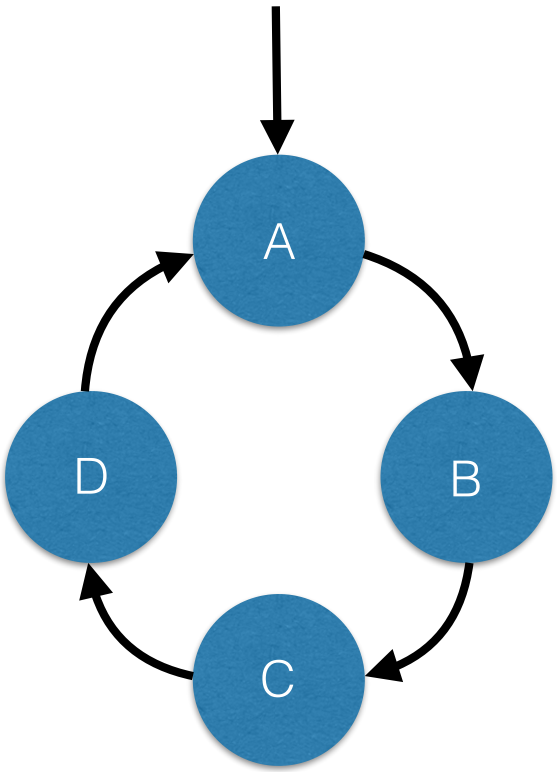
Finite-State Machines (Automata) lecture 12

cl

- a simple form of computation
- used widely
- one way to find patterns



current

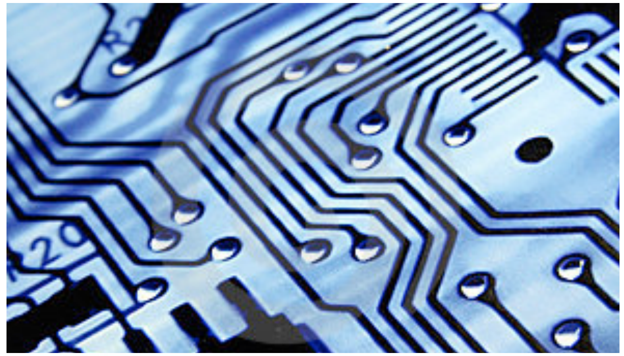
A	B	C	D
B	C	D	A

next

Application Fields

Industry

- real-time control, vending machines, cash dispensers, etc.



Electronic circuits

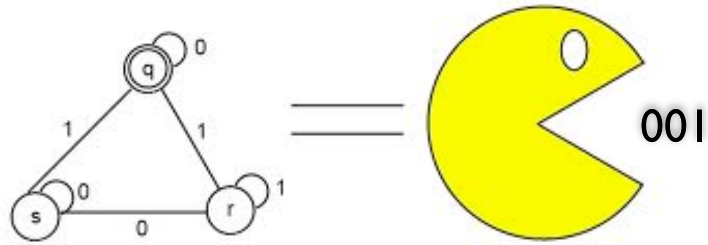
- data path / control path
- memory / cache handling
- protocols, USB, etc.

Communication protocols

- initiation and maintenance of communication links
- error detection and handling, packet retransmission

Language analysis

- natural languages
- programming languages
- search engines

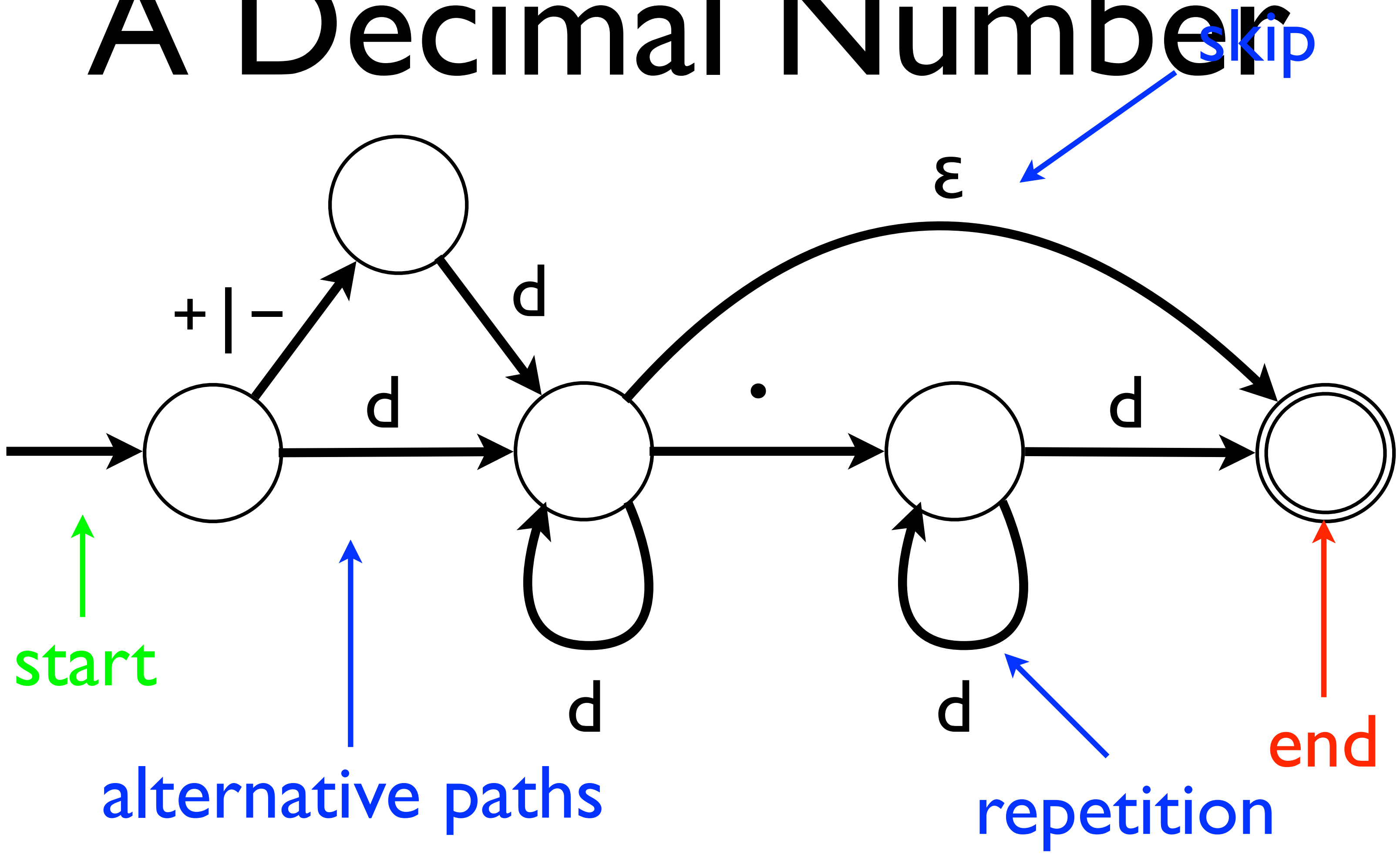


Finite State Machines

- A conceptual tool for modelling reactive systems.
- Not limited to software systems.
- Used to specify required system behaviour in a precise way.
- Then implement as software/hardware (and perhaps verify behaviour against FSM).

- Finite state machines can also be viewed as rules for generating sets of strings
(= sets of finite sequences of symbols)

