FLT: Characterization of some classes of languages and automata

Dr Giuditta Franco

Department of Computer Science, University of Verona, Italy

Chomsky grammar

$$G = (A, T, S, R)$$

A alphabet, $T \subset A$ terminal [1] alphabet ($N = A \setminus T$), $S \in N$ starting symbol, R binary relation on $A^\star$.
Namely, if $(\alpha, \beta) \in R$, then $\alpha \notin T^\star$, and we denote it $\alpha \to \beta$.

One-step rewriting: $\phi \Rightarrow_G \psi$, if $\phi = x\alpha y$, $\psi = x\beta y$, $(\alpha, \beta) \in R$
Multiple-steps rewriting [2] (transitive closure): $\phi \Rightarrow^\star_G \phi$, if $\exists \phi_1, \phi_2, \ldots, \phi_n$ such that $\phi = \phi_1$, $\psi = \phi_n$, and $\phi_i \Rightarrow_G \phi_{i+1} \forall i$

$$L(G) = \{\beta \mid \beta \in T^\star, S \Rightarrow^\star_G \beta\}$$

[1] halt criterion

[2] Conventiona rewriting systems are non-deterministically sequencial. Unconventional ways to apply these rules: probabilistic, by priority relation, (maximal, conditionated) parallel.

Examples

$\{S \to a, S \to aS\}$ generates monosomatic language

$\{S \to ab, S \to aSb\}$ generates bisomatic language

$\{S \to aS, S \to Sb, S \to \lambda\}$ generates bipartite language

$\{S \to abc, S \to aSBc, cB \to Bc, bB \to bb\}$ generates trisomatic language

$\{S \to abc, S \to aaBbc, Bb \to bbC, Cb \to bC, Cc \to cc, Cc \to Dcc, bD \to Db, aD \to aaB\}$ .. homework

Four types of grammars/languages

Type 0: $\alpha \in A^\star N A^\star$ (general form)

Type 1: (type 0 AND) $|\alpha| \leq |\beta|$ (monotonic form)

Type 2: (type 1 AND) $|\alpha| = 1$ (context-free form)

Type 3: (type 2 AND) $\beta \in T \cup TN$ (regular form).

For i=0,1,2,3, a grammar is of type *i* if all its rules are of type *i*.

$\mathcal{L}_i$ is the class of languages generated by grammars of type *i*.

A language L is of type *i* if $L \in \mathcal{L}_i \backslash \mathcal{L}_{i-1}$.

By definition, $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$. With strict inclusions, this is the Chomsky hierarchy.

Bottom and top of the Chomsky hierarchy

Definition: FIN is the class of finite languages.

Since $a^\star$ is an infinite regular language (see grammar $S \to a$ and $S \to aS$), then $\{a\}^\star \in \mathcal{L}_3 \setminus FIN$, and we have proved that $FIN \subset REG$.

Def: a language is infinite, if in its grammar there exists an *autoduplicating rule*: $\alpha \to \beta$ with $X \in N$ such that X occurs both in α and β. In formal terms: $\exists X \in N$ and $\alpha \to \beta$, such that $\alpha = \gamma X \delta$ and $\beta = \eta X \epsilon$ for some $\gamma, \delta, \eta, \epsilon \in A^\star$.

Remark: $|\mathcal{P}(A^\star)| > |\mathcal{L}_0|$. Here we notice $RE \subset \mathcal{P}(A^\star)$. We will see a language outside RE, in the proof of Post theorem.

Regular languages: REG

REG is defined as $Close(A, ;+, \star)$.

Theorem (Kleene): $REG = \mathcal{L}_3$

Theorem: $\mathcal{L}_3 = L(FSA)$, where $L(FSA) = \{\alpha \mid q_0\alpha \Rightarrow^{\star}_{M} q_f, q_f \in F\}$.
Proof: $REG \subseteq L(FSA)$ easy (by def of REG). Viceversa, it is evident by comparing FSA transitions and grammars of type 3.

Then, FSA recognize all and only regular languages.

$DFSA \equiv NDFSA$, FSA are equivalent to λ-transition FSA, and to one-final-state FSA.

Remark: $\mathcal{L}_3$ is closed by $\star$, by union, by complement, by intersection (to prove, homework).

Examples of regular patterns

Examples of regular languages: strings over $\{a, b, c\}$ starting with b and ending not with a: xyz, where x=b, $y \in \{a, b, c\}^\star$, $z \in \{b, c\}$, or strings starting with a, followed by an arbitrary number of bs and ending with the string bab or aba: $ab^\star(bab + aba)$.

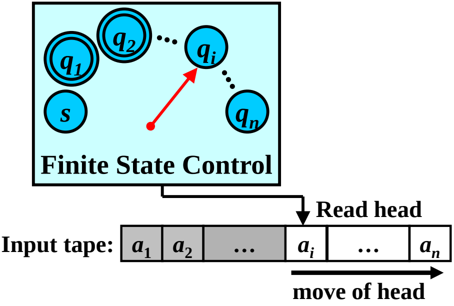
$a^\star(b + c)c^4$

$a^\star bbc^\star$

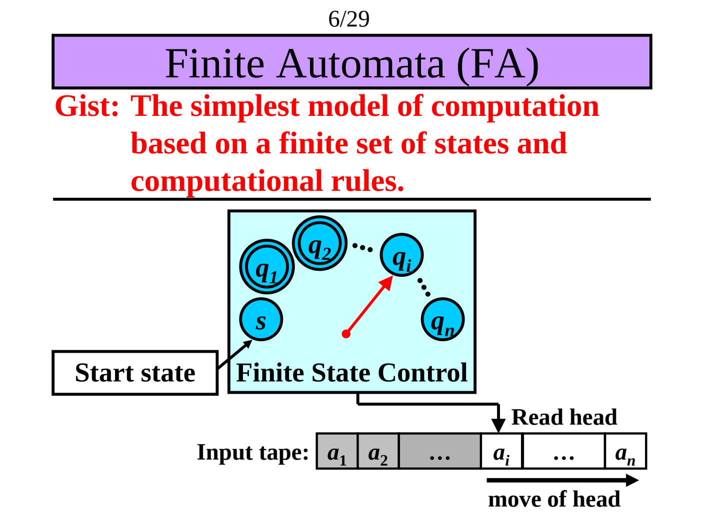
6/29

Finite Automata (FA)

Gist: The simplest model of computation based on a finite set of states and computational rules.



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

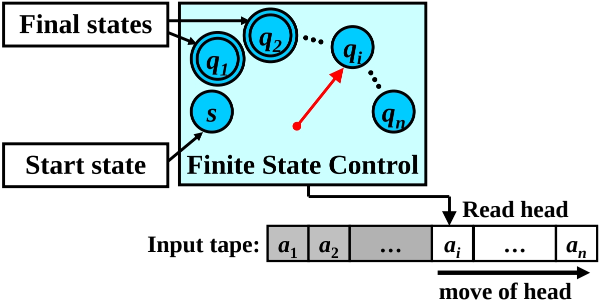


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

6/29

Finite Automata (FA)

Gist: The simplest model of computation based on a finite set of states and computational rules.

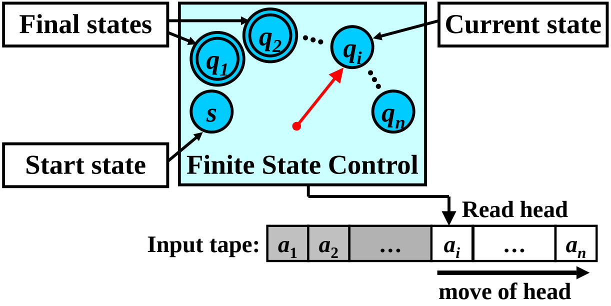


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

6/29

Finite Automata (FA)

Gist: The simplest model of computation based on a finite set of states and computational rules.



