Automata and Grammars (BIE-AAG)

7. Conversions between 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

Algorithm Construction of FA for a given regular expression – Glushkov's method

Input: Regular expression E.

Output: Finite automaton $M = (Q, T, \delta, q_0, F)$, $h(E) = L(M)$.

Method:

1. We assign numbers $1, 2, \dots, n$ to all occurrences of symbols from T in expression E. The resultant regular expression is denoted E'.
2. Set of symbols at the beginning $B = \{x_i : x \in T$, symbol x_i is the first symbol of some string in $h(E')\}$.
3. Set of neighbours $P = \{x_i y_j :$ symbols x_i and y_j occur next to each other in some string from $h(E')\}$.
4. Set of final symbols $F = \{x_i :$ symbol x_i is the last symbol of some string from $h(E)\} \cup \{q_0 : \varepsilon \in h(E)\}$.
5. $Q = \{q_0\} \cup \{x_i : x \in T, i \in \langle 1, n \rangle\}$.

Relationship between RE and FA

Algorithm (continued):

6. Mapping δ:
   (a) $\delta(q_0, x)$ contains x_i, $\forall x_i \in B$.
   (b) $\delta(x_i, y)$ contains y_j, $\forall x_i y_j \in P$
7. Set F is the set of final states. $\square$

Relationship between RE and FA

Example

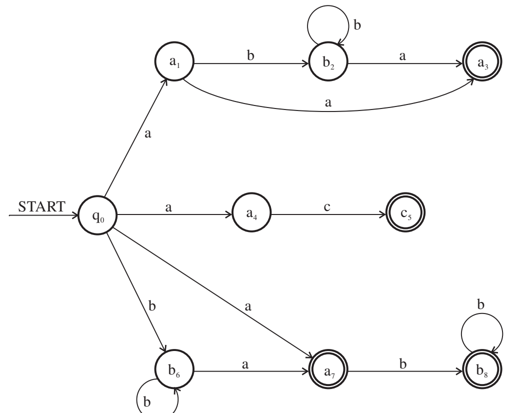
$E = ab^*a + ac + b^*ab^*$, $T = \{a, b, c\}$.

$E' = a_1 b_2^* a_3 + a_4 c_5 + b_6^* a_7 b_8^*$, $B = \{a_1, a_4, b_6, a_7\}$.

$P = \{a_1b_2, a_1a_3, b_2b_2, b_2a_3, a_4c_5, b_6b_6, b_6a_7, a_7b_8, b_8b_8\}$.

$F = \{a_3, c_5, a_7, b_8\}$.

$M = (\{q_0, a_1, b_2, a_3, a_4, c_5, b_6, a_7, b_8\}, \{a, b, c\}, \delta, q_0, \{a_3, c_5, a_7, b_8\})$



Relationship between RE and FA

Algorithm Construction of DFA for a given regular expression – method of derivatives, Janusz A. Brzozowski.
Input: Regular expression E over alphabet T.
Output: Finite automaton $M = (Q, T, \delta, q_0, F)$, $h(E) = L(M)$.
Method:

1. $Q = \{E\}, Q_0 = \{E\}, i := 1.$
2. $Q_i = \{\frac{dU}{da} : U \in Q_{i-1}, a \in T\} \setminus Q.$
3. If $Q_i \neq \emptyset$, $Q = Q \cup Q_i$, $i := i + 1$, go to step 2.
   If $Q_i = \emptyset$, we create automaton M thusly:
   $M = (Q, T, \delta, E, F)$,
   $\delta(\frac{dU}{dx}, a) = \frac{dU}{d(xa)}$.
   Set $F = \{\frac{dU}{dx} : \varepsilon \in h(\frac{dU}{dx})\}$. □

