Course 2

Formal Languages
- basic notions-

Examples of languages

- natural (ex. English, Romanian)
- programming (ex. C,C++, Java, Python)
- formal

A formal language is a set
Ex.:
$L = \{a^n b^n | \ n>0\}$ $L = \{ab, aabb, aaabbb, \ldots\}$
$L' = \{01^n | n>=0\}$ $L' = \{0, 01, 011, \ldots\}$

Example

a boy has a dog

S→PV
P→ *a* N
N→ *boy* or N→ *dog*
(N→*boy*|*dog*)
V → QC
Q → *has*
C → BN
B → *a*

- A→ α = **rule**
- S,P,V,N,Q,C,B = **nonterminal symbols**
- *a, boy,dog,has* = **terminal symbols**

Remarks

1. Sentence = word, sequence (contains only terminal symbols) ; denoted w.
2. S⇒PV⇒a NV⇒a NQC⇒a N has C - sentential form

   In general : $w=a_1a_2...a_n$

3. The rule guarantees syntactical correctness, but <u>not</u> the semantical correctness (*A dog has a boy*)

Grammar

- ***Definition***: A (formal) **grammar** is a 4-tuple: $G=(N,\Sigma,P,S)$ with the following meanings:
  - N – set of nonterminal symbols and $|N| < \infty$
  - Σ - set of terminal symbols (alphabet) and $|\Sigma|<\infty$
  - P – finite set of productions (rules), with the propriety:

    $P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \mathbf{x} (N \cup \Sigma)^*$
  - $S \in N$ – start symbol /axiom

Remarks :

1. $(\alpha,\beta) \in P$ is a production denoted $\alpha \rightarrow \beta$
2. $N \cap \Sigma = \emptyset$

A^* = transitive and reflexive closure = $\{a,aa,aaa,\ldots\}$ $\{a^0\}$

$A = \{a\}$

$A+ = \{a,aa,aaa,\ldots\}$

$X^0 = \varepsilon$

Binary relations defined on $(N \cup \Sigma)^*$

- ***Direct derivation***
  $\alpha \Rightarrow \beta$, $\alpha,\beta \in (N \cup \Sigma)^*$ ***if*** $\alpha = x1xy1$, $\beta = x1yy1$ ***and*** $x \rightarrow y \in P$
  (x is transformed in y)

- ***k derivation***

$\alpha \overset{k}{\Rightarrow} \beta$,$\alpha,\beta \in (N \cup \Sigma)^*$

sequence of k direct derivations $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_{k-1} \Rightarrow \beta$, $\alpha,\alpha_1,\alpha_2,\ldots\alpha_{k-1},\beta \in (N \cup \Sigma)^*$

- ***+ derivation***

$\alpha \overset{+}{\Rightarrow} \beta$ ***if*** $\exists$ k>0 ***such that*** $\alpha \overset{k}{\Rightarrow} \beta$ (there exists at least one direct derivation)

- **** derivation***

$\alpha \overset{*}{\Rightarrow} \beta$ ***if*** $\exists$ k≥0 ***such that*** $\alpha \overset{k}{\Rightarrow} \beta$ namely, $\alpha \overset{*}{\Rightarrow} \beta \Leftrightarrow \alpha \overset{+}{\Rightarrow} \beta$ ***OR*** $\alpha \overset{0}{\Rightarrow} \beta$ $(\alpha = \beta)$

Definition: ***Language generated*** by a grammar G=(N,Σ,P,S) is:

$$L(G)=\{w\in\Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

L1 = {a,b,aa}
L2 = {c,d,cd}
L1L2 = {ac,ad,acd,bc,bd,bcd,aac,aad,aacd}

Remarks:

1. $S\overset{*}{\Rightarrow} \alpha, \alpha\in(N\cup\Sigma)^*$ = sentential form
   $S\overset{*}{\Rightarrow} w, w\in\Sigma^*$ = word / sequence
2. Operations defined for languages (sets) :
   $L1\cup L2$, $L1\cap L2$, L1-L2 , $\overline{L}$ (complement) , $L^+=\bigcup_{k>0} L^k$, $L^*=\bigcup_{k\geq 0} L^k$
   Concatenation: $L=L_1L_2 = \{w_1w_2 \mid w_1\in L_1 , w_2\in L_2\}$
3. $|w|=0$ (empty word - denoted ε)

Definition: Two grammar G_1 and G_2 are equivalent if they generate the same language

$$L(G_1)=L(G_2)$$

Chomsky hierarchy(based on form $\alpha \to \beta \in P$)

