

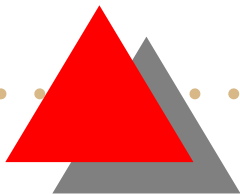


Regular Expressions

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science



Expressions

- **In arithmetic:**

1. expressions are constructed from numbers and variables using arithmetic operations and parentheses;
2. expressions represent computations with numbers;
3. results of expressions evaluation are numbers.

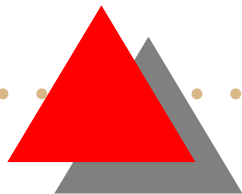
- **In automata theory:**

1. regular expressions are constructed from regular languages using regular operations and parentheses;
2. regular expressions represent computations with regular languages;
3. result of regular expressions evaluation are regular languages.



Example

- $(5 + 3) \times 4$ is an arithmetic expression constructed with operations $+$ and \times ; its value is the number 32.
- $(0 \cup 1)0^*$ is a regular expressions constructed from the languages 0 and 1 using regular operations; it evaluates to the language that contains all strings that start with 0 or 1 followed by zero or more 0-s.





Regular expression evaluation

Similar to the evaluation of arithmetic expression:

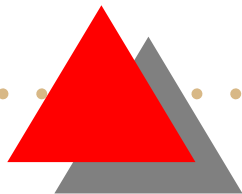
- Identify first the constant languages involved.

Example: 0 represents the language $\{0\}$ and 1 represents the language $\{1\}$

- Apply the regular operations on the languages thus identified following their set-theory definitions.

Example: $(0 \cup 1)$ represents the language $\{0, 1\}$; 0^* represents the language $\{\epsilon, 0, 00, \dots\}$ and $(0 \cup 1)0^*$ represents the language $\{0, 1, 00, 10, 000, 100, \dots\}$

Note: symbol \circ of concatenation operation is not written in regular expressions (convention).

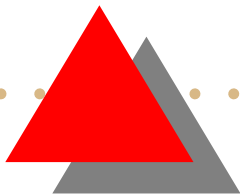




Applications

1. Regular expressions provide a powerful method to describe patterns searched in various texts;
2. Utilities such as AWK, GREP in Unix, modern programming languages, such as PERL, text editors, all use regular expressions for their pattern description;
3. The lexicon of programming languages is specified by regular expressions. Hence, lexical analysis is done using regular expressions.

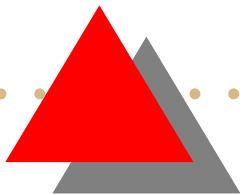
Other examples?





For a given alphabet Σ

- The regular expression Σ describes the language consisting of all strings of length 1 over Σ ;
- Σ^* describes the language of all strings over the alphabet Σ ;
- Σ^*1 is the language of all strings over Σ that ends in 1;
- The language $(0\Sigma^*) \cup (\Sigma^*1)$ describes the language that consists of all strings that either start with 0 or end in 1.





Precedence relation

Denoting by \preceq the rule describing the order of operations in the evaluation of arithmetic and regular expressions we have:

- In arithmetic expressions: $() \preceq \{\times, /\} \preceq \{+, -\}$
- In regular expressions: $() \preceq \{*\} \preceq \{\cup, \circ\}$

Note: $()$ denotes expressions enclosed in parentheses.



Formal definition

Consider Σ an alphabet. A regular expression R over Σ is defined recursively by the rules:

Recursion basis:

1. For any $a \in \Sigma$, a is a regular expression describing the language $L(a) = \{a\}$
2. ϵ is a regular expression describing the language $L(\epsilon) = \{\epsilon\}$
3. \emptyset is a regular expression describing the empty language $L(\emptyset) = \emptyset$.

Formal definition (continuation)

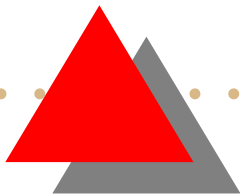
Recursion body:

4. If R_1 and R_2 are regular expressions evaluating to the languages $L(R_1)$ and $L(R_2)$ then:
 - (a) $(R_1 \cup R_2)$ is a regular expression that evaluates to the language $L(R_1) \cup L(R_2)$;
 - (b) $(R_1 \circ R_2)$ is a regular expression that evaluates to the language $L(R_1) \circ L(R_2)$;
 - (c) R_1^* is a regular expressions that evaluates to the language $\bigcup_{i \geq 0} L(R_1)^i$ where $L(R_1)^i = L(R_1)^{i-1} \circ L(R_1)$ and $L(R_1)^0 = \epsilon$.



Observation

A definition of the form expressed by rule (4) above is called an *inductive definition*.





Potential confusions

- Do not confuse regular expressions ϵ and \emptyset :
 1. ϵ represent the language that contains just one element, the empty string;
 2. \emptyset is the regular expression that represent the language with no elements, the empty language.
- Note the distinction between R and $L(R)$. R is an expression and $L(R)$ is the set of strings specified by R .

Example regular expressions

1. 0^*10^* , $L(0^*10^*) = \{w \mid w \text{ contains exactly a single } 1\}$;
2. $\Sigma^*1\Sigma^*$, $L(\Sigma^*1\Sigma^*) = \{w \mid w \text{ contains at least one } 1\}$;
3. $\Sigma^*001\Sigma^*$, $L(\Sigma^*001\Sigma^*) = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$;
4. $(\Sigma\Sigma)^*$, $L(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$;
5. $(\Sigma\Sigma\Sigma)^*$, $L(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of three}\}$;
6. $01 \cup 01$, $L(01 \cup 01) = \{01, 01\}$; (Is this right?)
7. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$, $L(0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1) = \{w \mid w \text{ starts and ends with the same symbol}\}$;
8. $(0 \cup \epsilon)1^*$, $L((0 \cup \epsilon)1^*) = \{01^* \cup 1^*\}$;
9. $(0 \cup \epsilon)(1 \cup \epsilon)$, $L((0 \cup \epsilon)(1 \cup \epsilon)) = \{\epsilon, 0, 1, 01\}$.



More example

10. $1^*\emptyset, L(1^*\emptyset) = \emptyset;$

Note: concatenating the empty set to any set yields the empty set.

11. $\emptyset^*, L(\emptyset^*) = \{\epsilon\};$

Note: by definition, ϵ is in the star operation applied on any language. If that language has no strings, ϵ becomes the only element of the resulting language.



Identities

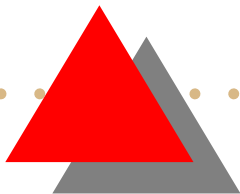
If R is a regular expression then the following identities take place:

- $L(R \cup \emptyset) = L(R)$;

Note: adding the empty language to any other language will not change that language.

- $L(R \circ \epsilon) = L(R)$;

Note: concatenating any string with the empty string does not change that string.



Facts

1. $L(R \cup \epsilon) \neq L(R)$;

Example: if $R = 0$ then $L(R) = 0$; $L(R \cup \epsilon) = \{0, \epsilon\}$.

2. $L(R \circ \emptyset) \neq L(R)$;

Example: if $R = 0$ then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$.



Application

- Regular expressions are useful tools for the design of programming languages and compilers;
- Lexicons of programming languages are described by regular expressions;

Example: numerical constants can be described by:

$\{+, -, \epsilon\}(DD^* \cup DD^*.D^* \cup D.DD^*)$ where

$D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

- From the lexicon description by regular expressions one can generate automatically lexical analyzers.

Terminology

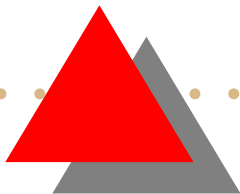
A regular expression R that evaluates to the language $L(R)$ is also said that **specifies the language $L(R)$** .



Equivalence with FA

Regular expressions and finite automata are equivalent in their descriptive power. That is:

1. Any regular expression E can be converted into a finite automaton, A_E , that recognizes the language specified by E ;
2. Vice-versa, any finite automaton recognizing a language A can be converted into a regular expression E_A that specifies the language A .





Theorem 1.54

A language is regular iff some regular expression specifies it.

Proof idea: this proof has two parts:

- **First part:** we show that a language specified by a regular expression is regular, i.e., there is a finite automaton that recognizes it.
- **Second part:** we show that if a language is regular then there is a regular expression that specifies it.

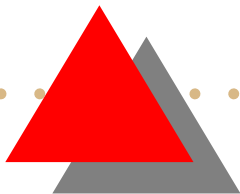


Lemma 1.55

If a language is specified by a regular expression, then it is regular.

Proof idea: Assume that we have a regular expression R that evaluate to the language A .

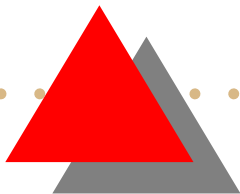
1. We will show how to convert R into an NFA that recognizes A .
2. Then by corollary 1.40, if an NFA recognizes A then A is regular.





Proof

Convert R into an NFA N by the following six-step procedure:



Step 1:

If $R = a \in \Sigma$, then $L(R) = \{a\}$ and the NFA N recognizing $L(R)$ is in Figure 1

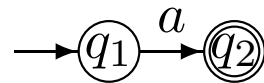


Figure 1: NFA N recognizing $\{a\}$

Note: this is an NFA but not a DFA because it has states with no exiting arrow for each possible input symbol.



Formal construction

Formally $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ where:

$$\delta(q_1, a) = \{q_2\},$$

$\delta(r, b) = \emptyset$ for $r \in \{q_1, q_2\}$, $r \neq q_1$ and $b \in \Sigma$, $b \neq a$.

Step 2:

If $R = \epsilon$ then $L(R) = \{\epsilon\}$ and the NFA N that recognizes $L(R)$ is in Figure 2

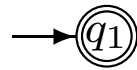


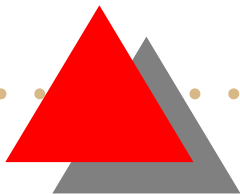
Figure 2: The NFA N recognizing $\{\epsilon\}$



Formal construction

Formally $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ where:

$\delta(r, b) = \emptyset$ for any $r \in \{q_1\}$ and $b \in \Sigma$



Step 3:

If $R = \emptyset$ then $L(R) = \emptyset$, and the NFA N that recognizes $L(R)$ is in Figure 3

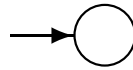


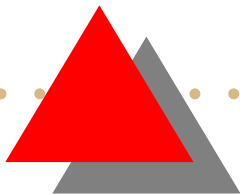
Figure 3: The NFA N recognizing \emptyset



Formal construction

Formally $N = (\{q\}, \Sigma, \delta, q, \emptyset)$ where:

$\delta(r, b) = \emptyset$ for any $r \in \{q\}$ and $b \in \Sigma$





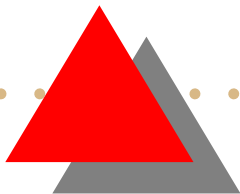
Step 4:

If $R = R_1 \cup R_2$ then $L(R) = L(R_1) \cup L(R_2)$.

Note: in view with the inductive nature of R we may assume that:

1. $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is an NFA that recognizes $L(R_1)$;
2. $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ is an NFA that recognizes $L(R_2)$.

The NFA N recognizes $L(R_1 \cup R_2)$ is given in Figure 4.



NFA recognizing $L(R_1) \cup L(R_2)$

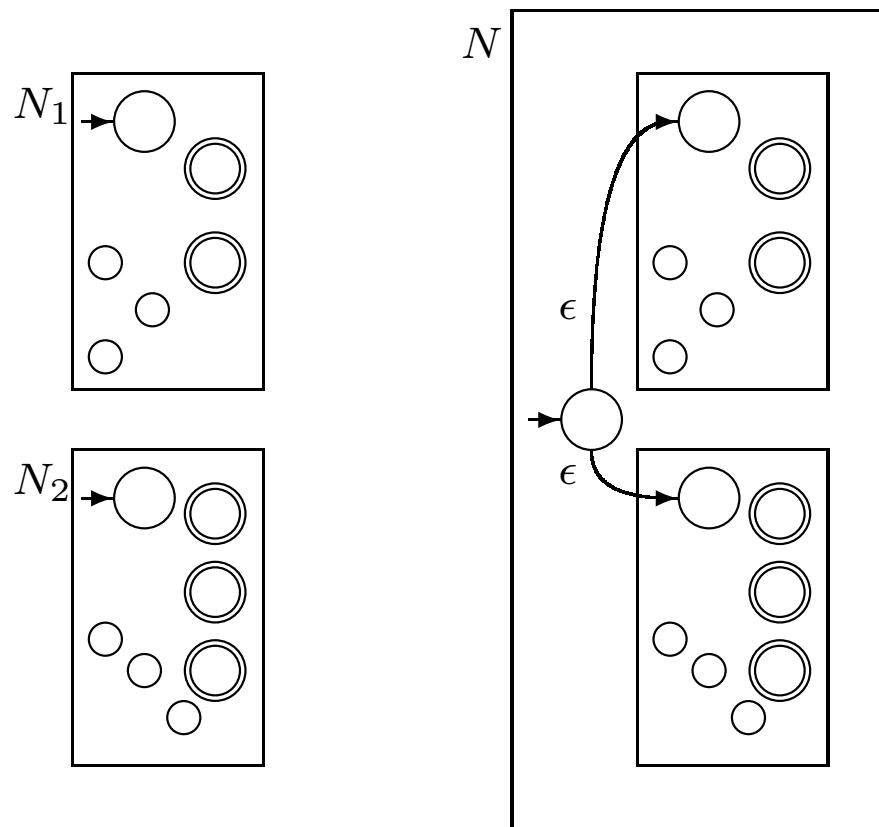


Figure 4: Construction of N to recognize $L(R_1 \cup R_2)$

Construction procedure

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$: That is, the states of N are all states on N_1 and N_2 with the addition of a new state q_0 ;
2. The start state of N is q_0 ;
3. The accept states of N are $F = F_1 \cup F_2$: that is, the accept states of N are all the accept states of N_1 and N_2 ;
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \\ \delta_2(q, a), & \text{if } q \in Q_2 \\ \{q_1, q_2\}, & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset, & \text{if } q = q_0 \text{ and } a \neq \epsilon. \end{cases}$$



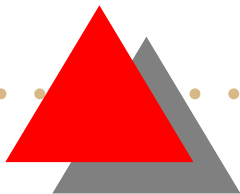
Step 5:

If $R = R_1 \circ R_2$ then $L(R) = L(R_1) \circ L(R_2)$.

Note: in view with the inductive nature of R we may assume that:

1. $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is an NFA that recognizes $L(R_1)$;
2. $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ is an NFA that recognizes $L(R_2)$.

The NFA N that recognizes $L(R_1 \circ R_2)$ is given in Figure 5.



