Chapter 5: Regular Expressions

U.D. Computación

DSIC - UPV

Content

- Definitions
- Properties
- Constructions with regular expressions
- Synthesis of FA
- Analysis de FA

Definitions

- Inductively, a regular expression over Σ is defined:
  - $\emptyset$ denotes the empty language.
  - λ denotes the language $\{\lambda\}$
  - $\forall a \in \Sigma$, a denotes the language $\{a\}$
  - If r and s are regular expressions denoting L_r and L_s:
    - (r) denotes L_r
    - $r + s$ denotes $L_r \cup L_s$
    - rs denotes $L_r L_s$
    - $(r)^*$ denotes L_r^*
  - All the regular expressions are built using the previous steps finitely many times.

Properties

- Let α, β and γ be regular expressions
  1. $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
  2. $\alpha(\beta\gamma) = (\alpha\beta)\gamma$
  3. $\alpha + \beta = \beta + \alpha$
  4. $\alpha(\beta + \gamma) = (\alpha\beta) + (\alpha\gamma)$
  5. $(\alpha + \beta)\gamma = (\alpha\gamma) + (\beta\gamma)$
  6. $\alpha\lambda = \lambda\alpha = \alpha$
  7. $\alpha + \emptyset = \emptyset + \alpha = \alpha$
  8. $\alpha\emptyset = \emptyset\alpha = \emptyset$
  9. $\lambda^* = \lambda$
  10. $\emptyset^* = \lambda$
  11. $\alpha^* = \lambda + \alpha\alpha^*$
  12. $(\alpha^* + \beta^*)^* = (\alpha^*\beta^*)^* = (\alpha + \beta)^*$
  13. $(\alpha\beta)^*\alpha = \alpha(\beta\alpha)^*$
  14. $(\alpha^*\beta)^*\alpha^* = (\alpha + \beta)^*$
  15. $(\alpha^*\beta)^* = (\alpha + \beta)^*\beta + \lambda$

Constructions

- Homomorphism
- Reverse

Constructions

- Homomorphism
  Given a regular expression α and a homomorphism $h : \Sigma_\alpha \to \Delta$, to obtain a regular expression for $h(L(\alpha))$, one has to change every symbol a of α by its image $h(a)$

  For example, let us consider $\alpha = a(bb^* + (aa)^*)^* b$ and the homomorphism: $h(a) = 0$ y $h(b) = 11$. The regular expression for $h(L(\alpha))$ is:

$$0(11(11)^* + (00)^*)^* 11$$

Constructions

- Reverse
  Given a regular expression α, to obtain a regular expression for α^r such that $L(\alpha^r) = (L(\alpha))^r$, we recursively apply the following rules:
  - If $\alpha = \emptyset$, $\alpha = \lambda$ or $\alpha = a \in \Sigma$, then $\alpha^r = \alpha$
  - If $\alpha = \beta + \gamma$, then $\alpha^r = \beta^r + \gamma^r$
  - If $\alpha = \beta\gamma$, then $\alpha^r = \gamma^r\beta^r$
  - If $\alpha = \beta^*$, then $\alpha^r = (\beta^r)^*$

  For example, let us consider $\alpha = a(b(a+b)^* + (bba)^*)^*b$.
  The regular expression for $(L(\alpha))^r$ is:

$$\alpha^r = b((a+b)^*b + (abb)^*)^*a$$

Syntheses of FAs from ERs

- Position Automaton
- Follow Automaton
- Brzozowski's Algorithm

Position Automata

- Local Automata. Local Languages.
- Linearized regular expressions.
- DFA for a linearized regular expression.
- Position Automaton.

Local Automata. Local Languages.

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is *local* if and only if for every $a \in \Sigma$ the set $\{\delta(q, a) \ : \ q \in Q\}$ has at most one element.
- If there is no arrow of the form $q \rightarrow q_0$, the automaton is called *standard local*
- A language is local if and only if it is recognized by a standard local automaton.