Relationship between RE and FA

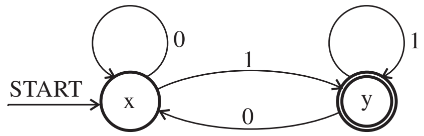
Example
$(0+1)^*1$

1. We set $Q = \{(0+1)^*1\}, Q_0 = \{(0+1)^*1\}$.
2. We compute Q_1:
   $\frac{d}{d1}((0+1)^*1) = (\emptyset + \varepsilon)(0+1)^*1 + \varepsilon = (0+1)^*1 + \varepsilon$
   $\frac{d}{d0}((0+1)^*1) = (\varepsilon + \emptyset)(0+1)^*1 + \emptyset = (0+1)^*1.$
   Since $\frac{d}{d0}((0+1)^*1) = (0+1)^*1$,
   $Q_1 = \{(0+1)^*1 + \varepsilon\}$ and $Q = \{(0+1)^*1, (0+1)^*1 + \varepsilon\}$.
3. We compute Q_2:
   $\frac{d}{d1}((0+1)^*1 + \varepsilon) = (\emptyset + \varepsilon)(0+1)^*1 + \varepsilon + \emptyset = (0+1)^*1 + \varepsilon$
   $\frac{d}{d0}((0+1)^*1 + \varepsilon) = (\varepsilon + \emptyset)(0+1)^*1 + \emptyset = (0+1)^*1.$
   Since both expressions are already in set Q, set Q_2 is empty.

Relationship between RE and FA

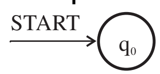
Example (continued)

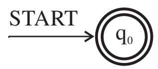
4. state $x = (0+1)^*1$,
   state $y = (0+1)^*1 + \varepsilon$,
   start state is x,
   final state is y, because $\varepsilon \in h((0+1)^*1 + \varepsilon)$
   $\frac{dx}{d1} = y$, $\frac{dx}{d0} = x$, $\frac{dy}{d1} = y$, $\frac{dy}{d0} = x$,

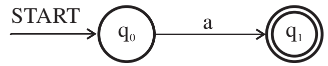


Relationship between RE and FA

Algorithm Construction of FA for a given regular expression – Thompson's method.
Input: Regular expression E over alphabet T.
Output: Finite automaton $M = (Q, T, \delta, q_0, F)$, $h(E) = L(M)$.
Method:

1. We create FA for elementary regular expressions:
   (a) For regular expression $E = \emptyset$:
   

   (b) For regular expression $E = \varepsilon$:
   

   (c) For regular expression $E = a$:
   

2. For parts of regular expression in form $E = E_1 + E_2,\ E = E_1E_2,\ E = E_1^*$ we create finite automata using appropriate algorithms.

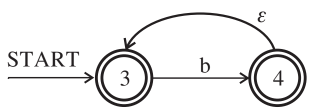
Relationship between RE and FA

Example

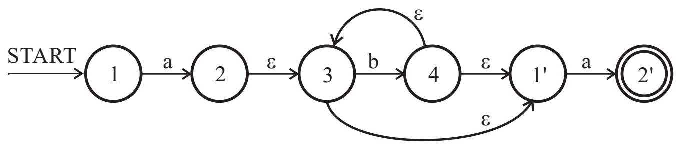
$E = ab^*a + ab$.

Necessary elementary expressions are:

We further create automaton for expression b^*:



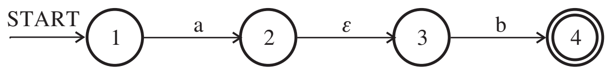
For expression ab^*a the FA is of the form:



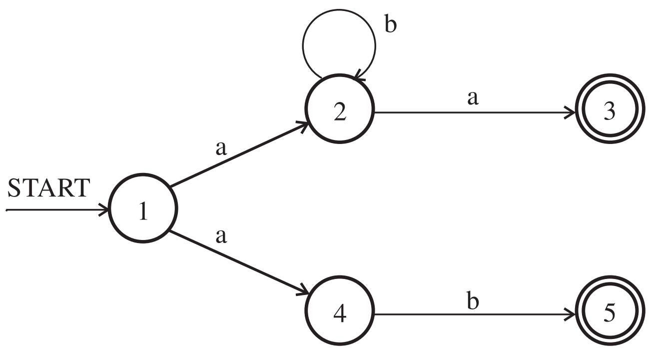
Relationship between RE and FA

Example (continued)

For expression ab the FA is of the form:



For expression $E = ab^*a + ab$:



Relationship between FA and RE

Theorem

For every finite automaton M a regular expression E can be constructed such that $L(M) = h(E)$. $\square$

Relationship between FA and RE

Definition

Extended finite automaton $M = (Q, T, \gamma, q_0, F)$, where
Q is a finite set of states,
T is a finite input alphabet,
γ is mapping from $Q \times Q$ into R_T,
$q_0 \in Q$ 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

