



**OPPA European Social Fund
Prague & EU: We invest in your future.**

Alphabet

Alphabet ... finite (unempty) set of symbols
 $|A|$... size of alphabet A

Examples: $A = \{ 'A', 'D', 'G', 'O', 'U' \}, |A| = 5$
 $A = \{ 0, 1 \}, |A| = 2$
 $A = \{ \bigcirc, \square, \triangle \}, |A| = 3$

word

**Word (over alphabet A) ... finite (maybe empty) sequence
also string of symbols of alphabet (A)**
 $|w|$... length of word w

Examples: $w = \text{OUAGADOUGOU}, |w| = 11$
 $w = 1001, |w| = 4$
 $w = \square\triangle\bigcirc\triangle\square, |w| = 5$

Language

Language ... set of words (=strings)
(not necessarily finite, can be empty)
 $|L|$... cardinality of language L

- ① Language specification -- List of all words of the language
(only for finite language!)

Examples: $A_1 = \{ 'A', 'D', 'G', 'O', 'U' \}$

$L_1 = \{ ADA, DOG, GOUDA, D, GAG \}, |L_1| = 5$

$A_2 = \{ 0, 1 \}$

$L_2 = \{ 0, 1, 00, 01, 10, 11 \}, |L_2| = 6$

$A_3 = \{ \bigcirc, \square, \triangle \}$

$L_3 = \{ \triangle\triangle, \bigcirc\square\bigcirc, \square\square\triangle\bigcirc \}, |L_3| = 3$

- ② Language specification -- Informal (but unambiguous) description in natural human language (usually for infinite language)

Examples: $A_1 = \{ 'A', 'D', 'G', 'O', 'U' \}$
 L_1 : Set of all words over A_1 , which begin with DA, end with G and do not contain subsequence AA.
 $L_1 = \{ DAG, DADG, DAGG, DAOG, DAUG, DADAG, DADDG... \}$
 $|L_1| = \infty$

$A_2 = \{ 0, 1 \}$
 L_2 : Set of all words over A_2 , which contain more 1s than 0s and each 0 is followed by at least two 1s.
 $L_2 = \{ 1, 11, 011, 0111, 1011, 1111, \dots, 011011, 011111, \dots \}$
 $|L_2| = \infty$

3 Language specification -- By finite automaton

Finite automaton
is a five-tuple (A, Q, σ, S_0, Q_F) , where:

A ... alphabet ... finite set of symbols
|A| ... size of alphabet

Q ... set of states (often numbered) (what is „a state“ ?)

σ ... transition function ... $\sigma: Q \times A \rightarrow Q$

S_0 ... start state $S_0 \in Q$

Q_F ... unempty set of final states $\emptyset \neq Q_F \subseteq Q$

Automaton FA1:

A ... alphabet ... $\{0,1\}$, $|A| = 2$

Q ... set of states $\{S, A, B, C, D\}$

σ ... transition function ... $\sigma: Q \times A \rightarrow Q : \{$

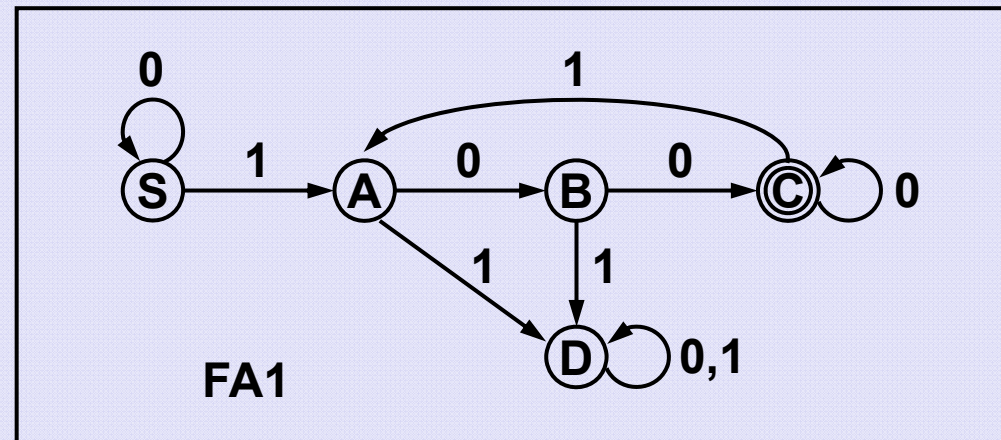
$\sigma(S,0) = S, \sigma(A,0) = B, \sigma(B,0) = C, \sigma(C,0) = C, \sigma(D,0) = D,$

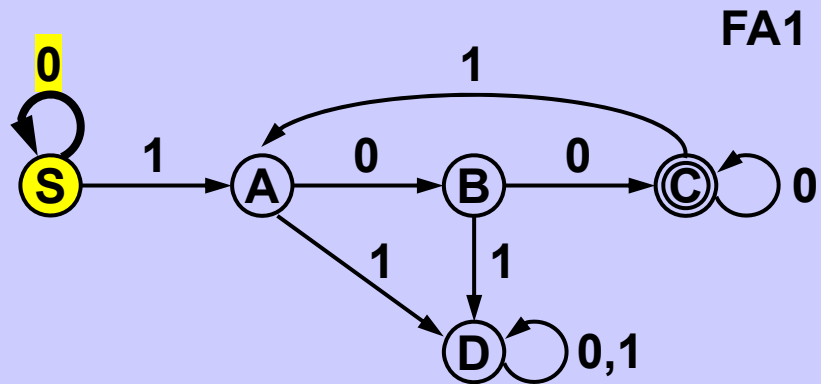
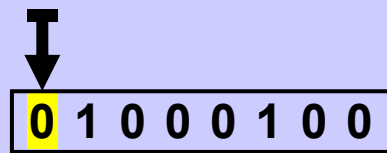
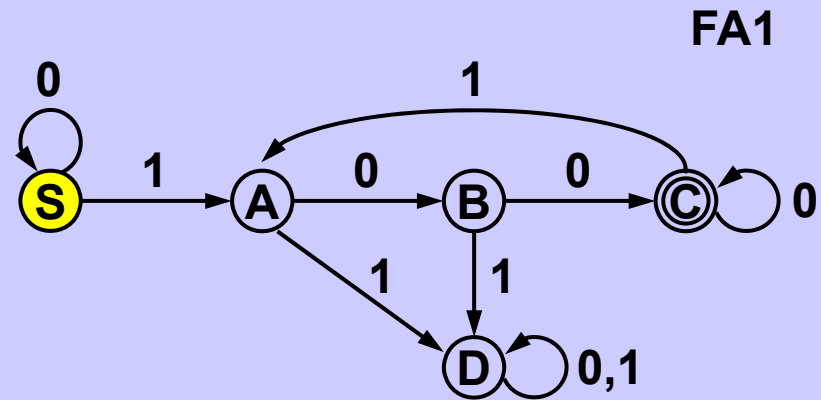
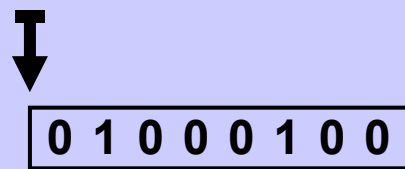
$\sigma(S,1) = A, \sigma(A,1) = D, \sigma(B,1) = D, \sigma(C,1) = A, \sigma(D,1) = D \}$

