Languages, grammars, automata

English sources:

[1] B. Melichar, J. Holub, T. Polcar: **Text Search Algorithms**
http://cw.felk.cvut.cz/lib/exe/fetch.php/courses/a4m33pal/melichar-tsa-lectures-1.pdf
Chapters 1.4 and 1.5, it is probably reasonably short, there is nothing to skip.

[2] J. E. Hopcroft, R. Motwani, J. D. Ullman: **Introduction to Automata Theory**
folow the link at http://cw.felk.cvut.cz/doku.php/courses/a4m33pal/literatura_odkazy
Chapters 1., 2., 3., there is a lot to skip, consult the teacher preferably.

Czech instant sources:

[3] M. Demlová: **A4B01JAG**
http://math.feld.cvut.cz/demlova/teaching/jag/
Pages 1-27, in PAL, you may wish to skip: Proofs, chapters 2.4, 2.6, 2.8.

[4] I. Černá, M. Křetínský, A. Kučera: **Automaty a formální jazyky I**
http://is.muni.cz/do/1499/el/estud/fi/js06/ib005/Formalni_jazyky_a_automaty_I.pdf
Chapters 1 and 2, skip same parts as in [1].

For more references see PAL links pages
https://cw.fel.cvut.cz/wiki/courses/be4m33pal/references (EN)
http://cw.felk.cvut.cz/doku.php/courses/b4m33pal/odkazy-zdroje (CZ)

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 = {○, □, △}, |A| = 3

word

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

Examples: w = OUAGADOUGOU, |w| = 11
w = 1001, |w| = 4
w = □△○△□, |w| = 5

Language

Language … set of words (=strings)
(not necessarily finite, can be empty)
over a given alphabet
|L| … cardinality of language L

---

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

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

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

$A_2 = \{0,1\}$

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

$A_3 = \{$ ○, □, △ $\}$

$L_3 = \{$ △△, ○□○, □□△○ $\}, \quad |L_2| = 3$

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 a and do not contain subsequence AA.

$L_1 = \{DAG, DADG, DAGG, DAOG, DAUG, DADAG, DADDG\ldots\}$

$|L_1| = \infty$

$A_2 = \{0,1\}$

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.

$L_2 = \{ 1, 11, 011, 0111, 1011, 1111, \ldots, 011011, 011111, \ldots \}$

$|L_2| = \infty$

3 Language specification -- By finite automaton

Definition

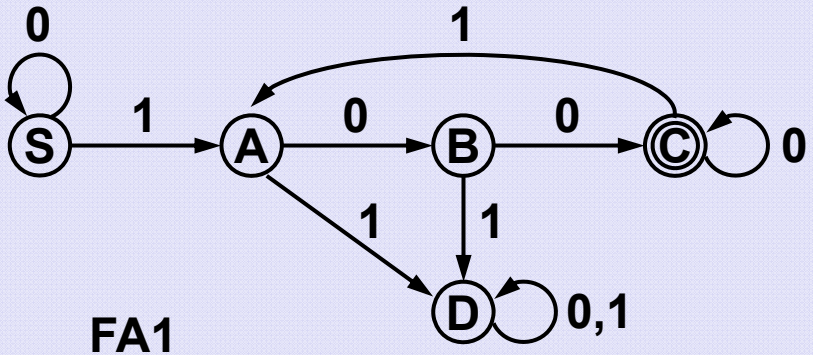
Finite automaton **is a five-tuple (A, Q, σ, S_0, Q_F), where:**

- **A** … **alphabet** … finite set of symbols
  - $|A|$ ... size of alphabet
- **Q** ... set of **states** (often numbered)
- **σ** ... **transition function** ... $\sigma: Q \times A \rightarrow Q$
- **S_0** ... **start state** $S_0 \in Q$
- **Q_F** ... unempty set of **final states** $\emptyset \neq Q_F \subseteq Q$

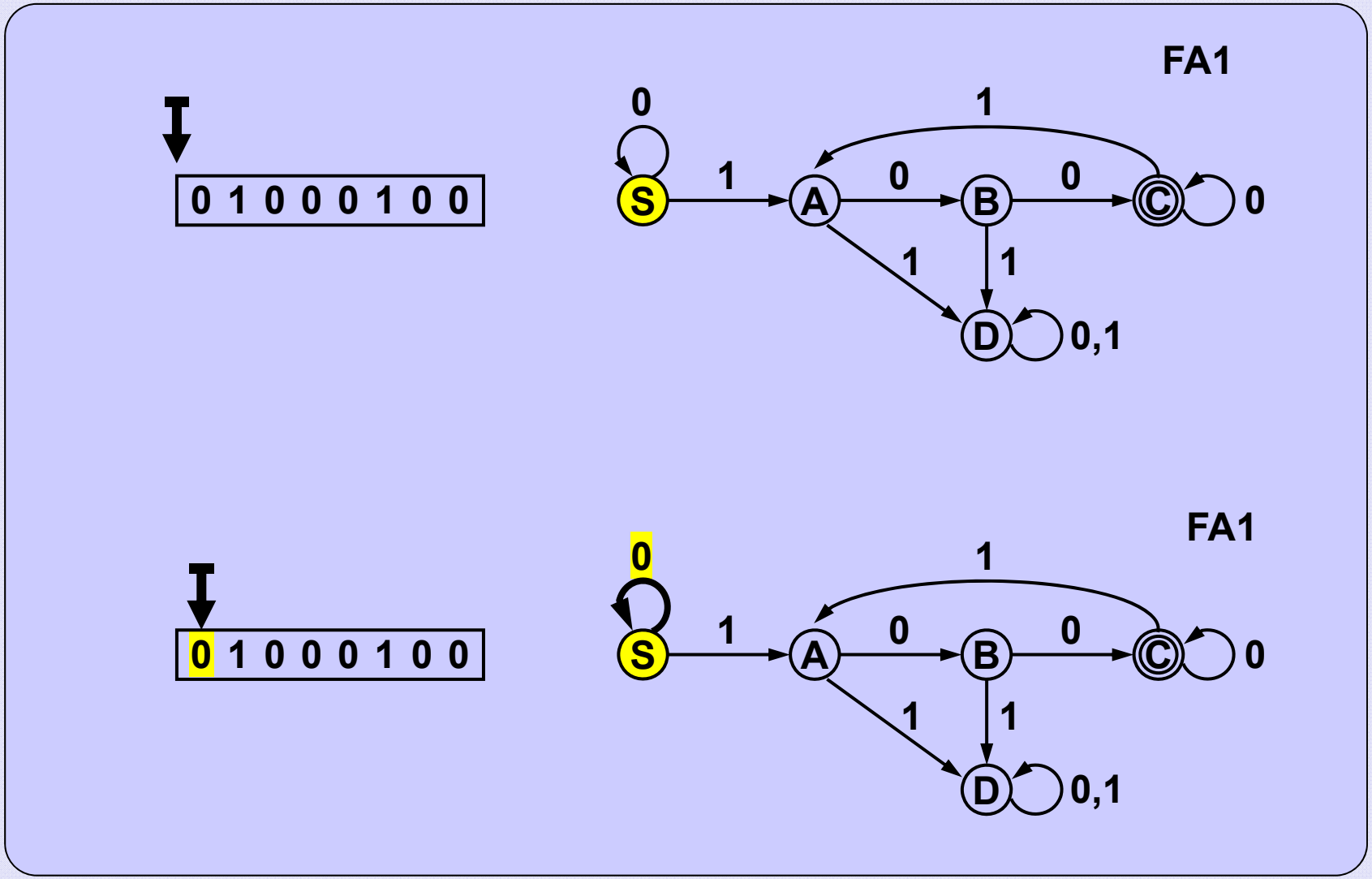
Automaton FA1:

