (Q, \Sigma, R, \delta, q_0, Z_0, F)$ where

- R is the alphabet of stack symbols,
- $\delta : Q \times (\Sigma \cup \{e\}) \times R \rightarrow \mathcal{P}(Q \times R^*)$ is the transition function
- Z_0 belonging to R is the starting symbol on the stack

Instantaneous Description

The evolution of the PDA is described by triples (q, w, γ) where;

- q is the current state
- w is the unread part of the input string or the remaining input
- γ is the current contents of the stack

A move from one instantaneous description to another

will be denoted by

$$(q_0, aw, Zr) \mapsto (q_1, w, \gamma r) \text{ iff } (q_1, \gamma) \text{ belongs to } \delta(q_0, a, Z)$$

The language accepted by a pushdown automaton

Two ways to define the accepted language:

- with empty stack (in this case F is the empty set)

$$L_p(M) = \{x \in \Sigma^* : (q_0, x, Z_0) \mapsto_M^* (q, \varepsilon, \varepsilon), q \in Q\}$$

- with explicit final states F

$$L_F(M) = \{x \in \Sigma^* : (q_0, x, Z_0) \mapsto_M^* (q, \varepsilon, \gamma), \gamma \in R^*, q \in F\}$$

Esempio

$L = \{ xcx^R \mid x \in \{a, b\}^* \}, \Sigma = \{a, b, c\}$
We will recognise the string when the input and stack are empty!

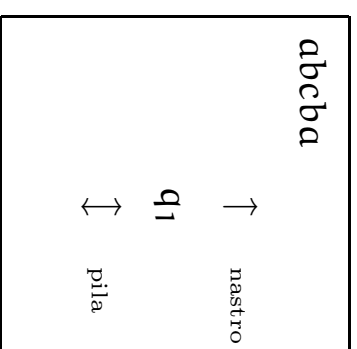
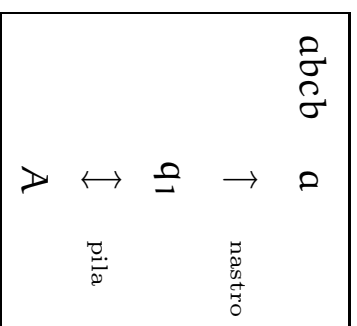
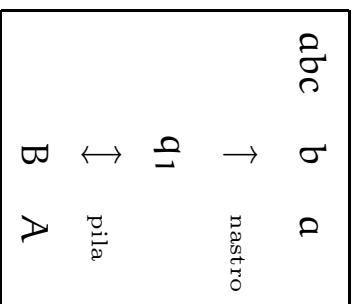
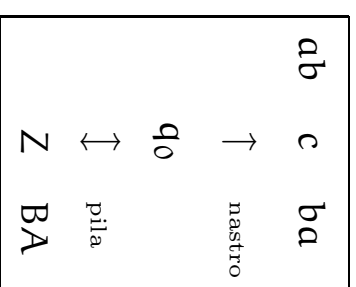
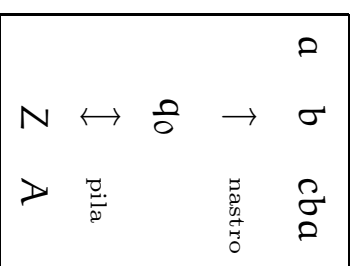
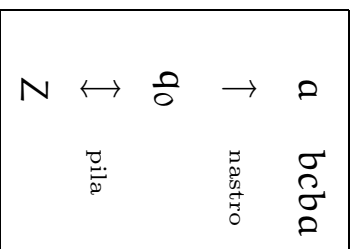
$\langle \{q_0, q_1\}, \{a, b, c\}, \{Z, A, B\}, \delta, q_0, Z, \emptyset \rangle$ *APND*

q_0	ϵ	a	b	c	q_1	ϵ	a	b	c
Z		q_0, ZA	q_0, ZB	q_1, ϵ	Z	q_1, Z	q_1, Z	q_1, Z	
A					A	q_1, ϵ	q_1, Z	q_1, Z	q_1, Z
B					B	q_1, Z	q_1, ϵ	q_1, Z	q_1, Z

Example: abcba

Remember: we will recognise the string when the input and stack are empty!

q_0	ϵ	a	b	c	q_1	ϵ	a	b	c
Z		q_0, ZA	q_0, ZB	q_1, ϵ	Z		q_1, Z	q_1, Z	
A					A		q_1, ϵ	q_1, Z	q_1, Z
B					B		q_1, Z	q_1, ϵ	q_1, Z