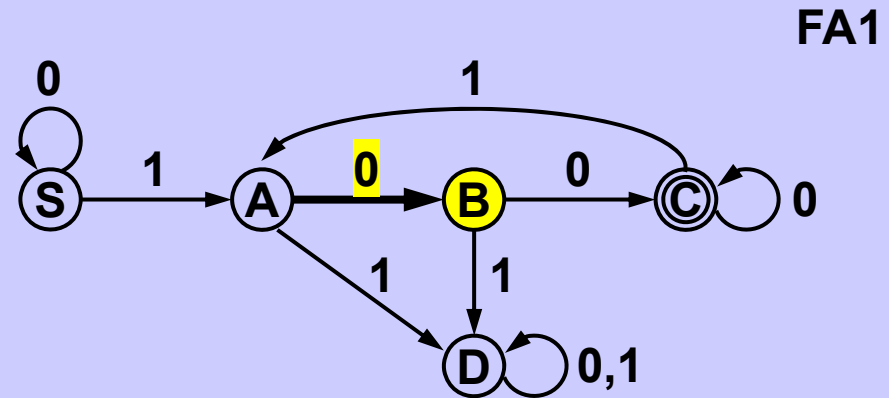
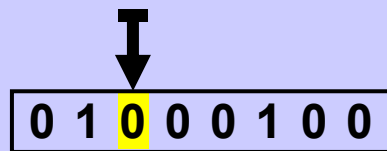
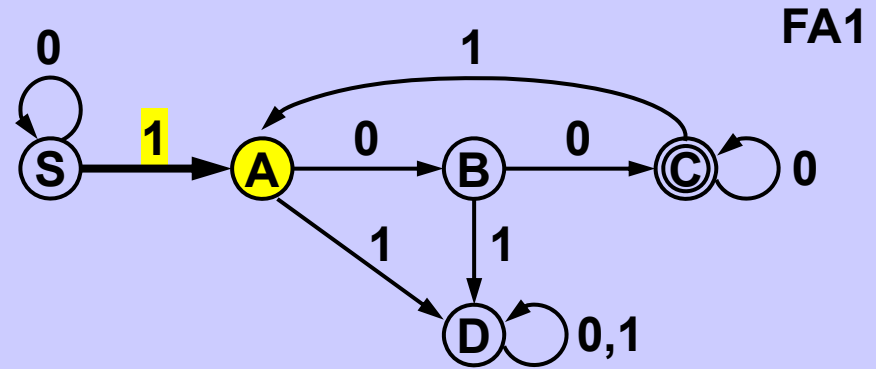
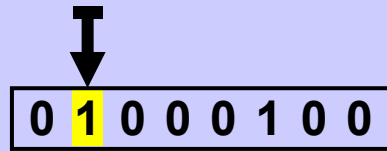
S_0 ... start state $S \in Q$

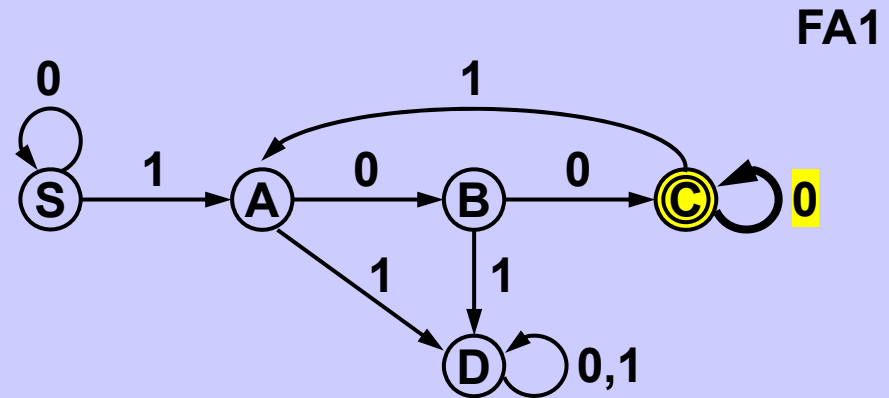
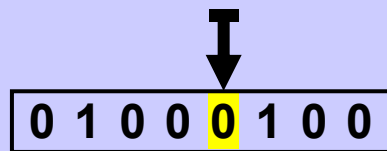
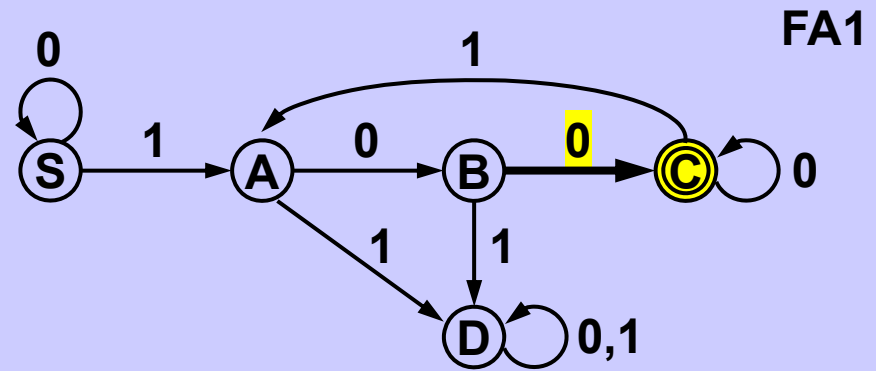
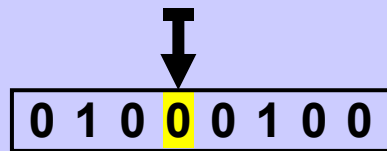
Q_F ... unempty set of final states $\emptyset \neq \{C\} \subseteq Q$