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Finite Automata: Definition

Definition: *A finite automaton* (FA) is a 5-tuple:

$$M = (Q, \Sigma, R, s, F), \text{ where}$$

- Q is a *finite set of states*
- Σ is an *input alphabet*
- R is a *finite set of rules* of the form: $pa \to q$, where $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$
- $s \in Q$ is the *start state*
- $F \subseteq Q$ is a set of *final states*

Mathematical note on rules:

- Strictly mathematically, R is a relation from $Q \times (\Sigma \cup \{\varepsilon\})$ to Q
- Instead of (pa, q), however, we write the rule as $pa \to q$

- $pa \to q$ means that with a, M can move from p to q
- if $a = \varepsilon$, no symbol is read

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation

q denotes a state $q \in Q$

$\rightarrow s$ denotes the start state $s \in Q$

f denotes a final state $f \in F$

$p \xrightarrow{a} q$ denotes $pa \rightarrow q \in R$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

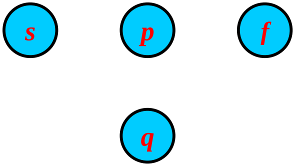
$M = (Q, \Sigma, R, s, F)$,
where:

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;

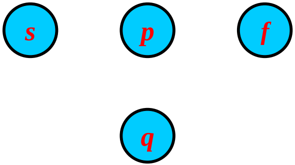


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;

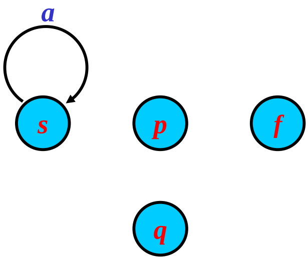


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \rightarrow s$,

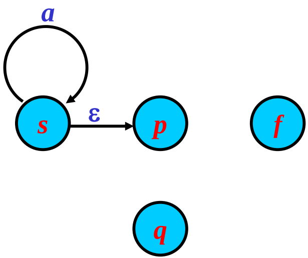


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$

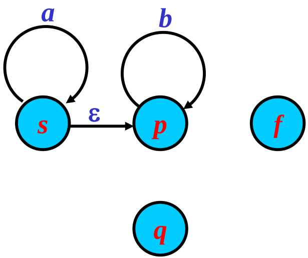


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \rightarrow s,$
  $s \rightarrow p,$
  $pb \rightarrow p,$

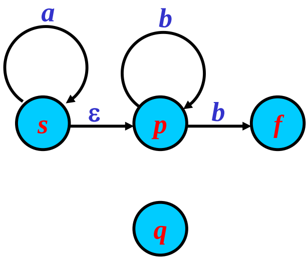


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$

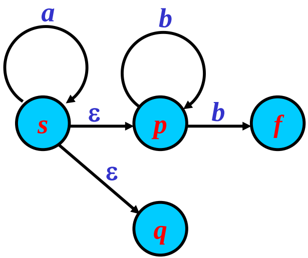


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$, where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$
  $s \to q,$

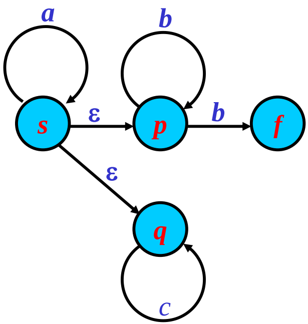


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \rightarrow s,$
  $s \rightarrow p,$
  $pb \rightarrow p,$
  $pb \rightarrow f,$
  $s \rightarrow q,$
  $qc \rightarrow q,$



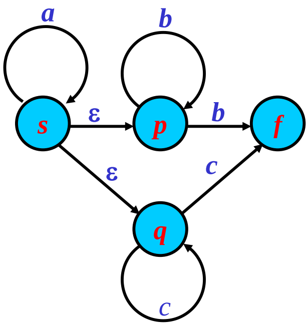
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

9/29

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$
  $s \to q,$
  $qc \to q,$
  $qc \to f,$

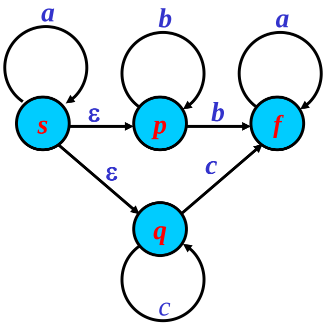


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$
  $s \to q,$
  $qc \to q,$
  $qc \to f,$
  $fa \to f\}$;

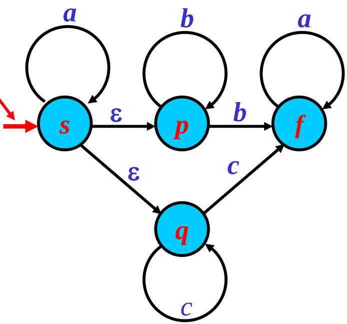


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$
  $s \to q,$
  $qc \to q,$
  $qc \to f,$
  $fa \to f\}$;

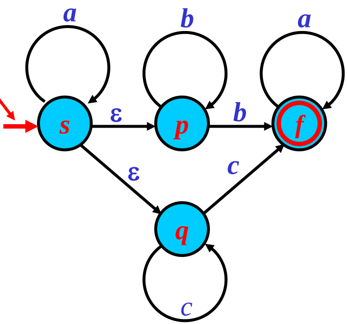


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, R, s, F)$,
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b, c\}$;
- $R = \{sa \to s,$
  $s \to p,$
  $pb \to p,$
  $pb \to f,$
  $s \to q,$
  $qc \to q,$
  $qc \to f,$
  $fa \to f\}$;
- $F = \{f\}$



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Accepted Language

Gist: *M* accepts *w* if it can completely read *w* by a sequence of moves from *s* to a final state

Definition: Let $M = (Q, \Sigma, R, s, F)$ be a FA. The *language accepted by M*, $L(M)$, is defined as:

$$L(M) = \{w: w \in \Sigma^*, sw \vdash^* f,\ f \in F\}$$

$M = (Q, \Sigma, R, s, F)$:

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

16/29

Accepted Language

Gist: ***M*** **accepts** ***w*** **if it can completely read** ***w*** **by a sequence of moves from** ***s*** **to a final state**

Definition: Let $M = (Q, \Sigma, R, s, F)$ be a FA. The *language accepted by M*, $L(M)$, is defined as:

$$L(M) = \{w: w \in \Sigma^*, sw \vdash^* f, f \in F\}$$

$M = (Q, \Sigma, R, s, F)$:

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

16/29

Accepted Language

Gist: ***M* accepts *w* if it can completely read *w* by a sequence of moves from *s* to a final state**

Definition: Let $M = (Q, \Sigma, R, s, F)$ be a FA. The *language accepted by M*, $L(M)$, is defined as:

$$L(M) = \{w: w \in \Sigma^*, sw \vdash^* f,\ f \in F\}$$

$M = (Q, \Sigma, R, s, F)$:

$$s\underbrace{a_1a_2\ldots a_n}_{w} \vdash q_1a_2\ldots a_n \vdash \ldots \vdash q_{n-1}a_n \vdash q_n$$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

16/29

Accepted Language

Gist: M accepts w if it can completely read w by a sequence of moves from s to a final state

Definition: Let $M = (Q, \Sigma, R, s, F)$ be a FA. The *language accepted by M*, $L(M)$, is defined as:

$$L(M) = \{w : w \in \Sigma^*, sw \vdash^* f,\ f \in F\}$$

$M = (Q, \Sigma, R, s, F)$:

$$s\underbrace{a_1a_2\ldots a_n}_{w} \vdash q_1a_2\ldots a_n \vdash \ldots \vdash q_{n-1}a_n \vdash q_n$$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Accepted Language

Gist: *M* accepts *w* if it can completely read *w* by a sequence of moves from *s* to a final state

