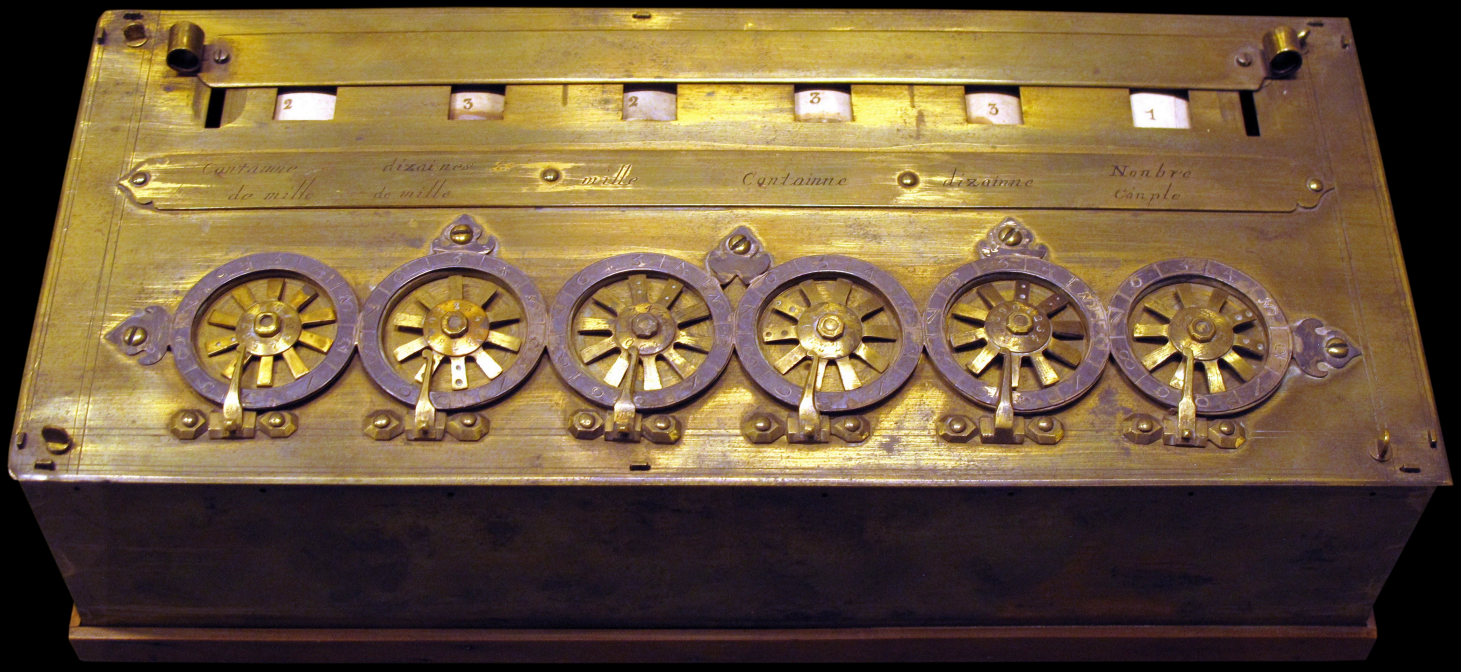
# "Pascaline"


2
3
2
3
3
1
Centaine de mille
dizaines de mille
mille
Centaine
dizaine
Nombre Simple

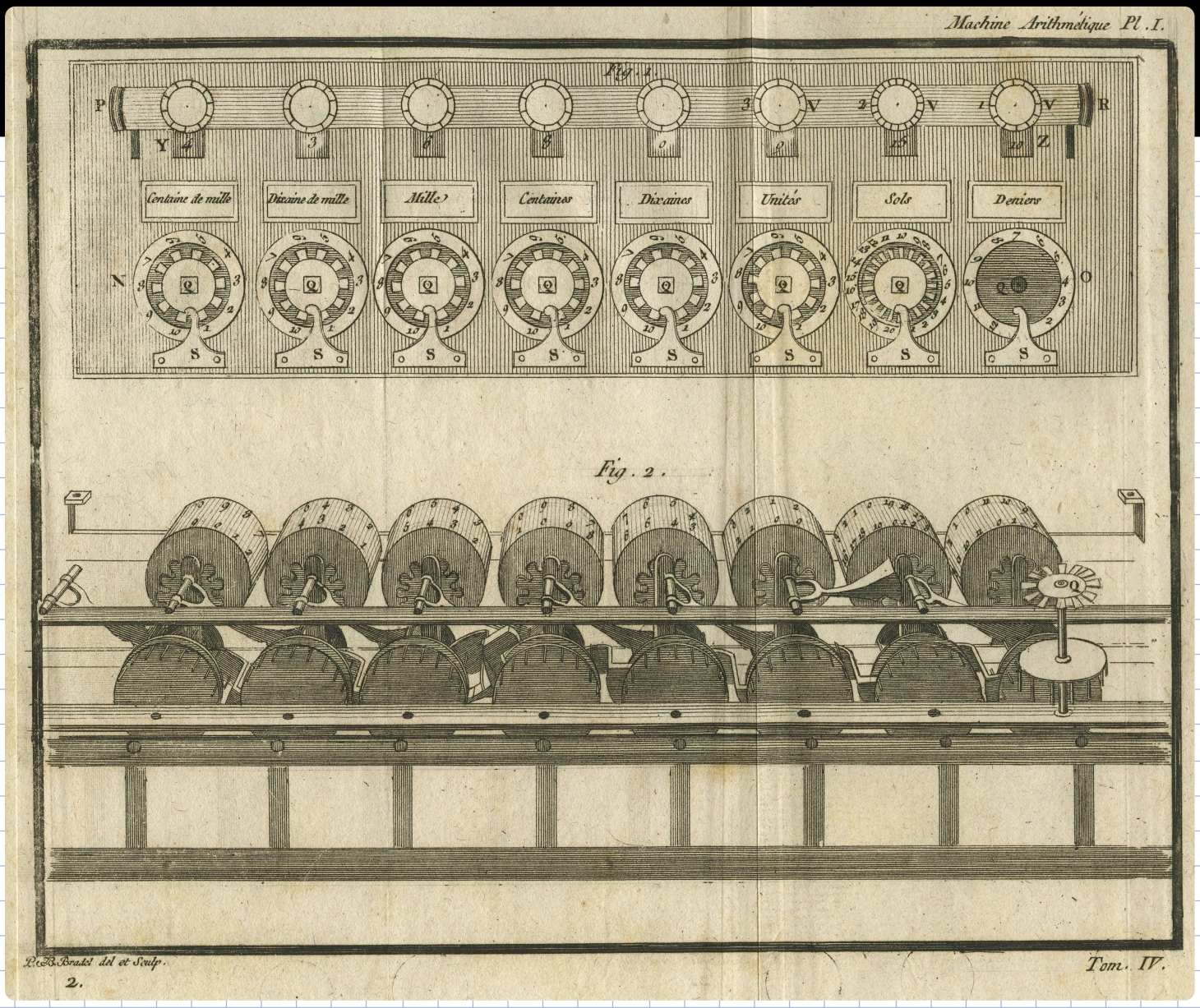

Machine Arithmétique Pl. I.
Fig. 1.
Centaine de mille
Dizaine de mille
Mille
Centaines
Dizaines
Unités
Sols
Deniers
Fig. 2.
P. & B. Bradel del et Sculp.
2.
Tom. IV.

Fig. 3.

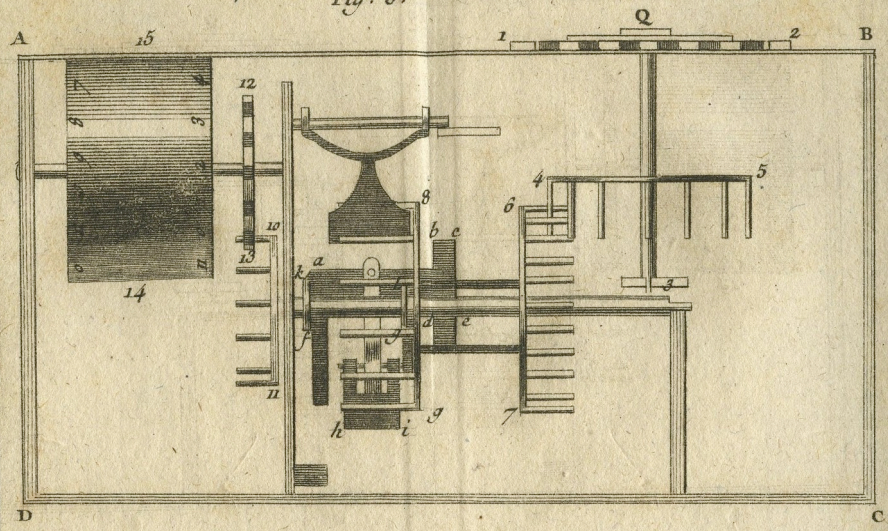



Fig. 6.

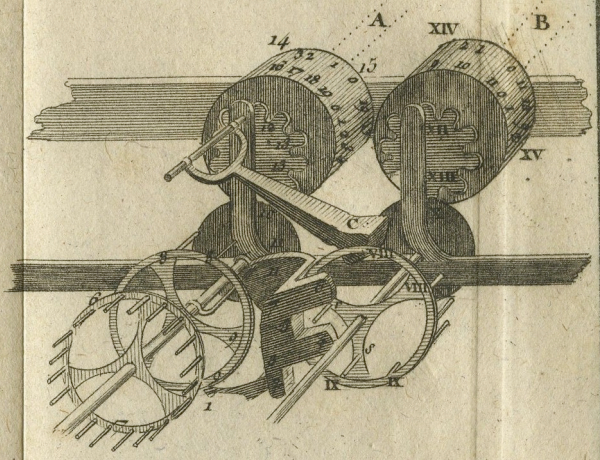



Fig. 4.

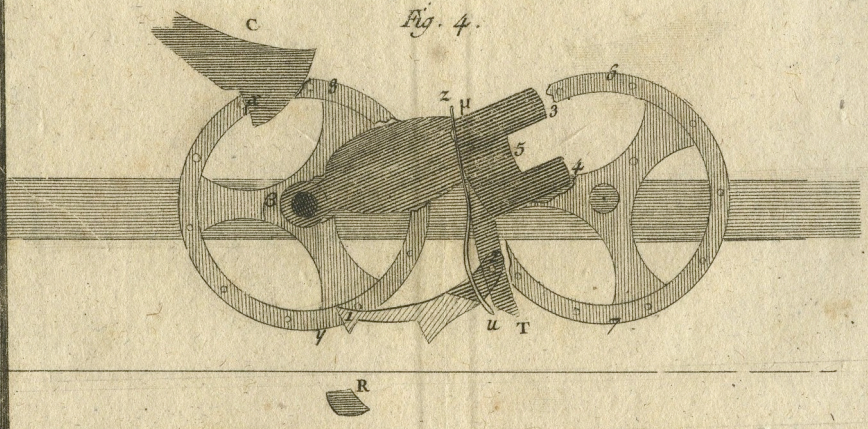
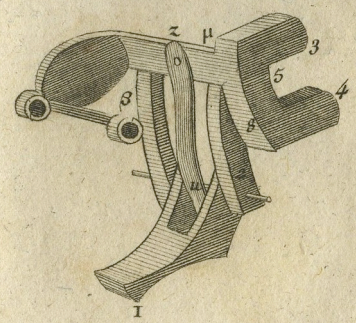


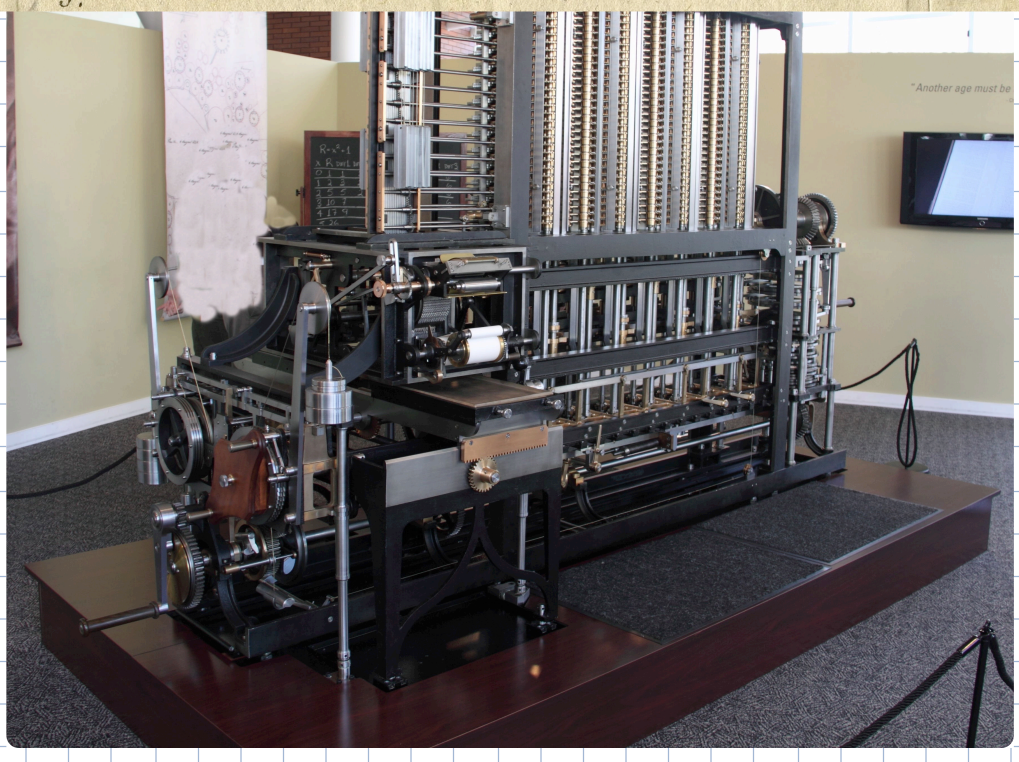

Fig. 5.