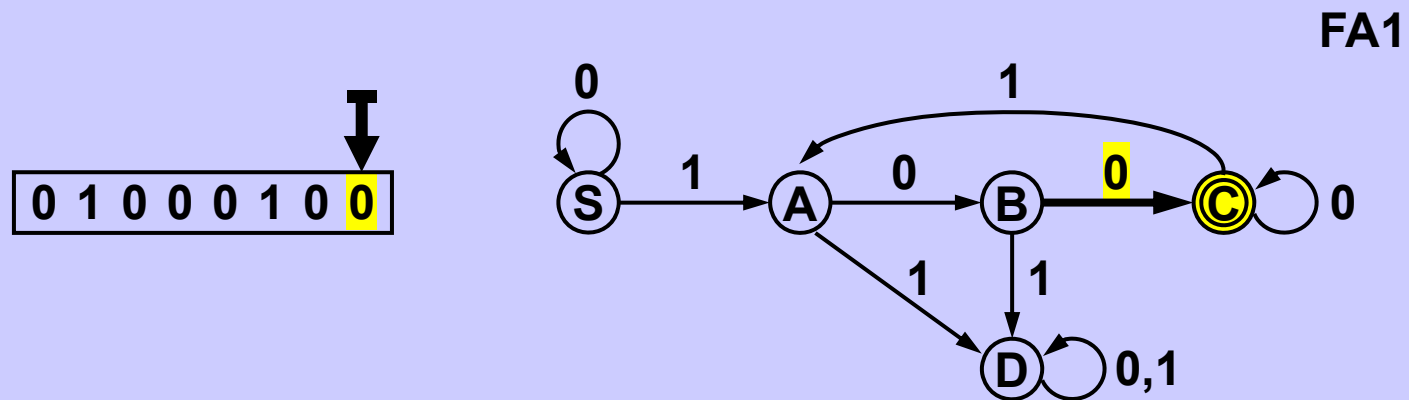
**Transition diagram
of the automaton FA1**







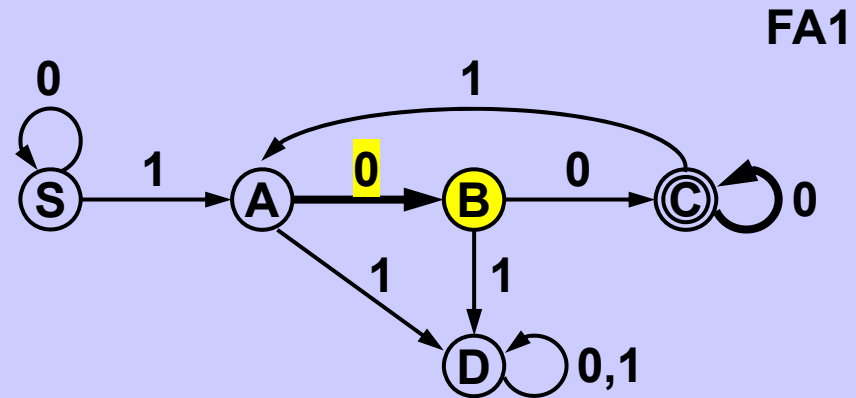
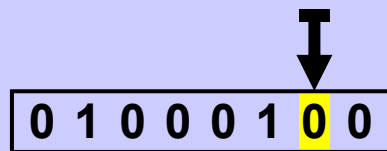
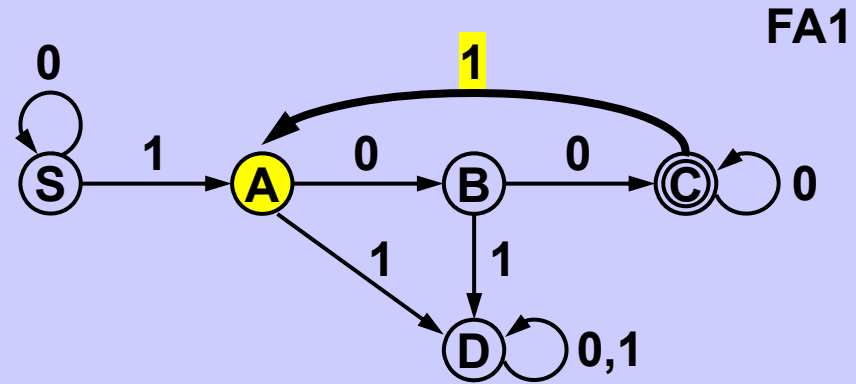
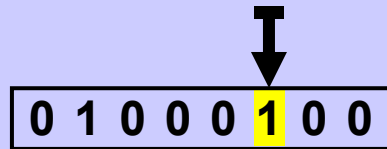


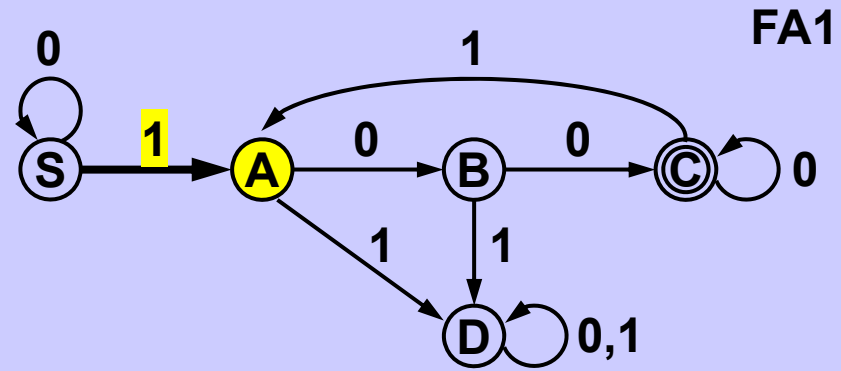
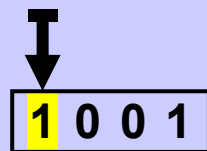
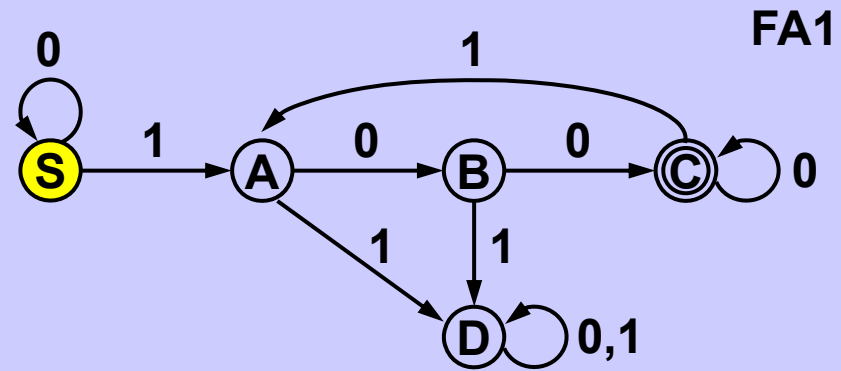
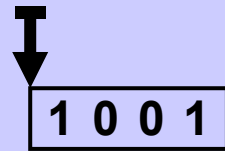


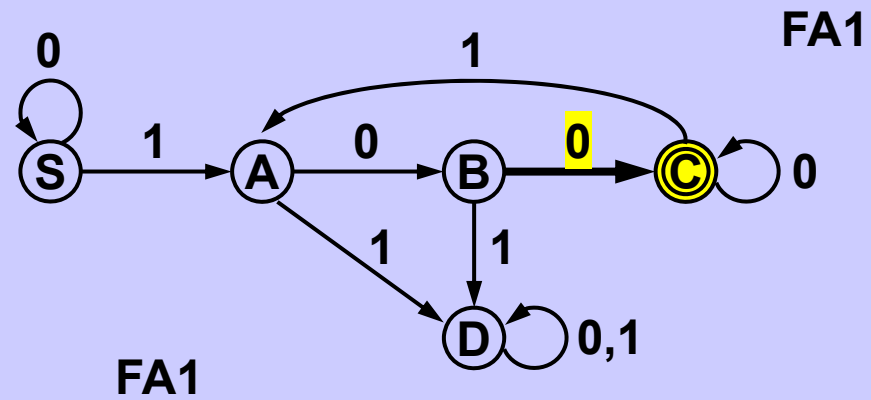
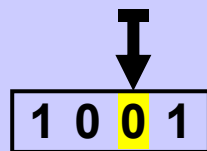
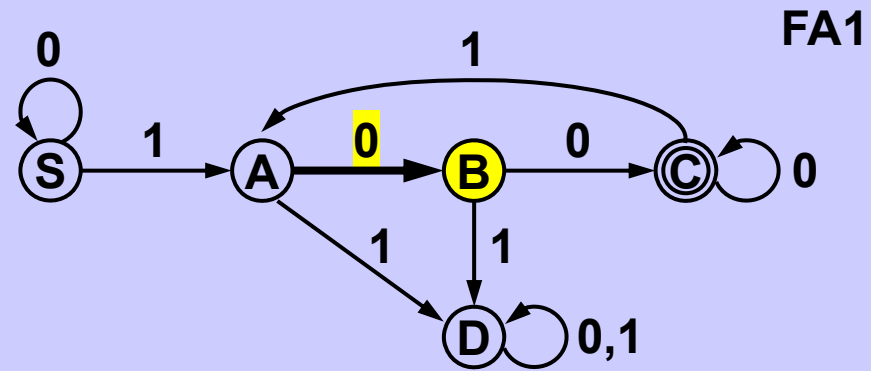
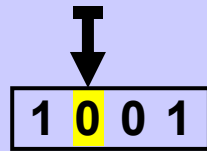
When the last word symbol is read automaton FA1 is in final state \odot



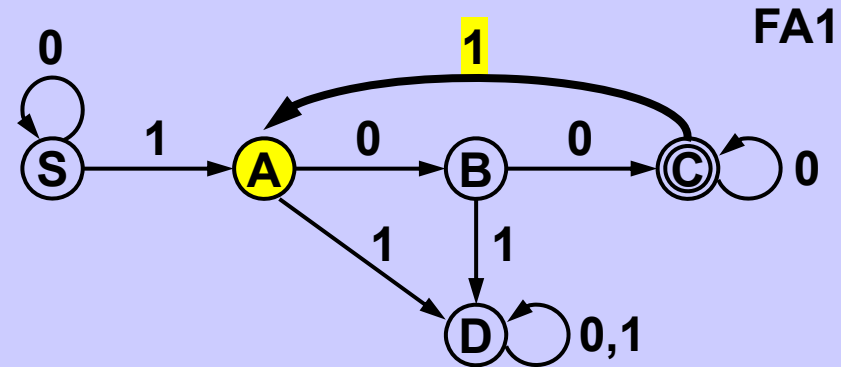
Word 0 1 0 0 0 1 0 0 is accepted by automaton FA1







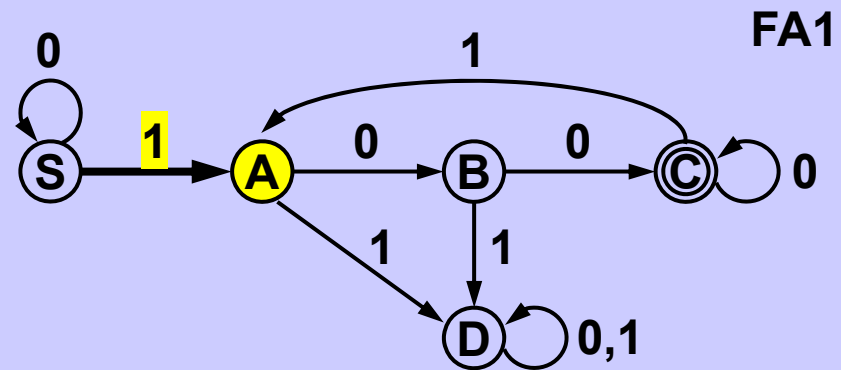
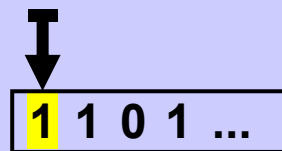
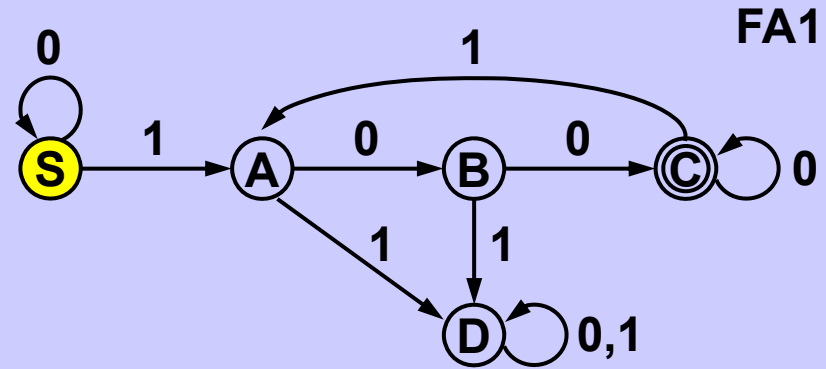
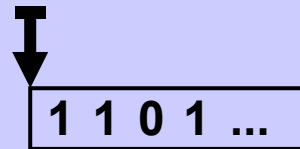
↓
1 0 0 1