Definition: Let $M = (Q, \Sigma, R, s, F)$ be a FA. The *language accepted by M*, $L(M)$, is defined as:

$$L(M) = \{w : w \in \Sigma^*, sw \vdash^* f, \ f \in F\}$$

$M = (Q, \Sigma, R, s, F)$:

if $q_n \in F$ then $w \in L(M)$;
otherwise, $w \notin L(M)$

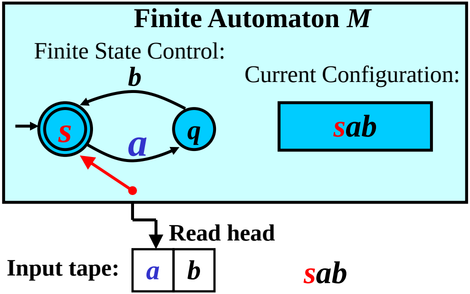
$$s\underbrace{a_1a_2\ldots a_n}_{w} \vdash q_1a_2\ldots a_n \vdash \ldots \vdash q_{n-1}a_n \vdash q_n$$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

FA: Example 1/3

$M = (Q, \Sigma, R, s, F)$, where:
$Q = \{s, q\}$, $\Sigma = \{a, b\}$, $R = \{sa \rightarrow q, qb \rightarrow s\}$, $F = \{s\}$

Question: $ab \in L(M)$ **?**



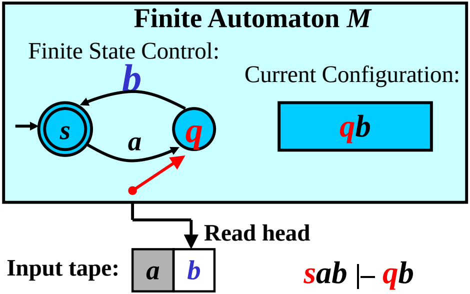
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

FA: Example 2/3

$M = (Q, \Sigma, R, s, F)$, where:

$Q = \{s, q\}$, $\Sigma = \{a, b\}$, $R = \{sa \rightarrow q, qb \rightarrow s\}$, $F = \{s\}$

Question: $ab \in L(M)$ **?**



$sab \vdash qb$

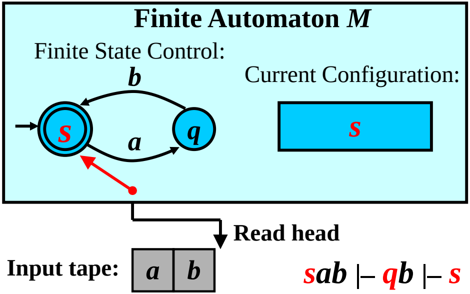
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

19/29

FA: Example 3/3

$M = (Q, \Sigma, R, s, F)$, where:
$Q = \{s, q\}$, $\Sigma = \{a, b\}$, $R = \{sa \to q, qb \to s\}$, $F = \{s\}$

Question: $ab \in L(M)$ **?**



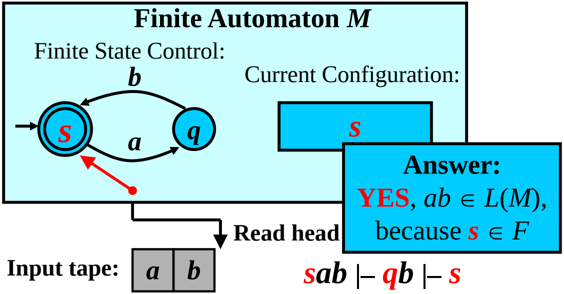
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

FA: Example 3/3

$M = (Q, \Sigma, R, s, F)$, where:

$Q = \{s, q\}$, $\Sigma = \{a, b\}$, $R = \{sa \rightarrow q, qb \rightarrow s\}$, $F = \{s\}$

Question: $ab \in L(M)$ **?**

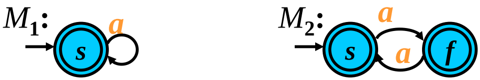


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Equivalent Models

Definition: Two models for languages, such as FAs, are equivalent if they both specify the same language.

Example:



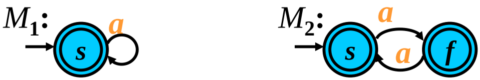
Question: Is M_1 equivalent to M_2 ?

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Equivalent Models

Definition: Two models for languages, such as FAs, are equivalent if they both specify the same language.

Example:



Question: Is M_1 equivalent to M_2 ?

Answer: M_1 and M_2 **are equivalent** because
$$L(M_1) = L(M_2) = \{a^n : n \geq 0\}$$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties 1/2

Definition: The family of regular languages is closed under an operation $\boldsymbol{o}$ if the language resulting from the application of $\boldsymbol{o}$ to any regular languages is also regular.

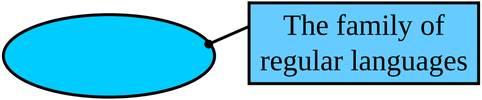
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties 1/2

Definition: The family of regular languages is closed under an operation ***o*** if the language resulting from the application of ***o*** to any regular languages is also regular.

Illustration:

- The family of regular languages is closed under ***union***. It means:



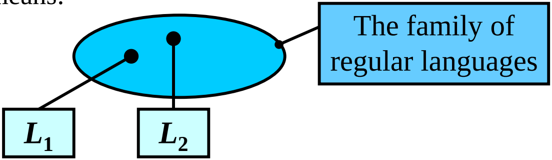
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties 1/2

Definition: The family of regular languages is closed under an operation ***o*** if the language resulting from the application of ***o*** to any regular languages is also regular.

Illustration:

- The family of regular languages is closed under ***union***. It means:



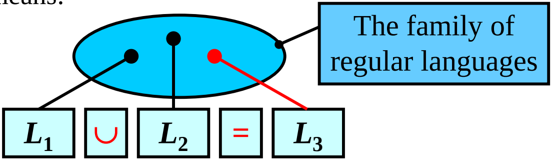
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties 1/2

Definition: The family of regular languages is closed under an operation ***o*** if the language resulting from the application of ***o*** to any regular languages is also regular.

Illustration:

- The family of regular languages is closed under ***union***. It means:



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties 2/2

Theorem: The family of regular languages is closed under **union**, **concatenation**, **iteration**.

Proof:

- Let L_1, L_2 be two **regular languages**
- Then, there exist two REs r_1, r_2: $L(r_1) = L_1$, $L(r_2) = L_2$;
- By the definition of regular expressions:
  - $r_1.r_2$ is a RE denoting $L_1 L_2$
  - $r_1 + r_2$ is a RE denoting $L_1 \cup L_2$
  - $r_1{}^*$ is a RE denoting $L_1{}^*$
- Every RE denotes regular language, so $L_1 L_2$, $L_1 \cup L_2$, $L_1{}^*$ are a **regular languages**

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties: Complement

Theorem: the family of regular languages is closed under complement.

Proof: Let L be a regular language. Then, there exists a complete DFA M: L(M) = L.

We can construct a complete DFA M'(having as final states the complement of the final states of M) s.t. $L(M') = \overline{L}$

Every FA defines a regular language, so the complement of L is a regular language.

Closure properties: Intersection

Theorem: The family of regular languages is closed under **intersection**.

Proof:

- Let L_1, L_2 be two **regular languages**
- $\overline{L_1}$, $\overline{L_2}$ are **regular languages**

(the family of regular languages is closed under complement)
- $\overline{L_1} \cup \overline{L_2}$ is a **regular language**

(the family of regular languages is closed under union)
- $\overline{\overline{L_1} \cup \overline{L_2}}$ is a **regular language**

(the family of regular languages is closed under complement)
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ is a **regular language** (DeMorgan's law)

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Boolean Algebra of Languages

Definition: Let a family of languages be closed under union, intersection, and complement. Then, this family represents a ***Boolean algebra of languages***.