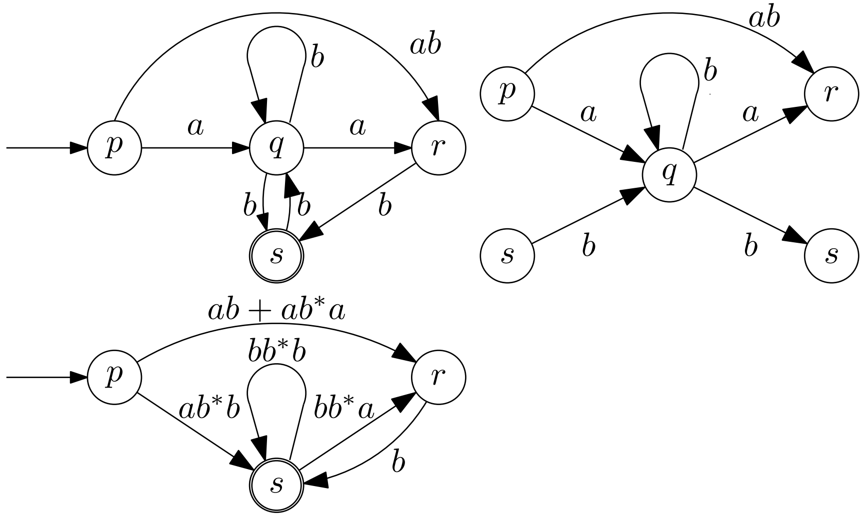
Language accepted by the extended FA M is $L(M) = \{x : x \in T^*,$
$x = x_1x_2 \dots x_n$ and there exists a sequence of states
$q_0, q_1, \dots, q_n,\ q_n \in F$ such that
$x_1 \in h(\gamma(q_0, q_1)),\ x_2 \in h(\gamma(q_1, q_2)), \dots,\ x_n \in h(\gamma(q_{n-1}, q_n))\}$.

Relationship between FA and RE

Theorem
Let $M = (Q, T, \gamma, q_0, F)$ be an extended finite automaton. Assume that q is neither a start state, nor a final state. Then the equivalent extended finite automaton is $M' = (Q \setminus \{q\}, T, \gamma', q_0, F)$, where mapping γ' is defined for every pair $p, r \in (Q \setminus \{q\})$ thusly:
$\gamma'(p, r) = \gamma(p, r) + \gamma(p, q)\gamma(q, q)^*\gamma(q, r)$. □

Relationship between FA and RE

Example



Relationship between FA and RE

Second term of expression
$\gamma'(p, r) = \gamma(p, r) + \gamma(p, q)\gamma(q, q)^*\gamma(q, r)$ will not apply if
$\gamma(p, q) = \emptyset$ or $\gamma(q, r) = \emptyset$.

Relationship between FA and RE

Algorithm Construction of regular expression for a given FA – elimination of states.
Input: Finite automaton $M = (Q, T, \delta, q_0, F)$.
Output: Regular expression E such that $h(E) = L(M)$.
Method:

1. We create $M_R = (Q, T, \gamma, q_0, F)$, where
   $\gamma(p, q) = \{a\},\ a \in T \cup \{\varepsilon\}$ if $\delta(p, a)$ contains q.
2. Automaton M_R will be extended:
   (a) If start state $q_0 \in F$ or $\gamma(q, q_0) \neq \emptyset$, then $Q = Q \cup \{q_0'\}$ a $\gamma(q_0', q_0) = \varepsilon$. q_0' is the start state.
   (b) If $|F| > 1$, then $Q = Q \cup \{f\}$ and $\gamma'(q, f) = \varepsilon, \forall q \in F$. Set of final states $F' = \{f\}$.

   After these modifications $M_R' = (Q', T, \gamma', q_0', F')$.

Relationship between FA and RE

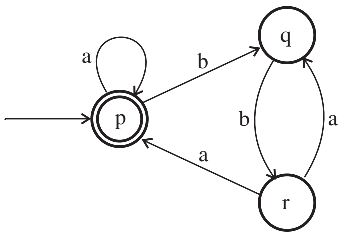
Algorithm (continued):

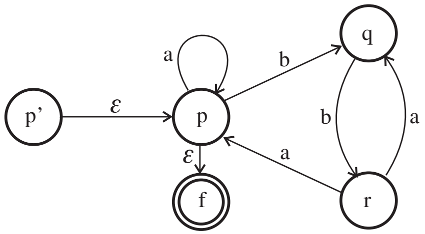
3. If $Q' = \{q'_0, f\}$, then regular expression is $\gamma'(q'_0, f)\gamma'(f, f)^*$. End.
   Else continue by step 4.
4. Choose $q \in Q', q \notin \{q'_0, f\}$. $Q' = Q' \setminus \{q\}$ and γ' are modified accordingly. Continue by step 3.