Example

$$L = \{ xx^R \mid x \in \{a, b\}^* \}, \Sigma = \{a, b\}$$

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $R = \{Z, A, B\}$

	q_0	ϵ	a	b
Z			q_0, AZ	q_0, BZ
A			q_0, AA	q_0, BA
			q_1, ϵ	
B			q_0, AB	q_0, BB
				q_1, ϵ

	q_1	ϵ	a	b
Z		q_1, ϵ		
A			q_1, ϵ	
B				q_1, ϵ

Exercises

Design a PDA to recognise the following languages:

$\{w \in \{0, 1\}^* \mid \text{every prefix has more 0's than 1's}\}$

$\{w \in \{0, 1\}^* \mid w \text{ has an equal number of 0's and 1's}\}$

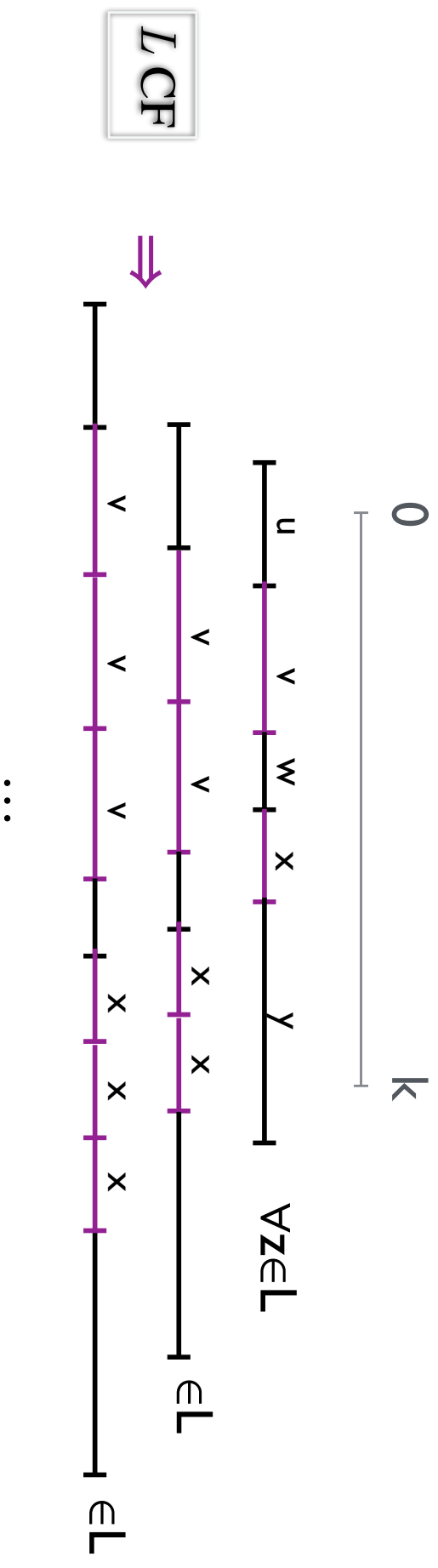
Unfortunately...

not all languages are Context Free !

Pumping Lemma for CF

Given a context free language L there exists an integer k such that for any string $z \in L$, $|z| \geq k$ it is possible to split z into 5 substrings

$z = uvwx^i y$ with $|vwx| \leq k$, $|vx| > 0$ such that $\forall i \in \mathbf{N}, uv^iwx^iy \in L$



Negating the PL for CF

The PL for CF gives a necessary condition, that can be used to prove that a language is not context free!