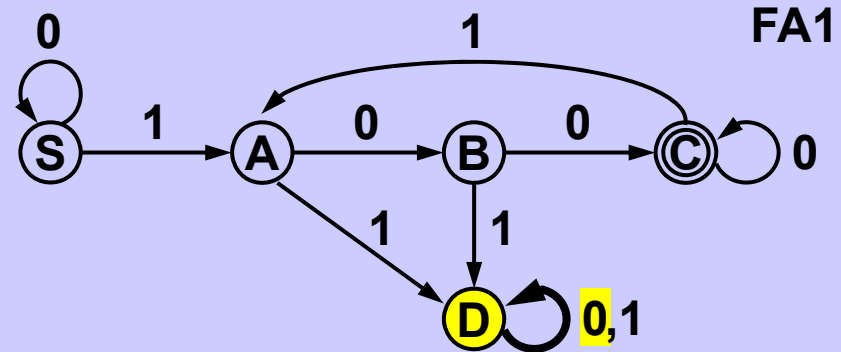
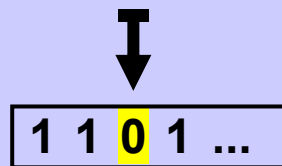
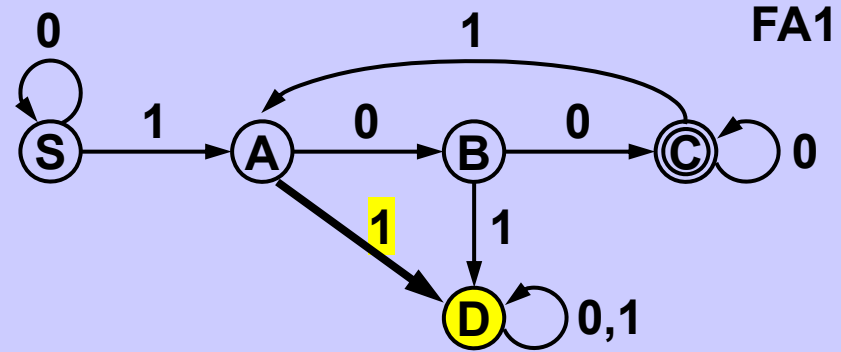
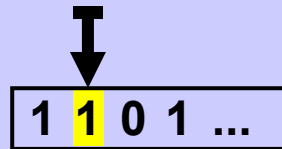


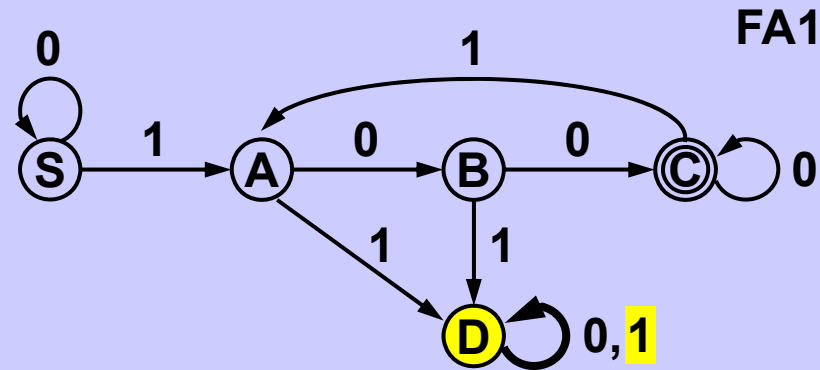
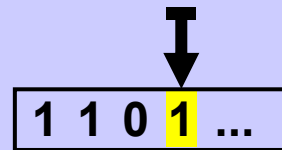
When the last word symbol is read automaton FA1 is in a state which is not final ○



Word **1 0 0 1** is not accepted by automaton FA1







No word starting with

1 1 ...

is accepted by automaton FA1

No word containing

... 1 1 ...

is accepted by automaton FA1

No word containing

... 1 0 1 ...

is accepted by automaton FA1

Automaton FA1 accepts only words -- containing at least one 1
 -- containing at least two 0s after each 1

Language accepted by automaton = set of all words accepted by automaton

Automaton activity:

At the beginning the automaton is in the start state.

Next it reads the input word symbol by symbol and transits to other states according to the transition function.

When the word is read the automaton is again in some state.

If it is in a final state, we say that it accepts the word,

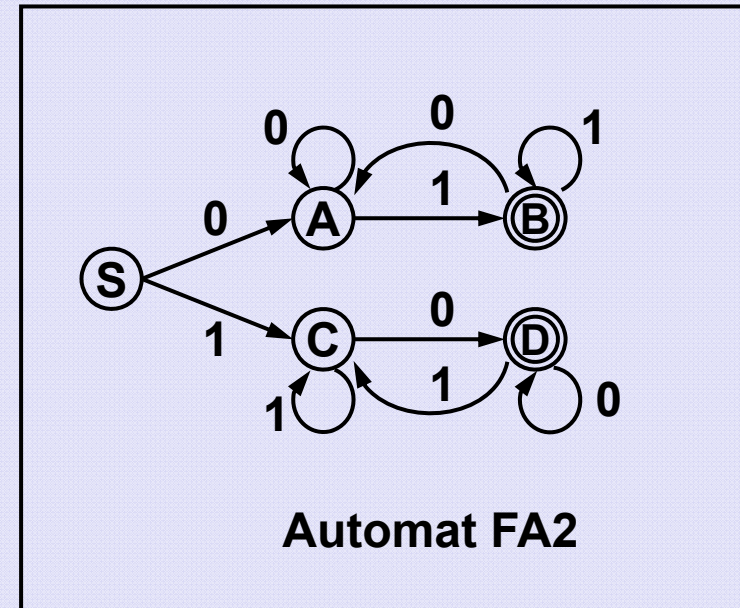
if it is not in a final state, we say that it does not accept the word.

All words accepted by the automaton represent

a language accepted (or recognized) by the automaton.