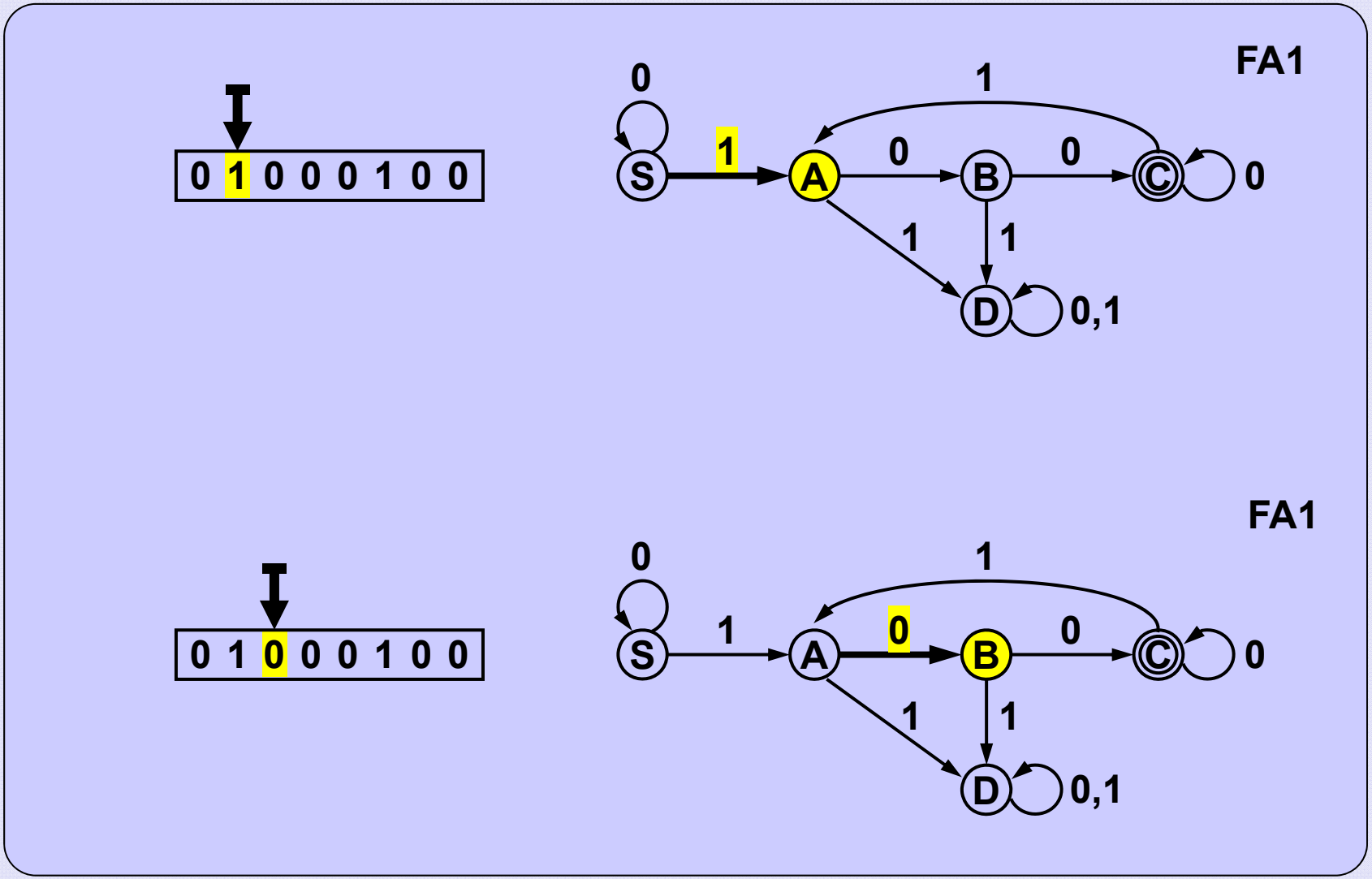
A … alphabet … {0,1}, |A| = 2
Q ... set of states {S, A, B, C, D }
σ ... transition function ... σ: Q×A → Q : {
σ(S,0) = S, σ(A,0) = B, σ(B,0) = C, σ(C,0) = C, σ(D,0) = D,
σ(S,1) = A, σ(A,1) = D, σ(B,1) = D, σ(C,1) = A, σ(D,1) = D }
S_0 ... start state S ∈ Q
Q_F … unempty set of final states ∅ ≠ { C } ⊆ Q