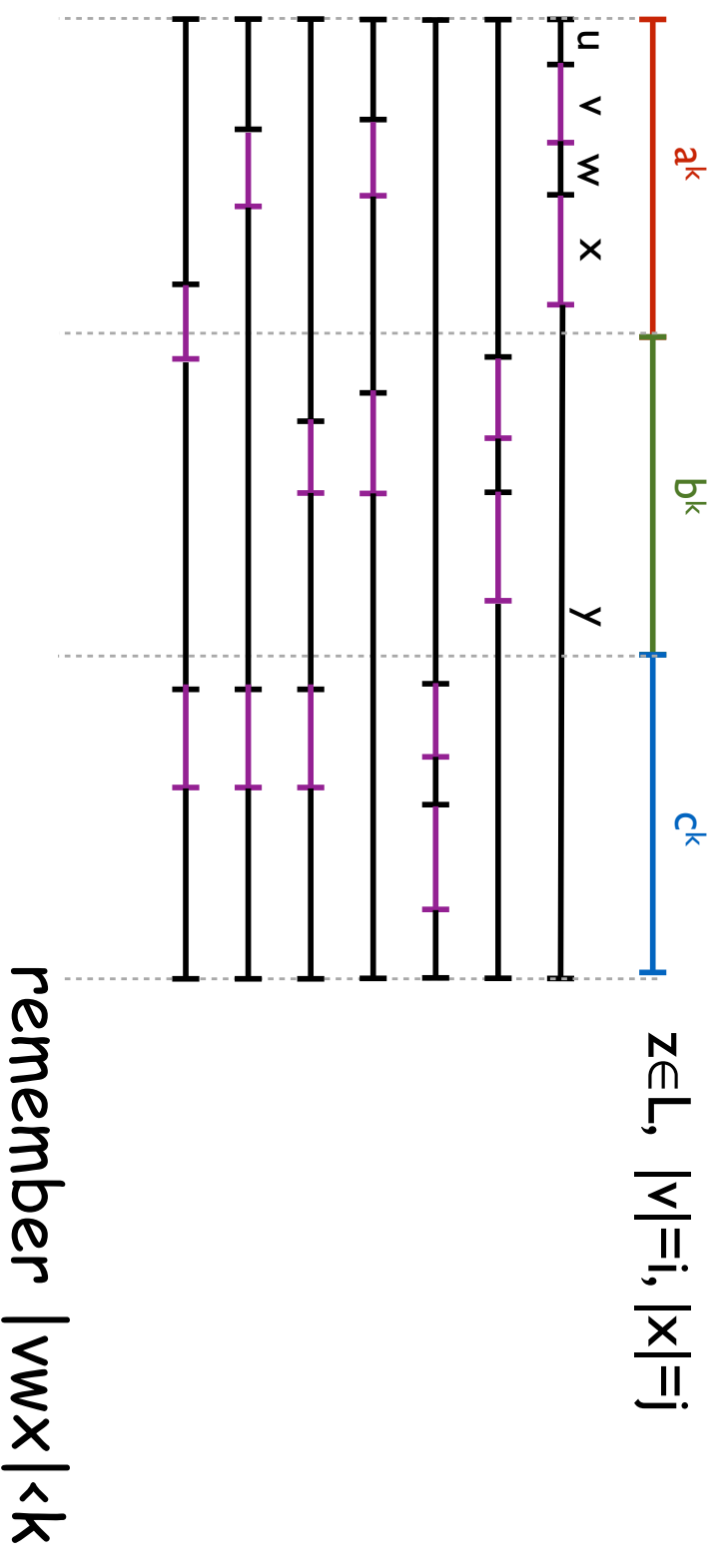
If $\forall k \in \mathbf{N} \exists z \in L. |z| \geq k$ for all possible splitting of the form

$z = uvwx$ with $|vwx| \leq k, |vx| > 0 \exists i \in \mathbf{N}$ such that $uv^iwx^iy \notin L$

then L is not context free!

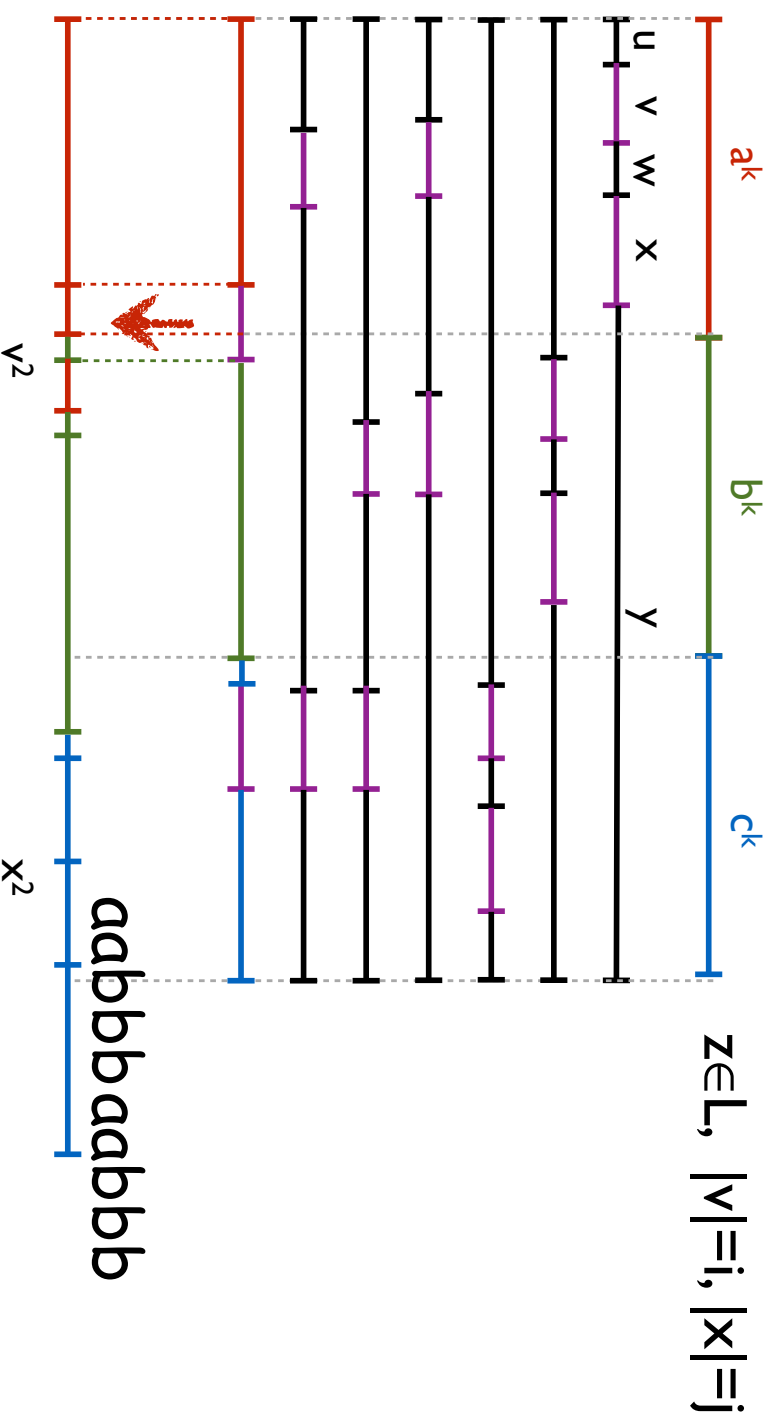
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



