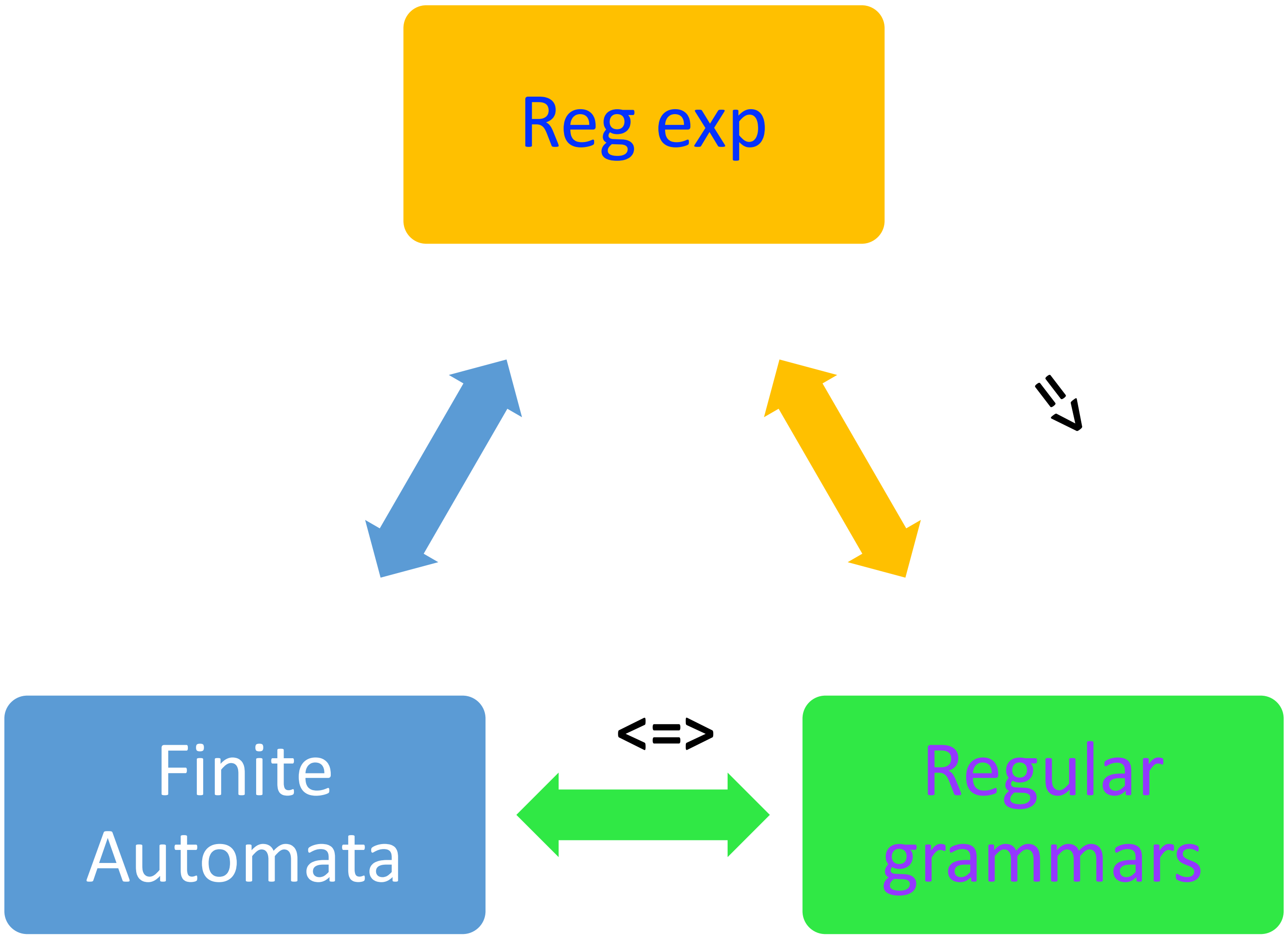
Course 4

Formal Languages

- Regular Languages -


Reg exp
=>
<=>
Finite Automata
Regular grammars

Theorem: *A language is a regular set if and only if is a right linear language*

Proof:

=> Apply lemma 1 and lemma 2

<= construct a system of regular exp equations where:

- Indeterminants – nonterminals
- Coefficients – terminals
- Equation for A: all the possible rewritings of A

Example: G=({S,A,B},{0,1}, P, S)