Theorem: The family of regular languages is a Boolean algebra of languages.

Proof:

• The family of regular languages is closed under union, intersection, and complement.

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma

By definition, $CF = \mathcal{L}_2$ and $CS = \mathcal{L}_1$.

Pumping Lemma Let L be an *infinite* CF language, $\exists$ two numbers $q \leq p$, such that, $\forall \alpha \in L$ with $|\alpha| \geq p$, $\exists u, v, w, x, y$, $vx \neq \lambda$, $|vwx| \leq q$ such that $\alpha = uvwxy$ and $\forall i, uv^i wx^i y \in L$.

v and x are said *ancillaries*. Proof by the application of a free-context autoreplicative rule.

This lemma characterizes CF languages, and allows us to show that $REG \subset CF \subset CS$. Examples of this string inclusions are the bisomatic and trisomatic languages.

For any $a^n b^n$, $\exists q \leq p$ even and different than zero, such that, $2n \geq p$, $w = \lambda, u = a^{\frac{2n-q}{2}}, v = a^{\frac{q}{2}}, x = b^{\frac{q}{2}}, y = b^{\frac{2n-q}{2}}$

Bisomatic is a context free and non-regular language

$$a^n b^n \in CF \backslash REG$$

By contradiction, let us assume M to be a *deterministic* FSA recognizing $a^n b^n$. $\exists n \neq m$ such that both a^n and a^m end up in one same state, then $a^n b^n$ and $a^m b^n$ end up in the same final state.

Pigeon(hole) principle (“dei cassetti”): if n items are put into m containers, with $n > m$, then at least one container must contain more than one item.

Pumping Lemma for CFL

• Let L be CFL. Then, there exists $k \geq 1$ such that:
if $z \in L$ and $|z| \geq k$ **then** there exist u, v, w, x, y so
$z = uvwxy$ and
1) $vx \neq \varepsilon$ **2)** $|vwx| \leq k$ **3)** for each $m \geq 0$, $uv^m wx^m y \in L$

Example:

$G = (\{S, A\}, \{a, b, c\}, \{S \to aAa, A \to bAb, A \to c\}, S)$
generate $L(G) = \{ab^n cb^n a : n \geq 0\}$, so $L(G)$ is CFL.
There is $k = 5$ such that **1)**, **2)** and **3)** holds:

• for $z = abcba$: $z \in L(G)$ and $|z| \geq 5$:

$uvwxy$

$uv^0wx^0y = ab^0cb^0a = aca \in L(G)$
$uv^1wx^1y = ab^1cb^1a = abcba \in L(G)$
$uv^2wx^2y = ab^2cb^2a = abbcbba \in L(G)$
⋮

$\boldsymbol{vx} = bb \neq \varepsilon$
$|\boldsymbol{vwx}| = 3$: $1 \leq 3 \leq 5$

• for $z = abbcbba$: $z \in L(G)$ and $|z| \geq 5$:
⋮

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Illustration

- L = any context-free language:

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Illustration

- L = any context-free language:

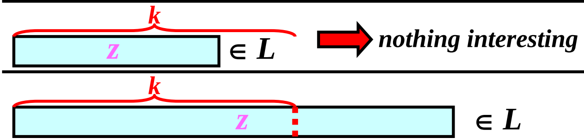
$\overbrace{z}^{k} \in L$ ⇒ *nothing interesting*

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

14/31

Pumping Lemma: Illustration

- L = any context-free language:

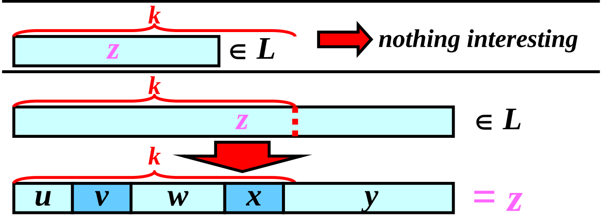


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

14/31

Pumping Lemma: Illustration

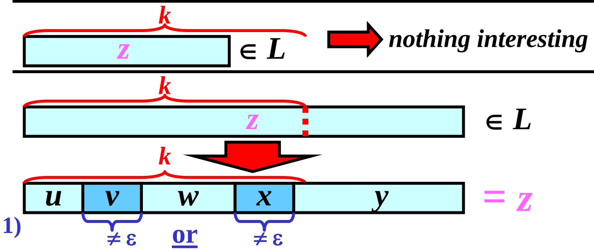
- L = any context-free language:



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Illustration

• L = any context-free language:

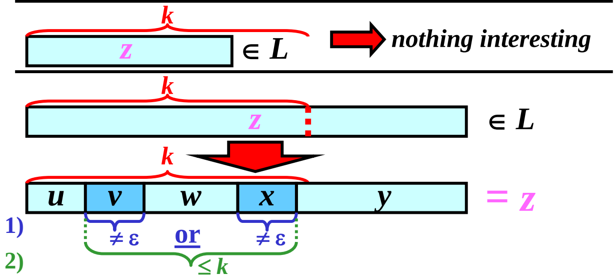


Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

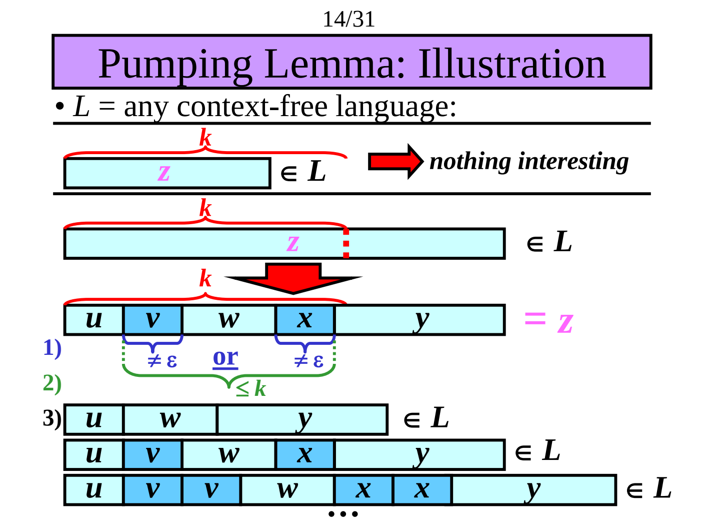
14/31

Pumping Lemma: Illustration

- L = any context-free language:



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Application

• Based on the pumping lemma for CFL, we often make a proof by contradiction to demonstrate that a language is **not** a CFL.

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Application

• Based on the pumping lemma for CFL, we often make a proof by contradiction to demonstrate that a language is **not** a CFL.

Assume that L is a CFL.

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

15/31

Pumping Lemma: Application

• Based on the pumping lemma for CFL, we often make a proof by contradiction to demonstrate that a language is **not** a CFL.

Assume that L is a CFL.

↓

Consider the PL constant $\boldsymbol{k}$ and select $\boldsymbol{z} \in L$, whose length depends on $\boldsymbol{k}$ so $|\boldsymbol{z}| \geq \boldsymbol{k}$ is surely true.

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Application

• Based on the pumping lemma for CFL, we often make a proof by contradiction to demonstrate that a language is **not** a CFL.

Assume that L is a CFL.

↓

Consider the PL constant $\boldsymbol{k}$ and select $\boldsymbol{z} \in L$, whose length depends on $\boldsymbol{k}$ so $|\boldsymbol{z}| \geq \boldsymbol{k}$ is surely true.