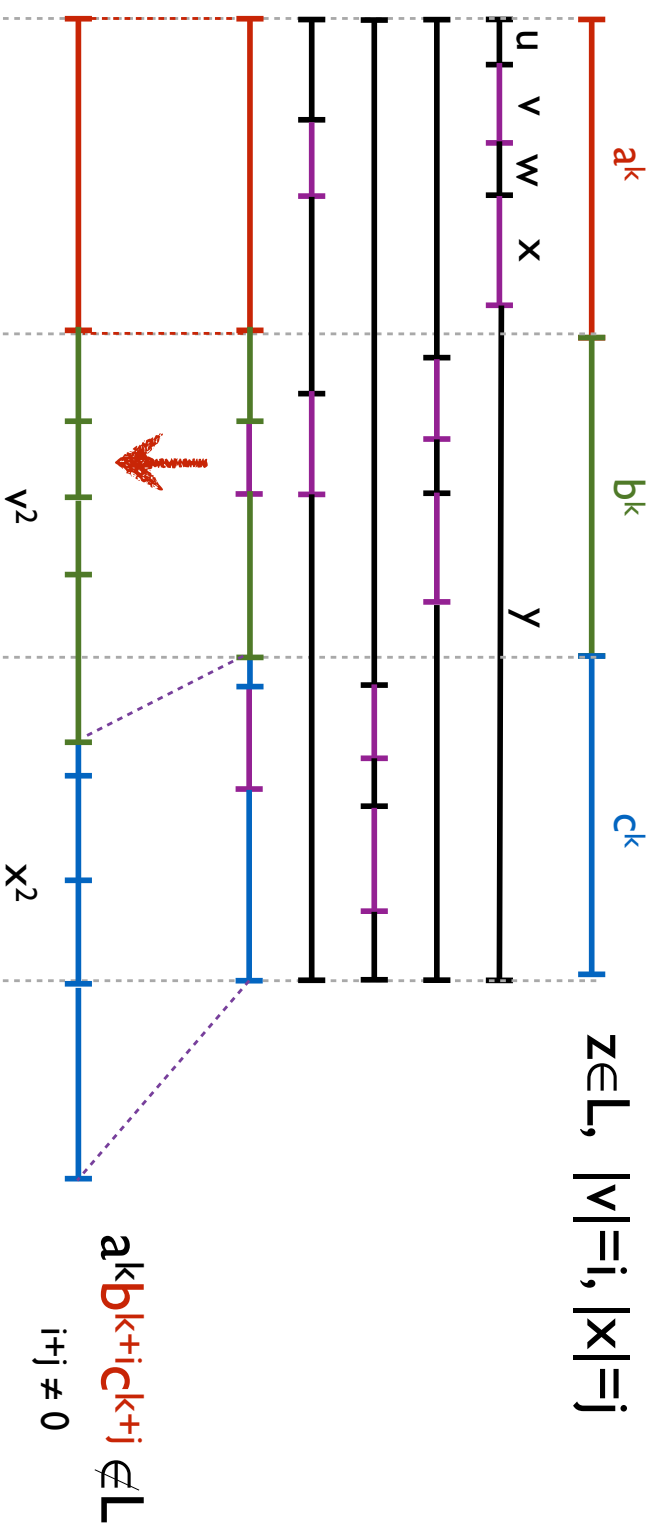
Example

- Let $L = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



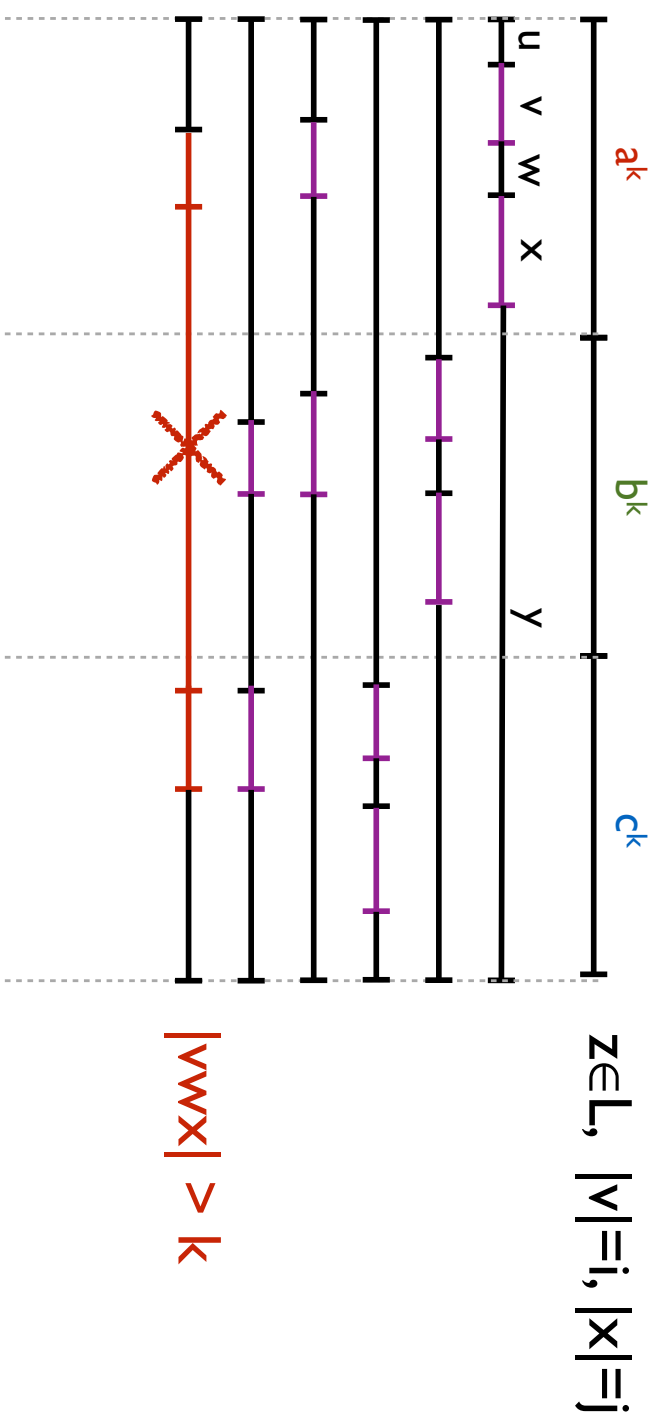
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