Relationship between FA and RE

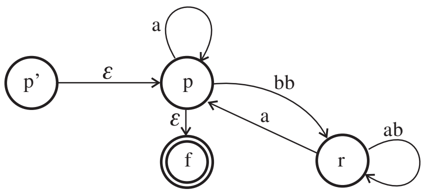
Example

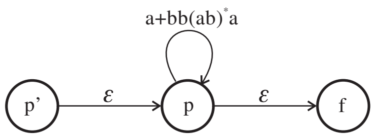
$M = (\{p, q, r\}, \{a, b\}, \delta, p, \{p\})$





We create the extended finite automaton and add a new initial and final state. In the next step we exclude states q and r.

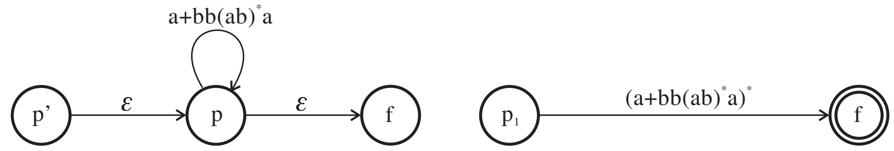




Relationship between FA and RE

Example (continued)

Eventually we exclude state p.



The resulting regular expression is $E = (a + bb(ab)^*a)^*$.

Relationship between FA and RE

Algorithm Construction of regular expression for a given finite automaton – solution of regular equations, incoming transitions

Input: Finite automaton $M = (Q, T, \delta, q_0, F)$.

Output: Regular expression E, $h(E) = L(M)$.

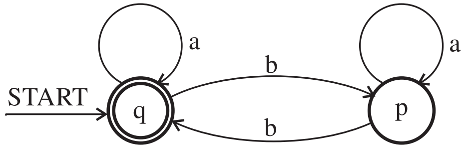
Method:

1. For every state q from Q:
   $X_q = X_{p_1}a_1 + X_{p_2}a_2 + \cdots + X_{p_n}a_n$, if $q \in \delta(p_i, a_i)$. In case that q is the start state, we add ε.
2. The system of left regular equations is solved through Gauss elimination.
3. Resulting regular expression:
   $E = X_{p_1} + X_{p_2} + ... + X_{p_n}$ if $p_i \in F,\ i = 1, 2, ..., n$.

Relationship between FA and RE

Example

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



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

$$X_p = X_q b a^*$$

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

Resulting regular expression: $E = X_q = (a + ba^*b)^*$

Relationship between FA and RE

Algorithm Construction of reg. expression for a given finite automaton – solution of regular equations, outgoing transitions

Input: Finite automaton $M = (Q, T, \delta, q_0, F)$.

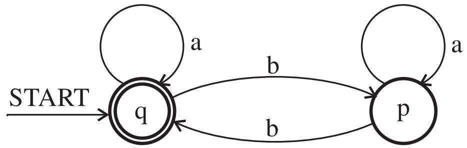
Output: Regular expression E, $h(E) = L(M)$

Method:

1. For each state q from Q:
   $X_q = a_1 X_{p_1} + a_2 X_{p_2} + ... + a_n X_{p_n}$, if $p_i \in \delta(q, a_i)$.
   In case that q is a final state, we add ε.
2. The system of right regular equations is solved through Gauss elimination.
3. Resulting regular expression is the expression for the initial state q_0 (for variable X_{q_0}).

Relationship between FA and RE

Example



$X_q = aX_q + bX_p + \varepsilon$

$X_p = aX_p + bX_q$

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

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

Relationship between FA and RE

Algorithm Construction of regular expression for a given finite automaton – integral method.

Input: Total DFA $M = (Q, T, \delta, q_0, F)$.

Output: Regular expression E, $h(E) = L(M)$.