↓

For all decompositions of $\boldsymbol{z}$ into $\boldsymbol{uvwxy}$: $vx \neq \varepsilon$, $|vwx| \leq k$, show that there exists $m \geq 0$ such that $\boldsymbol{uv^mwx^my} \notin \boldsymbol{L}$;
from the pumping lemma, $\boldsymbol{uv^mwx^my} \in \boldsymbol{L}$

$\Big\}$ **contradiction**

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

15/31

Pumping Lemma: Application

• Based on the pumping lemma for CFL, we often make a proof by contradiction to demonstrate that a language is **not** a CFL.

Assume that L is a CFL.

↓

Consider the PL constant $\boldsymbol{k}$ and select $\boldsymbol{z} \in L$, whose length depends on $\boldsymbol{k}$ so $|\boldsymbol{z}| \geq \boldsymbol{k}$ is surely true.

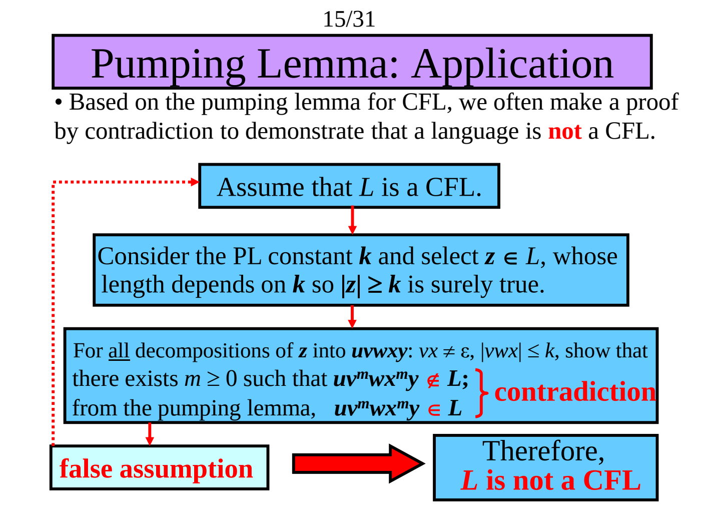
↓

For all decompositions of $\boldsymbol{z}$ into $\boldsymbol{uvwxy}$: $vx \neq \varepsilon$, $|vwx| \leq k$, show that
$\left.\begin{array}{l}\text{there exists } m \geq 0 \text{ such that } \boldsymbol{uv^m wx^m y} \notin \boldsymbol{L}; \\ \text{from the pumping lemma, } \boldsymbol{uv^m wx^m y} \in \boldsymbol{L}\end{array}\right\}$ **contradiction**

↓

false assumption

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Pumping Lemma: Example 1/2

Prove that $L = \{a^n b^n c^n : n \geq 1\}$ is not CFL.

1) Assume that L is a CFL. Let $k \geq 1$ be the pumping lemma constant for L.

2) Let $z = a^k b^k c^k$: $a^k b^k c^k \in L$, $|z| = |a^k b^k c^k| = 3k \geq k$

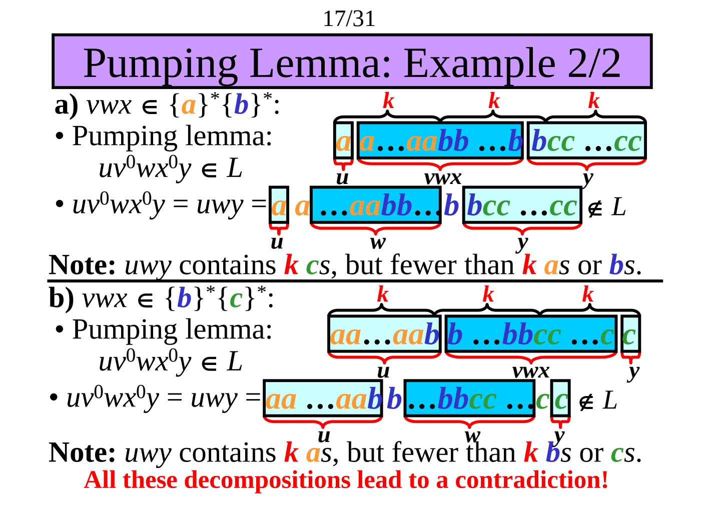
3) All decompositions of z into $uvwxy$; $vx \neq \varepsilon$, $|vwx| \leq k$:

$$\overbrace{aaaaa\ldots aa}^{k}\overbrace{bb\ldots bb\ldots bb}^{k}\overbrace{cc\ldots ccccc}^{k}$$

a) $vwx \in \{a\}^*\{b\}^*$, $vx \neq \varepsilon$

b) $vwx \in \{b\}^*\{c\}^*$, $vx \neq \varepsilon$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)



Pumping Lemma: Example 2/2

a) $vwx \in \{a\}^*\{b\}^*$:

- Pumping lemma: $uv^0wx^0y \in L$

$$\overbrace{a\,a\ldots aa}^{k}\overbrace{bb\ldots b}^{k}\overbrace{bcc\ldots cc}^{k}$$

$$\underbrace{a}_{u}\underbrace{a\ldots aabb\ldots b}_{vwx}\underbrace{bcc\ldots cc}_{y}$$

- $uv^0wx^0y = uwy = \underbrace{a}_{u}\,a\,\underbrace{\ldots aabb\ldots}_{w}\,b\,\underbrace{bcc\ldots cc}_{y} \notin L$

Note: uwy contains k cs, but fewer than k as or bs.

b) $vwx \in \{b\}^*\{c\}^*$:

- Pumping lemma: $uv^0wx^0y \in L$

$$\overbrace{aa\ldots aa}^{k}\overbrace{bb\ldots bb}^{k}\overbrace{cc\ldots cc}^{k}$$

$$\underbrace{aa\ldots aab}_{u}\underbrace{b\ldots bbcc\ldots c}_{vwx}\underbrace{c}_{y}$$

- $uv^0wx^0y = uwy = \underbrace{aa\ldots aab}_{u}\,b\,\underbrace{\ldots bbcc\ldots}_{w}\,c\,\underbrace{c}_{y} \notin L$

Note: uwy contains k as, but fewer than k bs or cs.

All these decompositions lead to a contradiction!

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

17/31

Pumping Lemma: Example 2/2

a) $vwx \in \{a\}^*\{b\}^*$:

- Pumping lemma: $uv^0wx^0y \in L$
- $uv^0wx^0y = uwy = \underbrace{a}_{u}\,a\,\underbrace{\ldots aabb\ldots}_{w}\,b\,\underbrace{bcc\ldots cc}_{y} \notin L$

$$\overbrace{\underbrace{a}_{u}\underbrace{a\ldots aabb\ldots b}_{vwx}\underbrace{bcc\ldots cc}_{y}}^{k\quad k\quad k}$$

Note: uwy contains k cs, but fewer than k as or bs.

b) $vwx \in \{b\}^*\{c\}^*$:

- Pumping lemma: $uv^0wx^0y \in L$
- $uv^0wx^0y = uwy = \underbrace{aa\ldots aab}_{u}\,b\,\underbrace{\ldots bbcc\ldots}_{w}\,c\,\underbrace{c}_{y} \notin L$

$$\overbrace{\underbrace{aa\ldots aab}_{u}\underbrace{b\ldots bbcc\ldots c}_{vwx}\underbrace{c}_{y}}^{k\quad k\quad k}$$

Note: uwy contains k as, but fewer than k bs or cs.

All these decompositions lead to a contradiction!

4) Therefore, L is not a CFL.

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties of CFL

Definition: The family of CFLs is closed under an operation ***o*** if the language resulting from the application of ***o*** to **any** CFLs is a CFL as well.

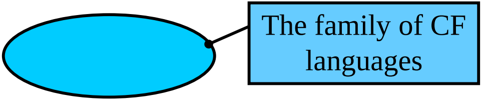
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties of CFL

Definition: The family of CFLs is closed under an operation ***o*** if the language resulting from the application of ***o*** to **any** CFLs is a CFL as well.