- type 0 : no restriction
- type 1 : context dependent grammar ($x_1Ay_1 \to x_1\gamma y_1$)
- type 2 : context free grammar ($A \to \alpha \in P$,where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$)
- type 3 : regular grammar ($A \to aB | a \in P$)

Remark :

type 3 ⊆ type 2 ⊆ type 1 ⊆ type 0

Regular grammars

- $G = (N, \Sigma, P, S)$ right linear grammar if

  $$\forall p \in P: A \rightarrow aB \text{ or } A \rightarrow b, \text{ where } A,B \in N \text{ and } a,b \in \Sigma$$

- $G = (N, \Sigma, P, S)$ regular grammar if
  - G is right linear grammar

  and
  - $A \rightarrow \varepsilon \notin P$, with the exception that $S \rightarrow \varepsilon \in P$, in which case S does not appear in the rhs (right hand side) of any other production

S->aA|ε; A-> a reg
S->aS|aA; A->bS|b reg
S->aA; A->aA|ε NOT reg
S->aA|ε; A->aS NOT reg

- $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$ - right linear language

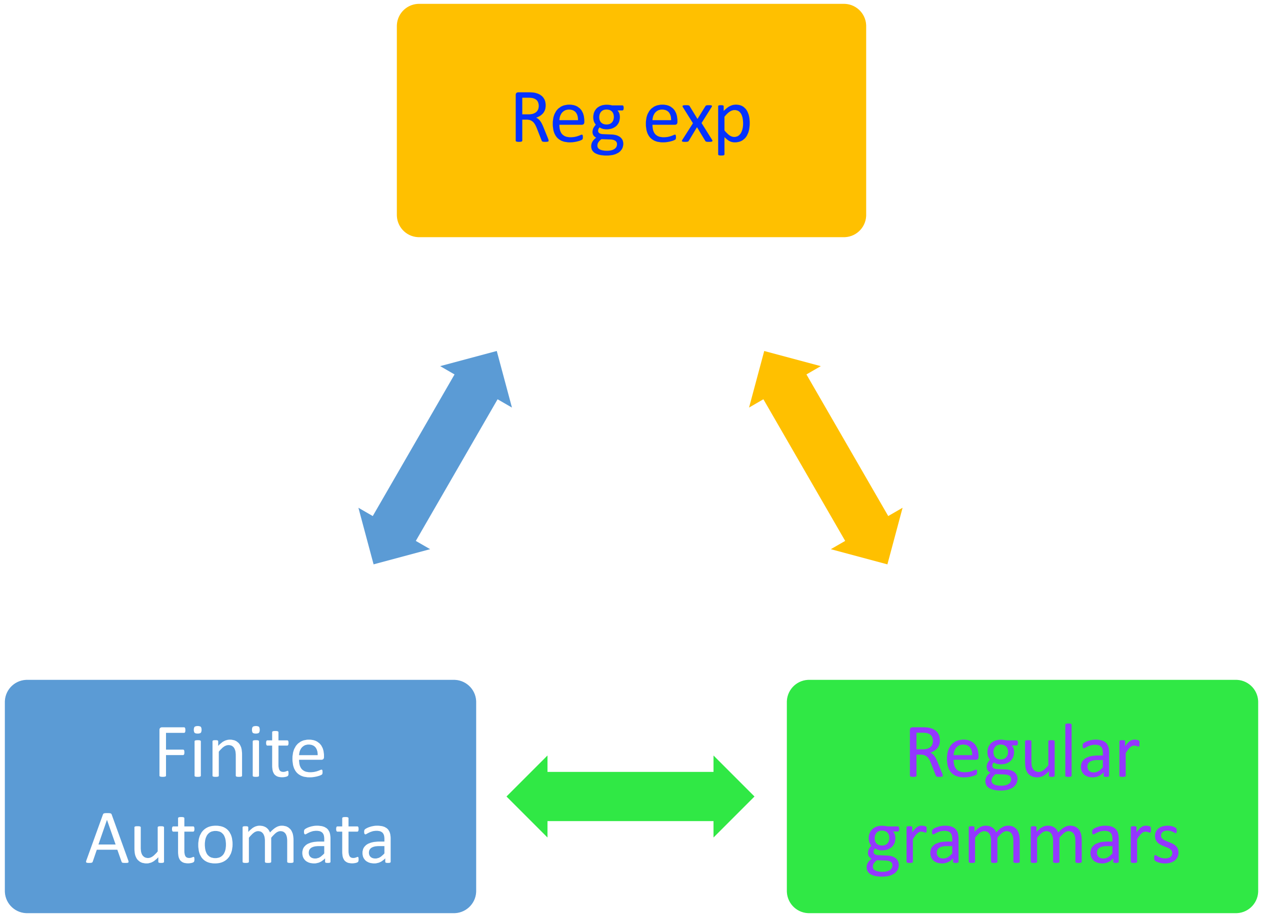
Notations

- A,B,C,... – nonterminal symbols
- $S \in N$ – start symbol
- $a,b,c,... \in \Sigma$ – terminal symbol
- $\alpha,\beta,\gamma \in (N \cup \Sigma)^*$ - sentential forms
- ε – empty word
- $x,y,z,w \in \Sigma^*$ - words
- $X,Y,U,... \in (N \cup \Sigma)$ – grammar symbols (nonterminal or terminal)

Regular languages

Why?

1. Search engine – success of Google
2. Unix commands
3. Programming languages – new feature


Reg exp
Finite Automata
Regular grammars

Regular grammars

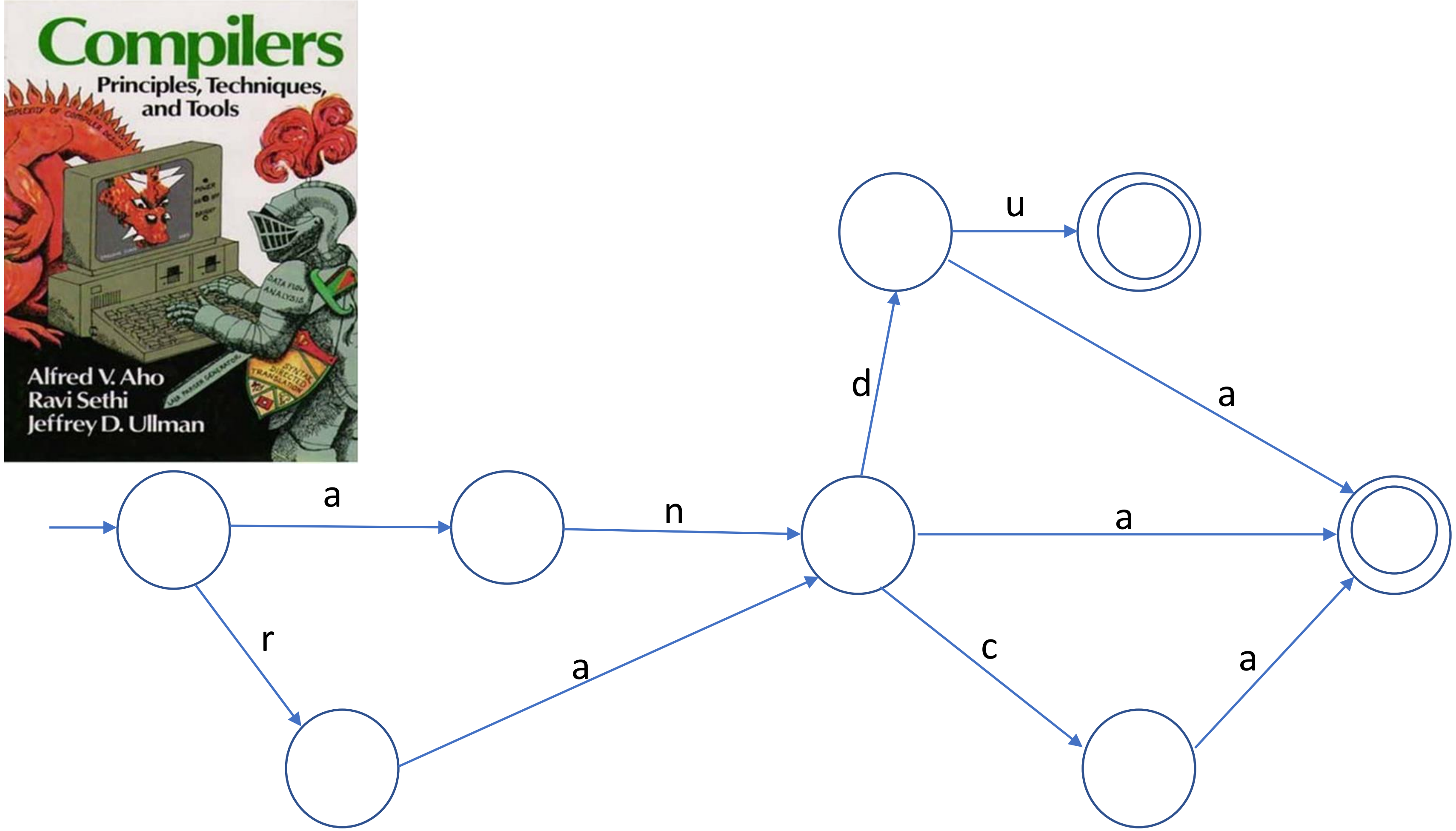
- $G = (N, \Sigma, P, S)$ right linear grammar if

  $$\forall p \in P: A \to aB \text{ or } A \to b, \text{ where } A,B \in N \text{ and } a,b \in \Sigma$$