A Decimal Number

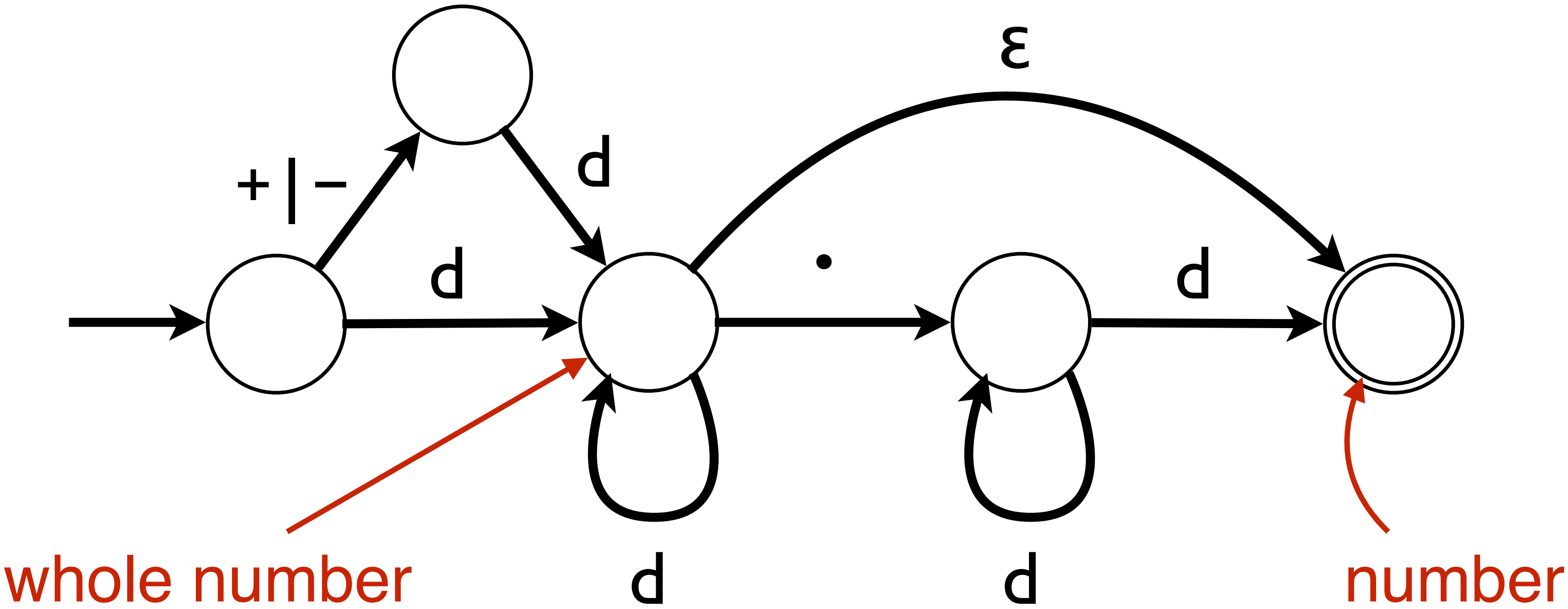


Each path from start to end, with a choice of one of the labels for each edge traversal, gives a decimal number.

d = {"0","1","2","3","4","5","6","7","8","9"}

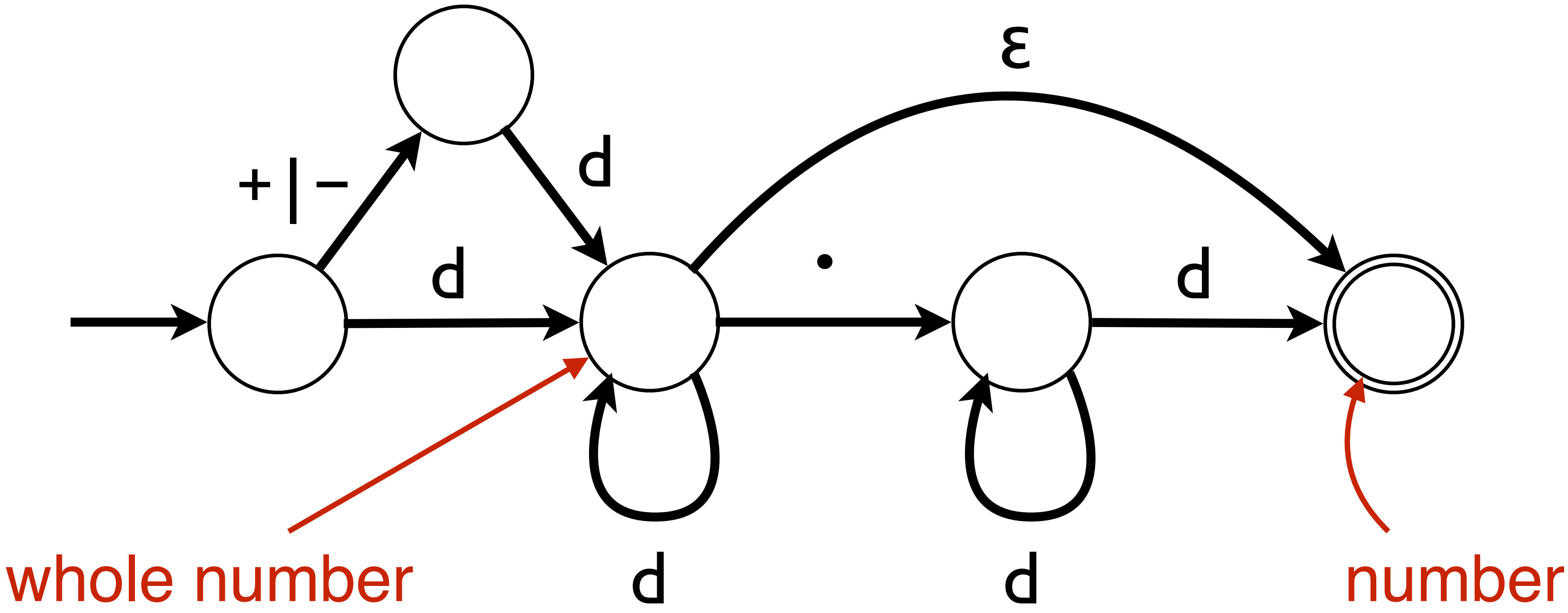
+|− = { "+","-" } ε = "" . = "."

This gives us a new way of defining sets by rules.



A whole number is a digit followed by any number of digits,
or a sign followed by a digit followed by any number of digits

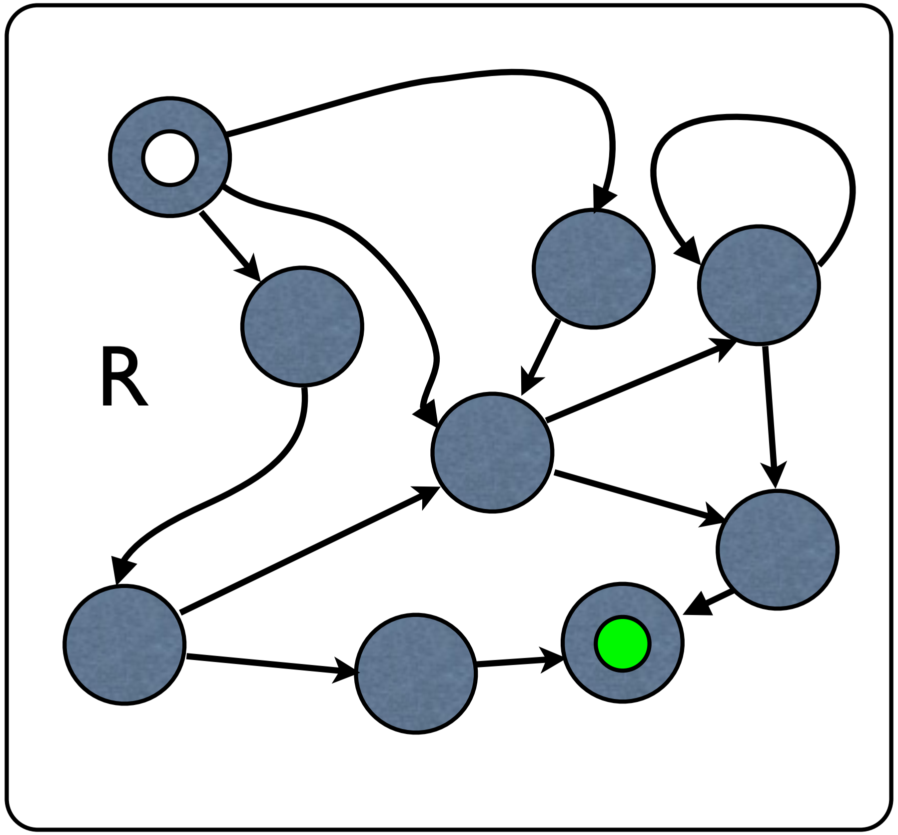
This gives us a new way of defining sets by rules.



A whole number is a digit followed by any number of digits,
or a sign followed by a digit followed by any number of digits.