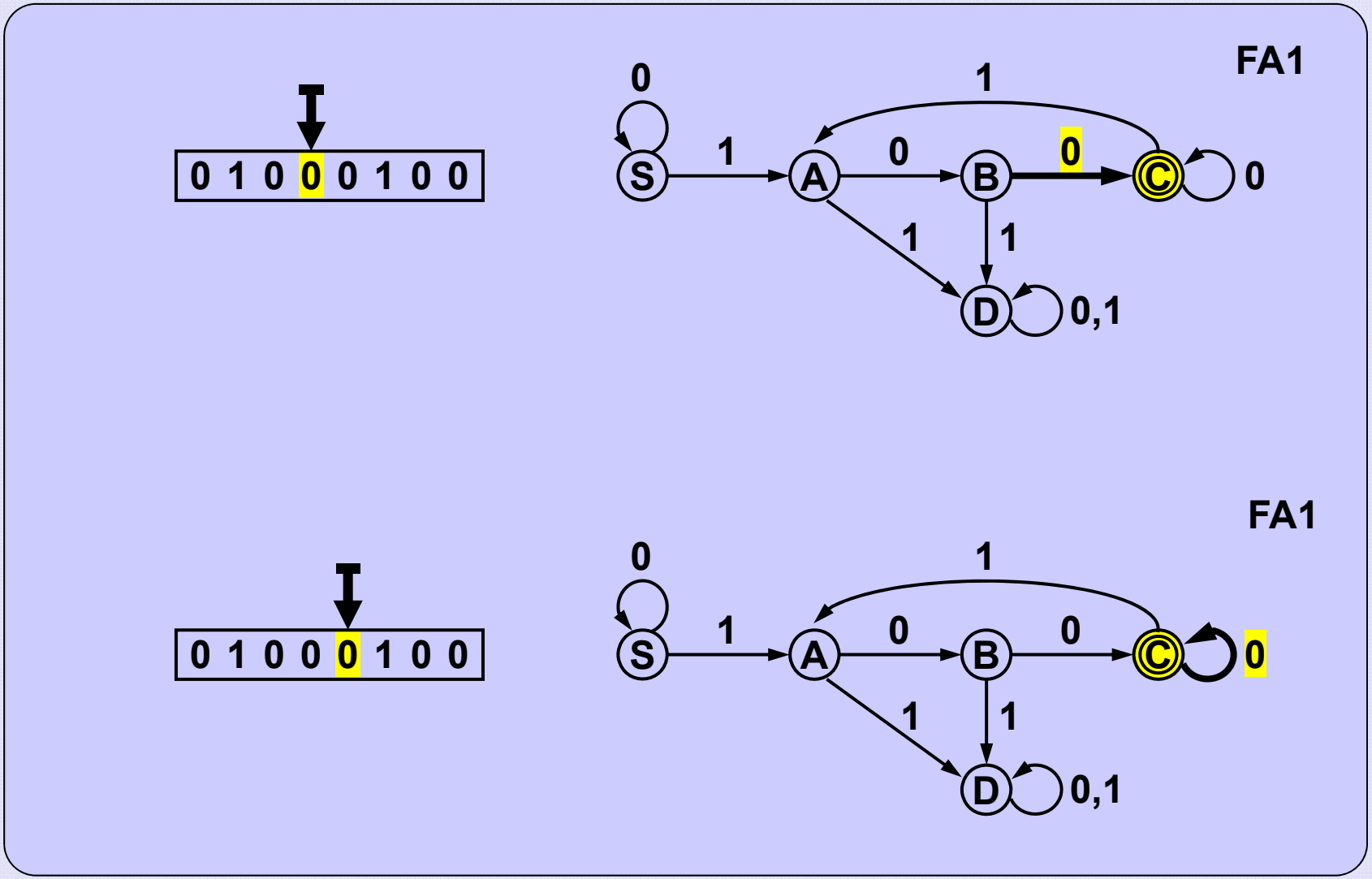
Transition diagram of the automaton FA1

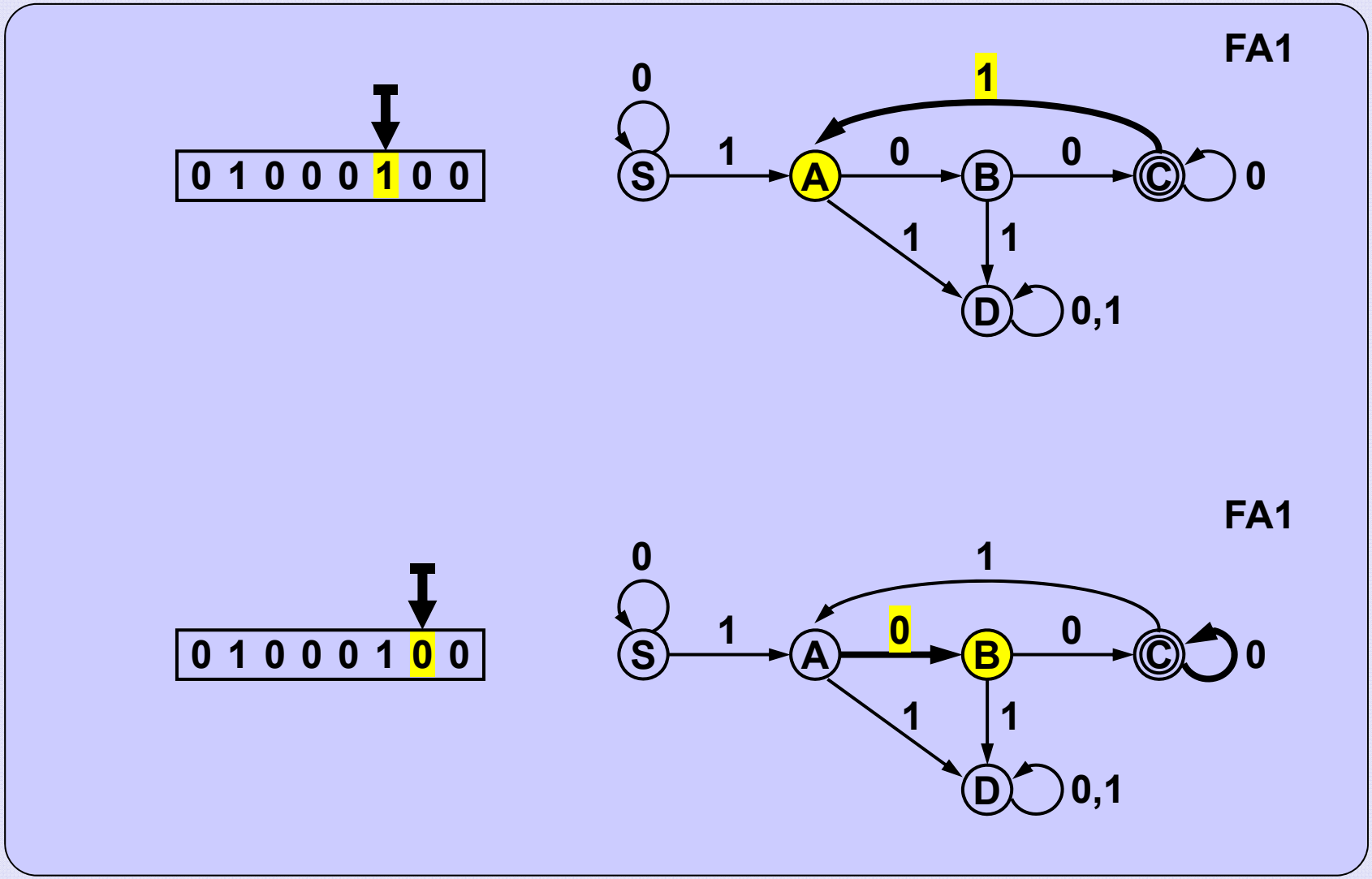


FA1

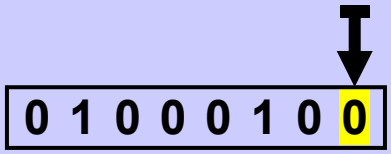

FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1
FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1

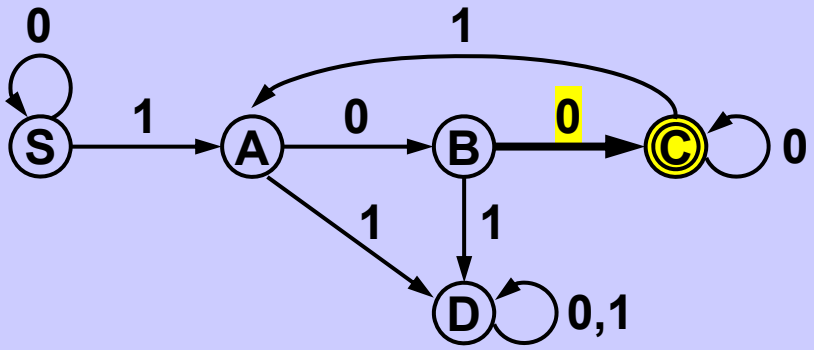

0 1 0 0 0 1 0 0
FA1
S
A
B
C
D
0
1
0,1
0 1 0 0 0 1 0 0
FA1
S
A
B
C
D
0
1
0,1


FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1
FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1


FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1
FA1
0 1 0 0 0 1 0 0
S
A
B
C
D
0
1
0,1