Linearized Regular Expression

- Let α be una regular expression and let n be the number of symbols in α without parenthesis and operation symbols. The linearized expression of α (denoted by $\overline{\alpha}$) is obtained placing a subindex $j \in \{1, \ldots, n\}$ to every symbol of α that indicates its position.
  Example: Consider

$$\alpha = (a + b)(a^* + ba^* + b^*)^*$$

the linearized expression is

$$\overline{\alpha} = (a_1 + b_2)(a_3^* + b_4 a_5^* + b_6^*)^*$$

Linearized Regular Expression

- If Σ_α and $\Sigma_{\overline{\alpha}}$ are the alphabets of α and $\overline{\alpha}$ respectively, and $h : \Sigma_{\overline{\alpha}}^* \to \Sigma_\alpha^*$ is a homomorphism that erases the subindexes, then:

$$h(L(\overline{\alpha})) = L(\alpha)$$

- An automaton for $L(\alpha)$ can be obtained:
  - Building an automaton for $L(\overline{\alpha})$ and
  - Removing the subindexes of this automaton (*position automaton*)

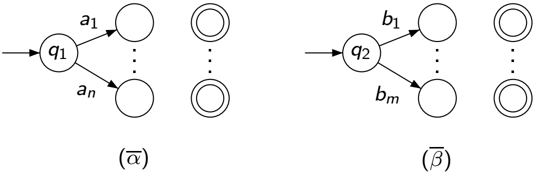
DFA for a Linearized Regular Expression
Base cases

- Every linearized regular expression denotes a local language (recognized by a standard local automaton) It can be seen by induction over the structure of the regular expressions .
- Base Cases:

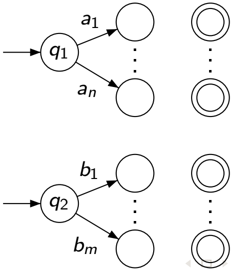
$\emptyset$ λ a

DFA for a Linearized Regular Expression
Building rules

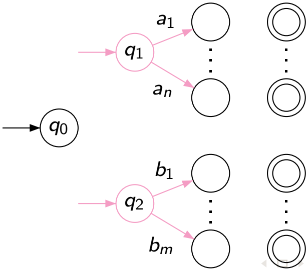
- Compound Expressions:
  Let $\overline{\alpha}$ and $\overline{\beta}$ be linearized regular expressions, and let $A(\overline{\alpha}) = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and $A(\overline{\beta}) = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$, with $\Sigma_1 \cap \Sigma_2 = \emptyset$, local automata accepting $L(\overline{\alpha})$ and $L(\overline{\beta})$ respectively:



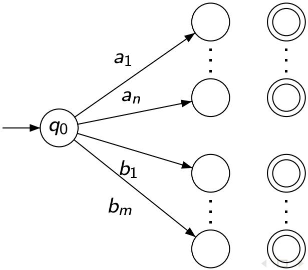
- Union $(\overline{\alpha}+\overline{\beta})$:
  - $Q=(Q_1-\{q_1\})\cup(Q_2-\{q_2\})\cup\{q_0\},\ q_0\notin Q_1\cup Q_2.$
  - $\delta=\{(q,a,q')\in\delta_1\cup\delta_2 : q\notin\{q_1,q_2\}\}\cup\{(q_0,a,q) : (q_1,a,q)\in\delta_1\vee(q_2,a,q)\in\delta_2\},$
  - $F=\begin{cases}F_1\cup F_2 & \text{si } q_1\notin F_1\wedge q_2\notin F_2\\ (F_1-\{q_1\})\cup(F_2-\{q_2\})\cup\{q_0\} & \text{en otro caso.}\end{cases}$



- Union ($\overline{\alpha} + \overline{\beta}$):
  - $Q = (Q_1 - \{q_1\}) \cup (Q_2 - \{q_2\}) \cup \{q_0\},\ q_0 \notin Q_1 \cup Q_2.$
  - $\delta = \{(q, a, q') \in \delta_1 \cup \delta_2 : q \notin \{q_1, q_2\}\} \cup \{(q_0, a, q) : (q_1, a, q) \in \delta_1 \vee (q_2, a, q) \in \delta_2\},$
  - $F = \begin{cases} F_1 \cup F_2 & \text{si } q_1 \notin F_1 \wedge q_2 \notin F_2 \\ (F_1 - \{q_1\}) \cup (F_2 - \{q_2\}) \cup \{q_0\} & \text{en otro caso.} \end{cases}$