A number is a whole number **or** a whole number followed by a decimal point followed by a digit followed by any number of digits

finite state machines

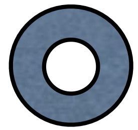


A labelled directed graph

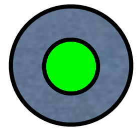
nodes 

edges 

with

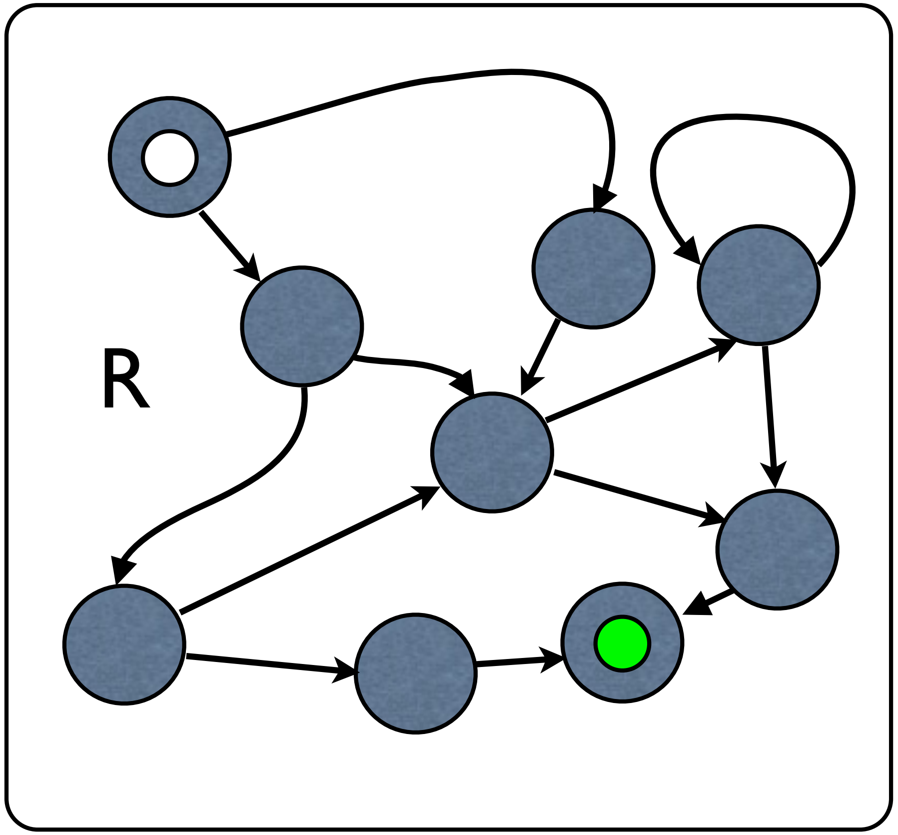
a set of start nodes 

and

a set of end nodes 

Each edge is labelled with a set of strings

finite state machines



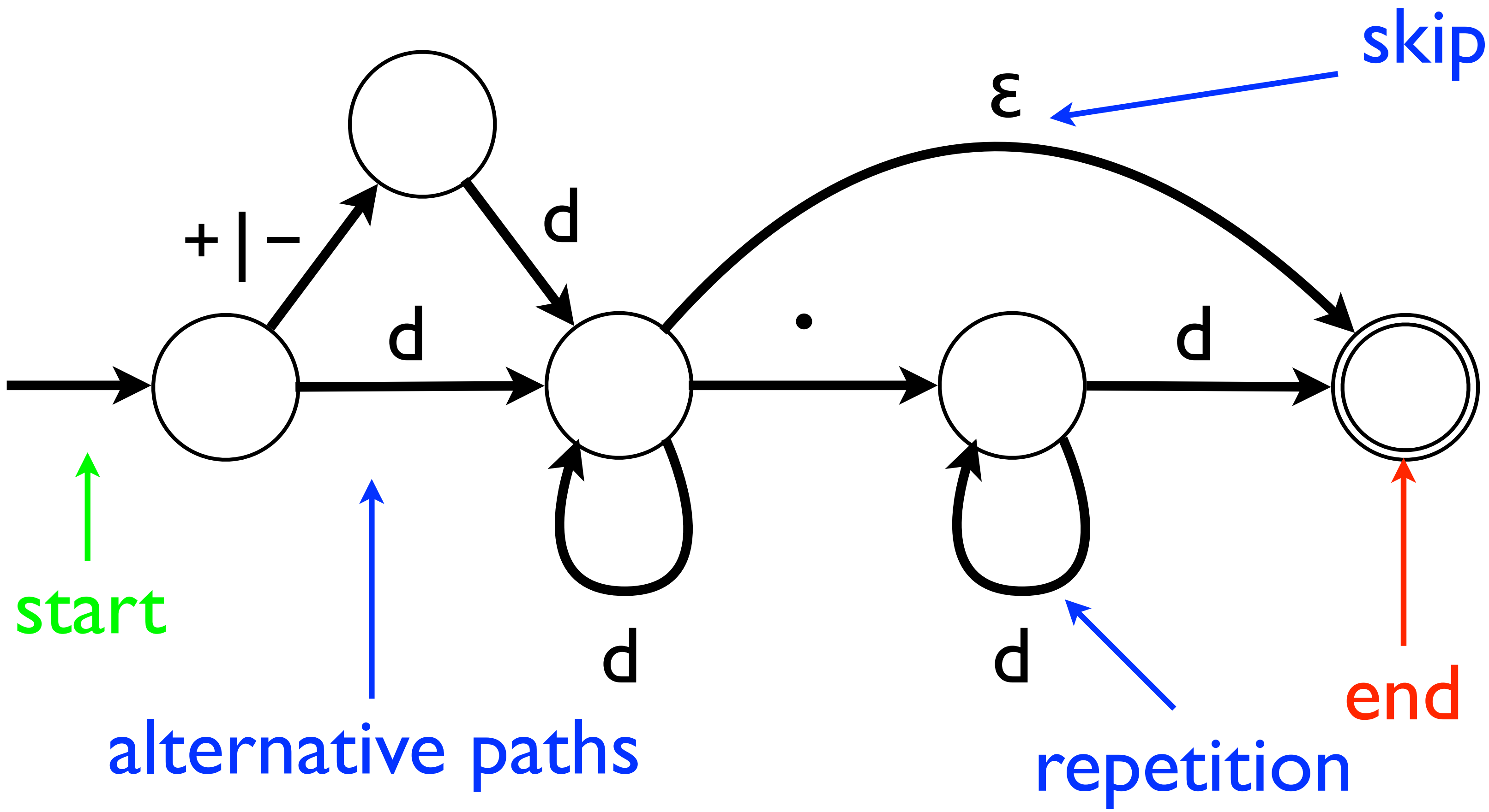
A labelled directed graph
nodes
edges
with
a set of start nodes
and
a set of end nodes

Each edge is labelled with
a set of strings

Each path from a start node to an end node, with a choice of one of the labels for each edge traversal, gives, by concatenation, a string.

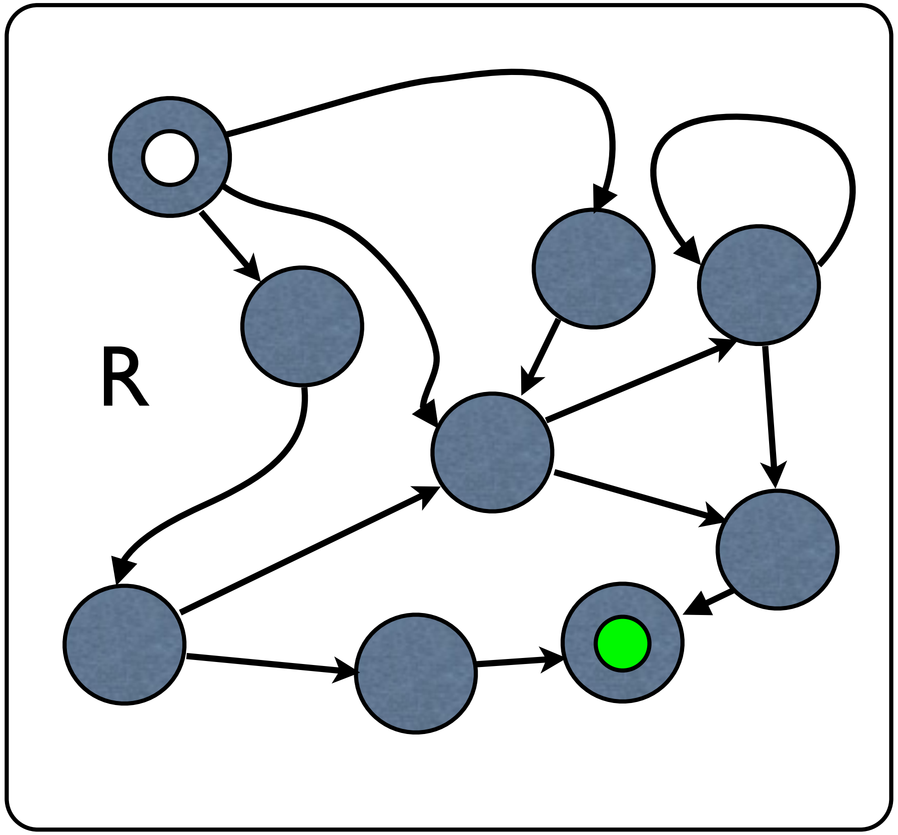
The **language defined by the machine**
is the **set of all such strings**

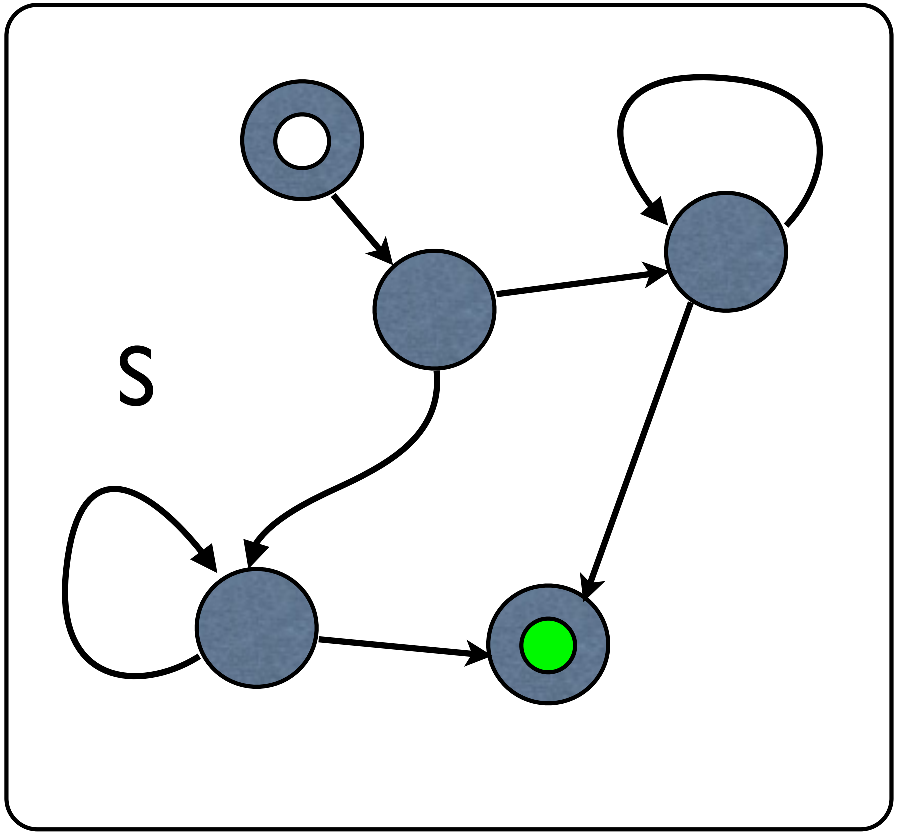
A Decimal Number



The language defined by this machine
is the set of all decimal numerals

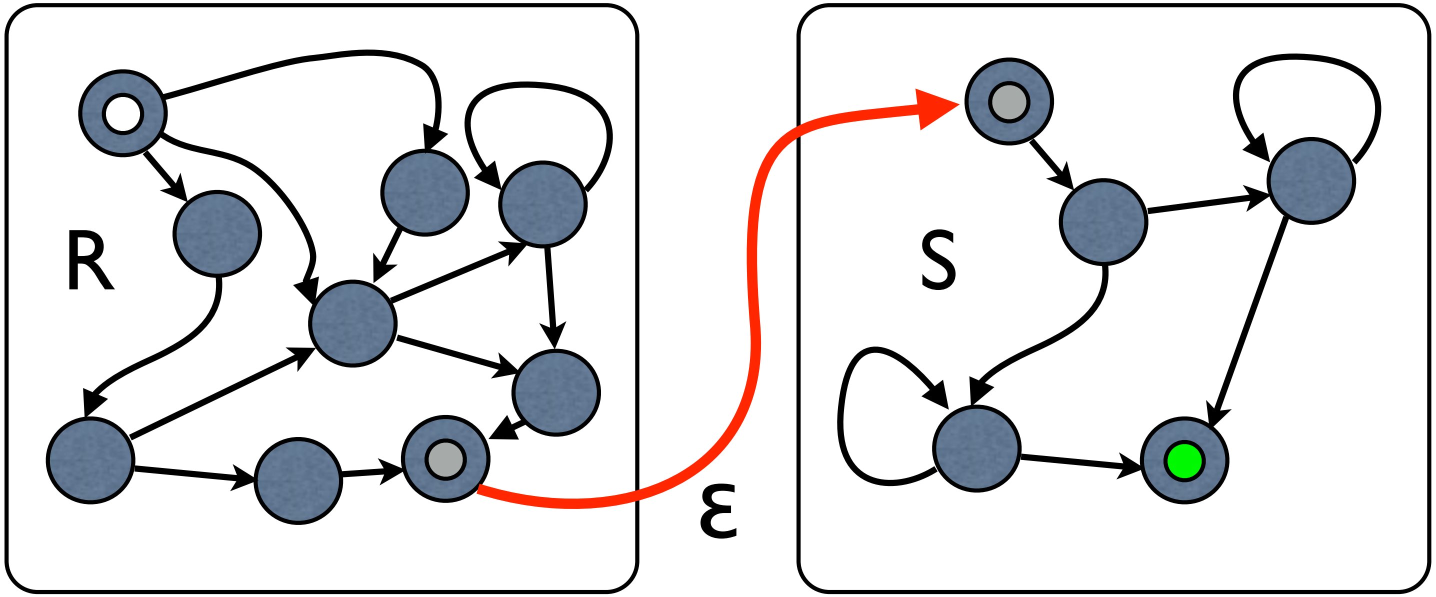
finite state machines





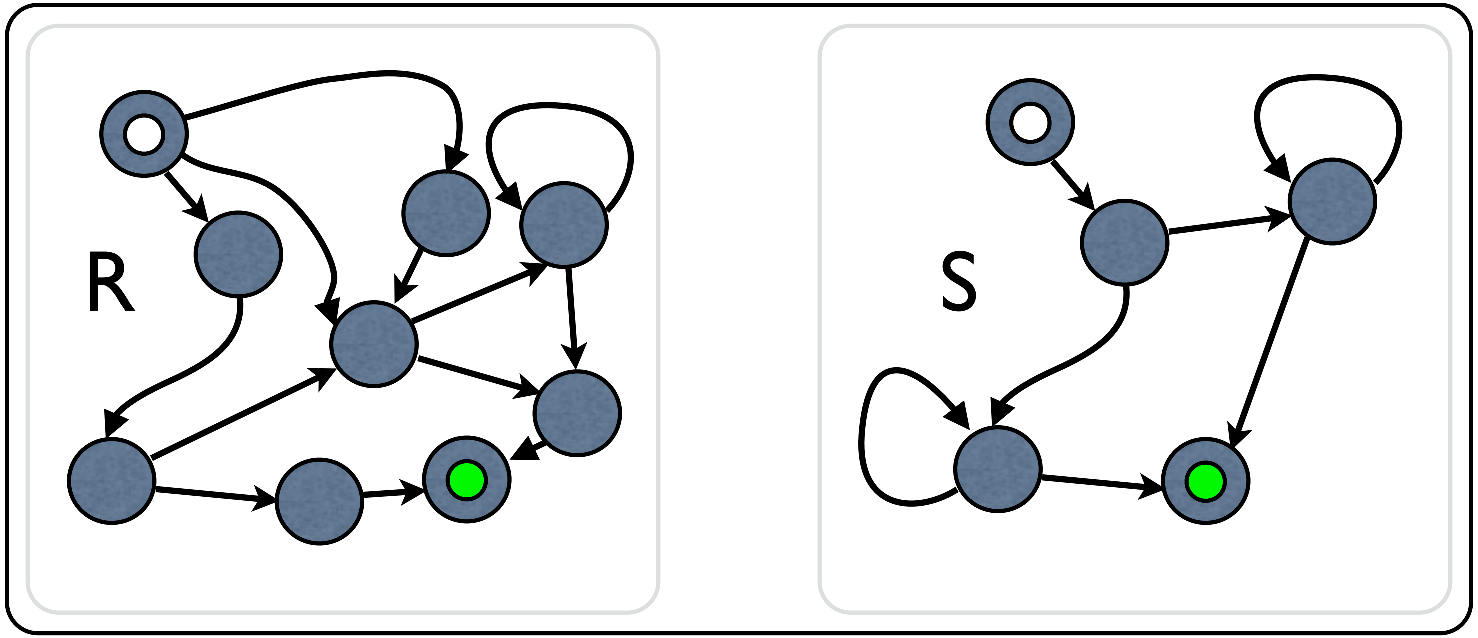
What happens when we link two machines together?