Method:

1. For each state q we find the shortest string x_q for which there exists a path from state q_0 into state q; $x_{q_0} = \varepsilon$.
2. A sequence of expressions $\frac{dE}{dx_q} = Y_q$ ordered by the length of minimal strings x_q beginning with string x_{q_0}.
3. We create a tree for all derivatives by $x_q, q \in Q$:
   (a) $x_{q_o} = \varepsilon$: $E = \frac{dE}{d\varepsilon} = q_0$
   (b) $\forall a_i \in T, \forall x_q, q \in Q$, we add edges labeled by symbols a_i leading into nodes $Y_{qa_i} = \frac{dE}{dx_q a_i} = \delta(q, a_i)$
   If $\delta(q, a_i)$ is an error state, then $Y_{qa_i} = \emptyset$.

Relationship between FA and RE

Algorithm (continued):

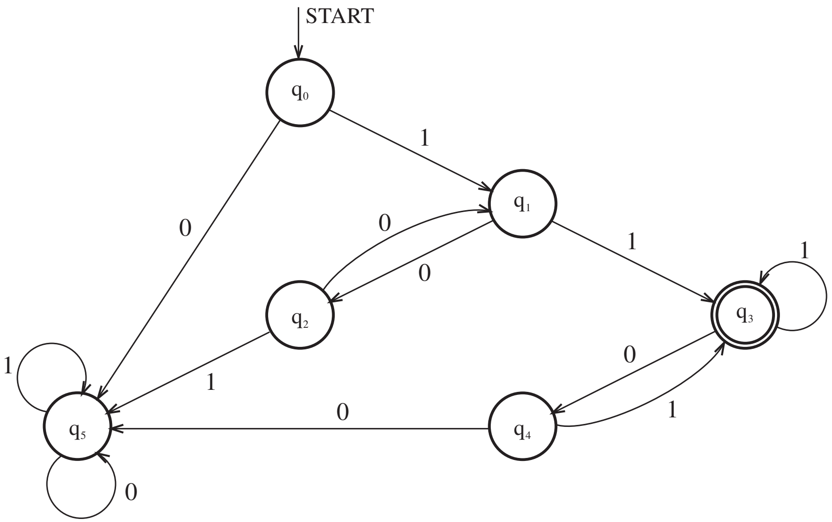
4. For the last minimal string in the ordering by the step 2 we perform integration of expressions by which the targets of the string are labeled.
5. If the node corresponding to state q_i has descendants for which we found in step 4 the values of integrals $p_1, p_2, \dots, p_k$ by $a_1, a_2, \dots, a_k$, then we construct the following equation:
   $q_i = a_1p_1 + a_2p_2 + \ldots + a_kp_k + \varepsilon$, if $q_i \in F$,
   $q_i = a_1p_1 + a_2p_2 + \ldots + a_kp_k$, otherwise.
   We solve this equation and remove nodes for which we performed the integration from the tree.
6. We repeat steps 4 and 5 for every minimal string by the ordering from step 2, until the integration is over.
7. The output is regular expression for the start state q_0. □

Relationship between FA and RE

Example

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, q_0, \delta, \{q_3\})$.

We add error state $q_\emptyset = q_5$.



Relationship between FA and RE

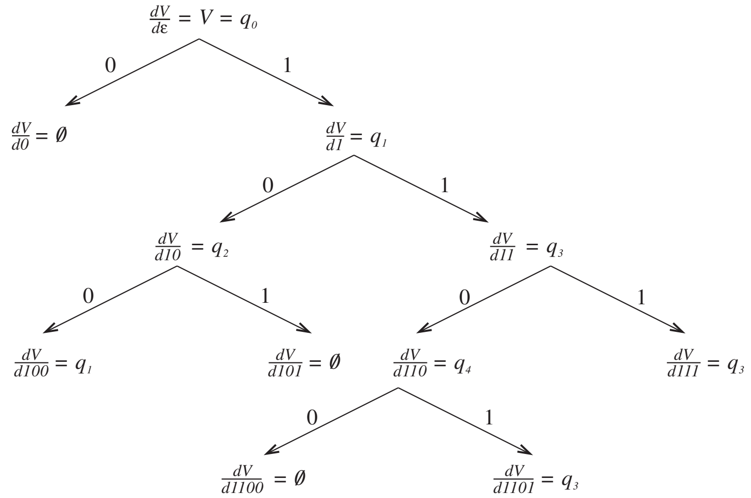
Example (continued)

1. Shortest strings for individual states are these:
   $x_{q_0} = \varepsilon$, $x_{q_1} = 1$,
   $x_{q_5} = 0$, $x_{q_2} = 10$,
   $x_{q_3} = 11$, $x_{q_4} = 110$.
2. This is the ordered sequence of expressions:
   $Y_{q_0} = \frac{dE}{d\varepsilon} = E$,
   $Y_{q_1} = \frac{dE}{d1}$,
   $Y_{q_5} = \frac{dE}{d0} = \emptyset$,
   $Y_{q_2} = \frac{dE}{d10}$,
   $Y_{q_3} = \frac{dE}{d11}$,
   $Y_{q_4} = \frac{dE}{d110}$.