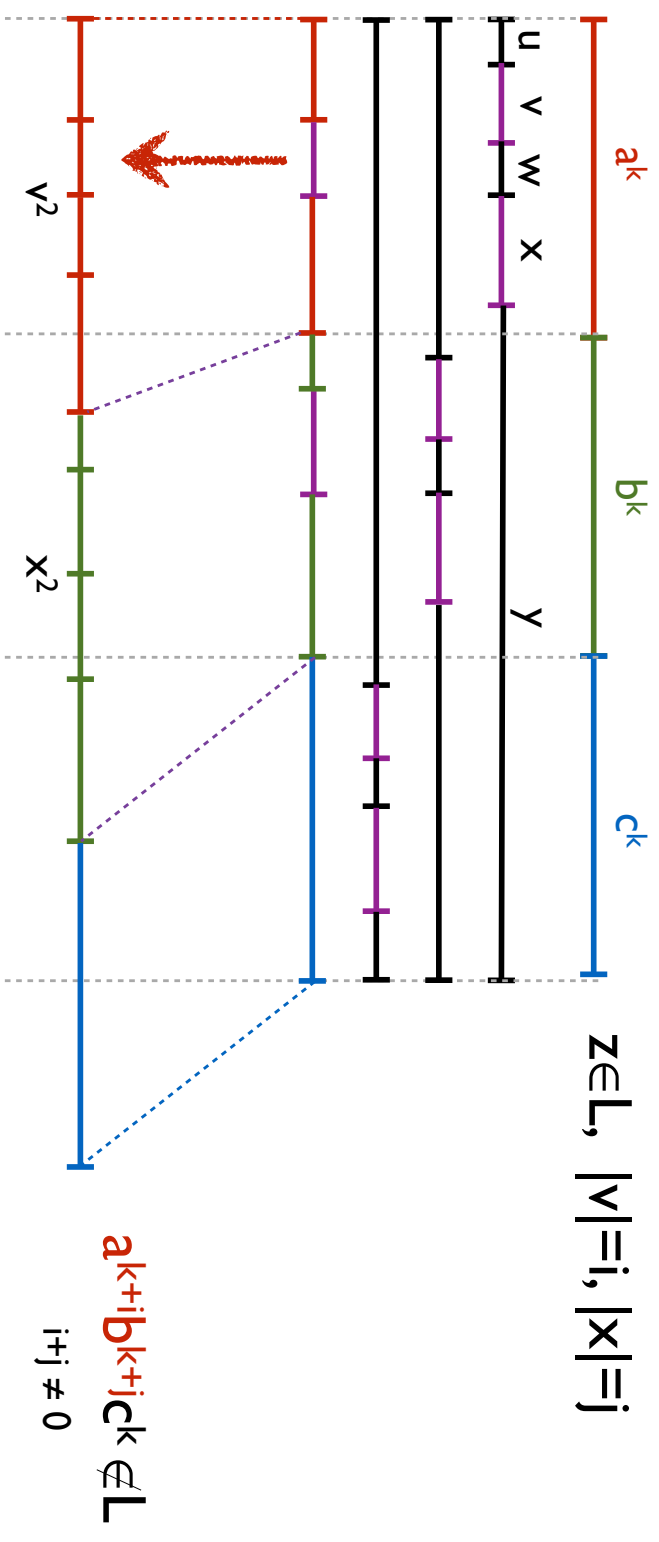
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



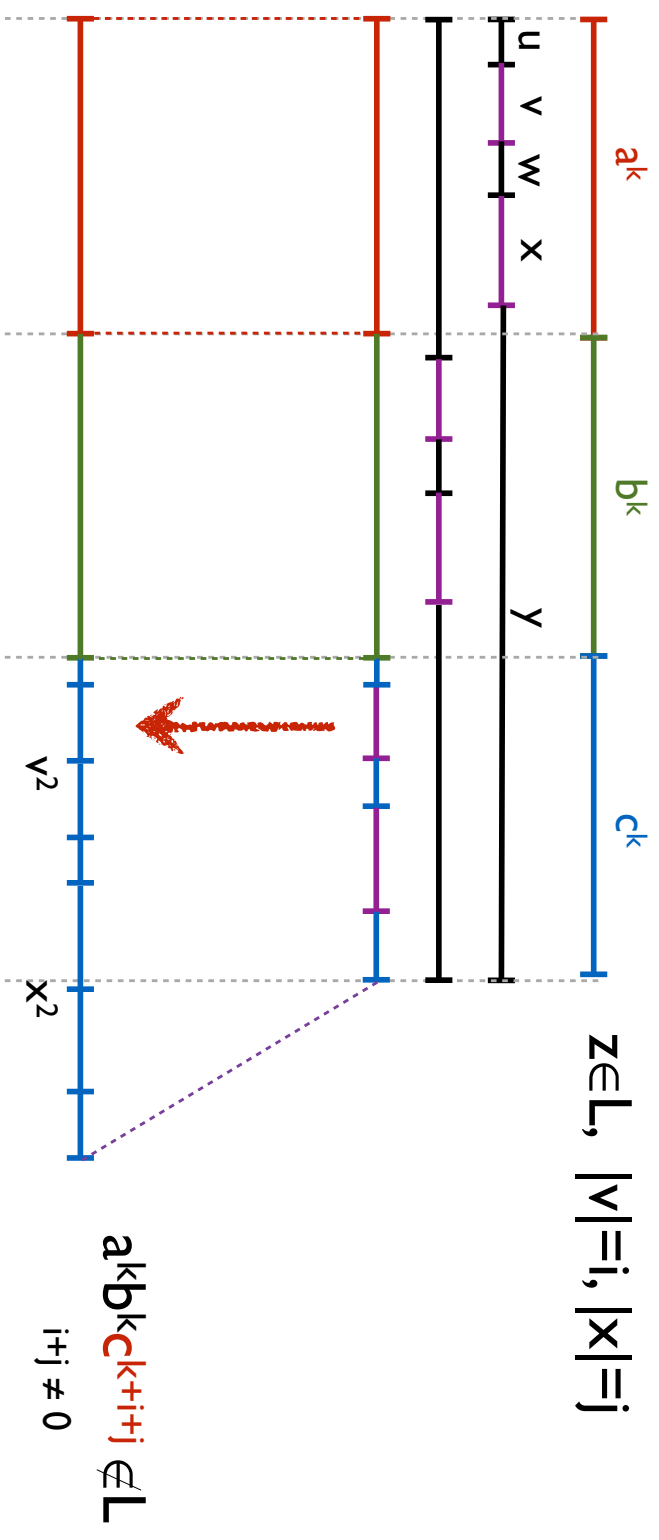
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



