Some non-regular languages are:

- Palindromes = $\{w = w^R | w \in \Sigma^*\}$
- Copy Language = $\{ww | w \in \Sigma^*\}$
- $\{a^n b^n | n \geq 0\}$.

Some non-regular languages are:

- Palindromes = $\{w = w^R | w \in \Sigma^*\}$
- Copy Language = $\{ww | w \in \Sigma^*\}$
- $\{a^n b^n | n \geq 0\}$.

- But, $\{a^n b^n | 0 \leq n \leq 5\}$ is regular.
- Every finite language is regular. {Can you prove this?}.
- Σ^* is regular, ϕ is regular, …

Regular Expressions

1.3

REGULAR EXPRESSIONS

In arithmetic, we can use the operations $+$ and $\times$ to build up expressions such as

$$(5+3)\times 4\,.$$

Similarly, we can use the regular operations to build up expressions describing languages, which are called ***regular expressions***. An example is:

$$(0 \cup 1)0^*.$$

- What are values of these expressions?

Where used

- Very useful to describe a set of strings having certain patterns.
  - In UNIX, rm *.c → removes all files ending with .c
  - Lex, a tool used in compiler generators
  - grep, awk available utilities in UNIX use regular expressions.

Operators

0 means the language $\{0\}$

1 means $\{1\}$

$(0 \cup 1)$ means $\{0\} \cup \{1\}$

0^* means $\{0\}^*$

$(0 \cup 1)0^*$ actually is shorthand for $(0 \cup 1) \circ 0^*$

$$\boxed{(0 \cup 1)0^* = \{0, 1\}\{0\}^*}$$

Inductive Definition

DEFINITION 1.52

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.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

Don’t confuse between null string and null set

Don’t confuse the regular expressions ε and $\emptyset$. The expression ε represents the language containing a single string—namely, the empty string—whereas $\emptyset$ represents the language that doesn’t contain any strings.

Parentheses in an expression may be omitted. If they are, evaluation is done in the precedence order: star, then concatenation, then union.

- Precedence is $*, \circ, U$
- So, aUb* is different from (aUb)*

  aUb* = (a U (b)*)
- aUb*c is same as aU(b*c)

- Many authors use + for U

  So, aUb = a+b But + is overloaded.

  Sipser reserved the + to mean only one thing.

+ (positive closure)

$$R^* = R^0 \cup R^1 \cup R^2 \cup \ldots \cup R^i \cup \ldots$$

$$R^+ = R^1 \cup R^2 \cup \ldots \cup R^i \cup \ldots$$

$$R^+ = RR^*$$

$$R^* = R^+ \cup \epsilon$$

- The value of a regular expression R is nothing but the language represented by R
- When we want to distinguish between R.E. and the language represented by it. We use *L(R)* to mean language represented by *R*.

We can write Σ as shorthand for regular expression $(0 \cup 1)$

From the context it should be clear for us by saying Σ do we mean alphabet or the language consisting of all possible strings of length 1.

Σ^* is a regular expression which is the language of all strings (including ϵ) over the alphabet Σ.

1. $0^*10^* =$

2. $\Sigma^*1\Sigma^* =$

3. $\Sigma^*001\Sigma^*$

1. $\texttt{0}^*\texttt{1}\texttt{0}^* = \{w \mid w \text{ contains a single } \texttt{1}\}$.

2. $\Sigma^*\texttt{1}\Sigma^* = \{w \mid w \text{ has at least one } \texttt{1}\}$.

3. $\Sigma^*\texttt{001}\Sigma^* = \{w \mid w \text{ contains the string } \texttt{001} \text{ as a substring}\}$.

Can you describe the language?

$$1^*(01^+)^* =$$

$$(\Sigma\Sigma)^* =$$

$$(\Sigma\Sigma\Sigma)^* =$$

$$01 \cup 10 =$$

$$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$$

$$(0 \cup \varepsilon)(1 \cup \varepsilon) =$$

$$1^*\emptyset =$$

$$\emptyset^* =$$

Can you describe the language?

$$1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}.$$

$$(\Sigma\Sigma)^* =$$

$$(\Sigma\Sigma\Sigma)^* =$$

$$01 \cup 10 =$$

$$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$$

$$(0 \cup \varepsilon)(1 \cup \varepsilon) =$$

$$1^*\emptyset =$$

$$\emptyset^* =$$

Can you describe the language?

$$1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}.$$

$$(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}.$$

$$(\Sigma\Sigma\Sigma)^* =$$

$$01 \cup 10 =$$

$$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$$

$$(0 \cup \varepsilon)(1 \cup \varepsilon) =$$

$$1^*\emptyset =$$

$$\emptyset^* =$$

$$1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}.$$

$$(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}.$$

$$(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of } 3\}.$$

$$01 \cup 10 = \{01, 10\}.$$

$$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}.$$

$$(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}.$$

$$1^*\emptyset = \emptyset.$$

$$\emptyset^* = \{\varepsilon\}.$$

Understood?