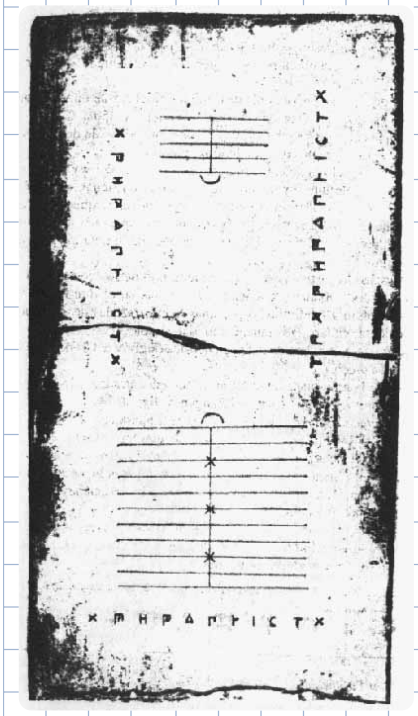


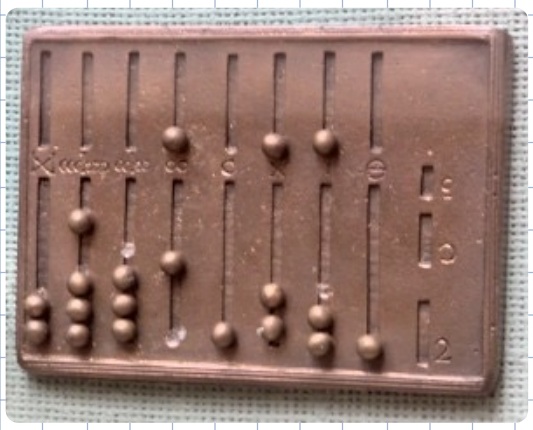

P. JB. Bradel del et Sculp.

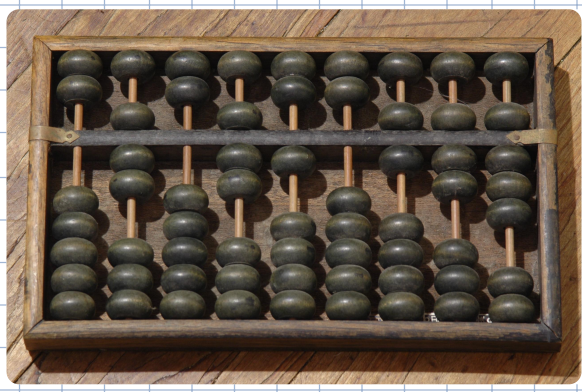
Tom. IV.

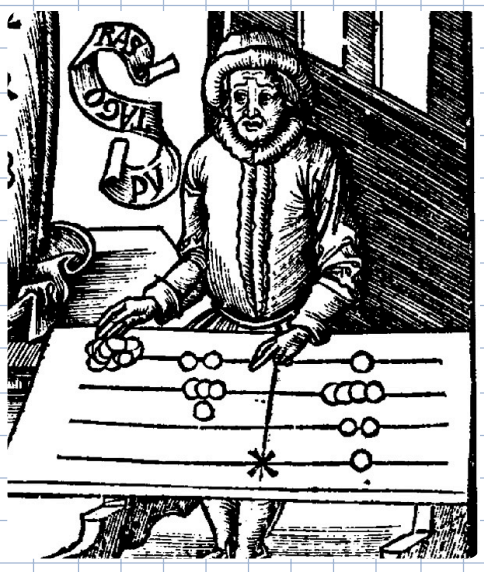
3.

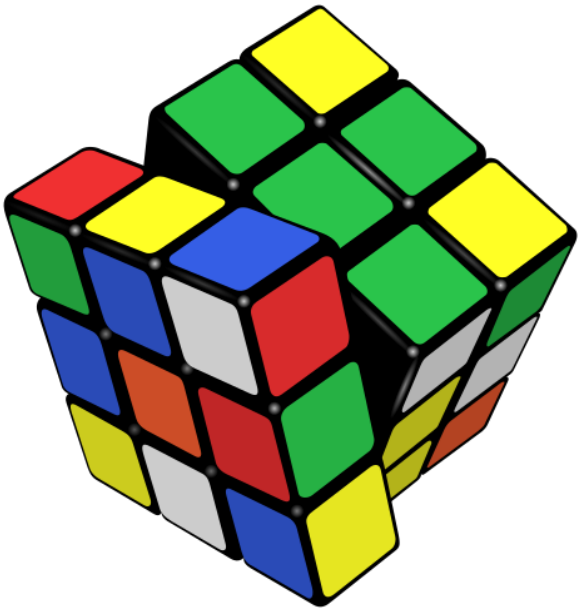










| | |
|---|---|
| Finite set of states | $Q$ |
| Initial/start state | $s$ |
| Input alphabet | $\Sigma$ |
| Transition function | $\delta : Q \times \Sigma \to Q$ |
| Accepting states | $A$ |