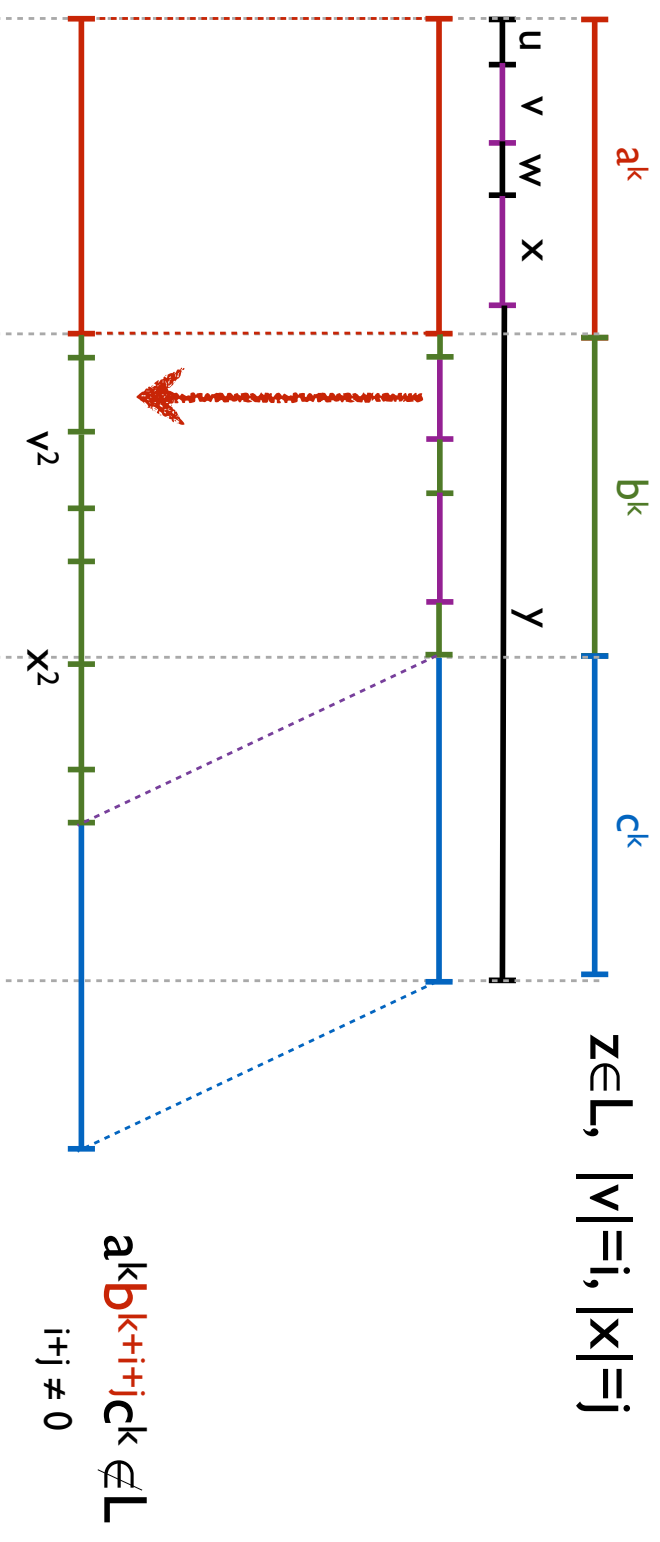
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