Language over alphabet $\{0,1\}$:

If the word starts with 0, it ends with 1,
If the word starts with 1, it ends with 0.



Example of analysis of different words by FA2:

$0\ 1\ 0\ 1\ 0$: $(S),0 \rightarrow (A),1 \rightarrow (B),0 \rightarrow (A),1 \rightarrow (B),0 \rightarrow (A)$

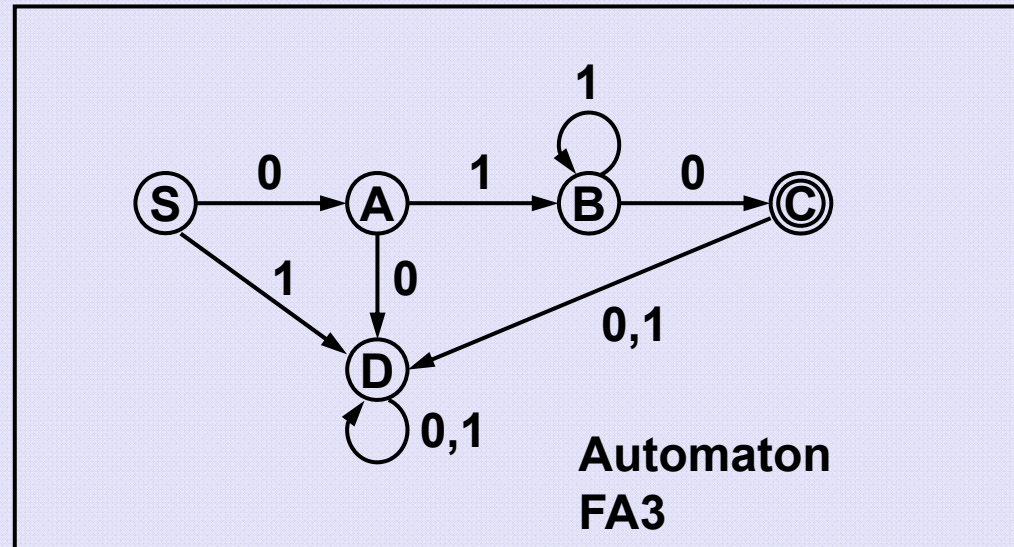
(A) is not a final state, word $0\ 1\ 0\ 1\ 0$ is rejected by FA2.

$1\ 0\ 1\ 1\ 0$: $(S),1 \rightarrow (C),0 \rightarrow (D),1 \rightarrow (C),1 \rightarrow (C),0 \rightarrow (D)$

(D) is a final state, word $1\ 0\ 1\ 1\ 0$ is accepted by FA2.

Language:

```
{
0 1 0,
0 1 1 0,
0 1 1 1 0,
0 1 1 1 1 0,
0 1 1 1 1 1 0,
... }
```



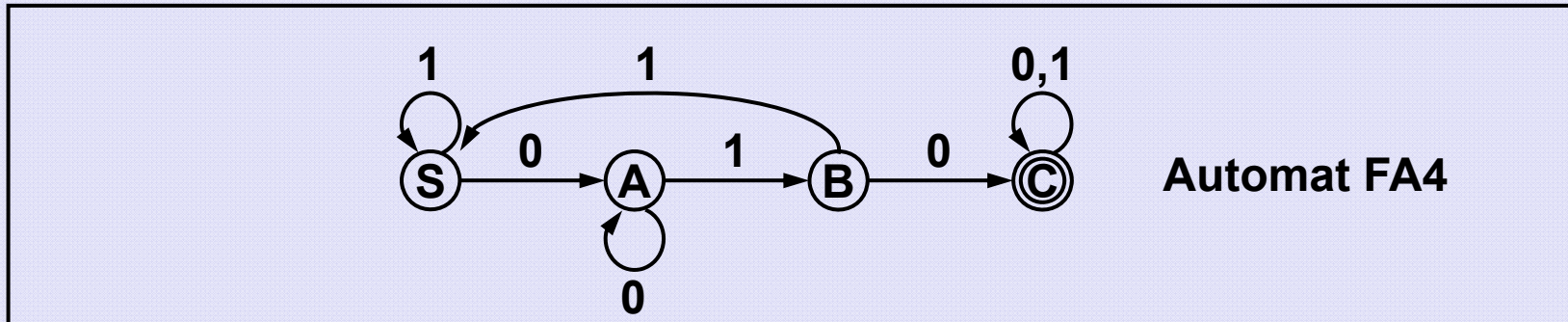
Example of analysis of different words by FA3:

0 1 0 1 0 : (S),0 → (A),1 → (B),0 → (C),1 → (D),0 → (D)

(D) is not a final state, word 0 1 0 1 0 is rejected by FA3.

0 1 1 1 0 : (S),0 → (A),1 → (B),1 → (B),1 → (B),0 → (C)

(C) is a final state, word 0 1 1 1 0 is accepted by FA3.



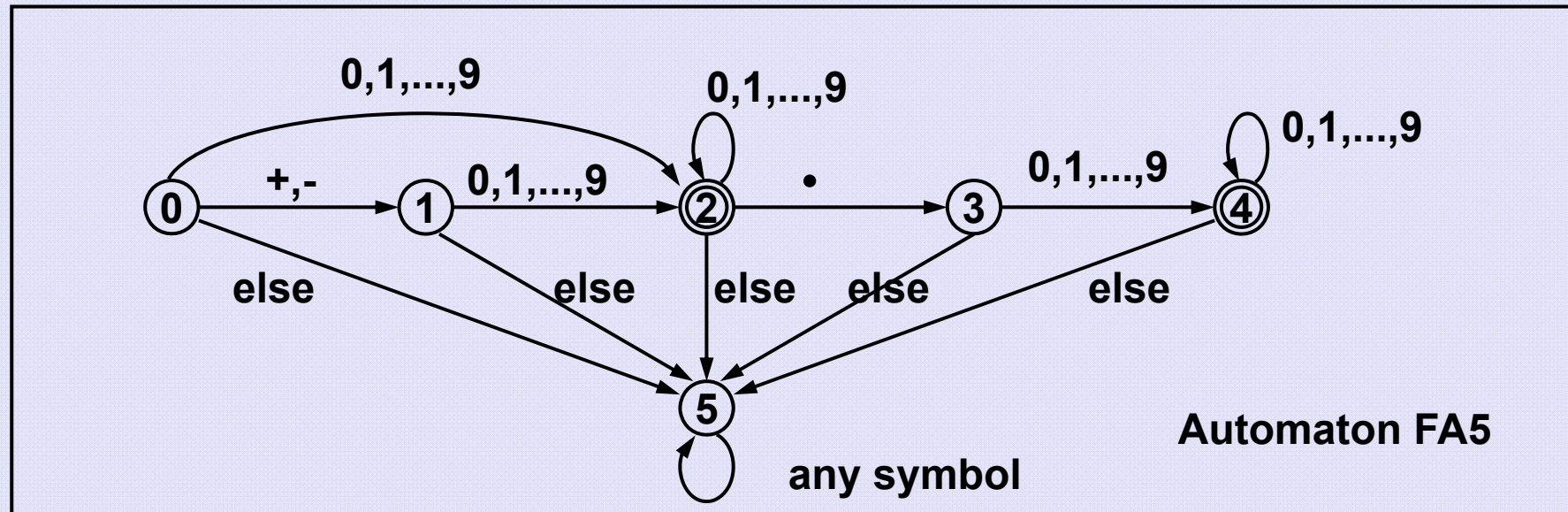
Automaton FA4 accepts each word over the alphabet $\{0,1\}$ which contains subsequence ... 0 1 0 ...