- $G = (N, \Sigma, P, S)$ regular grammar if
  - G is right linear grammar

  and
  - $A \to \varepsilon \notin P$, with the exception that $S \to \varepsilon \in P$, in which case S does not appear in the rhs (right hand side) of any other production

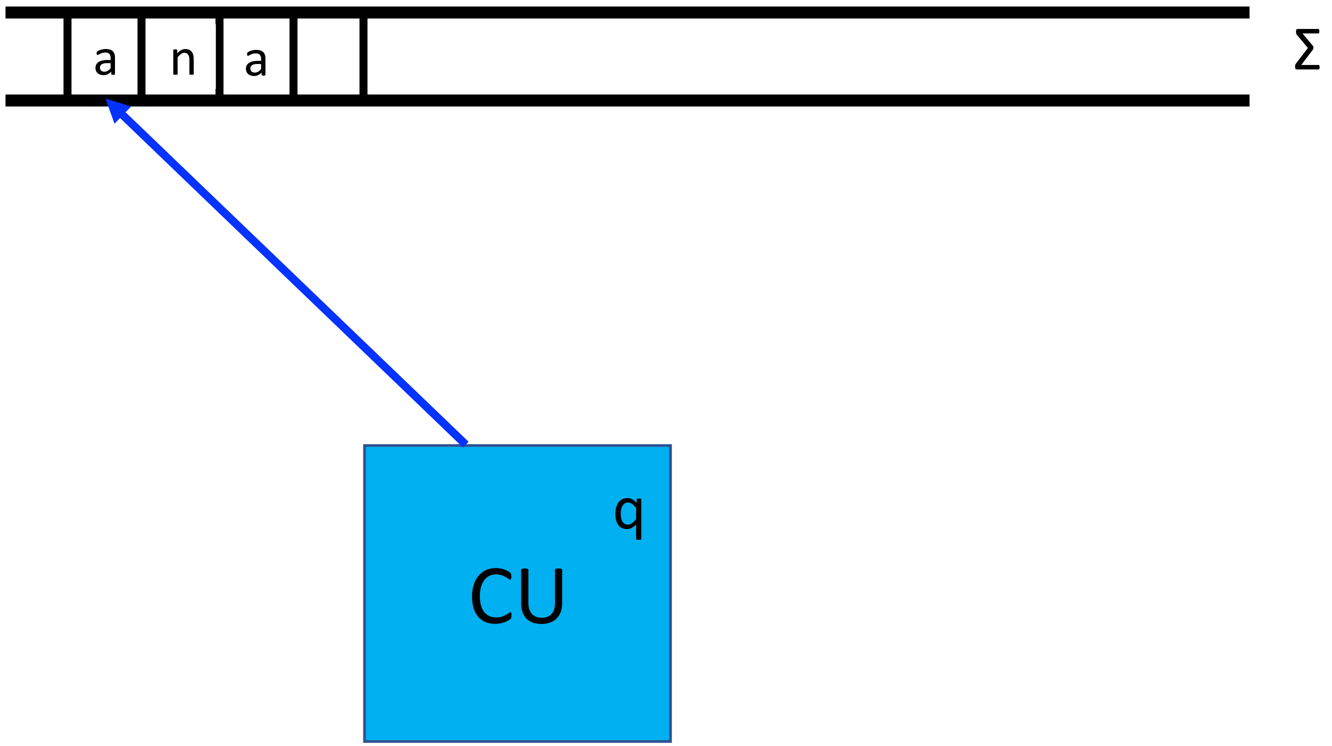
- $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$ - right linear language



Problem: The door to the tower is closed by the Red Dragon, using a complicated machinery. Prince Charming has managed to steal the plans and is asking for your help. Can you help him determining all the person names that can unlock the door

Finite Automata

- Intuitive model



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

$$M = (Q,\Sigma,\delta,q0,F)$$

where:

- Q - finite set of states ($|Q|<\infty$)
- Σ - finite alphabet ($|\Sigma|<\infty$)
- δ – transition function : $\delta:Q\times\Sigma\rightarrow P(Q)$
- q_0 – initial state $q_0 \in Q$
- $F\subseteq Q$ – set of final states