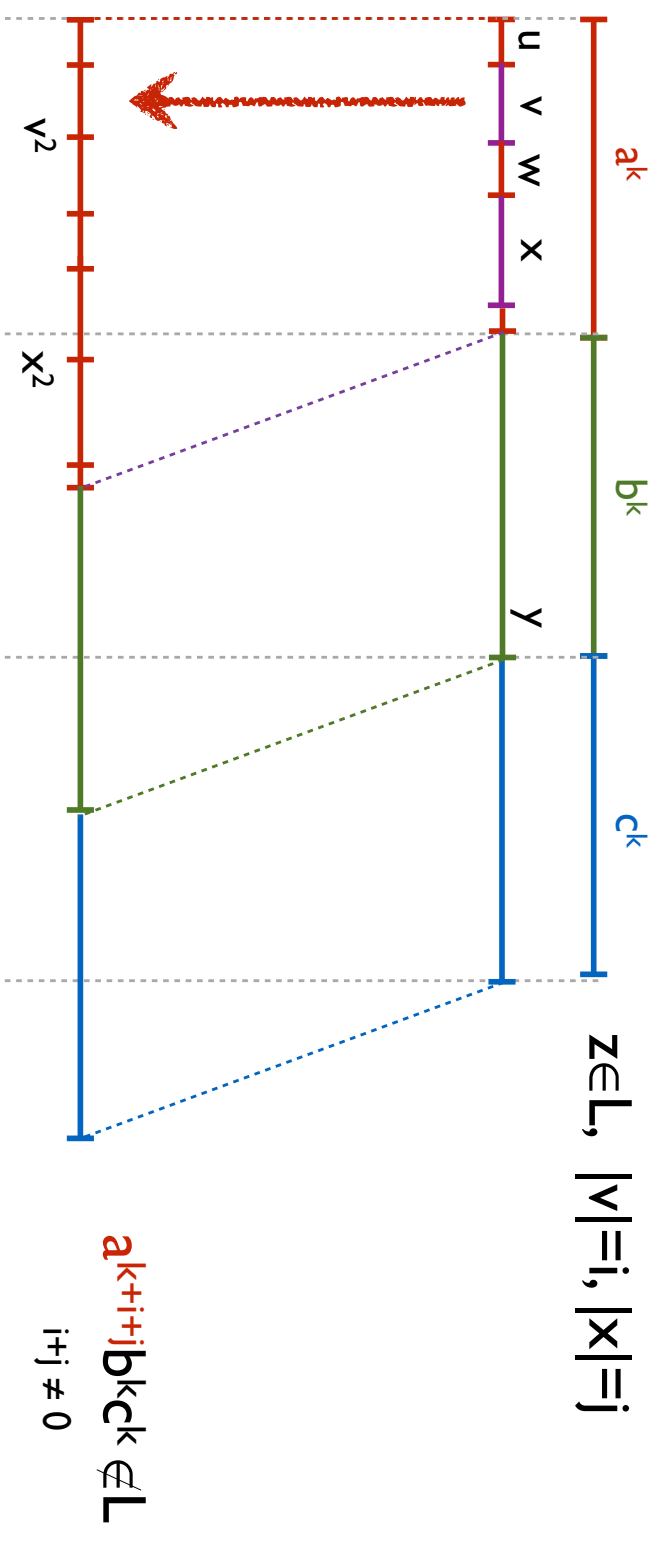
Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



Example

- Let $L = \{ a^n b^n c^n \mid n \in \text{Nat} \}$, consider $k \in \mathbb{N}$
- Let $z = a^k b^k c^k$



Exercises: are these languages context free?

$$\{0^n 1^3 n \mid n \geq 0\}$$

$$\{0^n 1^k n \mid n \geq 0 \text{ and } k \geq 0\}$$

$$\{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

$$\{a^i b^j c^k \mid k \neq i + j\}$$

$$\{w \in \{a, b\}^* \mid w \neq ww\}$$

Properties of the CF languages

The CF languages are closed with respect to the union, concatenation and Kleene closure.

The complement of CF language is not always CF.

- The CF language are not closed under intersection

Decision Properties:

Approximately all the properties are **decidable** in case of CF

- (i) Emptiness
- (ii) Non-emptiness
- (iii) Finiteness
- (iv) Infiniteness
- (v) Membership

Context Sensitive Grammar

Productions of the form $U \rightarrow V$ such that $|U| \leq |V|$

Example

$S \rightarrow aSBC \mid aBC$

$bC \rightarrow bc$

$CB \rightarrow BC$

$cC \rightarrow cc$

$bB \rightarrow bb$

$aB \rightarrow ab$

$\{a^i b^i c^i : i \geq 1\}$.

Complexity of Languages Problems

	Regular Grammar Type 3	Context Free Grammar Type 2	Context Sensitive Grammar Type 1	Unrestricted Grammar Type 0
Is $w \in L(G)$?	P	P	PSPACE	U
Is $L(G)$ empty?	P	P	U	U
Is $L(G_1) = L(G_2)$?	PSPACE	U	U	U

Examples of Language Hierarchy

The expressing expressive power:

regular \subset context-free \subset context-sensitive \subset phrase-structure

L1 = strings over $\{0, 1\}$ with an even number of 1's is regular

L2 = $\{a^n b^n \mid n \in \mathbf{N}\}$ is context-free, but not regular

L3 = $\{a^n b^n c^n \mid n \in \mathbf{N}\}$ is context-sensitive, but not context-free

Relationships between Languages and Automata

A language is :

regular	iff accepted by	finite-state automata
context-free		pushdown automata
context-sensitive		linear-bounded automata
phrase-structure		Turing machine

Chomsky's Hierarchy