Example of analysis of different words by FA4:

0 0 1 0 1 : (S),0 → (A),0 → (A),1 → (B),0 → (C),1 → (C)
 (C) is a final state, word 0 0 1 0 1 is accepted by FA4.

0 1 1 1 0 : (S),0 → (A),1 → (B),1 → (S),1 → (S),0 → (A)
 (A) is not a final state, word 0 1 1 1 0 is rejected by FA4.

Language over the alphabet $\{ +, -, ., 0, 1, \dots, 8, 9, \dots \}$ whose words represent a decimal numbers



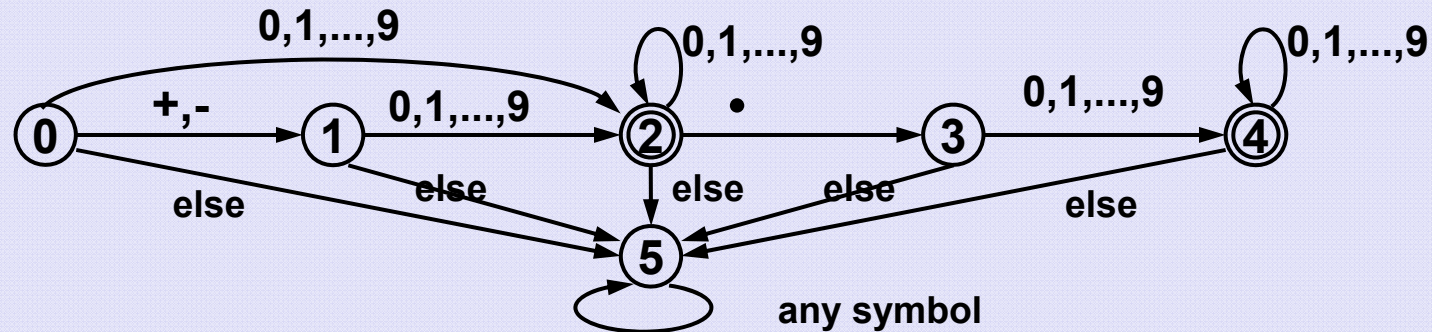
Example of word analysis

+87.09: (0),+ → (1),8 → (2),7 → (2),. → (3),0 → (4),9 → (4)

(4) is a final state, word **+87.05** is accepted by FA5.

76+2: (0),7 → (2),6 → (2),+ → (5),2 → (5)

(5) is not a final state, word **76+2** is not accepted by FA5.

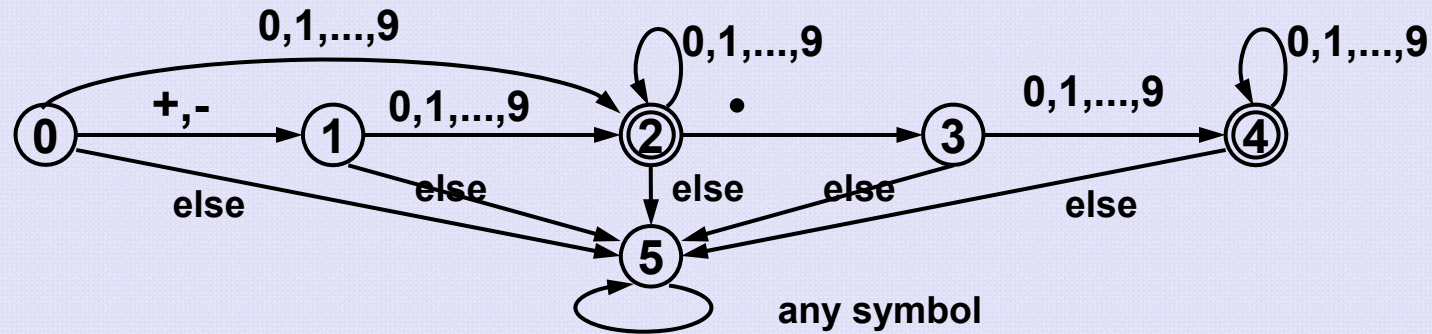


Code of the finite automaton

(The word which is being read is stored in the array arr[]):

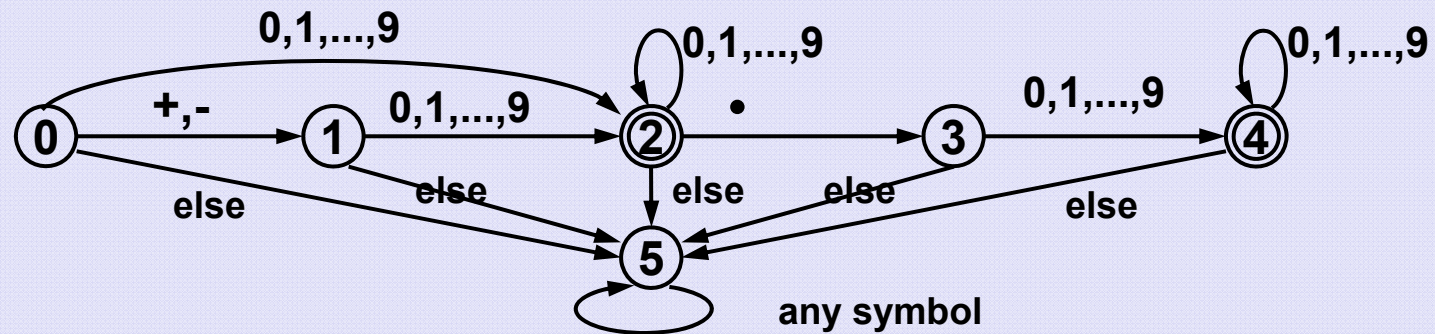
```
int isDecimal(char arr[], int length) {
    int i;
    int state = 0;

    for(i = 0; i < length; i++) { // check each symbol
        switch (state) {
            ...
        }
    }
}
```