P: S → 0A | 1B | **ε**

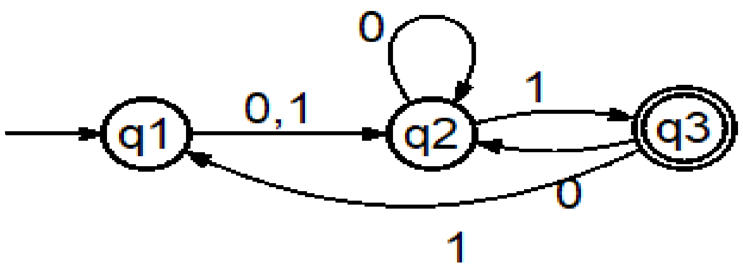
A → 0B | 1A

B → 0B | 1

$$\begin{cases} S = 0A + 1B + \boldsymbol{\varepsilon} \\ A = 0B + 1A \\ B = 0B + 1 \end{cases}$$

Regular exp = solution corresponding to S

Theorem: *A language is a regular set if and only if is accepted by a FA*



Proof:

=> Apply lemma 1 and lemma 2 (to follow, similar to RG)

<= construct a system of regular exp equations where:

- Indeterminants – states
- Coefficients – terminals
- Equation for A: all the possibilities that put the FA in state A
- Equation of the form: X=Xa+b => solution X=ba*

$$\begin{cases} q_1 = q_3 0 + \boldsymbol{\varepsilon} \\ q_2 = q_1 0 + q_1 1 + q_2 0 + q_3 0 \\ q_3 = q_2 1 \end{cases}$$

Regular exp = union of solutions corresponding to final states

Lemma 1': $\boldsymbol{\Phi}$,{$\boldsymbol{\varepsilon}$}, {a},$\forall a \in \boldsymbol{\Sigma}$ are accepted by FA

Reg exp	FA
$\boldsymbol{\Phi}$	$M = (Q, \boldsymbol{\Sigma}, \delta, q_{0,} \boldsymbol{\Phi})$
$\boldsymbol{\varepsilon}$	$M = (Q, \boldsymbol{\Sigma}, \boldsymbol{\Phi}, q_{0,} \{q_0\})$
a,$\forall a \in \boldsymbol{\Sigma}$	$M = (\{q_0,q_1\}, \boldsymbol{\Sigma}, \{\delta(q_0,a) = q_1\}, q_{0,} \{q_1\})$

Lemma 2': If L_1 and L_2 are accepted by a FA then: $L_1 \cup L_2$, L_1L_2 and L_1^* are accepted by FA

Proof:

$M_1 = (Q_1, \boldsymbol{\Sigma}_1, \delta_1, q_{01}, F_1)$ such that $L_1 = L(M_1)$

$M_2 = (Q_2, \boldsymbol{\Sigma}_2, \delta_2, q_{02}, F_2)$ such that $L_2 = L(M_2)$

$M_3 = (Q_3, \boldsymbol{\Sigma}_{1U}, \delta_3, q_{03}, F_3)$

$Q_3 = Q_1 \cup Q_2 \cup \{q_{03}\}$; $\Sigma_3 = \Sigma_1 \cup \Sigma_2$

$F_3 = F_1 \cup F_2 \cup \{q_{03} \mid \text{if } q_{01} \in F_1 \text{ or } q_{02} \in F_2\}$

$\delta_3 = \delta_1 \cup \delta_2 \cup \{\delta_3(q_{03},a) = p \mid \exists \delta_1(q_{01},a) = p\} \cup$

$\{\delta_3(q_{03},a) = p \mid \exists \delta_2(q_{02},a) = p\}$

$L(M_3) = L(M_1) \cup L(M_2)$

PROOF!!! Homework

$M_4 = (Q_4, \mathbf{\Sigma}_4, \delta_4, q_{04}, F_4)$

$Q_4 = Q_1 \cup Q_2$; $q_{04} = q_{01}$;

$F_3 = F_2 \cup \{q \in F_1 \mid \text{if } q_{02} \in F_2\}$

$$\delta_3(q,a) = \begin{cases} \delta_1(q,a), \text{ if } q \in Q_1\text{-}F_1 \\ \delta_1(q,a) \cup \delta_2(q_{02},a) \text{ if } q \in F_1 \\ \delta_2(q,a), \text{ if } q \in Q_2 \end{cases}$$

$L(M_3) = L(M_1)L(M_2)$

PROOF!!! Homework

$M_5 = (Q_5, \boldsymbol{\Sigma}_1, \delta_5, q_{05}, F_5)$ *//IDEA: concatenate with itself*

$Q_5 = Q_1$; $q_{05} = q_{01}$

$F_5 = F_1 \cup \{q_{01}\}$

$$\delta_5(q,a) = \begin{cases} \delta_1(q,a), \text{ if } q \in Q_1 - F_1 \\ \delta_1(q,a) \cup \delta_1(q_{01},a) \text{ if } q \in F_1 \end{cases}$$

$L(M_3) = L(M_1)^*$

PROOF!!! Homework

Pumping Lemma

- Not all languages are regular
- How to decide if a language is regular or not?