Illustration:

- The family of CF languages is closed under ***union***.
  It means:



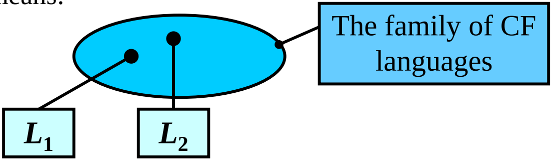
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties of CFL

Definition: The family of CFLs is closed under an operation ***o*** if the language resulting from the application of ***o*** to **any** CFLs is a CFL as well.

Illustration:

- The family of CF languages is closed under ***union***. It means:



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

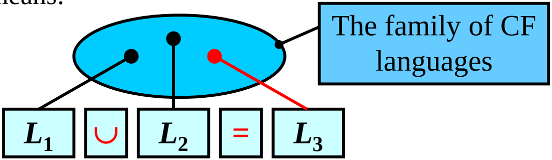
18/31

Closure properties of CFL

Definition: The family of CFLs is closed under an operation ***o*** if the language resulting from the application of ***o*** to **any** CFLs is a CFL as well.

Illustration:

- The family of CF languages is closed under ***union***.
  It means:



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Closure properties

Theorem: The family of CFLs is closed under **union**, **concatenation**, **iteration**.

Proof:

- Let L_1, L_2 be two **CFLs.**
- Then, there exist two CFGs G_1, G_2 such that $L(G_1) = L_1$, $L(G_2) = L_2$;
- Construct grammars
  - G_u such that $L(G_u) = L(G_1) \cup L(G_2)$
  - G_c such that $L(G_c) = L(G_2) \,.\, L(G_2)$
  - G_i such that $L(G_i) = L(G_1)^*$

  by using the previous three algorithms
- Every CFG denotes CFL, so
- $L_1 L_2$, $L_1 \cup L_2$, L_1^* are **CFLs.**

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Intersection: Not Closed

Theorem: The family of CFLs is **not** closed under **intersection**.

Proof:

- The intersection of some CFLs is not a CFL:

- $L_1 = \{a^m b^n c^n : m, n \geq 1\}$ is a CFL
- $L_2 = \{a^n b^n c^m : m, n \geq 1\}$ is a CFL
- $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 1\}$ is not a CFL
  (proof based on the pumping lemma) *QED*

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

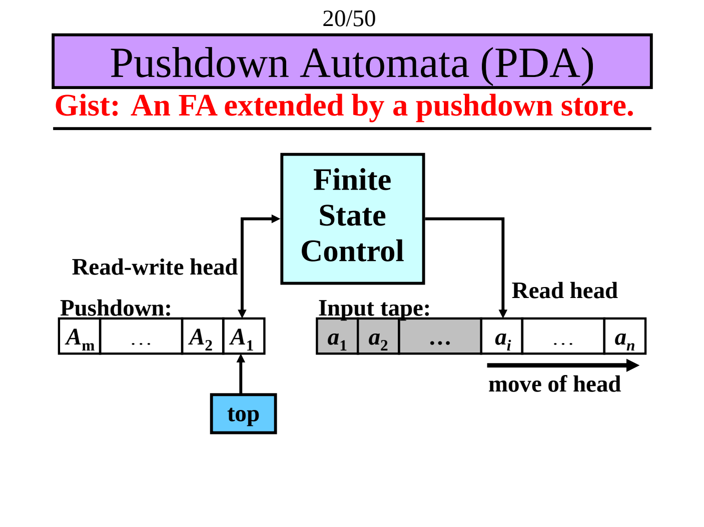
Complement: Not Closed

Theorem: The family of CFLs is not closed under **complement**.

Proof by contradiction:

- Assume that family of CFLs is closed under complement.
- $L_1 = \{a^m b^n c^n : m, n \geq 1\}$ is a **CFL**
- $L_2 = \{a^n b^n c^m : m, n \geq 1\}$ is a **CFL**
- $\overline{L_1}$, $\overline{L_2}$ are **CFLs**
- $\overline{L_1} \cup \overline{L_2}$ is a **CFL** (the family of CFLs is closed under union)
- $\overline{\overline{L_1} \cup \overline{L_2}}$ is a **CFL** (assumption)
- DeMorgan's law implies $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 1\}$ is a **CFL**
- $\{a^n b^n c^n : n \geq 1\}$ is not a **CFL** $\Rightarrow$ **Contradiction**

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

21/50

Pushdown Automata: Definition

Definition: *A pushdown automaton* (PDA) is a 7-tuple $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where

- Q is a *finite set of states*
- Σ is an *input alphabet*
- Γ is a *pushdown alphabet*
- R is a *finite set of rules* of the form: $Apa \to wq$ where $A \in \Gamma$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Gamma^*$
- $s \in Q$ is the *start state*
- $S \in \Gamma$ is the *start pushdown symbol*
- $F \subseteq Q$ is a set of *final states*

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Notes on PDA Rules

Mathematical note on rules:

- Strictly mathematically, R is a finite relation from $\Gamma \times Q \times (\Sigma \cup \{\varepsilon\})$ to $\Gamma^* \times Q$
- Instead of $(Apa, wq) \in R$, however, we write $Apa \rightarrow wq \in R$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

22/50

Notes on PDA Rules

Mathematical note on rules:

- Strictly mathematically, R is a finite relation from $\Gamma \times Q \times (\Sigma \cup \{\varepsilon\})$ to $\Gamma^* \times Q$
- Instead of $(\boldsymbol{Apa}, \boldsymbol{wq}) \in R$, however, we write $\boldsymbol{Apa} \rightarrow \boldsymbol{wq} \in R$

- **Interpretation of $\boldsymbol{Apa} \rightarrow \boldsymbol{wq}$**: if the current state is $\boldsymbol{p}$, current input symbol is $\boldsymbol{a}$, and the topmost symbol on the pushdown is $\boldsymbol{A}$, then M can read $\boldsymbol{a}$, replace $\boldsymbol{A}$ with $\boldsymbol{w}$ and change state $\boldsymbol{p}$ to $\boldsymbol{q}$.
- **Note**: if $\boldsymbol{a} = \varepsilon$, no symbol is read

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation

q represents $q \in Q$

$\rightarrow s$ represents the initial state $s \in Q$

f represents a final state $f \in F$

$p \xrightarrow{A/w,\, a} q$ denotes $Apa \rightarrow wq \in R$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

24/50

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

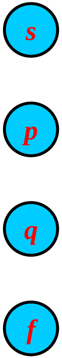
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;



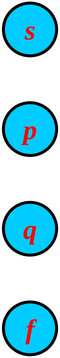
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;



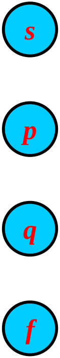
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;



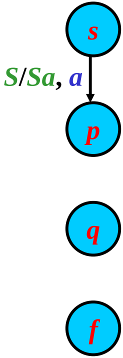
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$



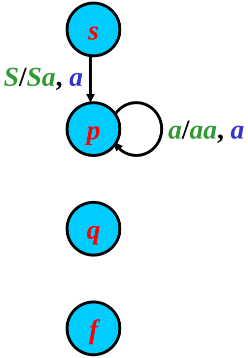
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$



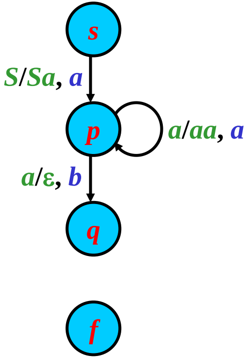
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$
  $apb \rightarrow q,$



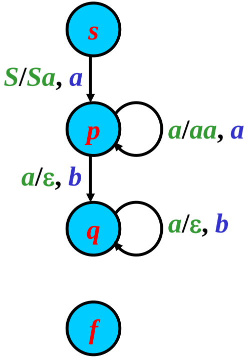
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$
  $apb \rightarrow q,$
  $aqb \rightarrow q,$



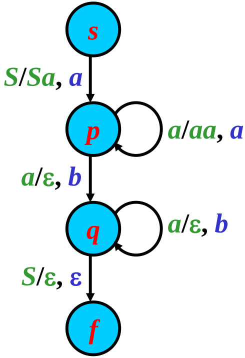
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$



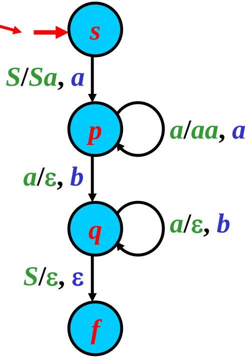
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

24/50

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$
  $apb \rightarrow q,$
  $aqb \rightarrow q,$
  $Sq \rightarrow f\}$



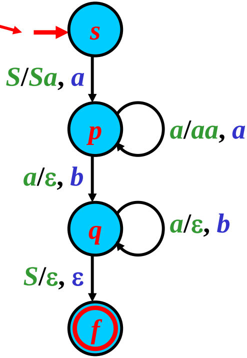
Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Graphical Representation: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Accepted Language: Three Types

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, S, F)$ be a PDA.

1) The *language that M accepts* ***by final state***, denoted by $\boldsymbol{L(M)_f}$, is defined as

$L(M)_f = \{w\colon w \in \Sigma^*, Ssw \vdash^* zf, z \in \Gamma^*, f \in F\}$

2) The *language that M accepts* ***by empty pushdown***, denoted by $\boldsymbol{L(M)_\varepsilon}$, is defined as

$L(M)_\varepsilon = \{w\colon w \in \Sigma^*, Ssw \vdash^* zf, z = \varepsilon, f \in Q\}$

3) The *language that M accepts* ***by final state and empty pushdown***, denoted by $\boldsymbol{L(M)_{f\varepsilon}}$, is defined as

$L(M)_{f\varepsilon} = \{w\colon w \in \Sigma^*, Ssw \vdash^* zf, z = \varepsilon, f \in F\}$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$**?**

S	s	a	a	b	b

$Ssaabb$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$**?**

$\boxed{S}\ \boxed{s}\ \boxed{a}\boxed{a}\boxed{b}\boxed{b}$

Rule: $Ssa \to Sap$

$\boxed{S}\boxed{a}\ \boxed{p}\ \boxed{a}\boxed{b}\boxed{b}$

$Ssaabb \vdash Sapabb$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$**?**

S | s | a a b b

Rule: $Ssa \to Sap$

S a | p | a b b

Rule: $apa \to aap$

S a a | p | b b

$Ssaabb \vdash Sapabb \vdash Saapbb$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$**?**

$\boxed{S}\ s\ \boxed{a\,a\,b\,b}$

Rule: $Ssa \to Sap$

$\boxed{S\,a}\ p\ \boxed{a\,b\,b}$

Rule: $apa \to aap$

$\boxed{S\,a\,a}\ p\ \boxed{b\,b}$

Rule: $apb \to q$

$\boxed{S\,a}\ q\ \boxed{b}$

$Ssaabb \vdash Sapabb \vdash Saapbb \vdash Saqb$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$
  $apb \rightarrow q,$
  $aqb \rightarrow q,$
  $Sq \rightarrow f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$**?**

$S\ s\ aabb$

Rule: $Ssa \rightarrow Sap$

$Sa\ p\ abb$

Rule: $apa \rightarrow aap$

$Saa\ p\ bb$

Rule: $apb \rightarrow q$

$Sa\ q\ b$

Rule: $aqb \rightarrow q$

$S\ q$

$Ssaabb \vdash Sapabb \vdash Saapbb \vdash Saqb \vdash Sq$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

30/50

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \rightarrow Sap,$
  $apa \rightarrow aap,$
  $apb \rightarrow q,$
  $aqb \rightarrow q,$
  $Sq \rightarrow f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$?

$\boxed{S}\ s\ \boxed{a\,a\,b\,b}$

Rule: $Ssa \rightarrow Sap$

$\boxed{S\,a}\ p\ \boxed{a\,b\,b}$

Rule: $apa \rightarrow aap$

$\boxed{S\,a\,a}\ p\ \boxed{b\,b}$

Rule: $apb \rightarrow q$

$\boxed{S\,a}\ q\ \boxed{b}$

Rule: $aqb \rightarrow q$

$\boxed{S}\ q$

Rule: $Sq \rightarrow f$

f

$Ssaabb \vdash Sapabb \vdash Saapbb \vdash Saqb \vdash Sq \vdash f$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

30/50

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$?

S | s | a a b b

Rule: $Ssa \to Sap$

S a | p | a b b

Rule: $apa \to aap$

S a a | p | b b

Rule: $apb \to q$

S a | q | b

Rule: $aqb \to q$

S | q |

Rule: $Sq \to f$

| f |

Empty pushdown

$Ssaabb \vdash Sapabb \vdash Saapbb \vdash Saqb \vdash Sq \vdash f$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

PDA: Example

$M = (Q, \Sigma, \Gamma, R, s, S, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, S\}$;
- $R = \{Ssa \to Sap,$
  $apa \to aap,$
  $apb \to q,$
  $aqb \to q,$
  $Sq \to f\}$
- $F = \{f\}$

Question: $aabb \in L(M)_{f\varepsilon}$?

$\boxed{S}\ \boxed{s}\ \boxed{a\,a\,b\,b}$

Rule: $Ssa \to Sap$

$\boxed{S\,a}\ \boxed{p}\ \boxed{a\,b\,b}$

Rule: $apa \to aap$

$\boxed{S\,a\,a}\ \boxed{p}\ \boxed{b\,b}$

Rule: $apb \to q$

$\boxed{S\,a}\ \boxed{q}\ \boxed{b}$

Rule: $aqb \to q$

$\boxed{S}\ \boxed{q}$

Rule: $Sq \to f$

Final state

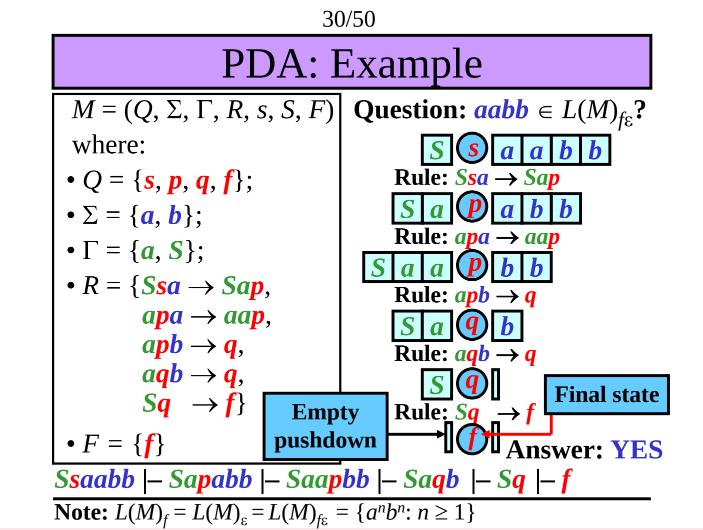
Empty pushdown

f

Answer: YES

$Ssaabb \vdash Sapabb \vdash Saapbb \vdash Saqb \vdash Sq \vdash f$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)



Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

Linear languages

Linear grammars have rules $X \to \beta$, with at most one non-terminal symbol in β. If we require β to be shorter or equal than 2, we have regular languages. In particular, we may define R-LIN ($X \to aY$) and L-LIN ($X \to Ya$).

Theorem: $L - LIN \equiv R - LIN \equiv REG$.
However, linear is strictly more general than regular, and $REG \subset LIN$. Indeed, the bisomatic language $a^n b^n$ is linear (see gramma $S \to ab, S \to aSb$) is an (infinite) linear language, which is not regular.

Btw, LIN equals languages generated by grammars having (possibly) both rigth linear and left right rules. See for example $S \to aA$ and $A \to Sb$.

Language of palindromes is not regular (CF, and linear), while the Dyck language is not even linear ($S \to xSyS$, and $S \to \lambda$ where x is the open parenthesis and y the closed one).

Decidable and Recursively Enumerable languages

Church thesis ('36): Turing machines are universal computable automata. A function/problem/language is computable if it is algorithmically generable/recognizable.

L is decidable, if there exists $M_L \in TM$ such that $\forall \alpha \in A^\star$ it answers whether $\alpha \in L$ or $\alpha \notin L$ in a finite number of steps. L is semi-decidable if the answer is given in a finite time only when it is positive.

By definition, $L \in RE$ iff there exists an algorithm to enumerate its strings. Then, $RE = L(TM)$ and $\mathcal{L}_0 \subseteq RE$.

REC is the class of decidable languages[3]. Halting (machine termination) problem is undecidable.

[3] Characteristic funcions, partial for RE and total for REC, are computable

Central theorem of representation: R$E = \mathcal{L}_0$.

For any RE language there exists a (general) grammar generating it.

RE is the class of **computable** and in general **semidecidable** languages – for them there exists a (Turing) machine which recognizes a string membership in a finite number of steps *only for positive answers*. If not decidable, for negative answers the machine takes an infinite number of steps to give the answer.

$$REC \subseteq RE$$

Chomsky hierarchy

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subseteq \mathcal{L}_0$$

We obtain this equivalent (strict?) hierarchy:

$$REG \subset CF \subset CS \subseteq RE$$

We are going to prove that:

- $CS \subseteq REC$ (context sensitive languages are decidable);
- $REC \subset RE$ (Post theorem characterizing REC)

Recursive languages

Context sensitive languages are decidable: $\forall \alpha \in L \in CS$, since the rules cannot shorten the word during computation, starting from S, there exists a finite number of steps within α may be generated. After that time I can give the positive/negative answer, in both cases.

Theorem (Post): $L \in REC$ iff $L, \overline{L} \in RE$.

Proof: $K = \{\alpha_i / \alpha_i \in L(G_i)\}$ if $\forall (i,j) \in \mathbb{N} \times \mathbb{N}$ G_i has generated α_i in j steps. Notice that $\overline{K} = \{\alpha_i \ / \ \alpha_i \notin L(G_i)\}$ is not RE (this is the language outside RE). Indeed, by contradiction, $\exists d$ such that $\overline{K} = L(G_d)$, then $\alpha_d \in \overline{K}$??

Notice that RE is not closed by complementation, and $CS \subset REC \subset RE \subset \mathcal{P}(A^\star)$, that is $\overline{K} \in \mathcal{P}(A^\star) \setminus RE$.

Recursive Language

Gist: Recursive Language accepts TM that always halt

Definition: Let L be a language. If $L = L(M)$, where M is DTM that always halts, then L is a *recursive language*.

Theorem: The family of recursive languages is closed under complement.

Proof: See page 693 in [Meduna: Automata and Languages]

Theorem: The family of recursively enumerable languages is **not** closed under complement.

Proof: See the $L_{\text{SelfAcceptance}}$

Courtesy of Alex Meduna (Brno University of Technology, Czech Republic) from his book *Formal Languages and Computation: Models and Their Applications* (2014)

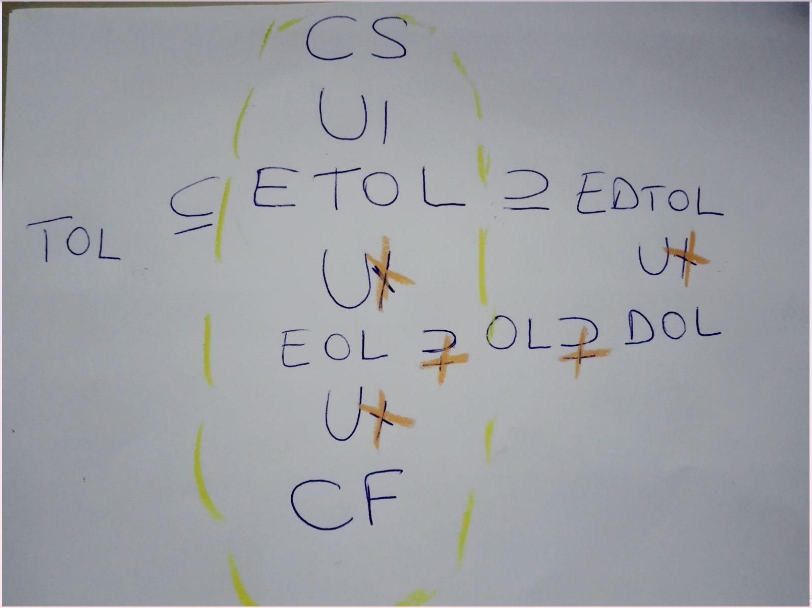
Parallel rewriting systems

There are infinitely many other classes between any couple of Chomsky classes. For example: $CF \subset EOL \subset ETOL \subseteq CS$, where OL are parallel rewriting systems, defined by an alphabet A, a starting word α and a morphism μ over A, rewriting in parallel each symbol a_i of the alphabet into one string $\beta_i \in A^\star$ (if deterministic: DOL), otherwise into a set of strings: $\beta_i \in \mathcal{P}(A^\star)$.

$$DOL \subset OL$$

EOL (Extended OL) is defined as OL with a set of terminals, while TOL (Table OL) introduces a non-deterministic choice among multiple morphisms. If this choice is deterministic, the system belongs to DTOL (or EDTOL, if extended).

Example of EDTOL generating trisomatic language, with two morphisms: $\{A \to aA, B \to bB, C \to cC, a \to a, b \to b, c \to c\}$ and $\{A \to a, B \to b, C \to c, a \to a, b \to b, c \to c\}$, start word ABC.


CS
TOL
ETOL
EDTOL
EOL
OL
DOL
CF

Comments on the Chomsky hierarchy

$$FIN \subset REG \subset LIN \subseteq CF \subset EOL \subset CS \subset REC \subset RE \subset \mathcal{P}(A^{\star})$$

Savitch theorem: $\forall L \in RE \exists L' \in CS$ such that $L = \{\alpha \mid \exists n \in \mathbb{N} : \alpha\#^n \in L', \# \notin A\}$

Boolean algebras (closed by $\cdot, +, C$): REG and CS (by Immerman theorem). RE is not Boolean, and CF is not closed by intersection (see trisomatic language): $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$

Another interesting property:
$\forall L \in \mathcal{L}_i, i = 0, 1, 2, 3$ and $\forall L' \in REG$, we have $L \cap L' \in \mathcal{L}_i$

Incomputable Functions

- The set of all rewriting systems is countable because each definition of a rewriting system is finite, so this set can be put into a bijection with $\mathbb{N}$.
- The set of all Turing Machines, which are defined as rewriting systems, is countable.
- The set of all functions is uncountable.
- Thus, there are more functions than Turing Machines.
- Some functions are incomputable.
- Even some simple total well-defined functions over $\mathbb{N}$ are incomputable.

Automata corresponding to Chomsky grammars

- Kleen theorem: REG = L(FSA)
- $CF = PDA = IFT_2$, $IFT_3 \subseteq CS$, $IFT_4 = RE$
- Kuroda theorem: CS = NLINSPACE
  (Savitch theorem: NPSPACE = PSPACE)
- RE=L(TM)

TM is universal, no matter if deterministic or not, the number of tapes, the number of final states.

Theorem (Shannon, '56): For any TM there exists one equivalent having two states, and (another) one equivalent having two symbols.