DFA
FSM

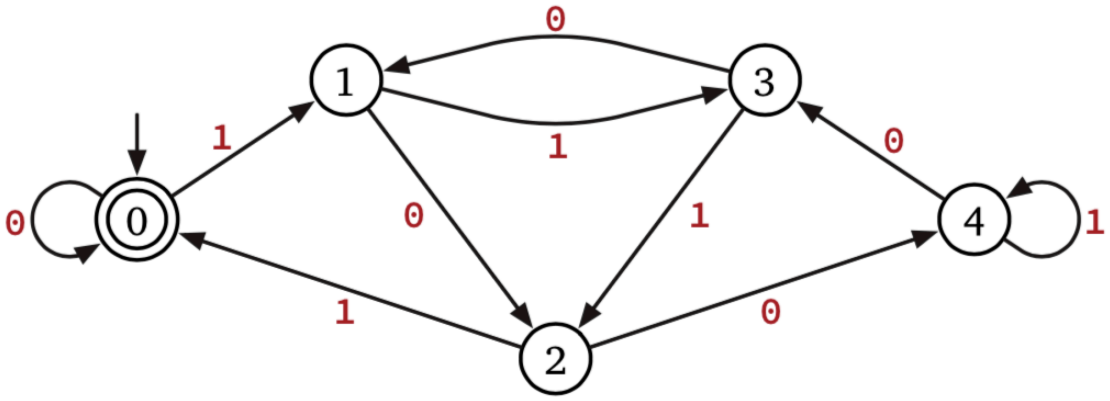
$Q = \{0,1,2,3,4\}$

$s = 0$

```
MULTIPLEOF5(w[1..n]):
  rem ← 0
  for i ← 1 to n
      rem ← (2 · rem + w[i]) mod 5
  if rem = 0
      return TRUE
  else
      return FALSE
```

$\delta(q,a) = (2q+a) \bmod 5$

$A = \{0\}$


State-transition graph for MULTIPLEOF5

| $q$ | $\delta[q,0]$ | $\delta[q,1]$ | $A[q]$ |
|---|---|---|---|
| → 0 | 0 | 1 | TRUE |
| 1 | 2 | 3 | FALSE |
| 2 | 4 | 0 | FALSE |
| 3 | 1 | 2 | FALSE |
| 4 | 3 | 4 | FALSE |

| w | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| q | 0 | 1 | 2 | 0 | 0 | 0 | 1 | 2 | 0 |

Accept!

State q means

(input so far) mod 5 = q

```
DoSomethingCool(w[1..n]):
  q ← s
  for i ← 1 to n
      q ← δ(q, w[i])
  return A[q]
```

Which strings does DFA accept?

Design DFA to accept given language $L$.

$$M = (Q, \Sigma, s, A, \delta)$$

$$\delta : Q \times \Sigma \to Q$$

Need $\delta^* : Q \times \Sigma^* \to Q$

Defined

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta^*(\delta(q,a), x) & \text{if } w = ax \end{cases}$$

$M$ accepts $w \iff \delta^*(s, w) \in A$

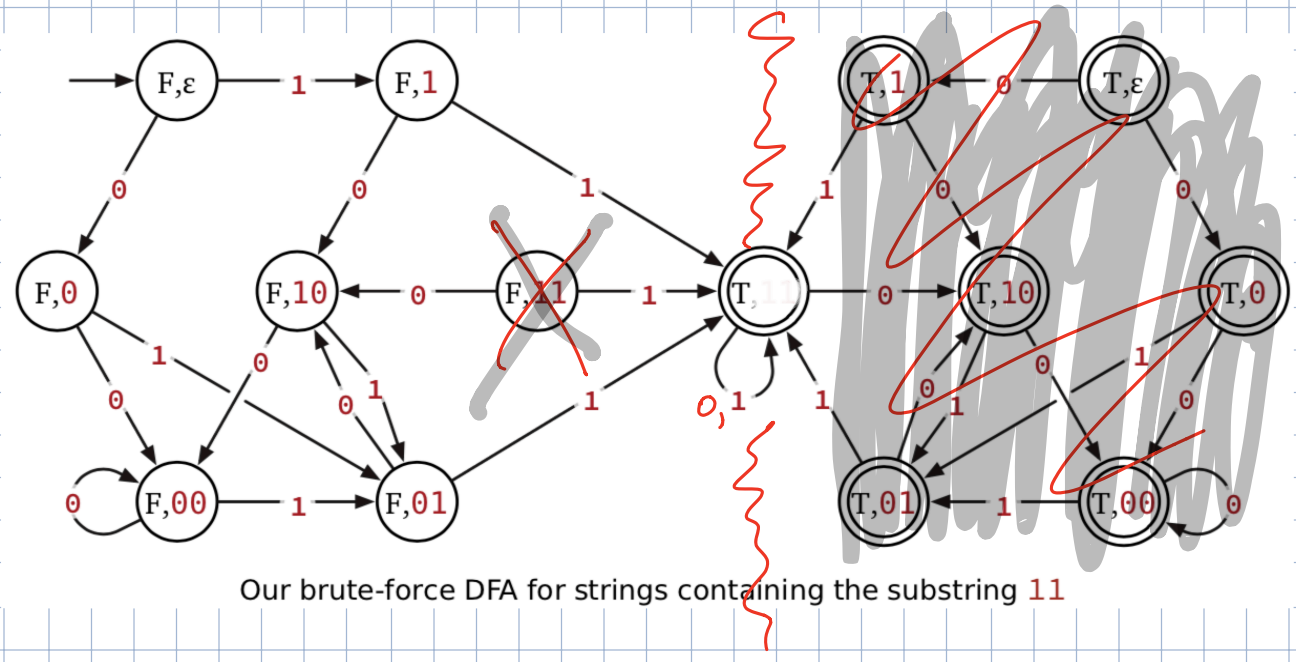
$$L(M) = \{ w \in \Sigma^* \mid \delta^*(s, w) \in A \}$$