- Idea: pump symbols

Example: $L = \{0^n1^n \mid n >= 0\}$

Theorem: (Pumping lemma, Bar-Hillel)

Let **L** be a regular language. $\exists \boldsymbol{p} \in \boldsymbol{N}$, such that if $\boldsymbol{w} \in \boldsymbol{L}$ with $\boldsymbol{|w|>p}$, then

$\boldsymbol{w = xyz}$, where $\boldsymbol{0<|y|<=p}$

and

$\boldsymbol{xy^iz \in L,\ \forall i \geq 0}$

Proof

L regular => $\exists$ M = (Q,$\boldsymbol{\Sigma}$,$\boldsymbol{\delta}$, q_0, F) such that L= L(M)

Let |Q| = p

If w $\in$ L(M): (q_0,w) $\vdash^*$ (q_f , $\boldsymbol{\varepsilon}$) , $q_f \in F$ process at least p+1 symbols

and

|w|>p p states

$\Rightarrow \exists$ q_1 that appear in at least 2 configurations

(q_0,xyz) $\vdash^*$ (q1,yz) $\vdash^+$ (q1,z) $\vdash^*$ (q_f , $\boldsymbol{\varepsilon}$) , $q_f \in F$ => 0<=|y|<=p

Proof (cont)

$(q_0,xy^iz) \overset{*}{\vdash} (q_1,y^iz)$

$\overset{*}{\vdash} (q_1,y^{i-1}z)$

$\overset{*}{\vdash} \dots$

$\overset{*}{\vdash} (q_1,yz)$

$\overset{*}{\vdash} (q_1, z)$

$\overset{*}{\vdash} (q_f , \varepsilon) , q_f \in F$

So, if $w=xyz \in L$ then $xy^iz \in L$, for all $i>0$

If $i=0$: $(q_0,xz) \overset{*}{\vdash} (q_1, z) \overset{*}{\vdash} (q_f , \varepsilon) , q_f \in F$

Example: $L = \{0^n1^n \mid n>= 0\}$

Suppose L is regular => $w= xyz = 0^n1^n$

Consider all possible decomposition =>

Case 1. $y = 0^k$

$$xyz = 0^{n-k}0^k1^n;\ xy^iz = 0^{n-k}0^{ik}1^n \notin L$$

Case 2. $y = 1^k$

$$xyz = 0^n1^k1^{n-k};\ xy^iz = 0^n1^{ik}1^{n-k} \notin L$$

Case 3. $y = 0^k\,1^l$

$$xyz = 0^{n-k}0^k1^l1^{n-l};\ xy^iz = 0^{n-k}(0^k1^l)^i1^{n-l} \notin L$$

Case 4. $y = 0^k\,1^K$

$$xyz = 0^{n-k}0^k1^k1^{n-k};\ xy^iz = 0^{n-k}0^k1^k0^k1^k\ldots1^{n-l} \notin L$$

=> L is not regular

Context free grammars (cfg)

Context free grammar (cfg)

- Procdutions of the form: A $\rightarrow\alpha$, A∈N, α∈(N∪$\boldsymbol{\Sigma}$)*

- More powerful

- Can model programming language:

  G = (N, $\boldsymbol{\Sigma}$,P,S) s.t. L(G) = programming language

Syntax tree

Definition: A syntax tree corresponding to a cfg $G = (N, \Sigma, P, S)$ is a tree obtained in the following way:

1. Root is the starting symbol S
2. Nodes $\in N \cup \Sigma$:
   1. Internal nodes $\in N$
   2. Leaves $\in \Sigma$
3. For a node A the descendants in order from left to right are $X_1, X_2, \ldots, X_n$ only if $A \rightarrow X_1X_2\ldots X_n \in P$

Remarks:

a) Parse tree = syntax tree – result of parsing (syntatic analysis)

b) Derivation tree – condition 2.2 not satisfied

c) Abstract syntax tree (AST) ≠ syntax tree (semantic analysis)

Syntax tree (cont)

Property: In a cfg $G = (N, \boldsymbol{\Sigma}, P, S)$, $w \in L(G)$ <u>if and only if</u> there exists a syntax tree with frontier w.

Proof: HomeWork

Example: S-> aSbS | c; w = aacbcbc

Leftmost derivations	**Rightmost derivations**
S => aSbS => aaSbSbS => aacbSbS => aacbcbS => aacbcbc	S => aSbS => aSbc => aaSbSbc => aaSbcbc => aacbcbc

Definition: A cfg G = (N, $\boldsymbol{\Sigma}$,P,S) is ambigous if for a w ∊ L(G) there exists 2 distinct syntax tree with frontier w.

Example: