Automata and Grammars (BIE-AAG)

5. Conversions among RG, RE, and FA

Jan Holub

Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague

Relationship between RE and FA

regular expression $\rightarrow$ finite automaton

- method of neighbours, Glushkov
- method of derivatives, Janusz A. Brzozowski
- method of incremental construction, Ken Thompson

Relationship between RE and FA

Algorithm NFA for a given regular expression – Glushkov's method

Input: Regular expression V over alphabet Σ.
Output: NFA M, $h(V) = L(M)$.

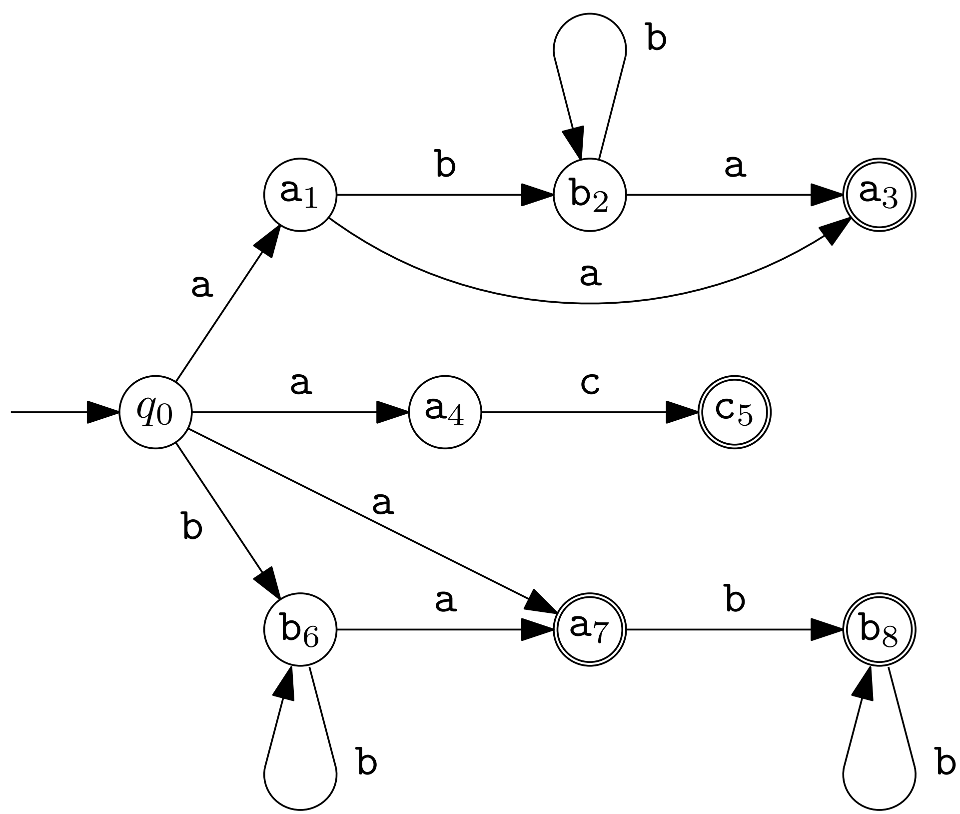
1: By assigning numbers $1, 2, \ldots, n$ to all occurrences of symbols from Σ in expression V we get RE V'.
2: $Z \leftarrow \{a_i : a \in \Sigma$, symbol a_i is the first symbol of a string in $h(V')\}$ ▷ the set of symbols at the beginning
3: $P \leftarrow \{a_i b_j : a, b \in \Sigma$, symbols a_i and b_j occur next to each other in some string from $h(V')\}$ ▷ the set of neighbours
4: $Q \leftarrow \{q_0\} \cup \{a_i : a_i \in V'\}$, $q_0 \notin V'$
5: $F \leftarrow \{a_i :$ symbol a_i is the last symbol of a string in $h(V')\}$ $\cup\{q_0 : \varepsilon \in h(V)\}$ ▷ the set of final symbols
6: $\delta(q, a) \leftarrow \emptyset$, $\forall q \in Q, \forall a \in \Sigma$,
7: $\delta(q_0, a) \leftarrow \delta(q_0, a) \cup \{a_i\}$, $\forall a_i \in Z$,
8: $\delta(a_i, b) \leftarrow \delta(a_i, b) \cup \{b_j\}$, $\forall a_i b_j \in P$
9: $M \leftarrow (Q, \Sigma, \delta, q_0, F)$
10: **return** M

Relationship between RE and FA

Example

$V = \mathtt{ab^*a} + \mathtt{ac} + \mathtt{b^*ab^*}$, $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

$V' = \mathtt{a}_1\mathtt{b}_2^*\mathtt{a}_3 + \mathtt{a}_4\mathtt{c}_5 + \mathtt{b}_6^*\mathtt{a}_7\mathtt{b}_8^*$, $Z = \{\mathtt{a}_1, \mathtt{a}_4, \mathtt{b}_6, \mathtt{a}_7\}$.
$P = \{\mathtt{a}_1\mathtt{b}_2, \mathtt{a}_1\mathtt{a}_3, \mathtt{b}_2\mathtt{b}_2, \mathtt{b}_2\mathtt{a}_3, \mathtt{a}_4\mathtt{c}_5, \mathtt{b}_6\mathtt{b}_6, \mathtt{b}_6\mathtt{a}_7, \mathtt{a}_7\mathtt{b}_8, \mathtt{b}_8\mathtt{b}_8\}$.
$F = \{\mathtt{a}_3, \mathtt{c}_5, \mathtt{a}_7, \mathtt{b}_8\}$.
$M = (\{\mathtt{q}_0, \mathtt{a}_1, \mathtt{b}_2, \mathtt{a}_3, \mathtt{a}_4, \mathtt{c}_5, \mathtt{b}_6, \mathtt{a}_7, \mathtt{b}_8\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, \delta, q_0, \{\mathtt{a}_3, \mathtt{c}_5, \mathtt{a}_7, \mathtt{b}_8\})$



Relationship between RE and FA

Algorithm DFA for a given regular expression – method of derivatives, Janusz A. Brzozowski

Input: Regular expression V over alphabet Σ.
Output: DFA M, $h(V) = L(M)$.

$Q \leftarrow \{V\}$; $q_0 \leftarrow V$
for $\forall U \in Q$ **do**
    for $\forall a \in \Sigma$ **do**
        $Q \leftarrow Q \cup \{\frac{\mathrm{d}U}{\mathrm{d}a} : \nexists U' \in Q, \frac{\mathrm{d}U}{\mathrm{d}a} \cong U'\}$
    end for
end for
$\delta(U, a) \leftarrow U'$, $\forall U \in Q$, $\forall a \in \Sigma$, $U' \in Q, U' \cong \frac{\mathrm{d}U}{\mathrm{d}a}$
$F \leftarrow \{U : U \in Q, \varepsilon \in h(U)\}$
$M \leftarrow (Q, \Sigma, \delta, q_0, F)$
return M

Relationship between RE and FA

Theorem (RE – bound of dissimilarities [Brz64])
Every regular expression has only a finite number of dissimilar derivatives.
Corollary
DFA can be constructed from RE using the algorithm above if only similarity ($\cong$) among derivatives is recognized.

Relationship between RE and FA

Example

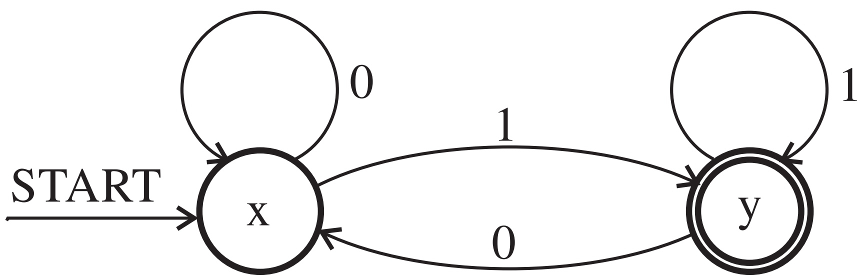
$(0+1)^*1$

1. $Q = \{(0+1)^*1\}$
2. $\frac{\mathrm{d}}{\mathrm{d}1}\,((0+1)^*1) = (\emptyset+\varepsilon)(0+1)^*1+\varepsilon = (0+1)^*1+\varepsilon$
   $\frac{\mathrm{d}}{\mathrm{d}0}\,((0+1)^*1) = (\varepsilon+\emptyset)(0+1)^*1+\emptyset = (0+1)^*1$
   $Q = \{(0+1)^*1, (0+1)^*1+\varepsilon\}$
3. $\frac{\mathrm{d}}{\mathrm{d}1}\,((0+1)^*1+\varepsilon) = (\emptyset+\varepsilon)(0+1)^*1+\varepsilon+\emptyset = (0+1)^*1+\varepsilon$
   $\frac{\mathrm{d}}{\mathrm{d}0}\,((0+1)^*1+\varepsilon) = (\varepsilon+\emptyset)(0+1)^*1+\emptyset+\emptyset = (0+1)^*1$
   $Q = \{(0+1)^*1, (0+1)^*1+\varepsilon\}$

Relationship between RE and FA

Example (continued)

4. state $x = (0+1)^*1$,
   state $y = (0+1)^*1 + \varepsilon$,
   the initial state is x,
   the final state is y, as $\varepsilon \in h((0+1)^*1 + \varepsilon)$
   $\frac{\mathrm{d}x}{\mathrm{d}1} = \mathrm{y}$, $\frac{\mathrm{d}x}{\mathrm{d}0} = x$, $\frac{\mathrm{d}y}{\mathrm{d}1} = y$, $\frac{\mathrm{d}y}{\mathrm{d}0} = x$.

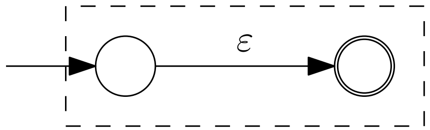


Relationship between RE and FA

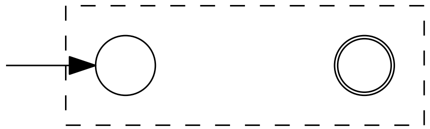
Algorithm NFA for a given regular expression – Thompson's method

Input: Regular expression V over alphabet Σ.
Output: NFA M, $h(V) = L(M)$.

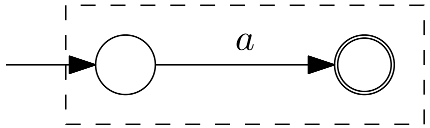
1: Construct NFA for regular expression $U = \varepsilon$:



2: Construct NFA for regular expression $U = \emptyset$:

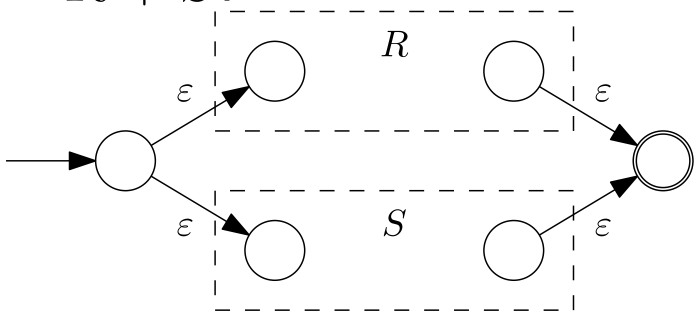


3: Construct NFA for regular expression $U = a, \forall a \in \Sigma$:

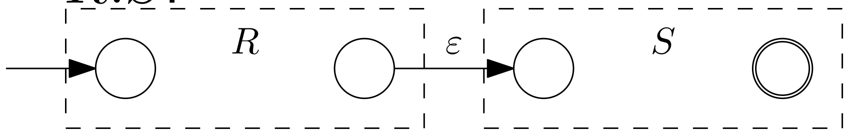


Relationship between RE and FA

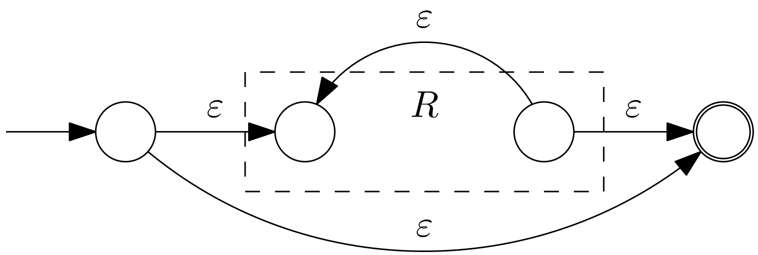
4: Given regular expressions R and S, construct NFA for regular expression $U = R + S$:



5: Given regular expressions R and S, construct NFA for regular expression $U = R.S$:



6: Given regular expression R, construct NFA for regular expression $U = R^*$:



7: $M \leftarrow (Q, \Sigma, \delta, q_0, F)$

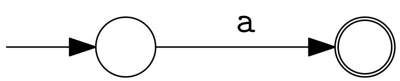
8: **return** M

Relationship between RE and FA

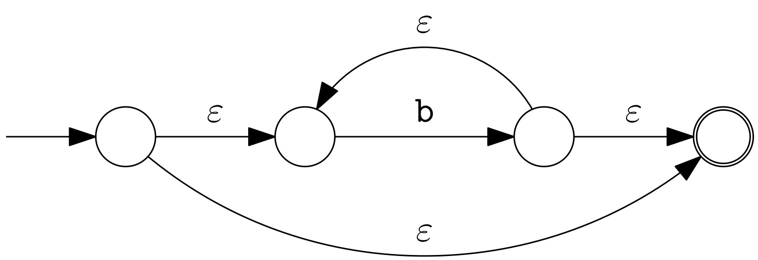
Example

$V = \mathtt{ab}^*\mathtt{a} + \mathtt{ab}$.

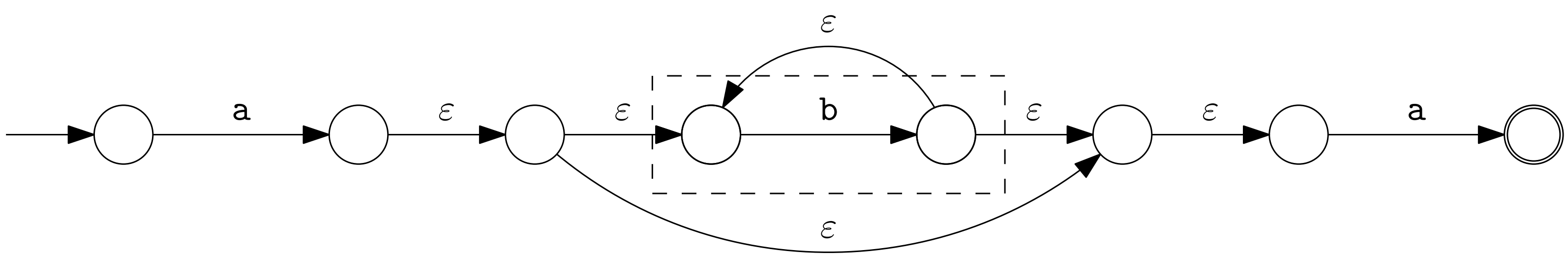
Necessary elementary regular expressions are $\mathtt{a}$ and $\mathtt{b}$:



We further create automaton for expression $\mathtt{b}^*$:



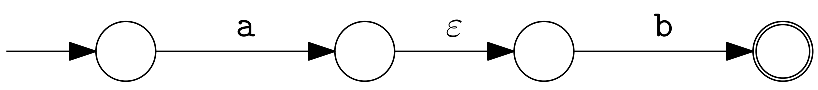
NFA for expression $\mathtt{ab}^*\mathtt{a}$:



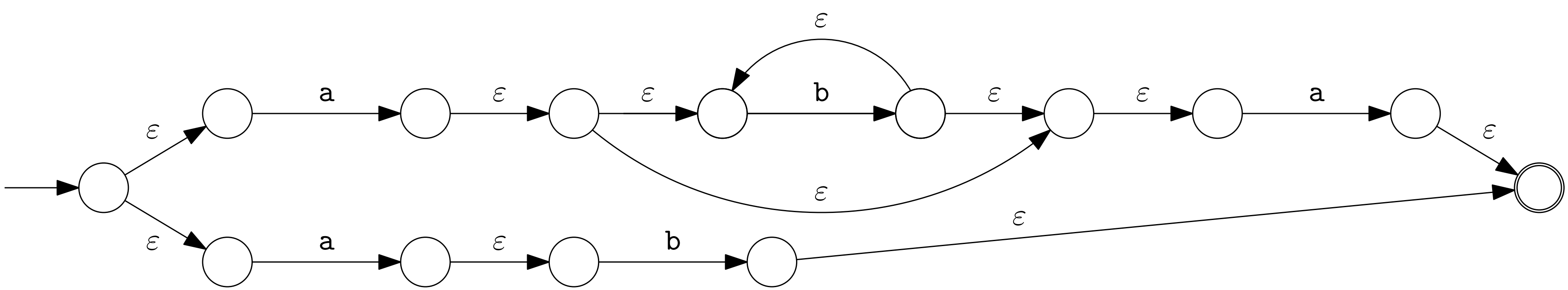
Relationship between RE and FA

Example (continued)

For expression ab the FA is of the form:



For regular expression $V = \mathtt{ab}^*\mathtt{a} + \mathtt{ab}$:



Relationship between FA and RE

finite automaton $\rightarrow$ regular expression

- state elimination method
- method of regular equations – incoming transitions
- method of regular equations – outgoing transitions

Relationship between FA and RE

Theorem
For every NFA M a regular expression V can be constructed such that $L(M) = h(V)$.

Relationship between FA and RE

Definition

Extended finite automaton (EFA) M is $(Q, \Sigma, \gamma, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite input alphabet,
- γ is mapping from $Q \times Q$ into R_T (R_T is a set of REs over Σ),
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states.

$\gamma(p, q) = \emptyset$, if transition from p to q is not defined.

Definition

The language accepted by EFA M is $L(M) = \{x : x \in \Sigma^*,\ x = x_1 x_2 \dots x_n,$ $x_i \in \Sigma^*$ and there exists a sequence of states $q_0, q_1, \dots, q_n,\ q_n \in F$ such that $x_1 \in h(\gamma(q_0, q_1)),\ x_2 \in h(\gamma(q_1, q_2)), \dots,\ x_n \in h(\gamma(q_{n-1}, q_n))\}$.

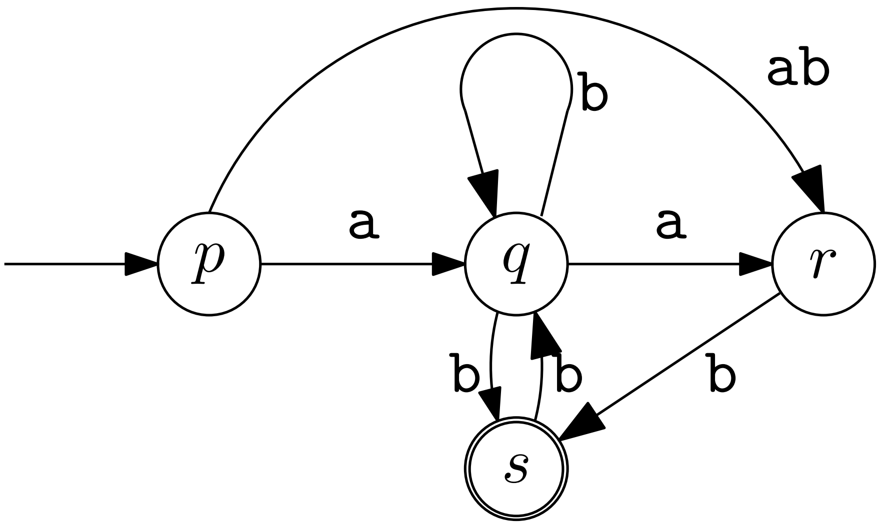
Relationship between FA and RE

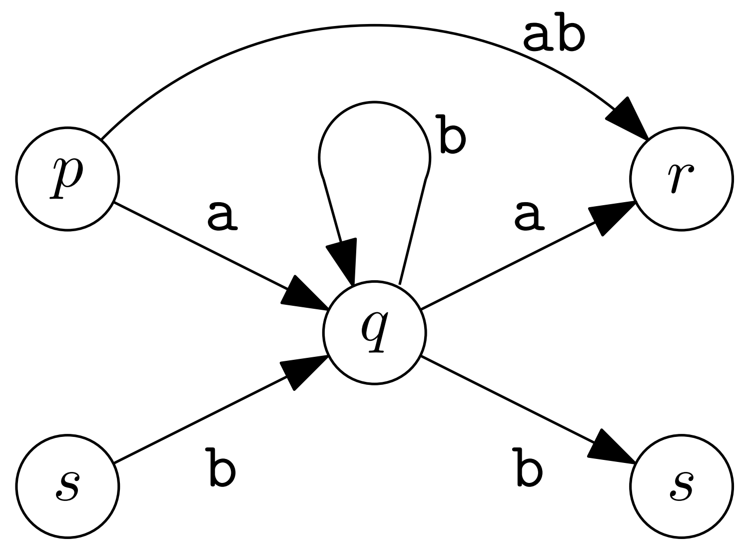
Theorem
Let $M = (Q, \Sigma, \gamma, q_0, F)$ be an EFA. Assume that $q \in Q$ is neither a start state, nor a final state. Then the equivalent EFA M' is $(Q \setminus \{q\}, \Sigma, \gamma', q_0, F)$, where mapping γ' is defined for every pair $p, r \in (Q \setminus \{q\})$ as follows:
$\gamma'(p, r) = \gamma(p, r) + \gamma(p, q)\gamma(q, q)^*\gamma(q, r)$.

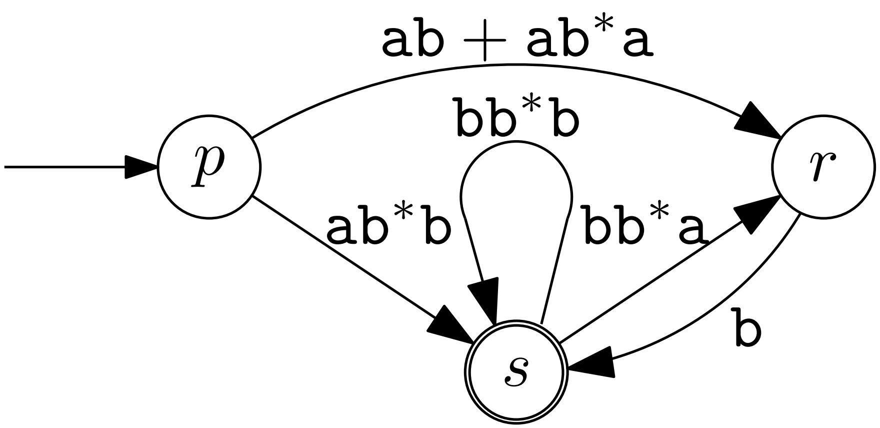
Note that the second term of expression
$\gamma'(p, r) = \gamma(p, r) + \gamma(p, q)\gamma(q, q)^*\gamma(q, r)$ does not apply if $\gamma(p, q) = \emptyset$ or $\gamma(q, r) = \emptyset$.

Relationship between FA and RE

Example







Relationship between FA and RE

Algorithm Regular expression for a given NFA – elimination of states.

Input: NFA $M = (Q, \Sigma, \delta, q_0, F)$.

Output: Regular expression V such that $h(V) = L(M)$.

$\gamma(p, q) \leftarrow +_{q \in \delta(p,a), a \in \Sigma \cup \{\varepsilon\}} a,\ \forall p, q \in Q$ ▷ r.e. for all transitions from p to q

$M_R \leftarrow (Q, \Sigma, \gamma, q_0, F)$ ▷ The initial EFA

if $q_0 \in F \vee \exists q, \gamma(q, q_0) \neq \emptyset$ **then**

  $Q \leftarrow Q \cup \{q_0''\}, q_0'' \notin Q$; $\gamma(q_0'', q_0) \leftarrow \varepsilon$; $q_0' \leftarrow q_0''$ ▷ q_0'' is a new initial state.

else

  $q_0' \leftarrow q_0$

end if

if $|F| > 1$ **then**

  $Q \leftarrow Q \cup \{f\}, f \notin Q$; $\gamma(q, f) \leftarrow \varepsilon, \forall q \in F$; $F' \leftarrow \{f\}$ ▷ a new final state

else

  $F' \leftarrow F$

end if

while $Q \neq \{q_0', f\}$ **do**

  Choose $q \in Q, q \notin \{q_0', f\}$; $Q \leftarrow Q \setminus \{q\}$

  $\gamma'(p, r) \leftarrow \gamma(p, r) + \gamma(p, q)\gamma(q, q)^*\gamma(q, r), \forall p, r \in Q$ ▷ See slide number 16

  $\gamma \leftarrow \gamma'$

end while

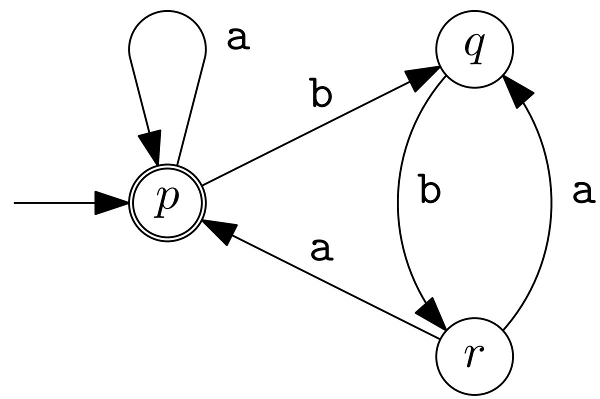
$V \leftarrow \gamma'(q_0', f)\gamma'(f, f)^*$

return V

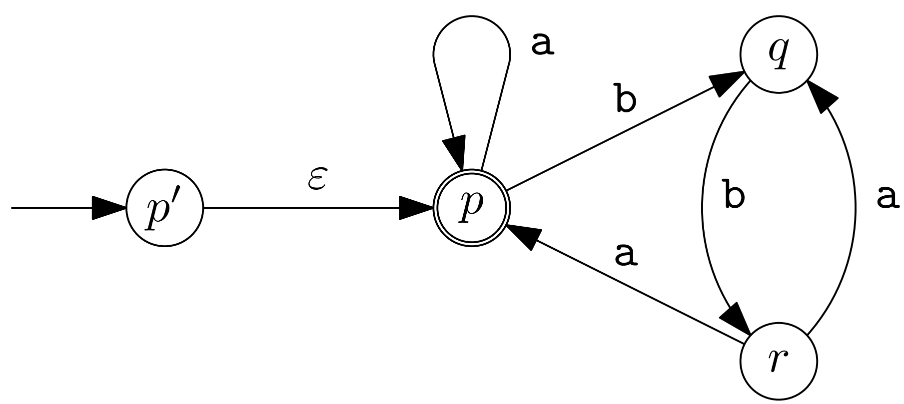
Relationship between FA and RE

Example

$M = (\{p, q, r\}, \{\mathtt{a}, \mathtt{b}\}, \delta, p, \{p\})$



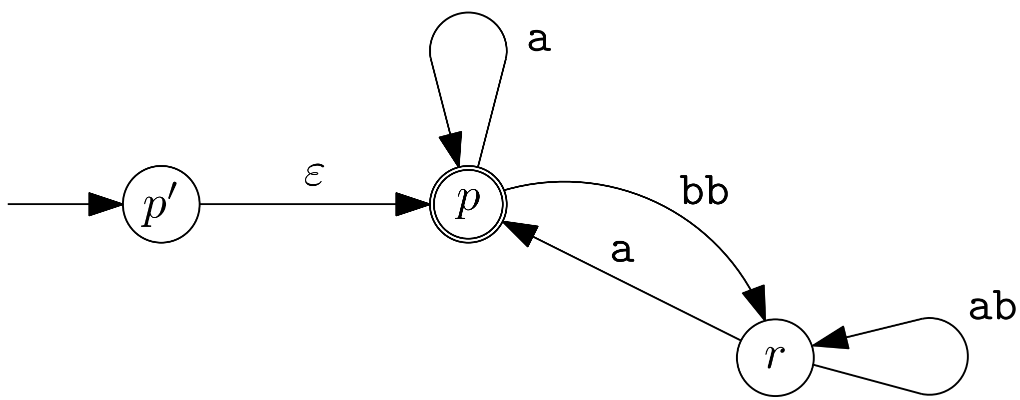
We construct the EFA and add a new initial state.



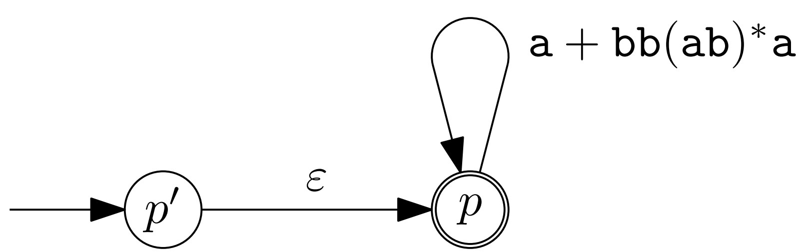
Relationship between FA and RE

Example (continued)

In the next step we exclude state q.



In the next step we exclude state r.



The resulting regular expression is $V = (\mathtt{a} + \mathtt{bb}(\mathtt{ab})^*\mathtt{a})^*$.

Relationship between FA and RE

Algorithm Regular expression for a given NFA – solution of regular equations, incoming transitions

Input: NFA $M = (Q, \Sigma, \delta, q_0, F)$.

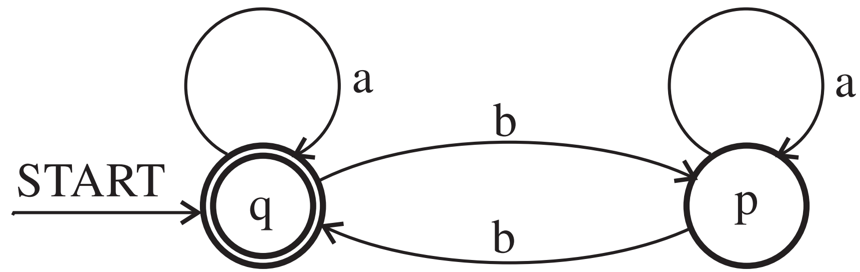
Output: Regular expression V such that $h(V) = L(M)$.

```
for ∀q ∈ Q \ {q0} do
    Insert equation X_q = X_{p1} a1 + X_{p2} a2 + ··· + X_{pn} an,
    ∀ai ∈ Σ, ∀pi ∈ Q, q ∈ δ(pi, ai)
end for
Insert equation X_{q0} = X_{p1} a1 + X_{p2} a2 + ··· + X_{pn} an + ε,
∀ai ∈ Σ, ∀pi ∈ Q, q0 ∈ δ(pi, ai)
Solve the system of left regular equations.
V ← X_{p1} + X_{p2} + ··· + X_{pn}, ∀pi ∈ F
return V
```

Relationship between FA and RE

Example

$M = (\{q, p\}, \{a, b\}, \delta, q, \{q\})$



$$X_q = X_q a \ + X_p b + \varepsilon$$
$$X_p = X_p a \ + X_q b$$

We express the variable X_p:

$$X_p = X_q b a^*$$

We substitute for X_p into the first equation:

$$X_q = X_q a \ + X_q b a^* b + \varepsilon = X_q(a + ba^*b) + \varepsilon$$

We express the variable X_q:

$$X_q = (a + ba^*b)^* = V$$

Relationship between FA and RE

Algorithm Regular expression for a given NFA – solution of regular equations, outgoing transitions

Input: NFA $M = (Q, \Sigma, \delta, q_0, F)$.

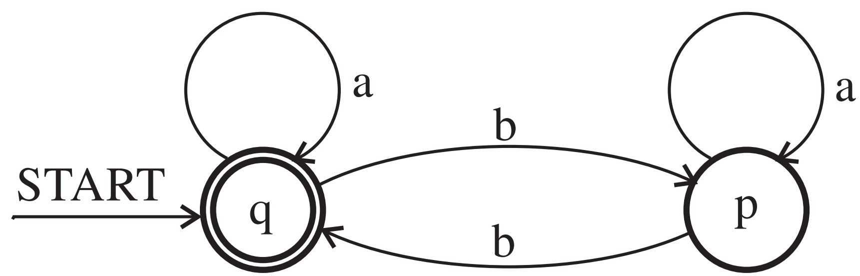
Output: Regular expression V such that $h(V) = L(M)$.

for $\forall q \in Q \setminus F$ **do**
    Insert equation $X_q = a_1 X_{p_1} + a_2 X_{p_2} + \cdots + a_n X_{p_n}$, $\forall a_i \in \Sigma, \forall p_i \in Q$, $p_i \in \delta(q, a_i)$
end for
for $\forall q \in F$ **do**
    Insert equation $X_q = a_1 X_{p_1} + a_2 X_{p_2} + \cdots + a_n X_{p_n} + \varepsilon$, $\forall a_i \in \Sigma, \forall p_i \in Q$, $p_i \in \delta(q, a_i)$
end for
Solve the system of right regular equations.
$V \leftarrow X_{q_0}$
return V

Relationship between FA and RE

Example

$M = (\{q, p\}, \{a, b\}, \delta, q, \{q\})$



$$X_q = aX_q + bX_p + \varepsilon$$
$$X_p = aX_p + bX_q$$

We express the variable X_p:

$$X_p = a^*bX_q$$

We substitute for X_p into the first equation:

$$X_q = aX_q + ba^*bX_q + \varepsilon$$
$$X_q = (a + ba^*b)X_q + \varepsilon$$

We express the variable X_q:

$$X_q = (a + ba^*b)^* = V$$

Relationship between RG and RE

regular grammar $\rightarrow$ regular expression

- nonterminal elimination method
- method of regular equations

Relationship between RG and RE

Theorem

For every regular grammar $G = (N, \Sigma, P, S)$ a regular expression V can be constructed such $L(G) = h(V)$.

Definition

Extended regular grammar (ERG) is the quadruple $G = (N, \Sigma, P, S)$, where

- N is the finite set of nonterminal symbols,
- Σ is the finite set of terminal symbols,
- P is the set of rules in the form $A \rightarrow \alpha B$ or $A \rightarrow \alpha,\ A, B \in N, \alpha \in R_T$,
- $S \in N$ is the start symbol.

Relationship between RG and RE

W.L.O.G. we assume that for given $A, B \in N$ there is at most one production rule of the form $A \rightarrow \alpha B$ and $A \rightarrow \alpha$ in the ERG.
(Modification: n-tuple of rules $A \rightarrow \alpha_1 B,\ A \rightarrow \alpha_2 B, \ldots, A \rightarrow \alpha_n B$ is replaced by rule $A \rightarrow (\alpha_1 + \alpha_2 + \cdots + \alpha_n)B$.)

Definition

Language defined by the ERG
$L(G) = \{x : x \in \Sigma^*, x = x_1 x_2 \ldots x_n$ and there exists derivation
$S \Rightarrow \alpha_1 A_1 \Rightarrow \alpha_1 \alpha_2 A_2 \Rightarrow \ldots \Rightarrow \alpha_1 \alpha_2 \ldots \alpha_{n-1} A_{n-1} \Rightarrow \alpha_1 \alpha_2 \ldots \alpha_{n-1} \alpha_n$ such that $x_i \in h(\alpha_i),\ 1 \leq i \leq n\}$.

Relationship between RG and RE

Theorem

Let $G = (N, \Sigma, P, S)$ be an ERG and A be a nonterminal symbol which is not the start symbol of G (i.e., $A \in N, A \neq S$). ERG $G' = (N \setminus \{A\}, \Sigma, P', S)$, where the rules in P' are formed as follows, is equivalent to G (i.e., $L(G) = L(G')$).

If the following rules are in G:

$$\begin{array}{ll} B \to \alpha_1 C & A \to \alpha_3 A \\ B \to \alpha_2 A & A \to \alpha_4 C, \end{array}$$

$\forall B \in N, \forall C \in N \cup \{\varepsilon\}$, then the following rules are in P' of grammar G':

$$B \to (\alpha_1 + \alpha_2 \alpha_3^* \alpha_4) C.$$

Note:

If some of the above rules does not occur in grammar G, then it is replaced by rule $X \to \emptyset Y$, where $X \in \{A, B\}$, $Y \in \{A, C\}$. The corresponding rules will not occur in G' either.

For example if G does not contain rule $A \to \alpha_4 C$, we replace it by $A \to \emptyset C$. The rule $B \to (\alpha_1 + \alpha_2 \alpha_3^* \emptyset) C$ actually changes to rule $B \to \alpha_1 C$.

Relationship between RG and RE

Algorithm Regular expression for a given regular grammar – elimination of nonterminal symbols

Input: Regular grammar $G = (N, \Sigma, P, S)$.

Output: Regular expression V such that $h(V) = L(G)$.

```
For all $n$-tuples of rules of the form $A \rightarrow \alpha_1 B, A \rightarrow \alpha_2 B, \ldots, A \rightarrow \alpha_n B$ in $P$ insert the following rule into $P'$: $A \rightarrow (\alpha_1 + \alpha_2 + \cdots + \alpha_n)B$, where $A \in N,\ B \in N \cup \{\varepsilon\}$
$N' \leftarrow N \cup \{S', F\}$, $S', F \notin N$
$P_R = \{A \rightarrow \alpha B : (A \rightarrow \alpha B) \in P'\}$
    $\cup \{A \rightarrow \alpha F : (A \rightarrow \alpha) \in P'\} \cup \{S' \rightarrow \varepsilon S,\ F \rightarrow \varepsilon\}$
while not($N' = \{S', F\} \wedge P_R = \{S' \rightarrow \alpha F, F \rightarrow \varepsilon\}$) do
    Select $A \in N' \setminus \{S', F\}$
    $N' \leftarrow N' \setminus \{A\}$
    $P_R \leftarrow (P_R \setminus \{B \rightarrow \alpha_1 C, B \rightarrow \alpha_2 A, A \rightarrow \alpha_3 A, A \rightarrow \alpha_4 C\}) \cup \{B \rightarrow (\alpha_1 + \alpha_2 \alpha_3^* \alpha_4)C\}, \forall B, C \in N$    ▷ See slide number 28
end while
$G_R \leftarrow (N', \Sigma, P_R, S')$    ▷ Build ERG
$V \leftarrow \alpha$
return $V$
```

Relationship between RG and RE

Example

$G = (\{S, A, B\}, \{a, b\}, P, S)$, where P:

$S \rightarrow bA \mid aS \mid a$
$A \rightarrow bB$
$B \rightarrow aA \mid aS \mid b.$

$S' \rightarrow S$, S' is a new nonterminal symbol.

$G_R = (\{S', S, A, B, F\}, \{a, b\}, P_R, S')$, where P_R:

$S' \rightarrow S$
$S \rightarrow bA \mid aS \mid aF$
$A \rightarrow bB$
$B \rightarrow aA \mid aS \mid bF$
$F \rightarrow \varepsilon.$

Relationship between RG and RE

Example (continued)

We exclude symbol A. $G_R^1 = (\{S', S, B, F\}, \{a, b\}, P_R^1, S')$, where P_R^1:

$S' \to S$
$S \to bbB \mid aS \mid aF$
$B \to abB \mid aS \mid bF$
$F \to \varepsilon.$

We exclude symbol B. $G_R^2 = (\{S', S, F\}, \{a, b\}, P_R^2, S')$, where P_R^2:

$S' \to S$
$S \to (a + bb(ab)^*a)S \mid (a + bb(ab)^*b)F$
$F \to \varepsilon.$

We exclude symbol S. $G_R^3 = (\{S', F\}, \{a, b\}, P_R^3, S')$, where P_R^3:

$S' \to (a + bb(ab)^*a)^*(a + bb(ab)^*b)F$
$F \to \varepsilon.$

$V = (a + bb(ab)^*a)^*(a + bb(ab)^*b)$

Relationship between RG and RE

Algorithm Regular expression for a given regular grammar – system of regular equations.

Input: Regular grammar $G = (N, \Sigma, P, S)$.
Output: Regular expression V such that $h(V) = L(G)$.

1: For every nonterminal symbol in N we construct a regular equation.
2: Solve the system of right regular equations for the start symbol S.
3: $V \leftarrow S$
4: **return** V

Relationship between RG and RE

Example

$G = (\{S, A\}, \{0, 1\}, P, S)$, where P:

$S \to 0S \mid 1A \mid 1$

$A \to 1S \mid 0A \mid 0$

The system of regular equations has the following form:

$S = 0S + 1A + 1$

$A = 1S + 0A + 0$

We solve the system:

$S = 0^*(1A + 1) = 0^*1(A + \varepsilon)$

$A = 10^*1A + 10^*1 + 0A + 0$

$A = (10^*1 + 0)A + 10^*1 + 0$

$A = (10^*1 + 0)^*(10^*1 + 0)$

$S = 0^*1((10^*1 + 0)^*(10^*1 + 0) + \varepsilon) = 0^*1(10^*1 + 0)^*$

The resulting regular expression describing language $L(G)$ is:

$S = 0^*1(10^*1 + 0)^*$

Relationship between RE and RG

regular expression $\rightarrow$ regular grammar

- method of incremental construction
- method of derivatives

Relationship between RE and RG

Theorem
For each regular expression V a regular grammar G can be constructed such that $L(G) = h(V)$.

Regular grammars and operations over them

Algorithm Grammar for a *union* of languages.

Input: Regular grammars $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, P_2, S_2)$ generating languages L_1 and L_2, $N_1 \cap N_2 = \emptyset$.

Output: Regular grammar G, $L(G) = L_1 \cup L_2$.

1: $N \leftarrow N_1 \cup N_2 \cup \{S\}, S \notin N_1 \cup N_2$
2: $P \leftarrow (P_1 \cup P_2 \cup \{S \to a : (X \to a) \in P_1 \cup P_2, X \in \{S_1, S_2\}, a \in \Sigma\} \cup \{S \to aB : (X \to aB) \in P_1 \cup P_2, X \in \{S_1, S_2\}, B \in N_1 \cup N_2, a \in \Sigma\} \cup \{S \to \varepsilon : (S_1 \to \varepsilon) \in P_1 \vee (S_2 \to \varepsilon) \in P_2\}) \setminus \{S_1 \to \varepsilon, S_2 \to \varepsilon\}$
3: $G \leftarrow (N, \Sigma, P, S)$
4: **return** G

W.L.O.G., the sets of terminal symbols are expected to be equal.

Regular grammars and operations over them

Algorithm Grammar for a *product* of languages.

Input: Regular grammars $G_1 = (N_1, \Sigma, P_1, S_1)$ a $G_2 = (N_2, \Sigma, P_2, S_2)$ generating languages L_1 and L_2, $N_1 \cap N_2 = \emptyset$.

Output: Regular grammar G such that $L(G) = L_1.L_2$.

$N \leftarrow N_1 \cup N_2$

$P \leftarrow \{A \to aB : (A \to aB) \in P_1, A, B \in N_1, a \in \Sigma\}$

$P \leftarrow P \cup \{A \to aS_2 : (A \to a) \in P_1, A \in N_1, a \in \Sigma\}$

$P \leftarrow P \cup \{A \to a : (A \to a) \in P_1, (S_2 \to \varepsilon) \in P_2, A \in N_1, a \in \Sigma\}$

$P \leftarrow P \cup \{A \to aB : (A \to aB) \in P_2, A, B \in N_2, a \in \Sigma\}$

$P \leftarrow P \cup \{A \to a : (A \to a) \in P_2, A \in N_2, a \in \Sigma\}$

$P \leftarrow P \cup \{S_1 \to \alpha : (S_2 \to \alpha) \in P_2, (S_1 \to \varepsilon) \in P_1, \alpha \in \Sigma.N \cup \Sigma \cup \{\varepsilon\}\}$

$G \leftarrow (N, \Sigma, P, S_1)$

return G

Regular grammars and operations over them

Algorithm Grammar for a *Kleene star* of a language.

Input: Regular grammar $G = (N, \Sigma, P, S)$ generating language L.
Output: Regular grammar G' such that $L(G') = L^*$.

$N' \leftarrow N \cup \{S'\}, S' \notin N$
$P' \leftarrow \{A \rightarrow aB : (A \rightarrow aB) \in P, A, B \in N, a \in \Sigma\}$
$P' \leftarrow P' \cup \{A \rightarrow a \mid aS : (A \rightarrow a) \in P, A \in N, a \in \Sigma\}$
$P' \leftarrow P' \cup \{S' \rightarrow aB : (S \rightarrow aB) \in P, B \in N, a \in \Sigma\}$
$P' \leftarrow P' \cup \{S' \rightarrow a \mid aS : (S \rightarrow a) \in P, a \in \Sigma\} \cup \{S' \rightarrow \varepsilon\}$
$G' \leftarrow (N', \Sigma, P', S')$
return G

Relationship between RE and RG

Algorithm Regular grammar for a given regular expression – incremental construction

Input: Regular expression V over alphabet Σ.
Output: Regular grammar $G = (N, \Sigma, P, S)$ such that $L(G) = h(V)$

1: $G_a = (\{A\}, \{a\}, \{A \to a\}, A)$, $\forall a \in \Sigma$
2: $G_\varepsilon = (\{E\}, \emptyset, \{E \to \varepsilon\}, E)$
3: $G_\emptyset = (\{B\}, \emptyset, \emptyset, B)$
4: We incrementally construct grammars for all subexpressions of type $x_1 + x_2$, $x_1 x_2$, x_1^* (i.e., for languages $h(x_1) \cup h(x_2)$, $h(x_1).h(x_2)$,$(h(x_1))^*$, respectively) of expression V from grammars for subexpressions x_1, x_2 (i.e., for languages $h(x_1)$, $h(x_2)$, respectively).
5: **return** G_V

Relationship between RE and RG

Example

$(ab + \varepsilon)^*$

1. $G_a = (\{A\}, \{a\}, \{A \to a\}, A)$
   $G_b = (\{B\}, \{b\}, \{B \to b\}, B)$.
2. $G_\varepsilon = (\{E\}, \emptyset, \{E \to \varepsilon\}, E)$.
3. We incrementally construct grammars for all languages which are values of expressions $ab, ab + \varepsilon, (ab + \varepsilon)^*$:
   $G(ab) = (\{A, B\}, \{a, b\}, \{A \to aB, B \to b\}, A)$

   $G(ab + \varepsilon) = (\{S, A, B, E\}, \{a, b\}, P_1, S)$, where P_1:
   $S \to aB \mid \varepsilon$, $\quad A \to aB$, $\quad B \to b$.

   $G((ab + \varepsilon)^*) = (\{S', S, A, B, E\}, \{a, b\}, P_2, S')$, where P_2:
   $S' \to aB \mid \varepsilon$, $\quad S \to aB$, $\quad A \to aB$, $\quad B \to b \mid bS$.

Relationship between RE and RG

Algorithm Regular grammar for a given regular expression – method of derivatives.

Input: Regular expression V over alphabet Σ.

Output: Regular grammar G such that $L(G) = h(V)$.

$N \leftarrow \{V\}$; $S \leftarrow V$
for $\forall U \in N$ **do**
    for $\forall a \in \Sigma$ **do**
        $N \leftarrow N \cup \{\frac{\mathrm{d}U}{\mathrm{d}a} : \nexists U' \in N, \frac{\mathrm{d}U}{\mathrm{d}a} \cong U'\}$
    end for
end for
$P \leftarrow \{U \rightarrow aU' : a \in \Sigma, U, U' \in N, U' \cong \frac{\mathrm{d}U}{\mathrm{d}a}\} \cup \{U \rightarrow a : a \in \Sigma, U \in N, \varepsilon \in h(\frac{\mathrm{d}U}{\mathrm{d}a})\}$
if $\varepsilon \in h(V)$ **then**
    if $(U \rightarrow aV) \in P, a \in \Sigma, U \in N$ **then** ▷ start symbol on righthand side
        $N \leftarrow N \cup \{V'\}, V' \notin N$ ▷ new start symbol
        $P \leftarrow P \cup \{V' \rightarrow \alpha : (V \rightarrow \alpha) \in P, \alpha \in N \cup N.\Sigma\} \cup \{V' \rightarrow \varepsilon\}$
        $S \leftarrow V'$
    else
        $P \leftarrow P \cup \{V \rightarrow \varepsilon\}$
    end if
end if
$G \leftarrow (N, \Sigma, P, S)$
return G

Relationship between RE and RG

Example

$(ab+\varepsilon)^*$

1. $N = \{(ab+\varepsilon)^*\}$
2. $\frac{\mathrm{d}(ab+\varepsilon)^*}{\mathrm{d}a} = (b+\emptyset)(ab+\varepsilon)^* = b(ab+\varepsilon)^*$
   $\frac{\mathrm{d}(ab+\varepsilon)^*}{\mathrm{d}b} = (\emptyset+\emptyset)(ab+\varepsilon)^* = \emptyset$
   $N = \{(ab+\varepsilon)^*, b(ab+\varepsilon)^*\}$
3. $\frac{\mathrm{d}(b(ab+\varepsilon)^*)}{\mathrm{d}a} = \emptyset$
   $\frac{\mathrm{d}(b(ab+\varepsilon)^*)}{\mathrm{d}b} = \varepsilon(ab+\varepsilon)^* = (ab+\varepsilon)^*$
   $N = \{(ab+\varepsilon)^*, b(ab+\varepsilon)^*\}$

If we label $A = (ab+\varepsilon)^*, B = b(ab+\varepsilon)^*$, we get productions P:

$$A \to aB \qquad B \to bA \mid b$$

As $\varepsilon \in h(A)$ and there is start symbol A on righthand side in production $B \to bA$, new start symbol A' is introduced and we get the resulting regular grammar $G = (\{A, A', B\}, \{a, b\}, P, A')$, where P:

$$A' \to aB \mid \varepsilon \qquad A \to aB \qquad B \to bA \mid b$$