Relationship between FA and RE

Example (continued) $Y_{q_0} = \frac{dE}{d\varepsilon} = E$, $Y_{q_1} = \frac{dE}{d1}$, $Y_{q_5} = \frac{dE}{d0} = \emptyset$, $Y_{q_2} = \frac{dE}{d10}$, $Y_{q_3} = \frac{dE}{d11}$, $Y_{q_4} = \frac{dE}{d110}$.

3. Tree constructed in step 3.



Relationship between FA and RE

Example (continued)

4. We integrate $\frac{dE}{d1100}$ a $\frac{dE}{d1101}$
   $\int \frac{dE}{d1100} d0 = \int \emptyset\, d0 = 0\emptyset = \emptyset,$
   $\int \frac{dE}{d1101} d1 = \int q_3 d1 = 1q_3.$
5. The equation has the form: $q_4 = 1q_3 + \emptyset$
   We further repeat steps 4 and 5:
   $\int \frac{dE}{d110} d0 = \int q_4 d0 = \int 1q_3 d0 = 01q_3,$
   $\int \frac{dE}{d111} d1 = \int q_3 d1 = 1q_3,$
   $q_3 = 01q_3 + 1q_3 + \varepsilon = (01+1)q_3 + \varepsilon = (01+1)^*,$
   $\int \frac{dE}{d100} d0 = \int q_1 d0 = 0q_1$
   $\int \frac{dE}{d101} d1 = \int \emptyset d1 = 1\emptyset = \emptyset,$
   $q_2 = 0q_1 + \emptyset$

Relationship between FA and RE

Example (continued)

$\int \frac{dE}{d10} d0 = \int q_2 d0 = \int 0q_1 d0 = 00q_1$
$\int \frac{dE}{d11} d1 = \int q_3 d1 = \int (01+1)^* d1 = 1(01+1)^*$
$q_1 = 00q_1 + 1(01+1)^* = (00)^*1(01+1)^*$
$\int \frac{dE}{d0} d0 = \int \emptyset d0 = 0\emptyset = \emptyset$
$\int \frac{dE}{d1} d1 = \int q_1 d1 = \int (00)^*(01+1)^* d1 = 1(00)^*1(01+1)^*$
$q_0 = 1(00)^*1(01+1)^* + \emptyset$

6. $L(M) = h(E) = 1(00)^*1(01+1)^*$

Relationship between RG and RE

Theorem
For every regular grammar $G = (N, T, P, S)$ a regular expression E such that $L(G) = h(E)$ can be constructed.

Definition
Extended (right) regular grammar is the quadruple $G = (N, T, P, S)$, where
N is the finite set of nonterminal symbols,
T 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 there are at most two production rules of the form $A \rightarrow \alpha B$ or $A \rightarrow \alpha$ in the extended regular grammar.
Modification: The pair of rules $A \rightarrow \alpha B,\ A \rightarrow \beta B$ is replaced by rule $A \rightarrow (\alpha + \beta)B$.

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

Relationship between RG and RE

Theorem
Let $G = (N, T, P, S)$ be an extended regular grammar. Assume that the nonterminal symbol A is not a start symbol. Then the equivalent extended regular grammar is $G' = (N \setminus \{A\}, T, P', S)$, where rules in P' are formed thusly:
If there are rules in G:

$B \to \alpha_1 C$

$B \to \alpha_2 A$

$A \to \alpha_3 A$

$A \to \alpha_4 C$,

then the following rules are in P' of G':

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

Relationship between RG and RE

Note:
If some of the given rules does not occur in grammar G, then it is replaced by rule $X \to \emptyset Y$, where $X \in \{A, B, C\},\ Y \in \{A, C\}$.

The corresponding rules will not occur in G' either.

For example if G misses a rule $A \to \alpha_4 C$, we replace it with a rule $A \to \emptyset C$ and then rule $B \to (\alpha_1 + \alpha_2 \alpha_3^* \emptyset)C$ degenerates to rule $B \to \alpha_1 C$.