FA1

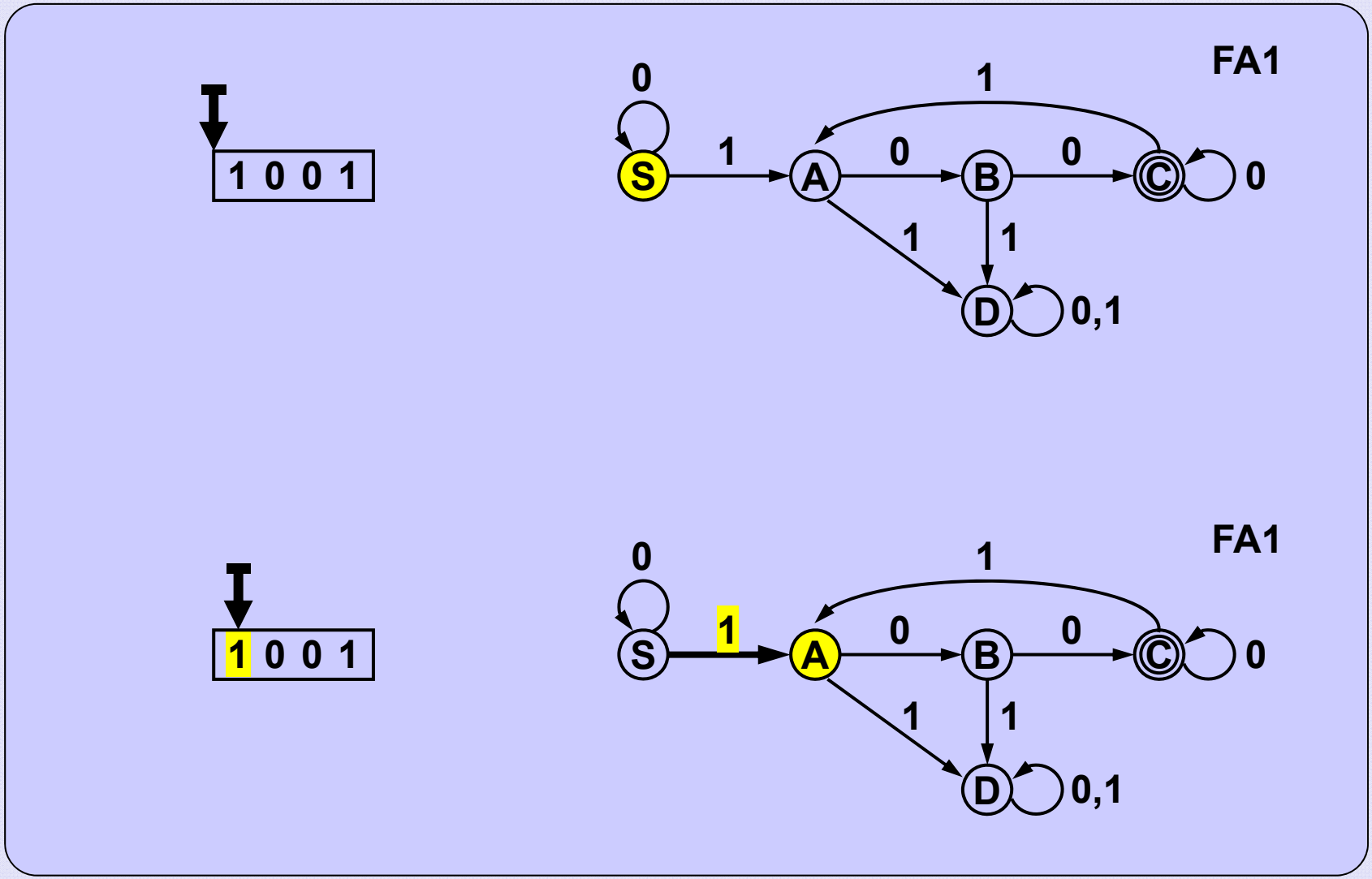


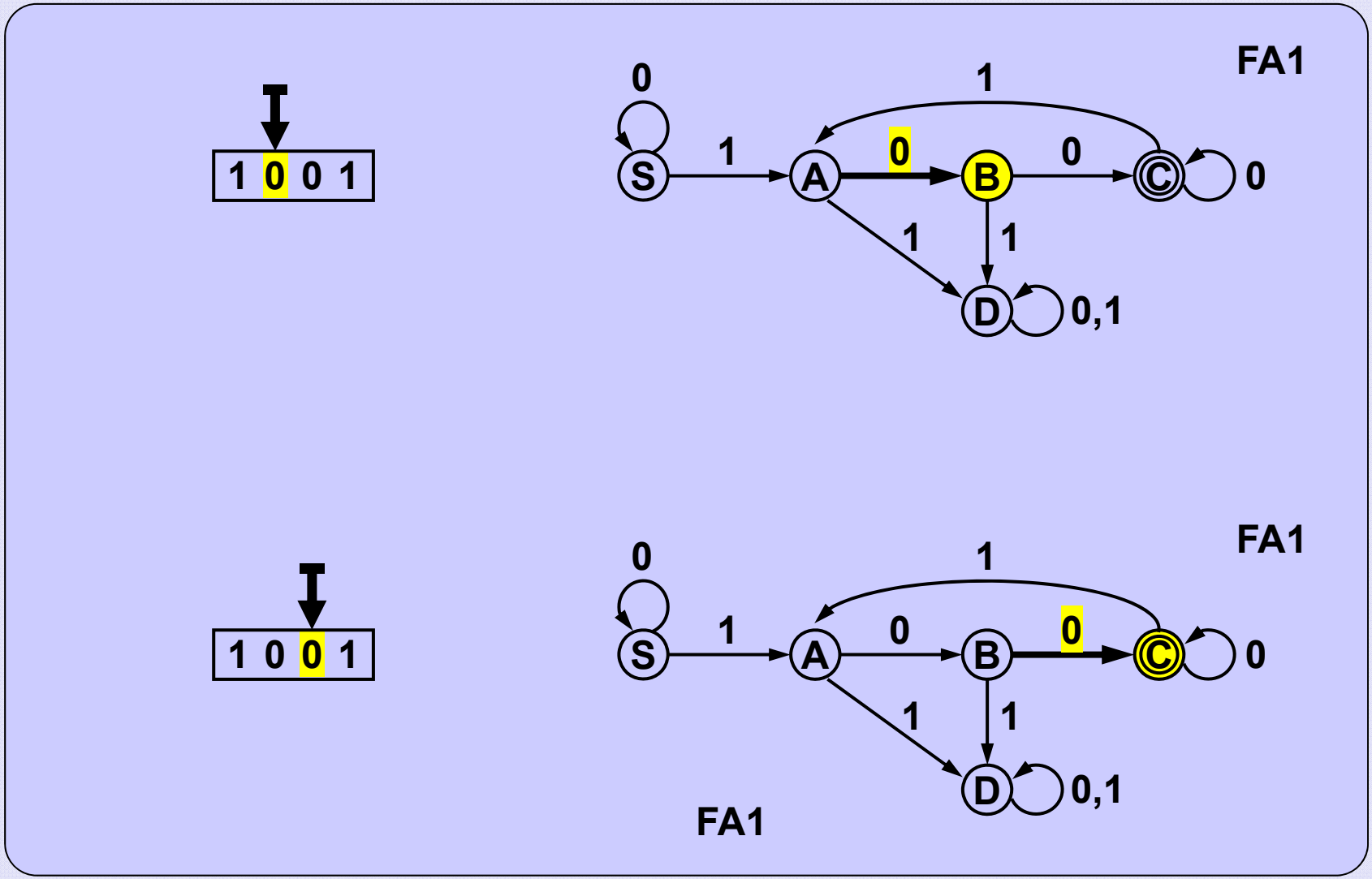


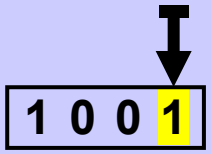
When the last word symbol is read automaton FA1 is in final state ◎