sequence
RS



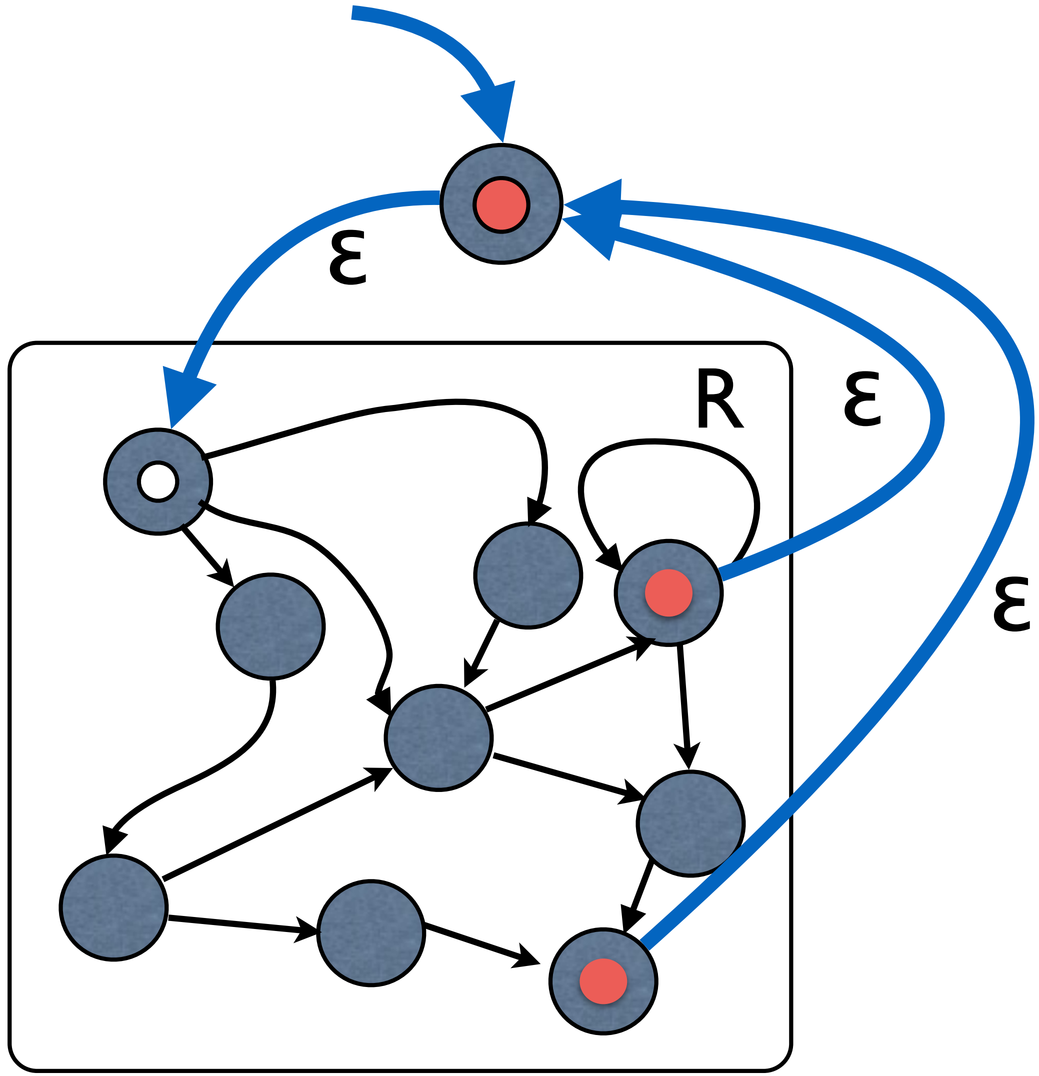
$$RS = \{ r +\!\!+ s \mid r \in R \wedge s \in S \}$$

alternation   R | S



R | S = R ∪ S

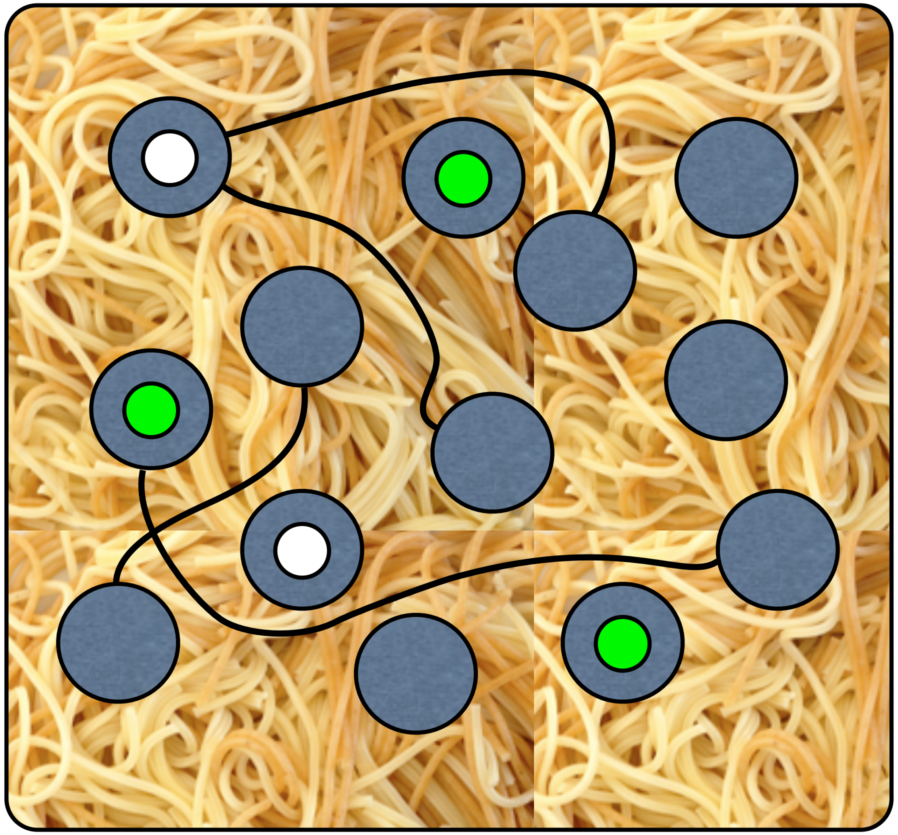
iteration R*



$$"" \in R^*$$

$$x \in R^*,\ r \in R \Rightarrow x \mathbin{++} r \in R^*$$

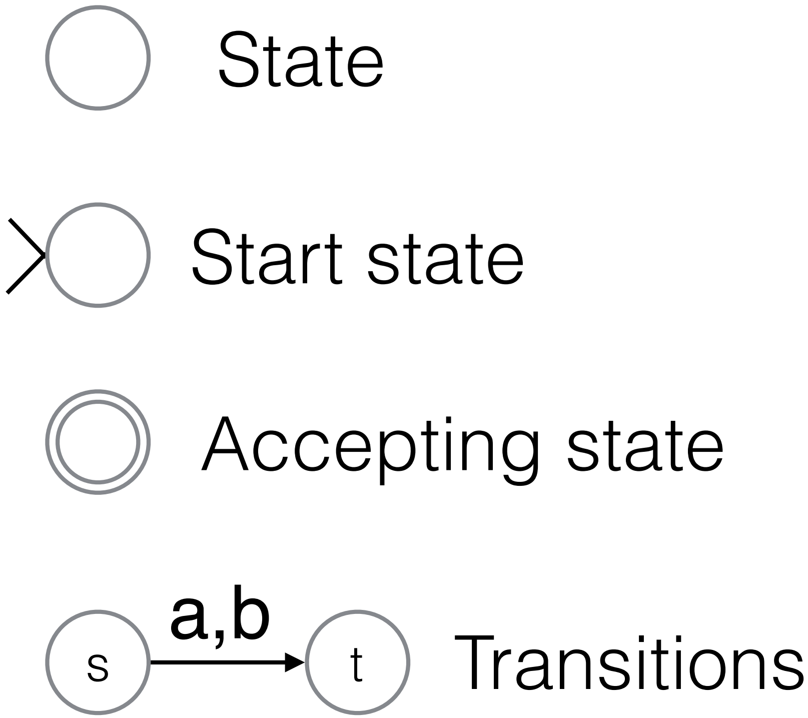
finite state spaghetti

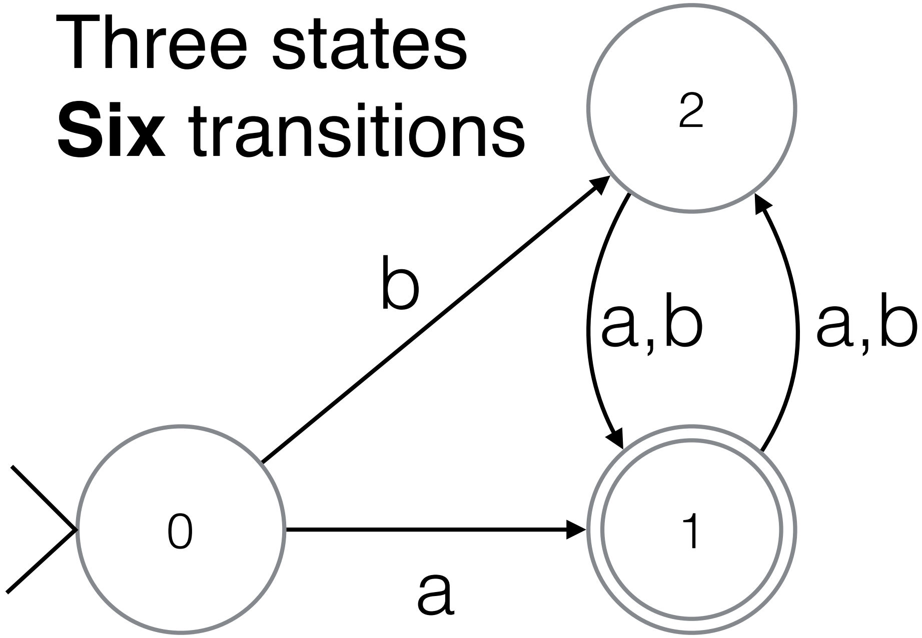


A language is a set of
finite sequences of words.