Relationship between RG and RE

Algorithm Construction of regular expression for a given regular grammar – elimination of nonterminal symbols.
Input: Right regular grammar $G = (N, T, P, S)$.
Output: Regular expression E such that $h(E) = L(G)$.
Method:

1. Extended right regular grammar $G_R = (N, T, P_R, S)$ equivalent with G. All n-tuples of rules in the form $A \to \alpha_1 B, A \to \alpha_2 B, \ldots, A \to \alpha_n B$ are replaced by the rule
   $A \to (\alpha_1 + \alpha_2 + \ldots + \alpha_n)B$, where $A \in N,\ B \in N \cup \{\varepsilon\}$.

Relationship between RG and RE

Algorithm (continued):

2. Grammar G_R is further extended thusly:
   (a) new start symbol $S' \notin N$ and we add $S' \to S$,
   (b) new final nonterminal symbol $F \notin N$,
   all rules $A \to \alpha$ are replaced by rules $A \to \alpha F$ and we add $F \to \varepsilon$.
   $G'_R = (N', T, P'_R, S')$, where $N' = N \cup \{S', F\}$,
   $P'_R = \{A \to \alpha B : A \to \alpha B \in P_R\}$
   $\cup \{A \to \alpha F : A \to \alpha \in P_R\} \cup \{S' \to S,\ F \to \varepsilon\}$.
3. If $N' = \{S'\}$ and $P'_R = \{S' \to \alpha F, F \to \varepsilon\}$, then $E = \alpha$ and the algorithm ends. Otherwise it continues by step 4.
4. We choose $A \in N'$ such that $A \notin S'$. $N' = N' \setminus \{A\}$ and rules P_R are modified correspondingly.
   We continue by step 3. $\square$

Relationship between RG and RE

Example

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

$S \rightarrow bA \mid aS \mid \varepsilon$

$A \rightarrow bB$

$B \rightarrow aA \mid aS.$

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

$S \rightarrow \varepsilon$ is replaced by rule $S \rightarrow F$ and for F we add rule $F \rightarrow \varepsilon$.

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

$S' \rightarrow S$

$S \rightarrow bA \mid aS \mid F$

$A \rightarrow bB$

$B \rightarrow aA \mid aS$

$F \rightarrow \varepsilon.$

Relationship between RG and RE

Algorithm Construction of regular expression for the given regular grammar

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

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

Method:

1. For every nonterminal symbol from N we construct a regular equation.
2. The resulting system of regular equations is solved using Gauss elimination for the start symbol of the grammar S. □

Relationship between RG and RE

Example (continued)

We exclude symbol A. $G^1_R = (\{S', S, B, F\}, \{a, b\}, P^1, S')$:

$S' \rightarrow S$

$S \rightarrow bbB \mid aS \mid F$

$B \rightarrow abB \mid aS$

$F \rightarrow \varepsilon.$

We exclude symbol B. $G^2_R = (\{S', S, F\}, \{a, b\}, P^2, S')$:

$S' \rightarrow S$

$S \rightarrow (a + bb(ab)^*a)S \mid F$

$F \rightarrow \varepsilon.$

We exclude symbol S. $S' \rightarrow (a + bb(ab)^*a)^*F$
$F \rightarrow \varepsilon.$

$E = (a + bb(ab)^*a)^*$

Relationship between RG and RE

Example

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

$S \rightarrow 0S \mid 1A \mid 1$

$A \rightarrow 1S \mid 0A \mid 0$

The system of regular equations has the following form:

$S = 0S + 1A + 1$

$A = 1S + 0A + 0$

We solve the system by elimination:

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

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

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

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

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

Resulting regular expression describing language L(G) has the form:

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

Relationship between RE and RG

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

Relationship between RE and RG

Algorithm Construction of right regular grammar for a given regular expression – iterative construction.
Input: Regular expression E over alphabet T.
Output: $G = (N, T, P, S)$, $L(G) = h(E)$
Method:

1. $\forall a \in T$ we create grammars:
   $G_a = (\{A\}, \{a\}, \{A \to a\}, A)$.
2. We create grammar $G_\varepsilon = (\{E\}, \emptyset, \{E \to \varepsilon\}, E)$.
3. Grammars for subexpressions of the form $x_1 + x_2$, x_1x_2, x_1^*, if we already have grammars for subexpressions x_1, x_2.
4. We exclude ε-rules and simple rules.