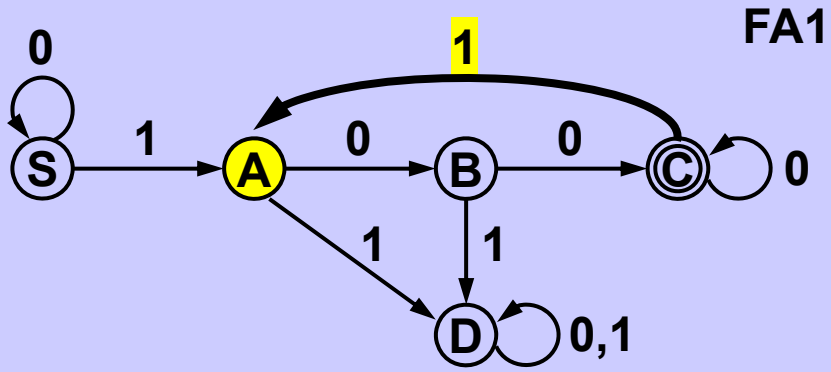
$\Rightarrow$

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


1 0 0 1
FA1
S
A
B
C
D
0
1
0,1
1 0 0 1
FA1
S
A
B
C
D
0
1
0,1


1 0 0 1
FA1
0
1
S
1
A
0
B
0
C
0
1
1
D
0,1
1 0 0 1
FA1
0
1
S
1
A
0
B
0
C
0
1
1
D
0,1
FA1

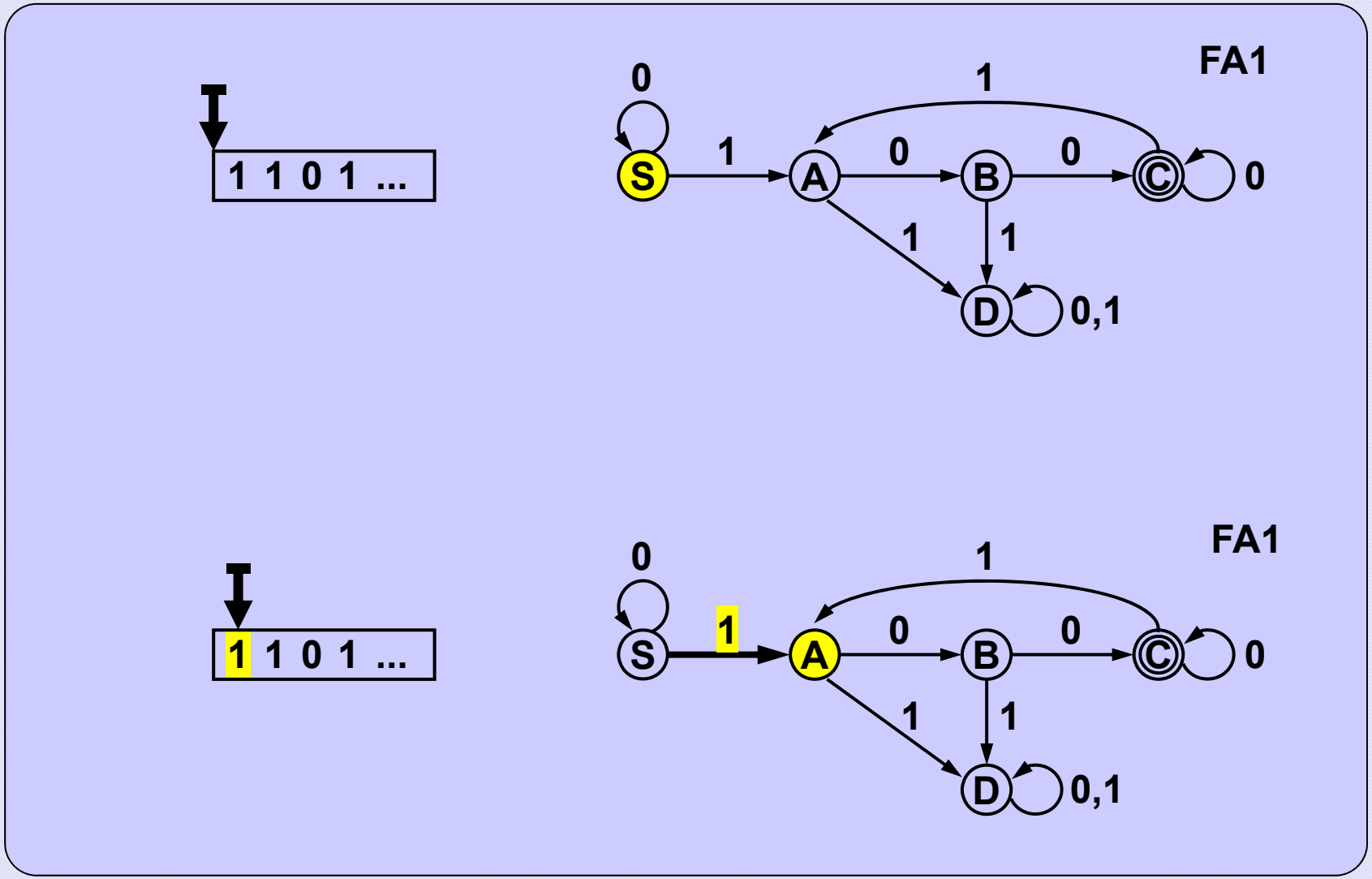


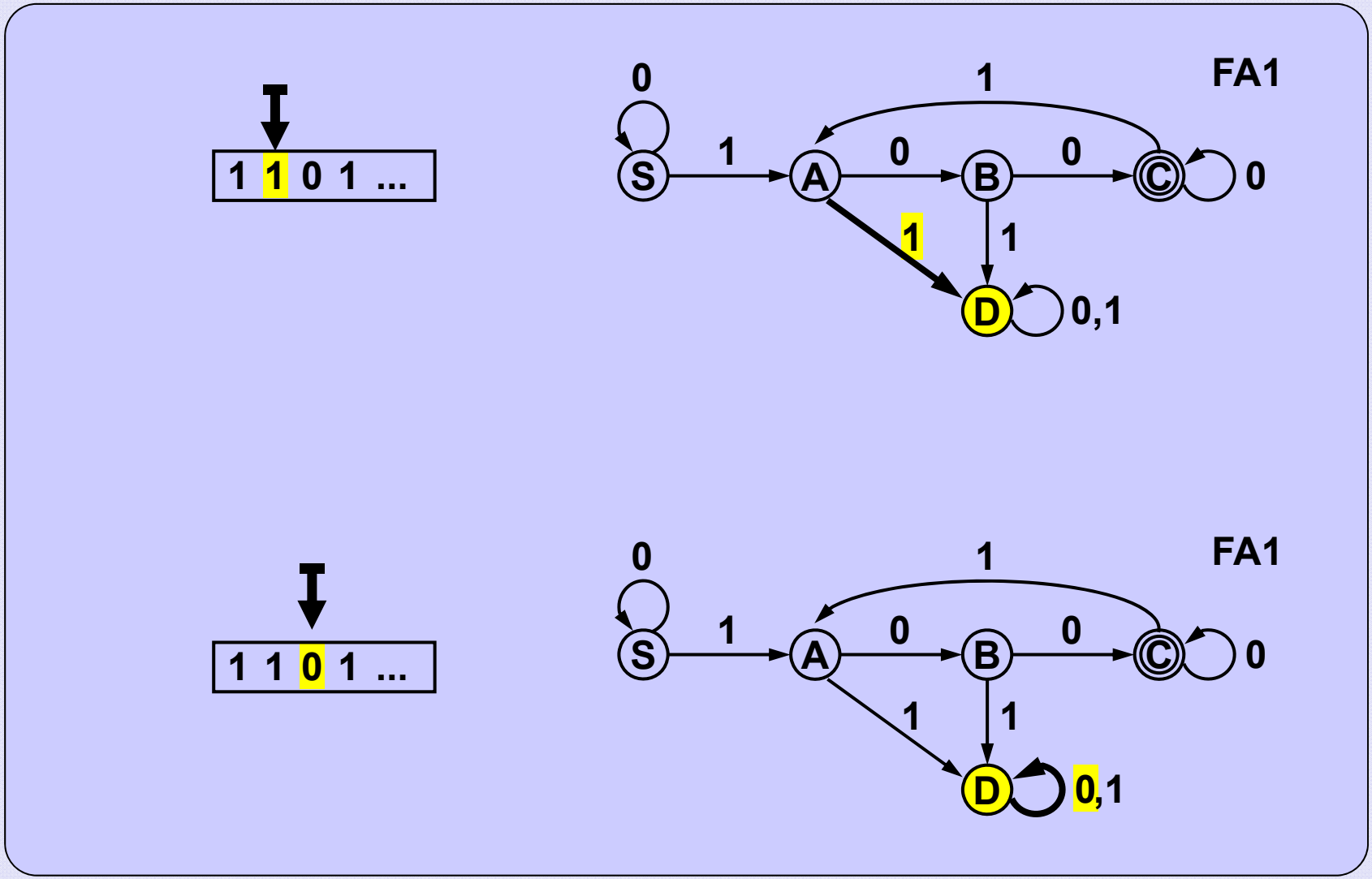


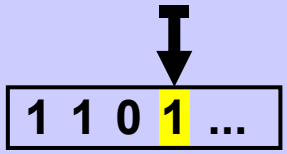
When the last word symbol is read automaton FA1 is in a state which is not final ◯

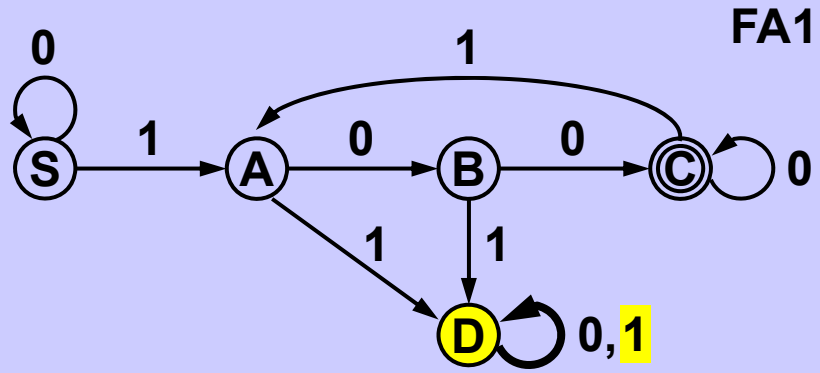
$\Rightarrow$

Word 1 0 0 1 **<u>is not accepted</u> by automaton FA1**


1 1 0 1 ...
FA1
S
A
B
C
D
0
1
0,1
1 1 0 1 ...
FA1
S
A
B
C
D
0
1
0,1


1 1 0 1 ...
FA1
0
1