Remarks

1. $Q \cap \Sigma = \emptyset$
2. $\delta : Q \times \Sigma \rightarrow P(Q)$, $\varepsilon \in \Sigma^0$ - relation $\delta(q,\varepsilon)=p$ **NOT** allowed
3. If $|\delta(q,a)| \leq 1$ => deterministic finite automaton (DFA)
4. If $|\delta(q,a)| > 1$ (more than a state obtained as result) => nondeterministic finite automaton (NFA)

Property: For any NFA *M* there exists a DFA *M'* equivalent to *M*

Configuration $C=(q,x)$

where:

- q state
- x unread sequence from input: $x \in \Sigma^*$

Initial configuration : (q_0,w) , w - whole sequence

Final configuration: (q_f ,ε) , $q_f \in F$, ε –empty sequence

(corresponds to accept)

Relations between configurations

- $\vdash$ **move / transition** (simple, one step)
  $(q,ax) \vdash (p,x)$, $p \in \delta(q,a)$

- $\overset{k}{\vdash}$ **k move** = a sequence of k simple transitions) $C_0 \vdash C_1 \vdash \ldots \vdash C_k$
- $\overset{+}{\vdash}$ **+ move**
  $C \overset{+}{\vdash} C'$: $\exists$ $k>0$ such that $C \overset{k}{\vdash} C'$
- $\overset{*}{\vdash}$ *** move (star move)**
  $C \overset{*}{\vdash} C'$: $\exists$ $k\geq 0$ such that $C \overset{k}{\vdash} C'$

Definition : ***Language*** accepted by FA $M = (Q,\Sigma,\delta,q0,F)$ is:

$$L(M)=\{ w \in \Sigma^* \mid (q_0,w) \vdash^* (q_f ,\varepsilon) , q_f \in F \}$$

Remarks

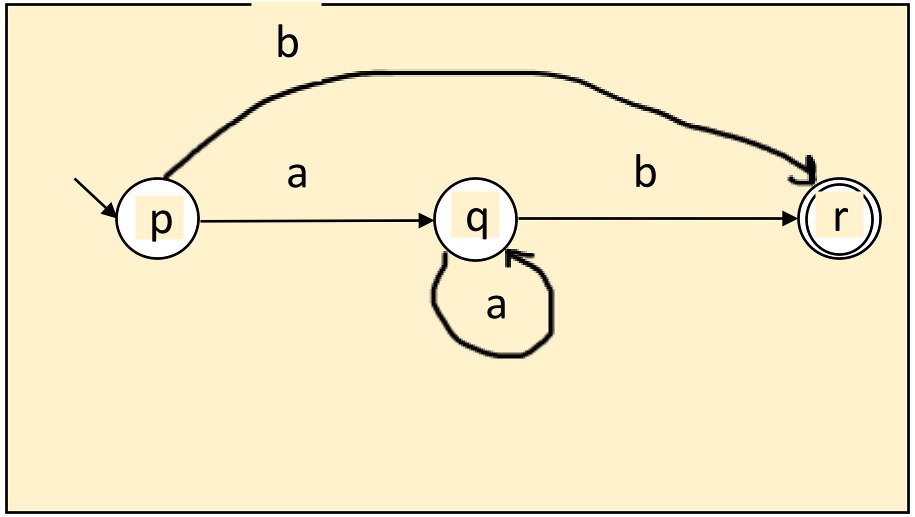
1. 2 finite automata M_1 and M_2 are equivalent if and only if they accept the same language

$$L(M_1)=L(M_2)$$

1. $\varepsilon \in L(M) \Leftrightarrow q_0 \in F$ (initial state is final state)

Representing FA

1. List of all elements
2. Table
3. Graphical representation



$M=(Q,\Sigma,\delta,p,F)$
$Q = \{p,q,r\}$
$\Sigma = \{a,b\}$
$\delta(p,a) = q$
$\delta(q,a)=q$
$\delta(q,b)=r$
$\delta(p,b)=r$
$F = \{r\}$

$M=(Q,\Sigma,\delta,p,F)$
$F = \{r\}$

	a	b
p	q	r
q	q	r
r	-	-

(p,aab)|-(q,ab)|-(q,b)|-(r,ε) => aab accepted
(p,aba)|-(q,ba)|-(r,a) => aba not accepted

Remember

- **Grammar**

$\mathbf{G=(N,\Sigma,P,S)}$

$\mathbf{L(G)=\{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}}$

- Finite automaton

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

$L(M)=\{ w \in \Sigma^* \mid (q_0,w) \vdash (q_f ,\varepsilon) , q_f \in F \}$