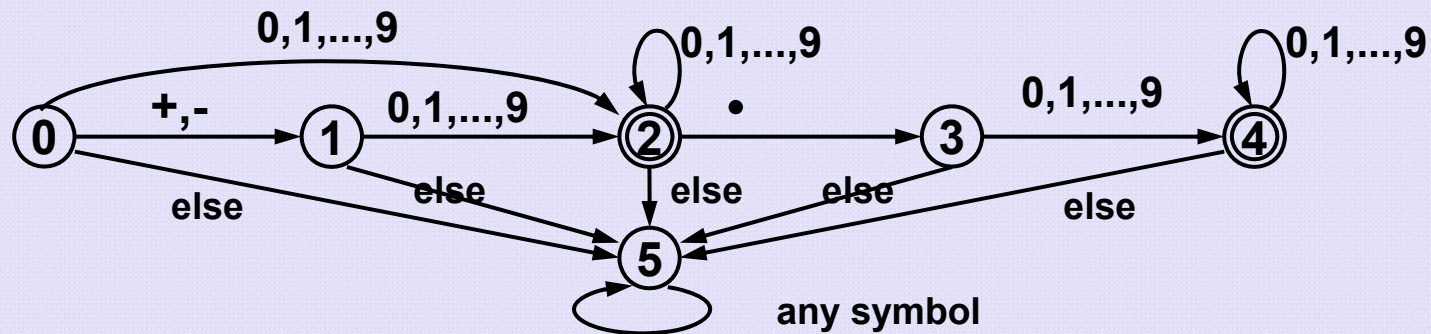
case 0:

```
if ((arr[i] == '+') || (arr[i] == '-')) state = 1;
else
if ((arr[i] >= '0') && (arr[i] <= '9')) state = 2;
else state = 5;
break;
```

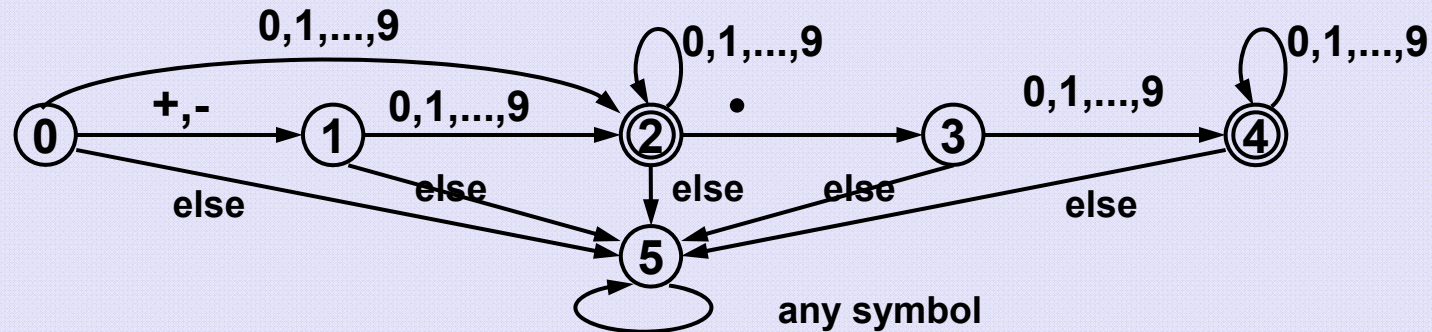
case 1:

```
if ((arr[i] >= '0') && (arr[i] <= '9')) state = 2;  
else state = 5;  
break;
```



case 2:

```
if ((arr[i] >= '0') && (arr[i] <= '9')) state = 2;  
else  
if (arr[i] == '.') state = 3;  
else state = 5;  
break;
```



③ case 3:

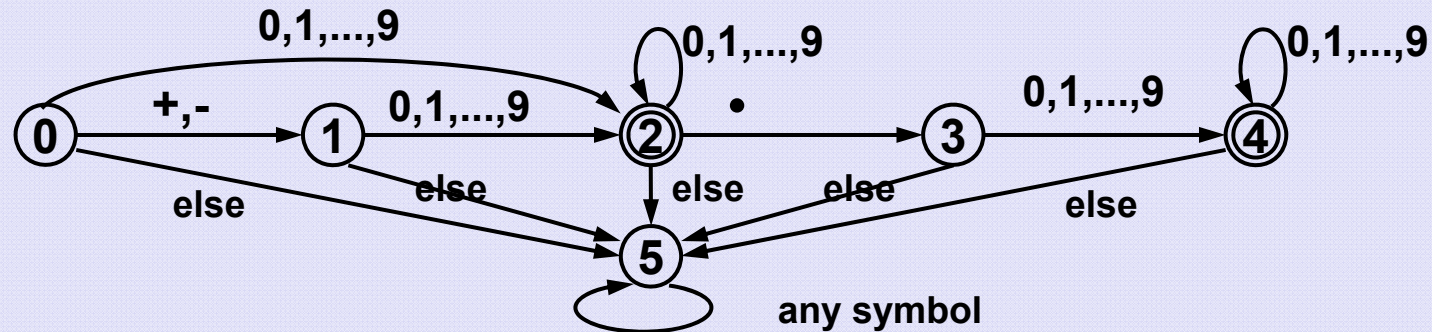
```
if ((arr[i] >= '0') && (arr[i] <= '9')) state = 4;
else state = 5;
break;
```

④ case 4:

```
if ((arr[i] >= '0') && (arr[i] <= '9')) state = 4;
else state = 5;
break;
```

⑤ case 5: break; // no need to react anyhow

```
default : break;
} // end of switch
```



```

} // end of for loop -- word has been read

if ((state == 2) || (state == 4)) // final states!!
    return 1;                    // success - decimal OK
else
    return 0;                    // not a decimal
} // end of function isDecimal()

```



**OPPA European Social Fund
Prague & EU: We invest in your future.**