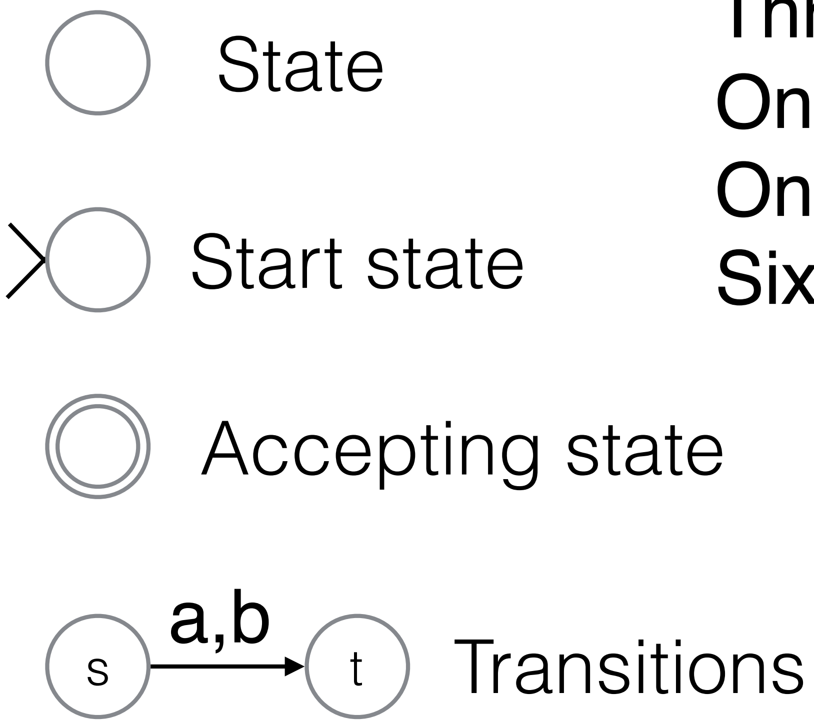
A language is a set of
finite sequences of symbols.

A language is **regular** iff
it is the language recognised
by some Finite State Machine.

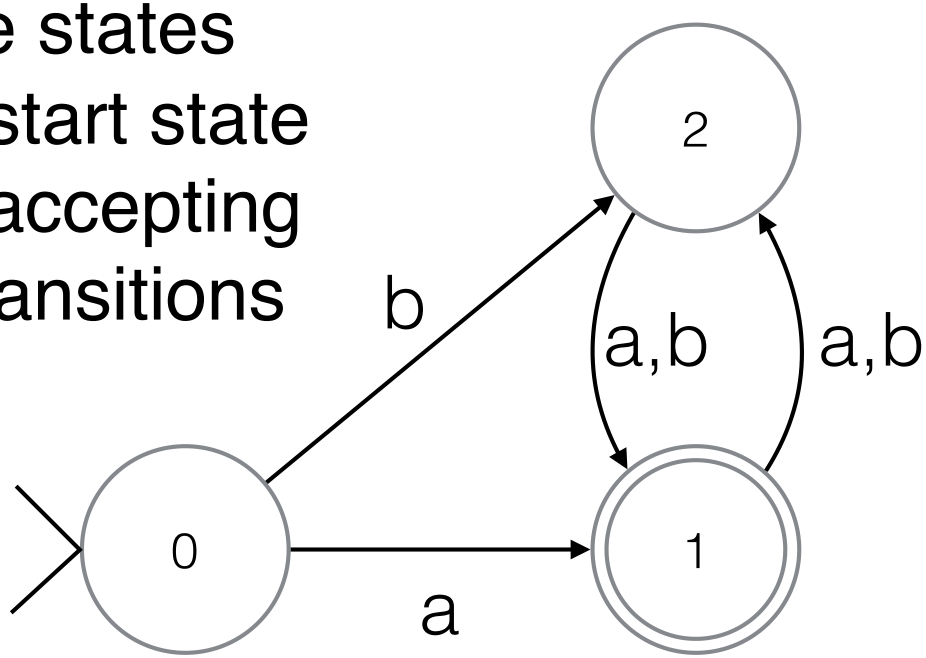




each transition is labelled with a finite sequence of tokens,
the tokens, depend on the application,
they may be letters, symbols, words, actions, …



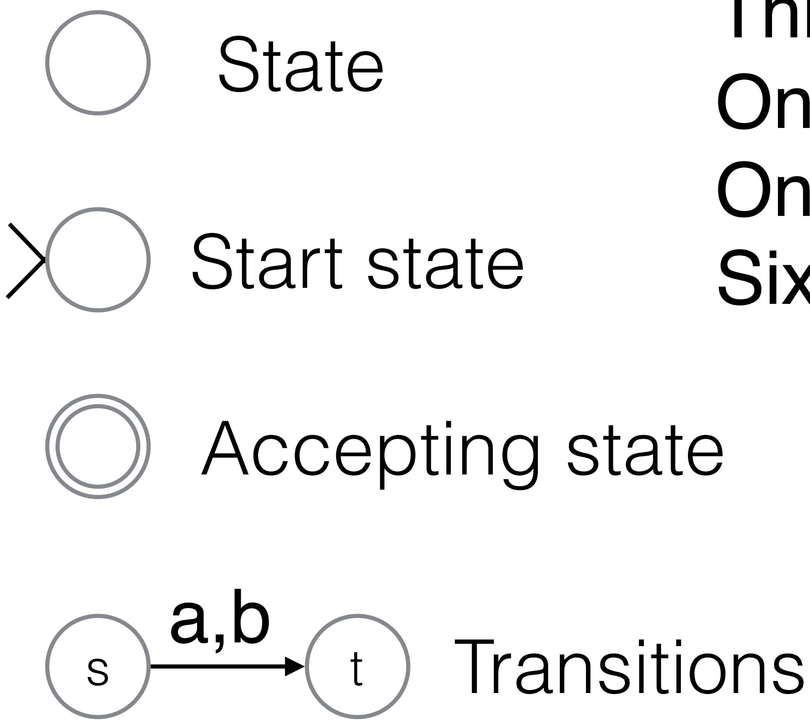
Three states
One start state
One accepting
Six transitions



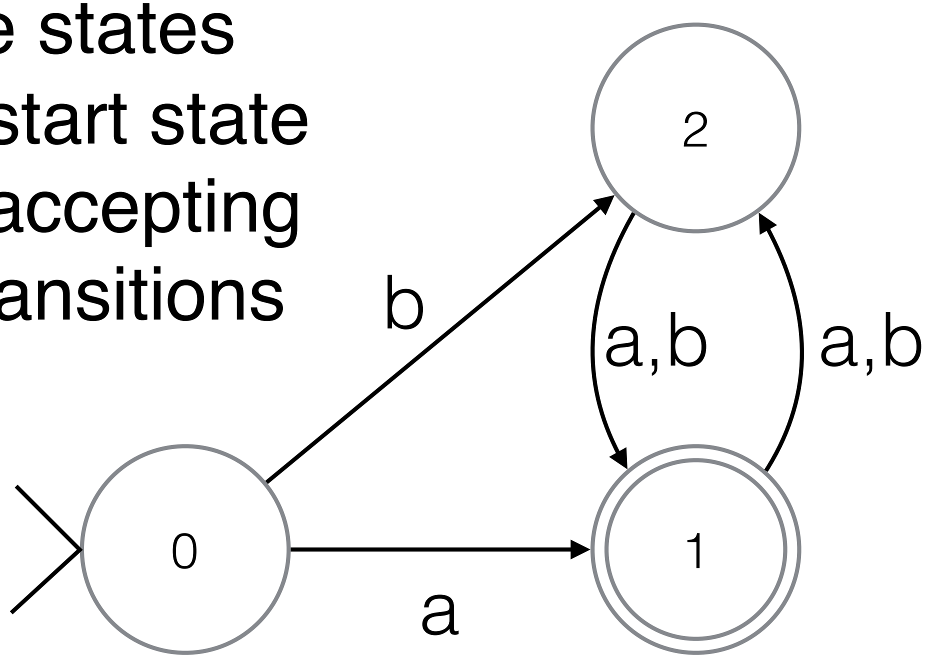
Normally we only use sequences of length 0 or 1 as labels.
Then we write **a** for **<a>** and **ε** for **<>**

We generate strings of tokens by tracing a path from a start state to an accepting state.

Which strings are accepted by this automaton?

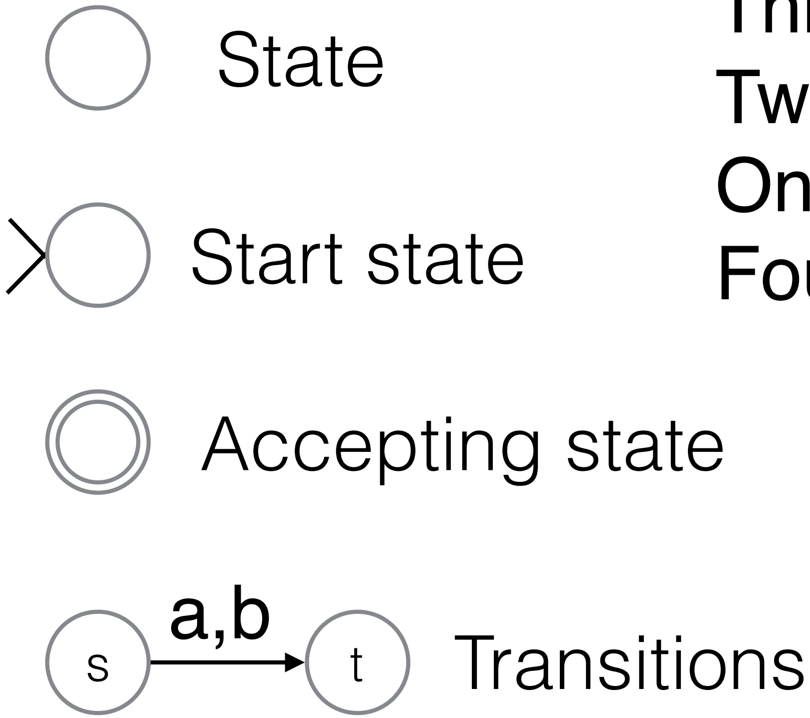


Three states
One start state
One accepting
Six transitions

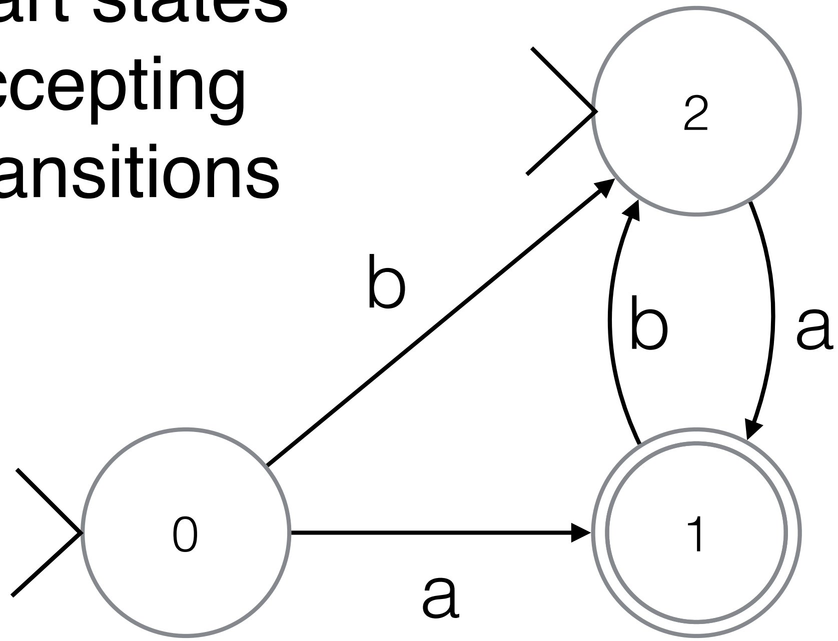


Normally we only use sequences of length 0 or 1 as labels.
Then we write **a** for **<a>** and **ε** for **<>**

Strings that
  begin with b and are of even length **or**
  begin with a and are of odd length

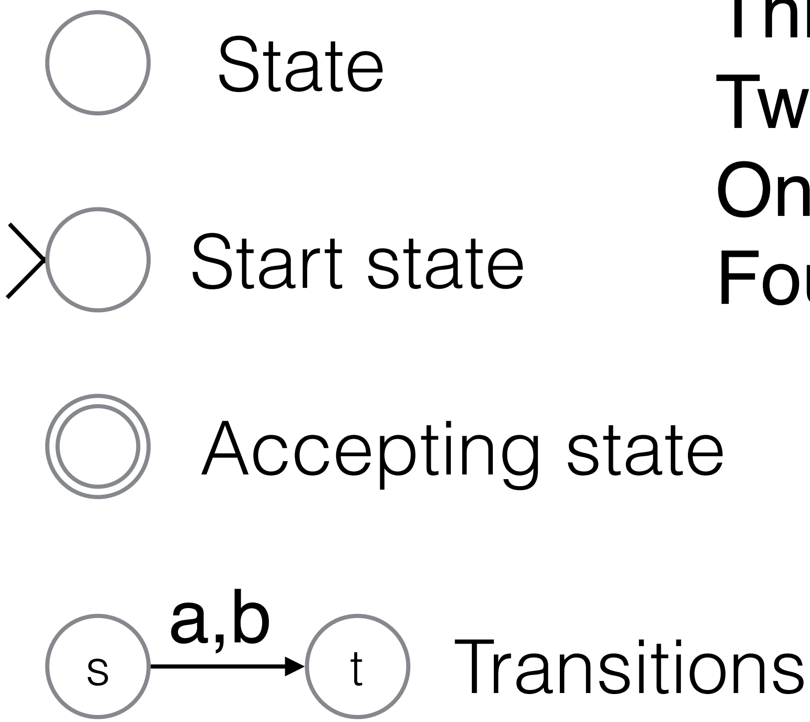


Three states
Two start states
One accepting
Four transitions

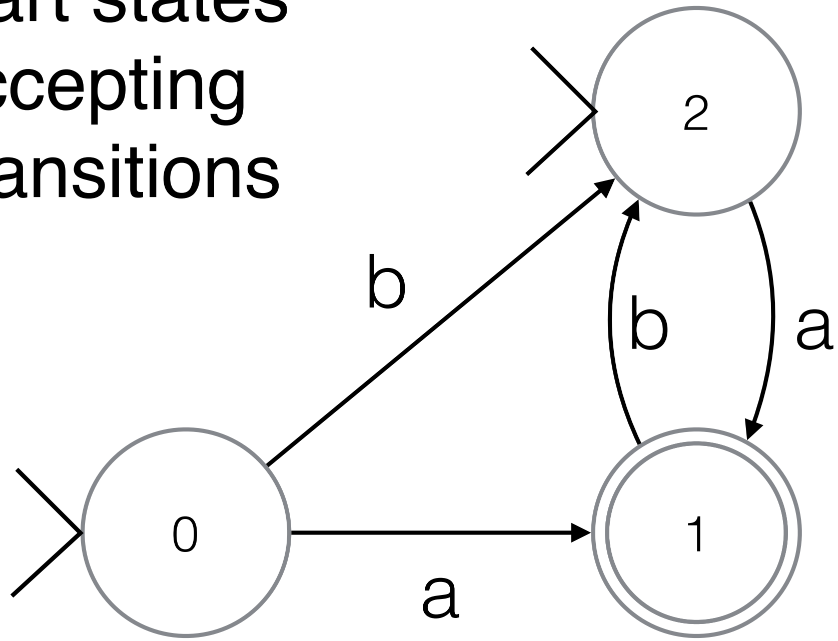


This machine is non-deterministic
some strings have more than one trace
some strings have no traces