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. Iteratively we construct grammars for languages that are values of expressions $ab, ab + \varepsilon, (ab + \varepsilon)^*$:
   $G(ab) = (\{A, B\}, \{a, b\}, \{A \to aB, B \to b\}, A)$
   $G(ab + \varepsilon) = (\{N_1, A, B, E\}, \{a, b\}, P_1, N_1)$, where P_1:
   $N_1 \to A \mid E$, $E \to \varepsilon$, $A \to aB$, $B \to b$.
   $G((ab + \varepsilon)^*) = (\{N_2, N_1, A, B, E\}, \{a, b\}, P_2, N_2)$, where P_2:
   $N_2 \to N_1 \mid \varepsilon$, $N_1 \to A \mid E$, $E \to N_2$, $A \to aB$, $B \to bN_2$.

Relationship between RE and RG

Example (continued)

4. We exclude ε-rules $\Big(N_\varepsilon = \{N_2, E, N_1\}\Big)$:
   $N_2' \to N_2 \mid \varepsilon$, $N_2 \to N_1$, $N_1 \to A \mid E$, $E \to N_2$, $A \to aB$,
   $B \to bN_2 \mid b$.
   Further we exclude simple rules:
   $N_{N2'} = \{N_2', N_2, N_1, A, E\}$, $N_{N2} = \{N_2, N_1, A, E\}$,
   $N_{N1} = \{A, E, N_2, N_1\}$, $N_E = \{E, N_2, N_1, A\}$, $N_A = \{A\}$,
   $N_B = \{B\}$ and we get rules:
   $N_2' \to \varepsilon \mid aB$, $N_2 \to aB$, $N_1 \to aB$, $E \to aB$, $A \to aB$,
   $B \to bN_2 \mid b$.

Symbols N_1, E, A are redundant:
$G = (\{N_2', N_2, B\}, \{a, b\}, P, N_2')$, where P:
$N_2' \to \varepsilon \mid aB$, $B \to bN_2 \mid b$, $N_2 \to aB$

Relationship between RE and RG

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

Input: Regular expression E over alphabet T.

Output: $G = (N, T, P, S)$, $L(G) = h(E)$.

Method:

1. $N = \{E\}, N_0 = \{E\}, i = 1$.
2. We create derivatives of all expressions from N_{i-1} by all alphabet symbols from alphabet T. Into the set N_i we put all expressions created through derivatives of expressions from N_{i-1}.
3. If $N_i \neq \emptyset$, we add N_i into N, $i = i + 1$ and we go to st. 2. Otherwise we create gr. $G = (N, T, P, E)$, where P:

   We add $\frac{dE}{dx} \rightarrow a\frac{dE}{d(xa)}$ into P.

   We add $\frac{dE}{dx} \rightarrow a$ into P in case that $\varepsilon \in h\left(\frac{dE}{d(xa)}\right)$.

   We add $E \rightarrow \varepsilon$ into P in case that $\varepsilon \in h(E)$.

Relationship between RE and RG

Example

$(ab + \varepsilon)^*$

1. $N = \{(ab + \varepsilon)^*\}, N_0 = \{(ab + \varepsilon)^*\}, i = 1.$
2. $\frac{d(ab+\varepsilon)^*}{da}$ =$(b + \emptyset)(ab + \varepsilon)^* = b(ab + \varepsilon)^*$

   $\frac{d(ab+\varepsilon)^*}{db}$ = $(\emptyset + \emptyset)(ab + \varepsilon)^* = \emptyset$

   $N_1 = \{b(ab + \varepsilon)^*\}, N = \{(ab + \varepsilon)^*, b(ab + \varepsilon)^*\}.$
3. $\frac{d(b(ab+\varepsilon)^*)}{da}$ = $\emptyset$

   $\frac{d(b(ab+\varepsilon)^*)}{db}$ = $\varepsilon(ab + \varepsilon)^* = (ab + \varepsilon)^*$

   $N_2 = \emptyset, N = \{(ab + \varepsilon)^*, b(ab + \varepsilon)^*\}$

If we label $A = (ab + \varepsilon)^*$, $B = b(ab + \varepsilon)^*$, we get grammar $G = (\{A, B\}, \{a, b\}, P, A)$, where P contains rules:

$$A \rightarrow aB \mid \varepsilon$$
$$B \rightarrow bA \mid b$$