SAPIENZA
Università di Roma
Dipartimento di Informatica

FOUNDATIONS OF COMPUTER SCIENCE
LECTURE 3: Regular expressions

Prof. Daniele Gorla

1

Expressions for denoting regular languages

SAPIENZA Università di Roma Dipartimento di Informatica

- Regular languages are closed under union, concatenation and star
- Here, we prove that, given an alphabet Σ, all regular languages over it can be expressed just by these three operations
- This suggests that we can use expressions built from characters of Σ and from the three regular operations to denote regular languages
- We shall also prove that all three operations are required for obtaining these languages
- This is another characterization of such languages, that are:
  - Those accepted by a DFA
  - Those accepted by a NFA
  - Those corresponding to a regular expression
- In the next class, we shall see a last characterization, based on so called *regular grammars*

2

2

Regular expressions

SAPIENZA Università di Roma Dipartimento di Informatica

DEFINITION

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

The order of operations is
* Star
* Concatenation
* Union

- The expression ε represents the language containing a single string (the empty one) whereas ∅ represents the language that doesn't contain any string
- The star operation is done first, followed by concatenation, and finally union, unless parentheses change this order
  → EX.: $a \cup b \circ c^*$ actually denotes $a \cup (b \circ (c^*))$
- We shall usually omit ∘ and write bc instead of $b \circ c$; furthermore, sometimes ∪ is denoted by + (so write $a+bc^*$ instead of $a \cup b \circ c^*$)
- R^+ is a shorthand for RR^* (hence, $R^* = R^+ \cup \varepsilon$)
- R^k denotes the concatenation of k copies of R's

3

3

Language of a regular expression

SAPIENZA Università di Roma Dipartimento di Informatica

It is straightforward to associate a language to a regular expression:

$L(a) = \{a\}$ $\quad L(\varepsilon) = \{\varepsilon\}$ $\quad L(\emptyset) = \emptyset$

$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$ $\quad L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$ $\quad L(R^*) = (\, L(R)\,)^*$

EXAMPLES: Take $\Sigma = \{0,1\}$. With abuse of notation, we write Σ to denote the r.e. $(0 \cup 1)$

- $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
- $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
- $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$
- $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of } 3\}$
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
  This distributivity law recalls distributivity of + over · in arithmetics; for this reason, union of regular expressions is usually denoted with +
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$

4

4

A few algebraic laws

SAPIENZA Università di Roma Dipartimento di Informatica

- $(R_1 \cup R_2)\, R_3 \;=\; R_1R_3 \cup R_2R_3$ $R_1\, (R_2 \cup R_3) = R_1R_2 \cup R_1R_3$
  → *distributivity* of $\cup$ over $\circ$
- $(R_1 \cup R_2) \cup R_3 \;=\; R_1 \cup (R_2 \cup R_3)$ $(R_1 \circ R_2) \circ R_3 \;=\; R_1 \circ (R_2 \circ R_3)$
  → *associativity* of $\cup$ and $\circ$
- $R \cup \emptyset \;=\; \emptyset \cup R \;= R$
  → $\emptyset$ is the *neutral element* of $\cup$
- $R \circ \varepsilon \;=\; \varepsilon \circ R \;= R$
  → ε is the *neutral element* of $\circ$
- $R \circ \emptyset = R \circ \emptyset = \emptyset$
  → $\emptyset$ is the *annihilator element* for concatenation
- $R_1 \cup R_2 = R_2 \cup R_1$
  → commutativity of $\cup$

These laws are natural, if you think at union as sum and concatenation as product in the algebra of reals → ATTENTION: $R_1 \circ R_2 \neq R_2 \circ R_1$

5

5

An example: syntax of numbers

SAPIENZA Università di Roma Dipartimento di Informatica

- Regular expressions are useful tools in the design of compilers for programming languages to describe basic objects such as variable names, constants, …
- For example, a numerical constant that may include a fractional part and/or a sign may be described by the regular expression

denote rationals with a , (or .)

assuming that is positive

$$(+ \cup - \cup \varepsilon)\ (D^{+} \cup D^{+}.D^{*} \cup D^{*}.D^{+})$$

nonempty sequence of digits

where $D = \{0,1,2,3,4,5,6,7,8,9\}$ is the alphabet of decimal digits.

- Examples of strings in the language generated by this expression are:
  72 3.14159 +7. -.01

characters:
\+ one or more of the preceding char or group
* zero or more of the preceding char or group
? zero or one of the preceding char or group
| replaces what is on the left or what is on the right
() defines a group

6

6

an example of regexp is:

ab+a

When you read a regular expression, the characters that appear must be matched exactly in the order specified

So, in this case the string must start with the letter a

The next character, the letter b, must also be matched; but it is followed by a metacharacter (one with special meaning)
The metacharacter + is a quantifier; it denotes 1 or more of the preceding character or group of characters

So, after the first letter, a, there must be one or more occurrences of the letter b

The final character is the letter a; it is not followed by a quantifier so this must be matched exactly
Therefore, the string must end with the letter a

examples of strings ab+a are aba, abba, abbba...

(+ U - U + eps) è un gruppo che definisce il segno (o negativo, o positivo o niente)
il resto è numero intero non vuoto con un numero in D concatenato a D stesso (insomma non vuoto) o numero in D concatenato a D stesso punto combinazione di tutto D (esempio 15.954) o il contrario (per ammettere anche il numero vuoto, il niente)
I gruppi sono poi concatenati

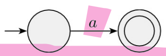
Equivalence with NFA (1)

SAPIENZA Università di Roma Dipartimento di Informatica

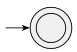
Thm.1: for every regular expression R there exists a NFA N such that $L(R) = L(N)$

PROOF Let's convert R into an NFA N. We consider the six cases in the formal definition of regular expressions.

1. $R = a$ for some $a \in \Sigma$. Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.

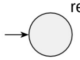


2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.



3. $R = \emptyset$. Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.

remember that the empty set is different from the eps



4. $R = R_1 \cup R_2$.
5. $R = R_1 \circ R_2$.
6. $R = R_1^*$.

we use the constructions given in the proofs that the class of regular languages is closed under the regular operations.

and by considering single operand

Q.E.D.

7

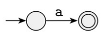
7

From a Reg. Expr. To a NFA: an example

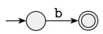
SAPIENZA Università di Roma Dipartimento di Informatica

Let's consider the R.E. $(ab \cup a)^*$ basically proceed step by step

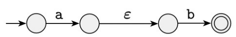
a



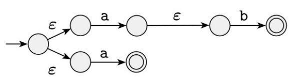
b



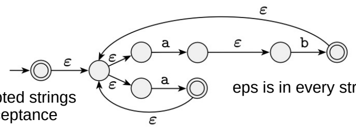
ab



ab ∪ a
accept ab or a



(ab ∪ a)*



repeat the accepted strings by restarting acceptance

eps is in every string star operator

8

8

Equivalence with NFA (2)

Thm.2: for every DFA M there exists a regular expression R such that $L(R) = L(M)$

To prove this result, we consider ***generalized nondeterministic finite automata*** (GNFA):

- GNFA are simply NFA wherein the transition arrows may have any regular expressions as labels, instead of only members of the alphabet or ε.
- So, a GNFA reads blocks of symbols from the input, not necessarily just one symbol at a time.
- The GNFA moves along a transition arrow connecting two states by reading a block of symbols from the input, which constitute a string described by the regular expression on that arrow.

DFA -> GNFA -> regexp

First we show how to convert DFAs into GNFAs, and then GNFAs into regular expr's.

For convenience, we require that GNFAs always have a special form:

- The start state has transition arrows going to every other state but no arrows coming in;
- There is only a single accept state (different from the starting one) and it has arrows coming in from other states but no arrows going to any other state;
- For all other states, one arrow goes to every other state (including itself).

9

Equivalence with NFA (3)

DEFINITION

A ***generalized nondeterministic finite automaton*** is a 5-tuple, $(Q, \Sigma, \delta, q_{\text{start}}, \{q_{\text{accept}}\})$ where

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta\colon (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

The symbol $\mathcal{R}$ is the collection of all regular expressions over the alphabet Σ

A GNFA accepts a string w in Σ^* if $w = w_1 w_2 \cdots w_k$, where each w_i is in Σ^* and a sequence of states $q_0, q_1, \ldots, q_k$ exists such that

1. $q_0 = q_{\text{start}}$ is the start state,
2. $q_k = q_{\text{accept}}$ is the accept state, and
3. for each i, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$; in other words, R_i is the expression on the arrow from q_{i-1} to q_i.

10

for each i, wi belongs to the language of the regexp

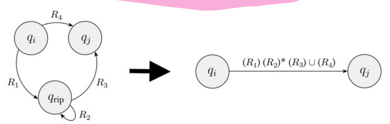
Equivalence with NFA (4)

Algorithm for converting a GNFA G into a regular expression R:

CONVERT(G):

1. Let k be the number of states of G.
2. If $k = 2$, then G must consist of a start state, an accept state, and a single arrow connecting them and labeled with a regular expression R. Return the expression R.
3. If $k > 2$, we select any state $q_{\text{rip}} \in Q$ different from q_{start} and q_{accept} and let G' be the GNFA $(Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})$, where
$$Q' = Q - \{q_{\text{rip}}\},$$
and for any $q_i \in Q' - \{q_{\text{accept}}\}$ and any $q_j \in Q' - \{q_{\text{start}}\}$, let
$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$
for $R_1 = \delta(q_i, q_{\text{rip}})$, $R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $R_3 = \delta(q_{\text{rip}}, q_j)$, and $R_4 = \delta(q_i, q_j)$.
4. Compute CONVERT(G') and return this value.

IDEA:



11

let qi be the starting state and qj be the final state
we can get from qi to qj with:
R4
R1 then R2* then R3
qed

Equivalence with NFA (5)

Lemma: For any GNFA G, CONVERT(G) is (language) equivalent to G.

Proof (by induction on k, the number of states of G)

Base (k=2): trivial, by step 2 of the algorithm

Induction step (k>2: assume the claim for k−1 states and prove the claim for k states):

Consider the first step performed by the algorithm, that reduces G to some G' by erasing some q_{rip}.

1. $w \in L(G)$ implies $w \in L(G')$:

- $w \in L(G)$ means that exists an accepting branch of G $q_{\text{start}}\, q_1\, q_2\, q_3 \ldots q_{\text{accept}}$ and w belongs to the regular expression obtained from the concatenation of the reg.expr's labeling the transitions
- If $q_{\text{rip}} \notin \{q_1\, q_2\, q_3 \ldots\}$, clearly G' also accepts w (the new regular expressions labeling the arrows of G' contains the old regular expression as part of a union)
- Otherwise, consider every occurrence of q_{rip}, say $q_{\text{rip}} = q_i$, and let q_h and q_j be the closest preceeding and following states in the sequence different from q_{rip}.
- Then, the reg.expr. labeling the arrow from q_h to q_j in G' contains in its union the concatenation of the reg.exp. from q_h to q_{rip}, from q_{rip} to itself (as many times as needed), and from q_{rip} to q_j.
- By repeating this for every occurrence of q_{rip} in $q_1\, q_2\, q_3 \ldots$, we show that w leads G' from q_{start} to q_{accept} (without passing from q_{rip}).

12

we can also do this by double inclusion

Equivalence with NFA (6)

2. $w \in L(G')$ implies $w \in L(G)$:
   - each arrow between any q_i and q_j in G' describes the collection of strings taking q_i to q_j in G, either directly or via q_{rip}
   - hence, also G accepts w.
3. Points 1 and 2 prove that $L(G) = L(G')$
4. Since G' has k–1 states, by inductive hyp. $L(G') = L(\text{CONVERT}(G'))$.
5. By definition of the algorithm, $\text{CONVERT}(G) = \text{CONVERT}(G')$.
6. So, $L(G) = L(\text{CONVERT}(G))$. Q.E.D.

13

Equivalence with NFA (7)

Proof of Thm.2:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the DFA.

Let q_{start} , $q_{accept} \notin Q$. We build the GNFA $G = (Q \cup \{q_{start} , q_{accept} \}, \Sigma, \delta', q_{start} , \{q_{accept}\})$ where δ' associates the transitions

- from q_{start} to q_0 with reg.exp. ε
- from q_{start} to every $q \in Q' - \{q_0 , q_{start}\}$ with reg.exp. $\emptyset$
- from every $q \in F$ to q_{accept} with reg.exp. ε
- from every $q \in Q \setminus F$ to q_{accept} with reg.exp. $\emptyset$
- from every $q \in Q$ to every $q' \in Q$ such that $\exists a.\delta(q, a) = q'$ with reg.exp. a
- from every $q \in Q$ to every $q' \in Q$ such that $\nexists a.\delta(q, a) = q'$ with reg.exp. $\emptyset$

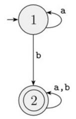
Trivially, $L(M) = L(G)$ and, by the Lemma, $L(G) = L(\text{CONVERT}(G))$.

Q.E.D.

14

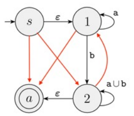
From a DFA to a Reg. Expr.: an example

Consider the DFA:



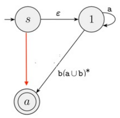
The associated GNFA is (red arrows are those labeled with ∅):

see algo definition for motivations behind red arrows



If we now remove state 2, we have three reg.expr.'s:

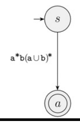
- One from s to a: $\emptyset(a\cup b)^*\varepsilon \cup \emptyset = \emptyset$
- One from s to 1: $\emptyset(a\cup b)^*\emptyset \cup \varepsilon = \varepsilon$
- One from 1 to a: $b(a\cup b)^*\varepsilon \cup \emptyset = b(a\cup b)^*$



Finally, if we remove state 1, the only reg.expr. left is

$\varepsilon\ (a)^*\ b(a\cup b)^* \cup \emptyset\ =\ a^*b(a\cup b)^*$

that generates the language of the original DFA.



15