If we let R be any regular expression, we have the following identities. They are good tests of whether you understand the definition.

$R \cup \emptyset = R$.
Adding the empty language to any other language will not change it.

$R \circ \varepsilon = R$.
Joining the empty string to any string will not change it.

Understood?

However, exchanging $\emptyset$ and ε in the preceding identities may cause the equalities to fail.

$R \cup \varepsilon$ may not equal R.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$.

$R \circ \emptyset$ may not equal R.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$.

Compilers -- Tokens

- Lexical Analysis
  - Automatic tools can be used (like Lex)
    - But , you need to describe what you want.
- For Decimal Numbers:

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

where $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is the alphabet of decimal digits. Examples of generated strings are: 72, 3.14159, +7., and -.01.

Equivalence of RE with DFA/NFA

- It is somewhat surprising to note that, RE can be used to describe any regular language.
  - This is not true for other higher level languages, like CFL {We can describe a CFL by a CFG, not by an expression}.
- Now, how is that we prove this?

Proof has two directions

- Given RE, show that we can build a DFA/NFA recognizing the language given by the RE.
- Given DFA/NFA, show that we can convert this in to a RE.

To Show

- Given RE, show that we can build a NFA recognizing the language given by the RE.
- We use inductive definition of RE.

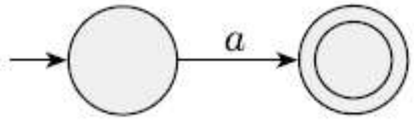
Inductive Definition of RE

DEFINITION 1.52

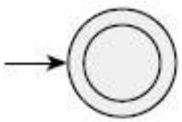
Say that R is a ***regular expression*** if R is

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.

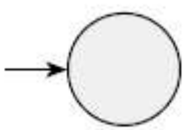
1. $R = a$ for some $a \in \Sigma$. Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.



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 $L(R)$.



- One should be able to do these formally !!

4. $R = R_1 \cup R_2$.

5. $R = R_1 \circ R_2$.

6. $R = R_1^*$.

- **Use the construction proofs.**

- Let $L(N_1) = A_1$, and $L(N_2) = A_2$

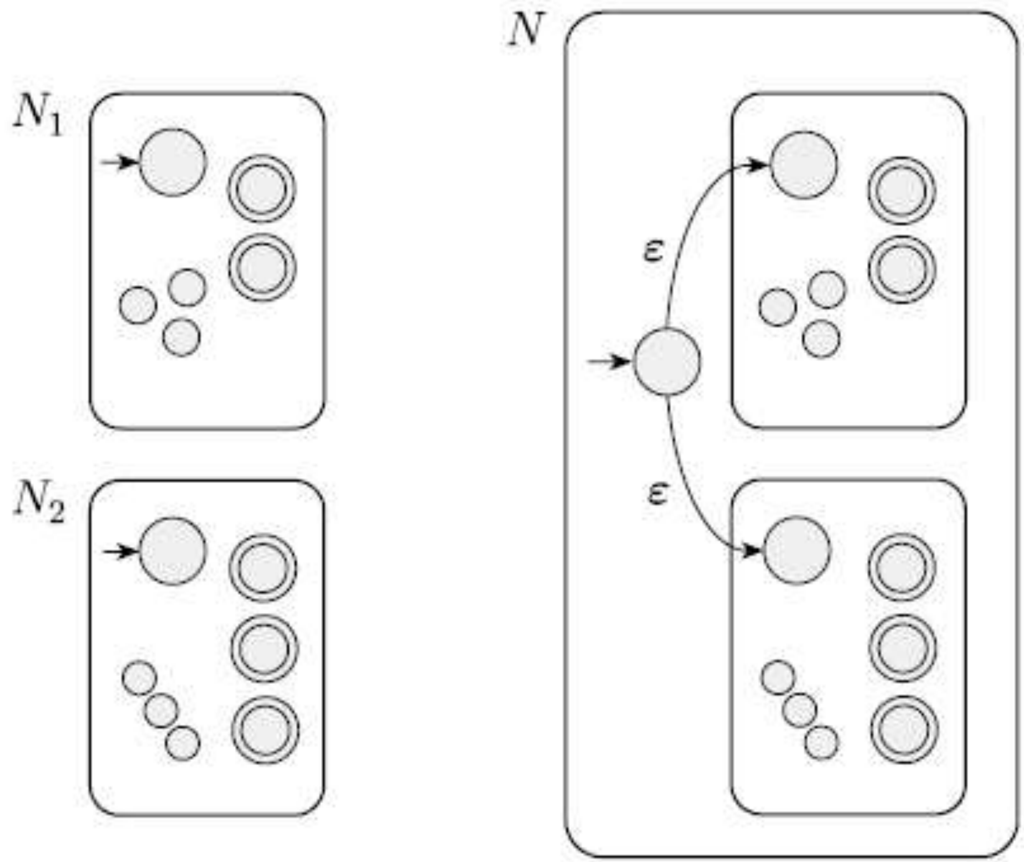


FIGURE 1.46

Construction of an NFA N to recognize $A_1 \cup A_2$

THEOREM **1.47**

The class of regular languages is closed under the concatenation operation.

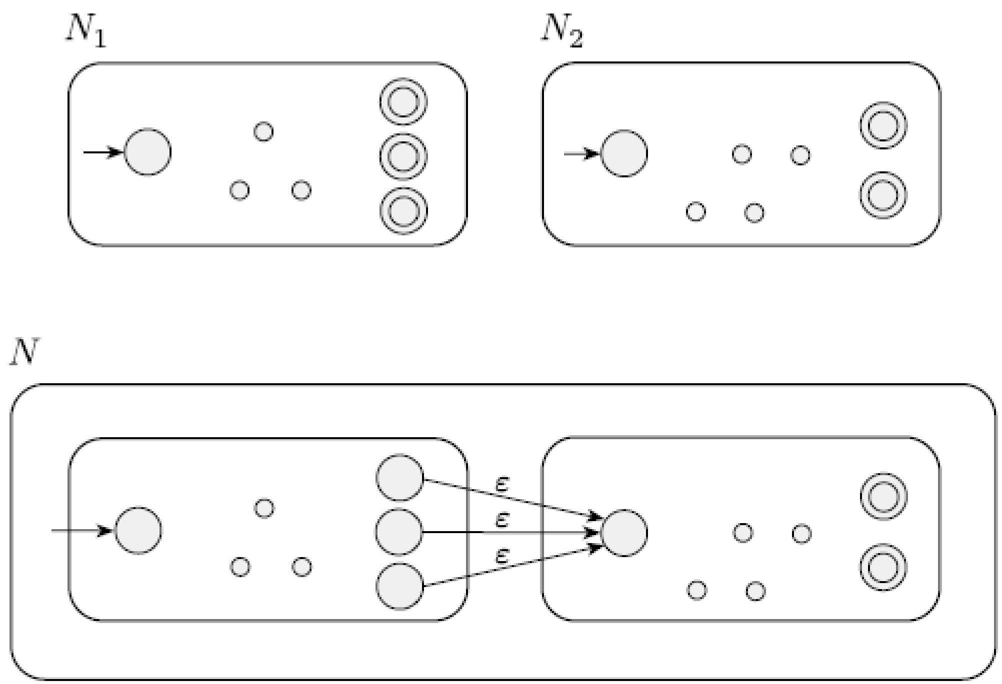
- 

FIGURE **1.48**
Construction of N to recognize $A_1 \circ A_2$

THEOREM 1.49

The class of regular languages is closed under the star operation.

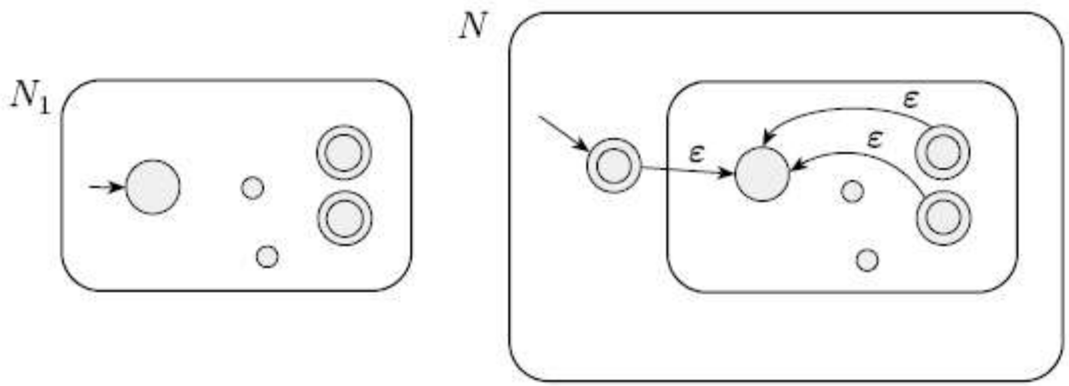
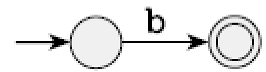
- 

FIGURE 1.50
Construction of N to recognize A^*

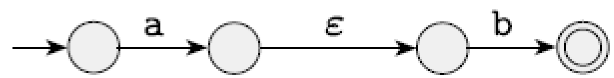
We convert the regular expression $(\texttt{ab} \cup \texttt{a})^*$ to an NFA in a sequence of stages.

a

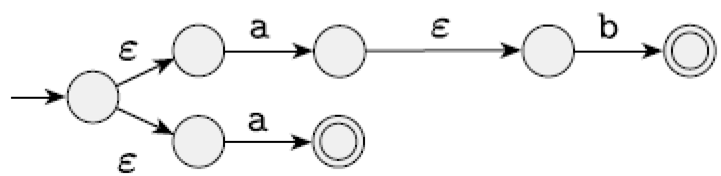
b



ab



ab ∪ a



(ab ∪ a)*