S
A
B
C
D
0,1
1 1 0 1 ...
FA1
0
1
S
A
B
C
D
0,1





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 begining 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,

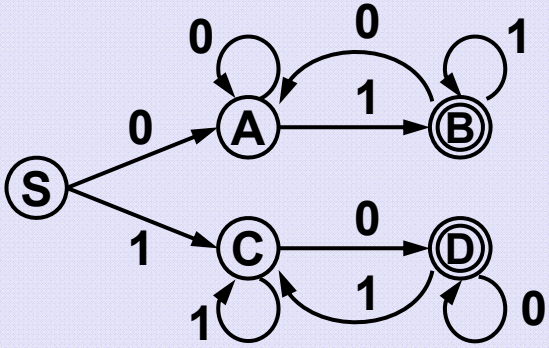
if it is not in a final state, we say that it does not accept the word.

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.



Automat FA2

Example of analysis of different words by FA2:

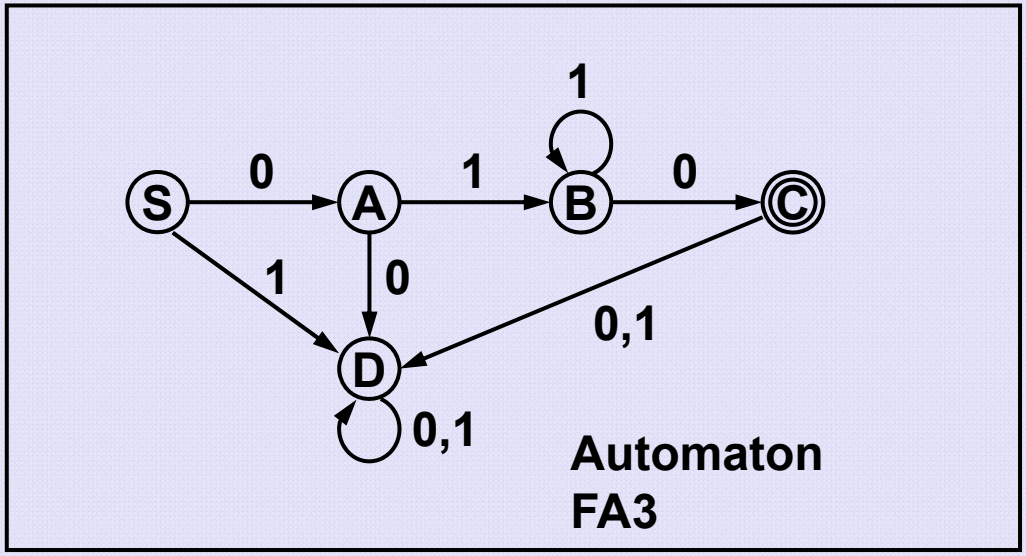
0 1 0 1 0 : (S),0 → (A),1 → (B),0 → (A),1 → (B),0 → (A)

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

1 0 1 1 0 : (S),1 → (C),0 → (D),1 → (C),1 → (C),0 → (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,
... **}**