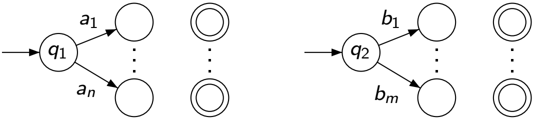


- Union ($\overline{\alpha} + \overline{\beta}$):
  - $Q = (Q_1 - \{q_1\}) \cup (Q_2 - \{q_2\}) \cup \{q_0\}$, $q_0 \notin Q_1 \cup Q_2$.
  - $\delta = \{(q, a, q') \in \delta_1 \cup \delta_2 : q \notin \{q_1, q_2\}\} \cup \{(q_0, a, q) : (q_1, a, q) \in \delta_1 \vee (q_2, a, q) \in \delta_2\}$,
  - $F = \begin{cases} F_1 \cup F_2 & \text{si } q_1 \notin F_1 \wedge q_2 \notin F_2 \\ (F_1 - \{q_1\}) \cup (F_2 - \{q_2\}) \cup \{q_0\} & \text{en otro caso.} \end{cases}$



DFA for a Linearized Regular Expression
Product

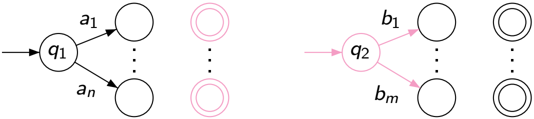
- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \notin F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q, a, q') \in \delta_2 : q \neq q_2\} \cup \{(q, a, q') : q \in F_1 \wedge (q_2, a, q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_2$



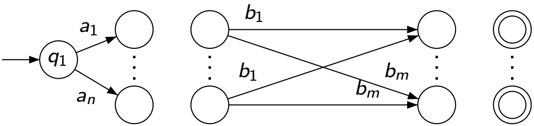
DFA for a Linearized Regular Expression

Product

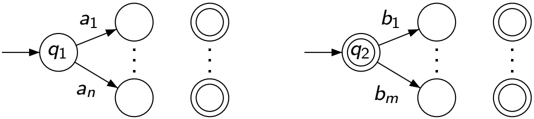
- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \notin F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q, a, q') \in \delta_2 : q \neq q_2\} \cup \{(q, a, q') : q \in F_1 \wedge (q_2, a, q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_2$



- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \notin F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q, a, q') \in \delta_2 : q \neq q_2\} \cup \{(q, a, q') : q \in F_1 \wedge (q_2, a, q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_2$



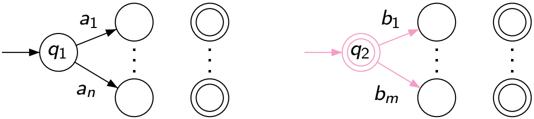
- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \in F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q, a, q') \in \delta_2 : q \neq q_2\} \cup \{(q, a, q') : q \in F_1 \wedge (q_2, a, q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_1 \cup (F_2 - \{q_2\})$



DFA for a Linearized Regular Expression

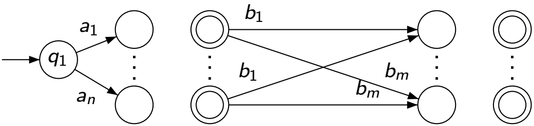
Product

- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \in F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q, a, q') \in \delta_2 : q \neq q_2\} \cup \{(q, a, q') : q \in F_1 \wedge (q_2, a, q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_1 \cup (F_2 - \{q_2\})$



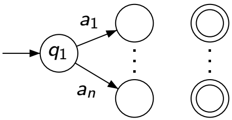
DFA for a Linearized Regular Expression
Product

- Product $(\overline{\alpha} \cdot \overline{\beta})$ $(q_2 \in F_2)$:
  - $Q = (Q_1 \cup Q_2) - \{q_2\})$,
  - $\delta = \delta_1 \cup \{(q,a,q') \in \delta_2 : q \neq q_2\} \cup \{(q,a,q') : q \in F_1 \wedge (q_2,a,q') \in \delta_2\}$,
  - $q_0 = q_1$
  - $F = F_1 \cup (F_2 - \{q_2\})$



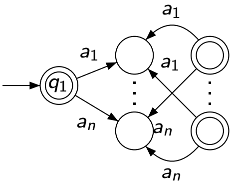
DFA for a Linearized Regular Expression
Star closure

- Closure ($\overline{\alpha}^*$):
  - $\delta' = \delta \cup \{(q, a, q') : q \in F \land (q_0, a, q') \in \delta\}$
  - $F = F_1 \cup \{q_1\})$



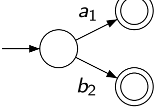
DFA for a Linearized Regular Expression
Star closure

- Closure ($\overline{\alpha}^*$):
  - $\delta' = \delta \cup \{(q, a, q') : q \in F \wedge (q_0, a, q') \in \delta\}$
  - $F = F_1 \cup \{q_1\})$

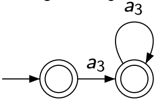


DFA for a Linearized Regular Expression
Example

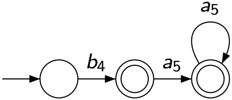
Let $\alpha = (a+b)(a^* + ba^* + b^*)^*$. Then
$\overline{\alpha} = (a_1 + b_2)(a_3^* + b_4 a_5^* + b_6^*)^*$



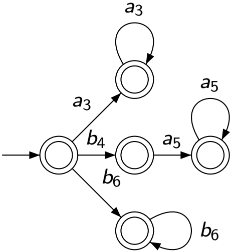
$a_1 + b_2$



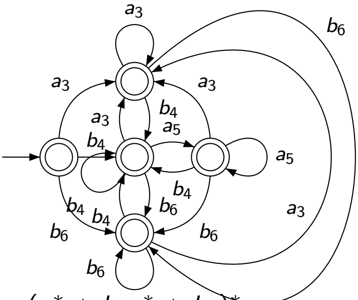
a_3^*



$b_4 a_5^*$



$a_3^* + b_4 a_5^* + b_6$



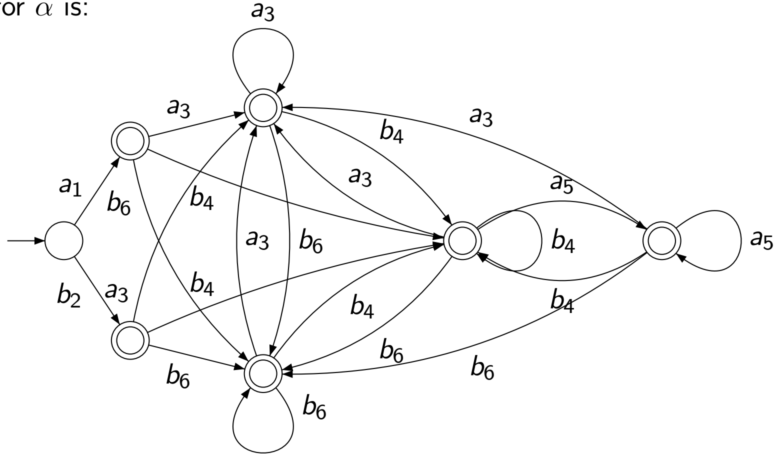
$(a_3^* + b_4 a_5^* + b_6)^*$

Position automaton
Algorithm

1: **Input:** α regular expression over Σ
2: **Output:** DFA for $L(\alpha)$
3: **Method:**
4: Obtain $\overline{\alpha}$ linearized version of α
5: Obtain A a Standard local automaton for $\overline{\alpha}$
6: $A_{pos} = h(A)$, where h is a homomorphism that erases subíndexes.
7: Return A_{pos}
8: **End Method**

Position automaton
Example

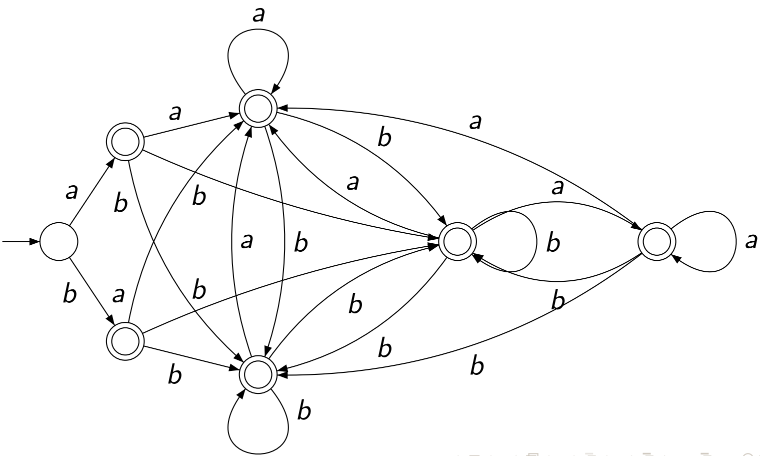
Given $\alpha = (a+b)(a^* + ba^* + b^*)^*$ and its linearized version $\overline{\alpha} = (a_1 + b_2)(a_3^* + b_4 a_5^* + b_6^*)^*$, the standard local automaton for $\overline{\alpha}$ is:



Position automaton

Example

and the Position automaton for $\alpha = (a+b)(a^* + ba^* + b^*)^*$ is:



Follow automaton

- Follow Relation
- Follow Automaton

Follow Automaton

- The *Follow Automaton* of a regular expression α is the quotient automaton of the position automaton by the following relation:

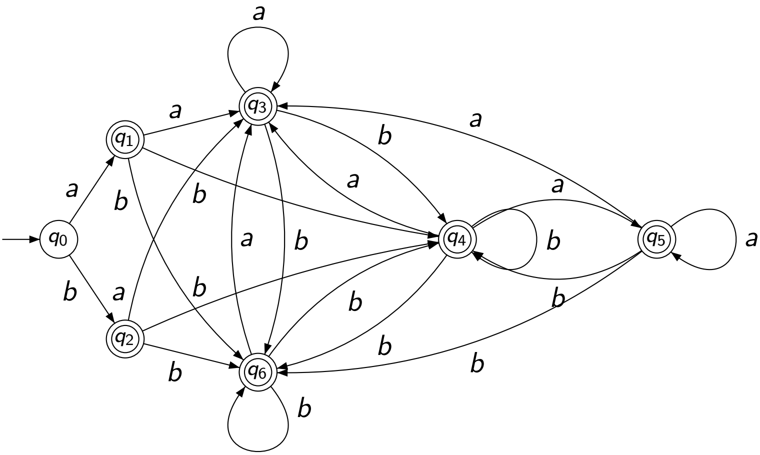
$$p \equiv_f q \Leftrightarrow \begin{cases} p, q \in F \ \mathit{or} \ p, q \in Q - F \\ \mathit{follow}(p) = \mathit{follow}(q) \end{cases}$$

where $\mathit{follow}(p) = \{q \in Q \ : \ \exists a \in \Sigma, \delta(p, a) = q\}$

- The resulting quotient automaton is a partial reduction of the position automaton.

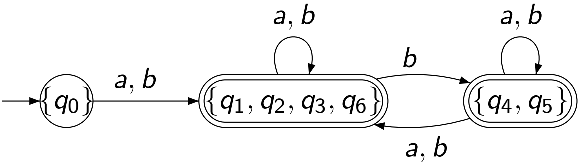
Follow Automaton

We recall the position automaton for
$\alpha = (a+b)(a^* + ba^* + b^*)^*$:



Follow Automaton

The equivalence classes are: $\{q_0\}, \{q_1, q_2, q_3, q_6\}, \{q_4, q_5\}$, thus the follow automaton for α is:



Brzozowski's algorithm

- Derivatives
- Brzozowski's algorithm

Derivatives

- Rules to calculate derivatives
  - With respect to symbols ($a, b \in \Sigma, \quad r, s$ E.R.)
    1. $a^{-1}\emptyset = \emptyset$
    2. $a^{-1}\lambda = \emptyset$
    3. $a^{-1}b = \begin{cases} \emptyset & \text{si } a \neq b \\ \lambda & \text{si } a = b \end{cases}$
    4. $a^{-1}(r+s) = a^{-1}r + a^{-1}s$
    5. $a^{-1}(rs) = \begin{cases} (a^{-1}r)s & \text{si } \lambda \notin r \\ (a^{-1}r)s + a^{-1}s & \text{si } \lambda \in r \end{cases}$
    6. $a^{-1}r^* = (a^{-1}r)r^*$
  - with respect to strings ($a \in \Sigma, \quad x \in \Sigma^*$)
    1. $\lambda^{-1}r = r$
    2. $(xa)^{-1}r = a^{-1}(x^{-1}r)$

Brzozowski's Algorithm

Input: α regular expression over Σ
Output: minimum DFA for $L(\alpha)$
Method:

$Q = \{\alpha\};\ q_0 = \alpha;\ F = \emptyset;\ \delta = \emptyset;$
if $\lambda \in L(\alpha)$ **then**
  $F = F \cup \{\alpha\}$
end if
$actives = \{\alpha\}$
while $actives \neq \{\}$ **do**
  $\beta = First(actives)$
  $actives = Rest(actives)$
  for all $a \in \Sigma$ **do**
    $\beta' = a^{-1}\beta$
    if $\nexists r \in Q\ :\ L(r) = L(\beta')$ **then**
      $Q = Q \cup \{\beta'\}$
      $\delta = \delta \cup \{(\beta, a, \beta')\}$
      $actives = actives \cup \{\beta'\}$
      if $\lambda \in L(\beta')$ **then**
        $F = F \cup \{\beta'\}$
      end if
    end if
  end for
end while
Return $(Q, \Sigma, \delta, q_0, F)$
End Method

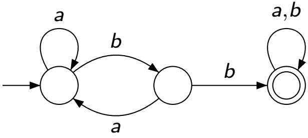
Brzozowski's Algorithm
Example

- Consider $\alpha = (a+b)^* bb(a+b)^*$:

- Consider $\alpha = (a+b)^*bb(a+b)^*$:
  - $q_0 = \alpha = (a+b)^*bb(a+b)^*$; $\lambda \notin L(q_0)$ and thus $F = \emptyset$
  - $a^{-1}q_0 = q_0$
    $b^{-1}q_0 = (a+b)^*bb(a+b)^* + b(a+b)^* = q_1$; $\lambda \notin L(q_1)$ and thus $F = \emptyset$.
  - $a^{-1}q_1 = q_0$
    $b^{-1}q_1 = (a+b)^*bb(a+b)^* + b(a+b)^* + (a+b)^* = (a+b)^* = q_2$; $\lambda \in L(q_2)$ and thus $F = \{q_2\}$.
  - $a^{-1}q_2 = b^{-1}q_2 = q_2$

Brzozowski's Algorithm
Example

- Consider $\alpha = (a+b)^*bb(a+b)^*$:
  - $q_0 = \alpha = (a+b)^*bb(a+b)^*$; $\lambda \notin L(q_0)$ and thus $F = \emptyset$
  - $a^{-1}q_0 = q_0$
    $b^{-1}q_0 = (a+b)^*bb(a+b)^* + b(a+b)^* = q_1$; $\lambda \notin L(q_1)$ and thus $F = \emptyset$.
  - $a^{-1}q_1 = q_0$
    $b^{-1}q_1 = (a+b)^*bb(a+b)^* + b(a+b)^* + (a+b)^* = (a+b)^* = q_2$; $\lambda \in L(q_2)$ and thus $F = \{q_2\}$.
  - $a^{-1}q_2 = b^{-1}q_2 = q_2$



FA Analysis

- Systems of equations with regular expressions
- Arden's Lemma
- FA Analysis

- equation with regular expressions: linear equation where both variables and coefficients are regular expressions.

$$X = rX + s$$

- Arden's Lemma: Let $X = rX + s$ be an equation with regular expressions. $X = r^*s$ is a solution for the ecuation. It is the only solution if $\lambda \notin r$
  - we prove that r^*s is a solution:

$$rX + s \underset{X=r^*s}{=} rr^*s + s = (rr^* + \lambda)s \underset{rr^*+\lambda=r^*}{=} r^*s$$

  - If $\lambda \in r$ there is an infinite number of solutions: $\forall t \subseteq \Sigma^*$, $r^*(s+t)$ is a solution:

$$\begin{aligned} X =& rX + s = rr^*(s+t) + s = rr^*s + rr^*t + s = \\ =&(rr^* + \lambda)s + rr^*t \underset{rr^*+\lambda=r^*}{=} r^*s + r^*t \underset{X=r^*(s+t)}{=} X \end{aligned}$$

Systems of equations with regular expressions

- Given a System of equations with regular expressions.:

$$\begin{cases} X_1 = r_{11}X_1 + r_{12}X_2 + & \ldots + r_{1n}X_n + s_1 \\ X_2 = r_{11}X_1 + r_{12}X_2 + & \ldots + r_{1n}X_n + s_2 \\ & \ldots \\ X_n = r_{11}X_1 + r_{12}X_2 + & \ldots + r_{1n}X_n + s_3 \end{cases}$$

The solution comes through applying Gauss' method, using Arden's Lemma to reduce r.e.

Chapter 5: Regular Expressions
U.D. Computación
Definitions
Properties
Constructions
Syntheses of FAs from ERs
Position Automata
Follow Automaton
Brzozowski's Algorithm
FA Analysis. Arden's Lemma

1: **Input:** Finite Automaton $A = (Q, \Sigma, \delta, q_1, F)$ con $Q = \{q_1, q_2, \dots, q_n\}$
2: **Output:** Regular Expression for $L(A)$
3: **Metodo:**
4: For every state q_i we introduce a variable X_i
5: If $q_i \in F$ then we add λ to the right side of the i-th equation
6: If $q_j \in \delta(q_i, a)$ then we add the term aX_j to the right side of the i-th equation, with $a \in \Sigma \cup \{\lambda\}$
7: Solve the system of equations with regular expressions using Arden's Lemma to reduce r.e.
8: Return the r.e. associated to the initial state
9: **End Method**