$L = \{ w \in \{0,1\}^* \mid w \text{ contains } 11 \}$

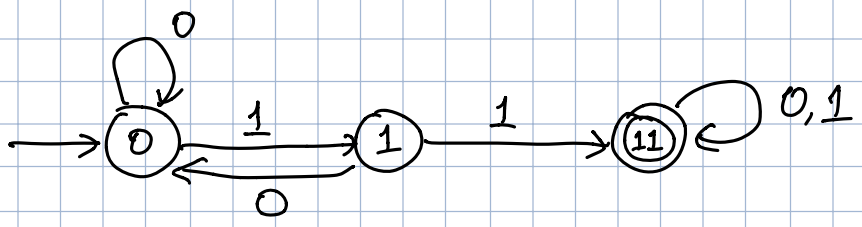
```
CONTAINS11(w[1..n]):
  found ← FALSE                    last2 ← ε
  for i ← 1 to n
      if i = 1
          last2 ← w[1]
      else
          last2 ← w[i−1] · w[i]
      if last2 = 11
          found ← TRUE
  return found
```

| $q$ | $\delta[q,0]$ | $\delta[q,1]$ |
|---|---|---|
| (FALSE, $\varepsilon$) | (FALSE, 0) | (FALSE, 1) |
| (FALSE, 0) | (FALSE, 00) | (FALSE, 01) |
| (FALSE, 1) | (FALSE, 10) | (TRUE, 11) |
| (FALSE, 00) | (FALSE, 00) | (FALSE, 01) |
| (FALSE, 01) | (FALSE, 10) | (TRUE, 11) |
| (FALSE, 10) | (FALSE, 00) | (FALSE, 01) |
| (FALSE, 11) | (FALSE, 10) | (TRUE, 11) |

| $q$ | $\delta[q,0]$ | $\delta[q,1]$ |
|---|---|---|
| (TRUE, $\varepsilon$) | (TRUE, 0) | (TRUE, 1) |
| (TRUE, 0) | (TRUE, 00) | (TRUE, 01) |
| (TRUE, 1) | (TRUE, 10) | (TRUE, 11) |
| (TRUE, 00) | (TRUE, 00) | (TRUE, 01) |
| (TRUE, 01) | (TRUE, 10) | (TRUE, 11) |
| (TRUE, 10) | (TRUE, 00) | (TRUE, 01) |
| (TRUE, 11) | (TRUE, 10) | (TRUE, 11) |




Our brute-force DFA for strings containing the substring 11




0 — last symbol (if any) was 0 and haven't seen 11

1 — last symbol was 1 and haven't seen 11

11 — seen 11