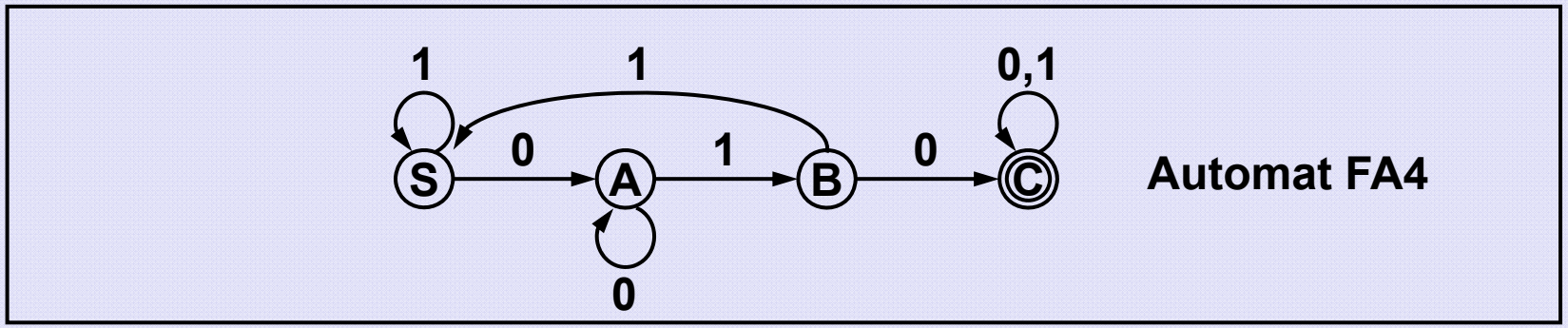
Example of analysis of different words by FA3:

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:

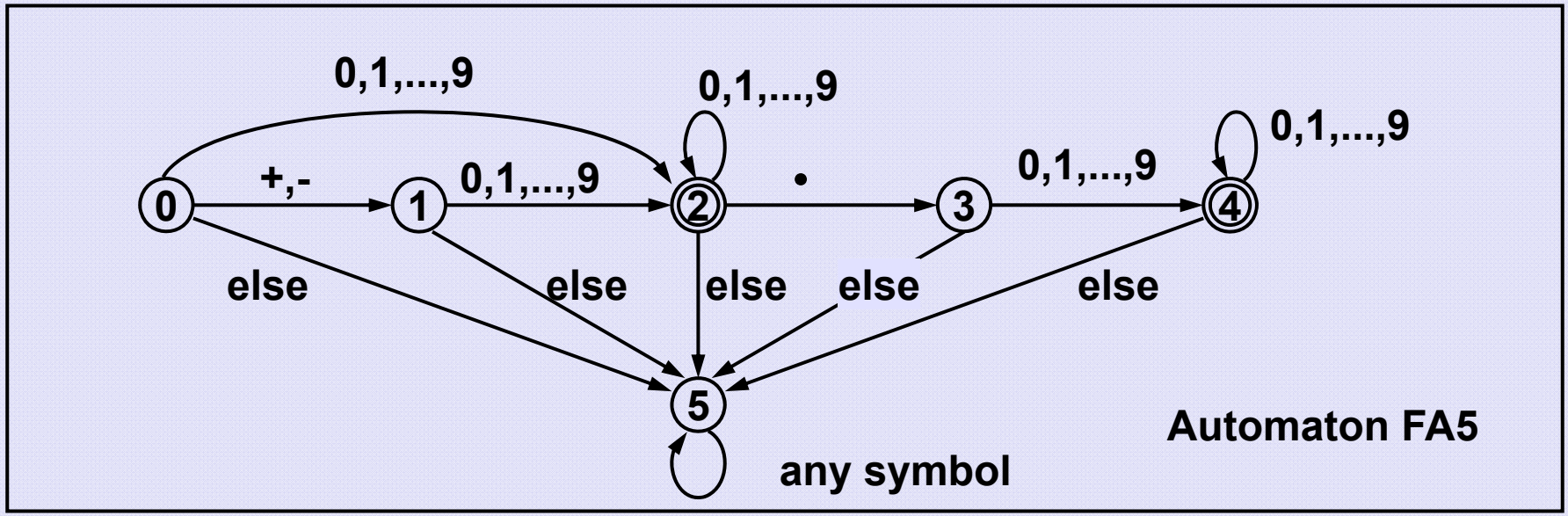
0 0 1 0 1 : (S),0 → (A),0 → (A),1 → (B),0 → (C),1 → (C)

(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, …, 8, 9, … } whose words represent decimal numbers



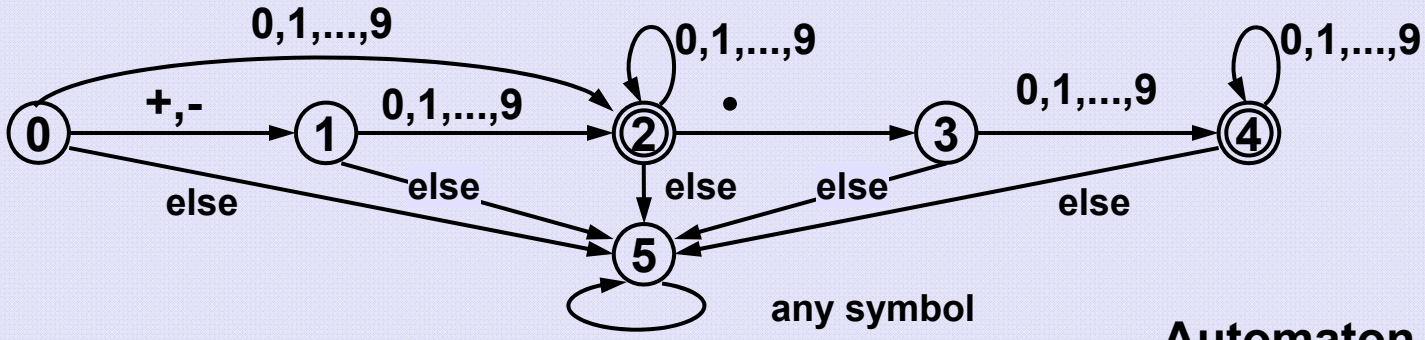
Example of word analysis

+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.



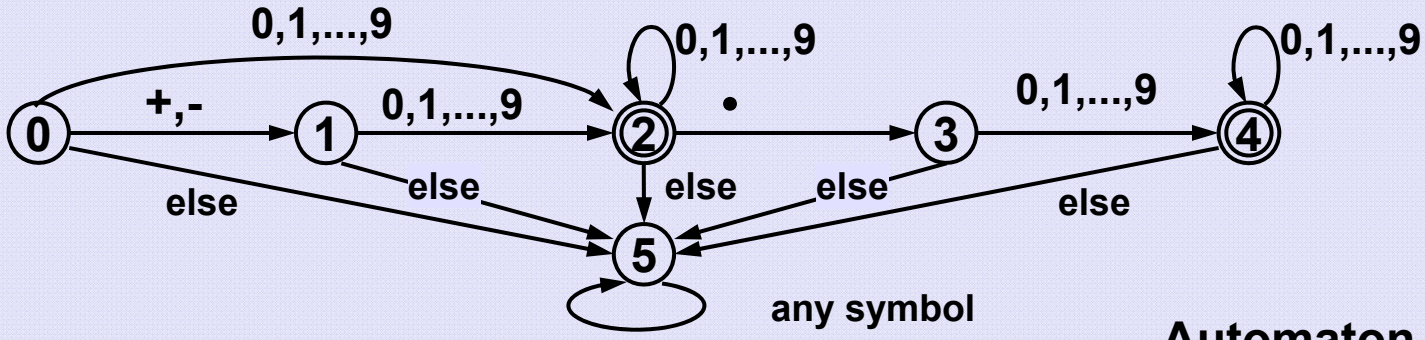
Automaton FA5

Code of the finite automaton

(The word which is being read is stored in the array text):