NFA recognizing $L(R_1) \circ L(R_2)$

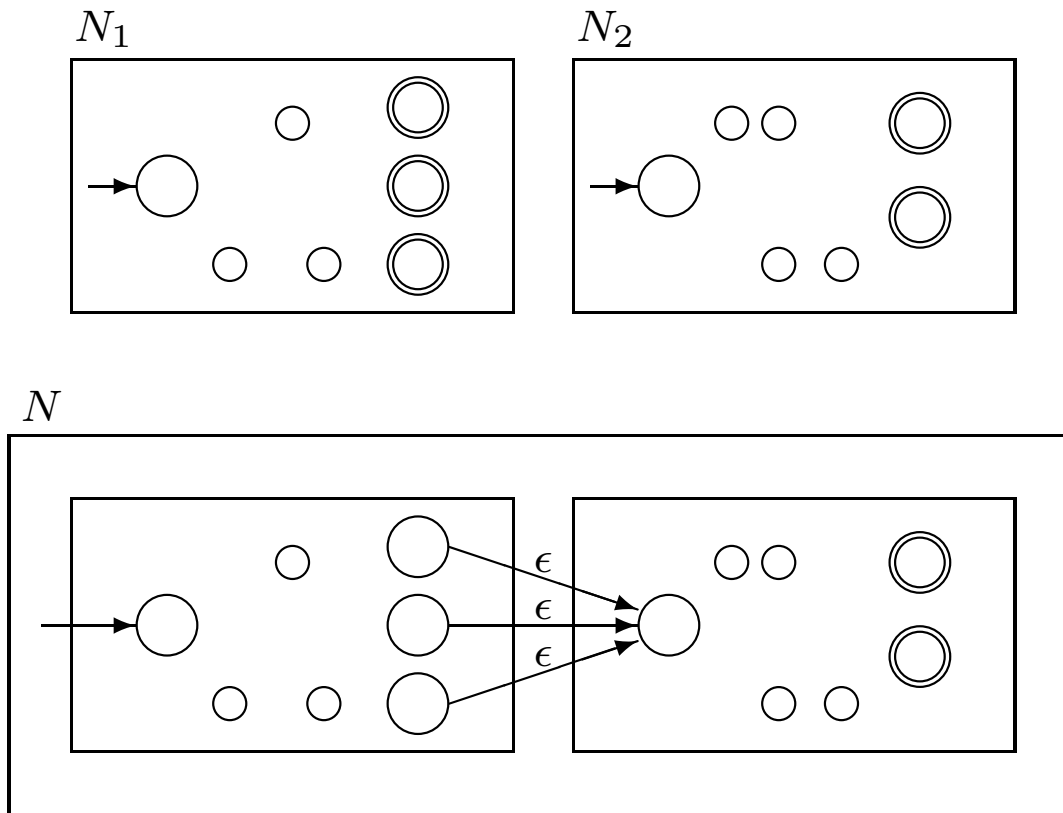


Figure 5: Construction of N to recognize $L(R_1 \circ R_2)$

Construction procedure

1. $Q = Q_1 \cup Q_2$. The states of N are all states of N_1 and N_2 ;
2. The start state is the state q_1 of N_1 ;
3. The accept states is the set F_2 of the accept states of N_2 ;
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & \text{if } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\}, & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a), & \text{if } q \in Q_2. \end{cases}$$



Step 6:

If $R = R_1^*$ then $L(R) = \cup_{i \geq 0} L(R_1)^i$.

Note: in view with the inductive nature of R we may assume that:

1. $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is an NFA that recognizes $L(R_1)$.

The NFA N that recognizes $L(R_1^*)$ is given in Figure 6.

NFA recognizing $L(R_1^*)$

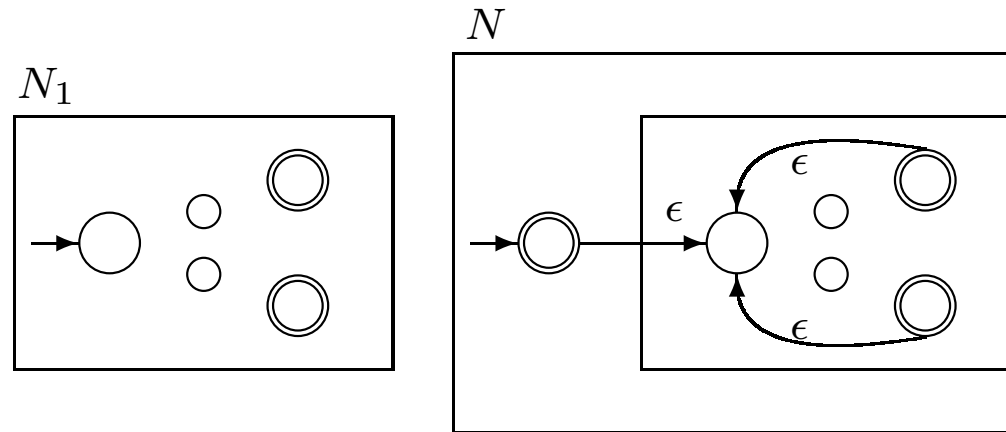


Figure 6: Construction of N to recognize $L(R_1^*)$

Construction procedure

1. $Q = \{q_0\} \cup Q_1$;
that is, states of N are the states of N_1 plus a new state q_0 .
2. Start state of N is q_0 ;
3. $F = \{q_0\} \cup F_1$;
that is, the accept states of N are the accept states of N_1 plus the new start state.
4. Define δ so that for any $q \in Q$ and $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & \text{if } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_1\}, & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \{q_1\}, & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset, & \text{if } q = q_0 \text{ and } a \neq \epsilon. \end{cases}$$



Examples conversion

Convert the following regular expressions into NFA-s following the procedure presented above:

1. $(ab \cup a)^*$;
2. $(a \cup b)^*aba$.

Hint: evaluate regular expressions and at each step of evaluation construct the NFA that recognizes the resulting language from the NFA-s that recognize the languages specified by the expression components.