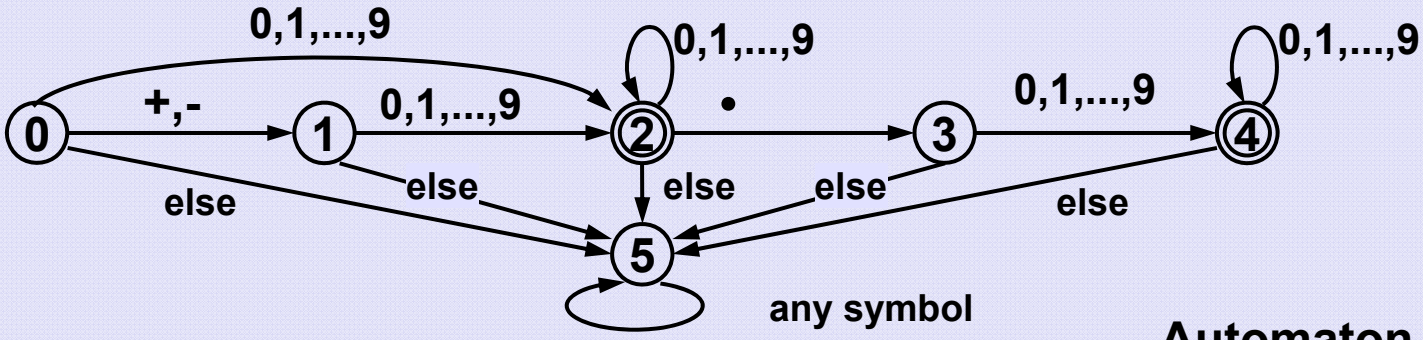
```
boolean isDecimal( char [] text ) {
int state = 0;

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



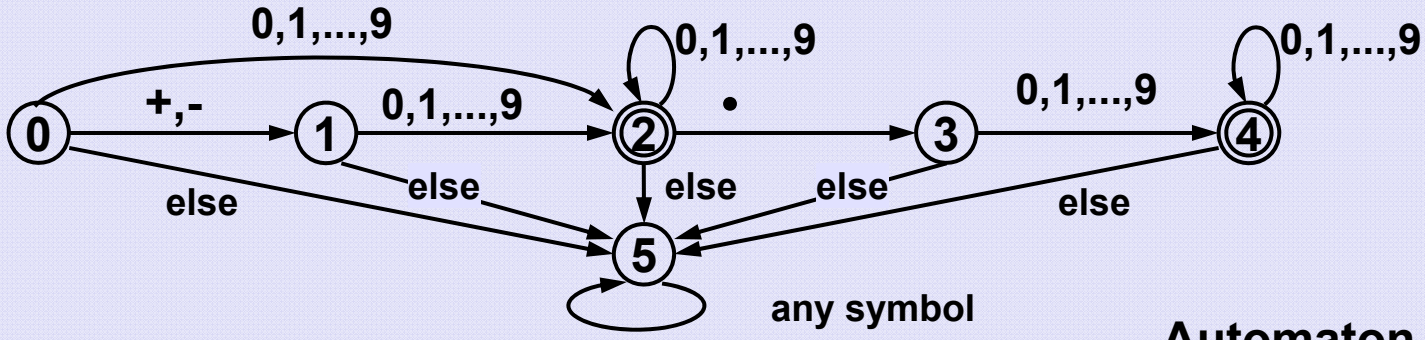
Automaton FA5

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



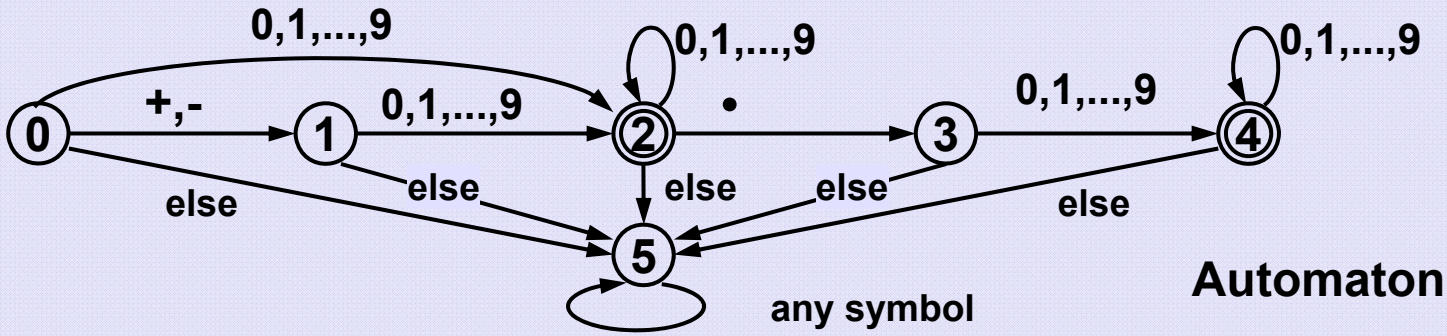
Automaton FA5

```
② case 2:
    if ((text[i] >= '0') && (text[i] <= '9')) state = 2;
    else
    if (text[i] == '.') state = 3;
    else state = 5;
    break;
③ case 3:
    if ((text[i] >= '0') && (text[i] <= '9')) state = 4;
    else state = 5;
    break;
```



Automaton FA5

```
④   case 4:
      if ((text[i] >= '0') && (text[i] <= '9')) state = 4;
      else state = 5;
      break;
⑤  case 5: break;                    // no need to react anyhow
    default : break;
    } // end  switch
  } // end for
return (state == 2) || (state == 4);   // final states
}
```



Automaton FA5

Transition table of automaton FA5

states	alphabet 0	1	2	3	4	5	6	7	8	9	.	+	–	%	=	...	}	final
0	2	2	2	2	2	2	2	2	2	2	5	1	1	5	5	...	5	0
1	2	2	2	2	2	2	2	2	2	2	5	5	5	5	5	...	5	0
2	2	2	2	2	2	2	2	2	2	2	3	5	5	5	5	...	5	1
3	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	...	5	0
4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	...	5	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	...	5	0

```
boolean isAccepted( char [] text, int [][] TT, boolean [] F ){
  int state = 0;   // start state
  for( char c: text ){
    state = TT[state][Integer.valueOf(c)];
  }
  return F[state];
}
```

Tables TT and F specify the automaton completely (provided start state is typically 0), their construction is problem/implementation dependent and should not influence the operation(s) of the automaton.

alphabet	0	1	2	3	4	5	6	7	8	9	.	+	–	%	=	...	}	final
states 0	2	2	2	2	2	2	2	2	2	2	5	1	1	5	5	...	5	0
1	2	2	2	2	2	2	2	2	2	2	5	5	5	5	5	...	5	0
2	2	2	2	2	2	2	2	2	2	2	3	5	5	5	5	...	5	1
3	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	...	5	0
4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	...	5	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	...	5	0

Pros:

Simplicity an completeness
An automaton defines all words in the language unambiguously.
There is no need of additional code/methods to check if a word is "correct" or "acceptable" or whatever.

Speed
Time spent on each symbol in an input word (text) is constant.
Input of length N is processed in

$O(N)$ time.

Cons:

Limited class of languages
A finite automaton can recognize only regular languages,
the smallest class of languages in Chomsky hierarchy.