Which of these strings are accepted?
abba abab baba aba bab ab ba a b

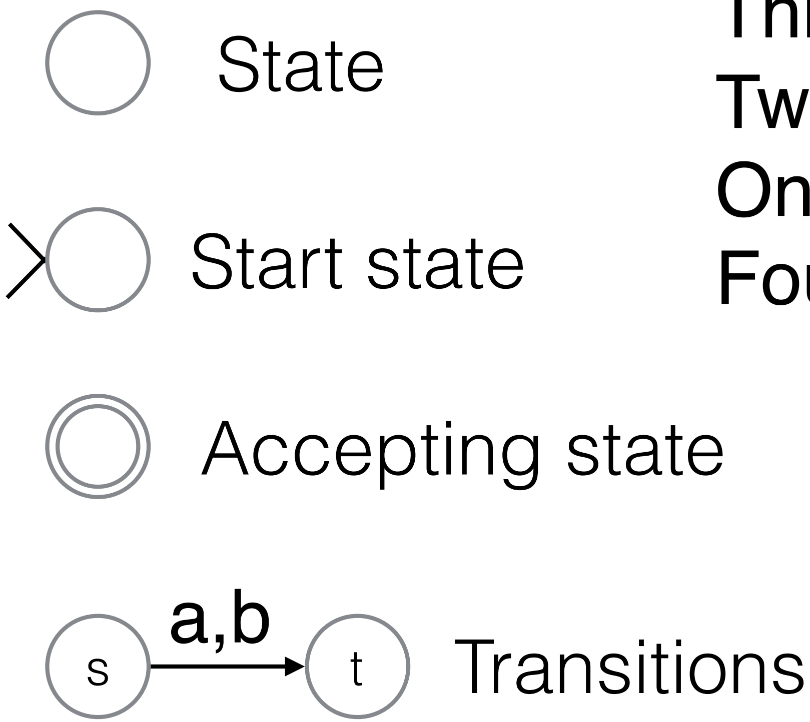


Three states
Two start states
One accepting
Four transitions

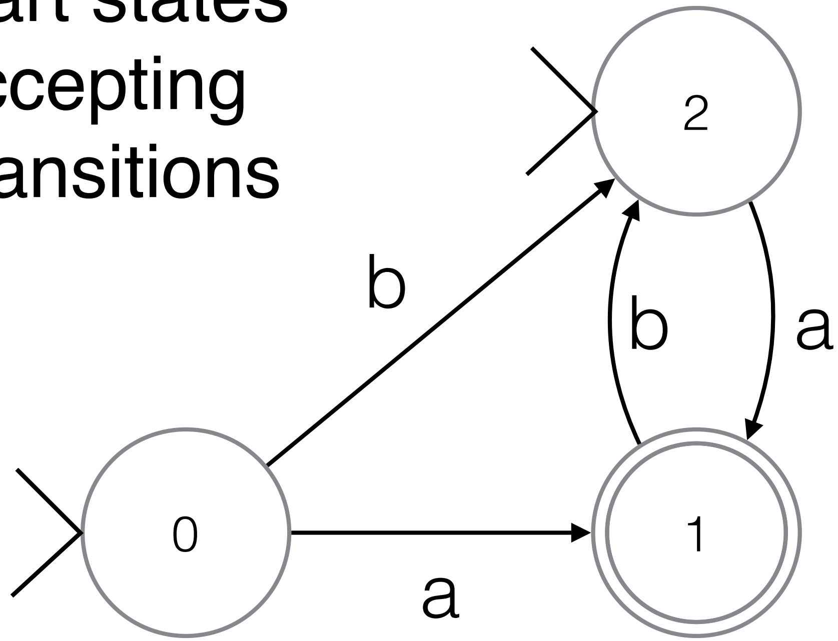


This machine is non-deterministic
 some strings have more than one trace
 some strings have no traces

Which of these strings are accepted?
abba abab **baba aba** bab ab **ba a** b



Three states
Two start states
One accepting
Four transitions



This machine is non-deterministic
some strings have more than one trace
some strings have no traces

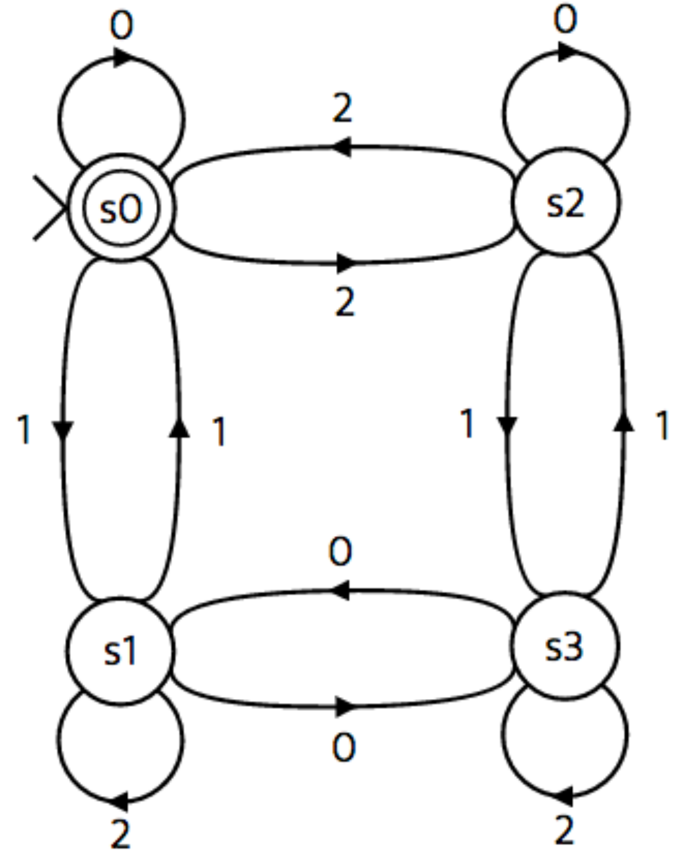
$a\ \varepsilon\ L$
$ba\ \varepsilon\ L$
$x\ \varepsilon\ L \rightarrow x\ ba\ \varepsilon\ L$

Which of these strings are accepted?
abba abab **baba aba** bab ab **ba a** b

KISS – DFA

Deterministic **F**inite **A**utomaton

Exactly one start state, and
  from each state, **q**,
    for each token, **t**,
      there is
        exactly one transition
        from **s** with label **t**



This machine has four states,

$Q = \{ s0, s1, s2, s3 \}$

Its alphabet has three symbols

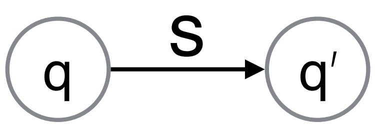
$\Sigma = \{ 0, 1, 2 \}$

For each

(state, symbol) pair, **(q,s)**

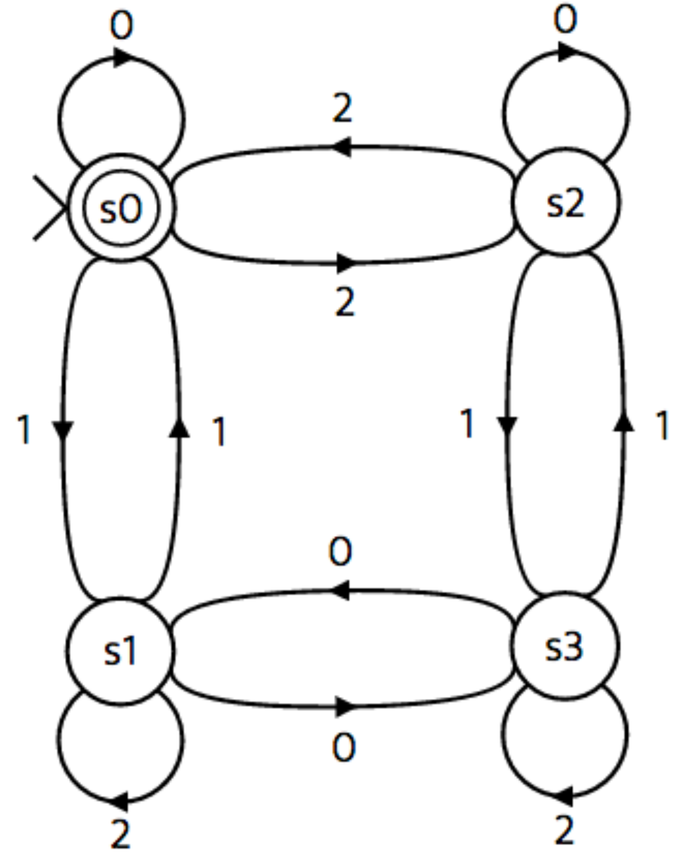
there is **exactly one** transition

from **q** with label **s**



This means that

q′ is **determined** by **(q,s)**

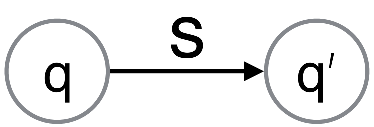


For each

(state, symbol) pair, **(q,s)**

there is **exactly one** transition

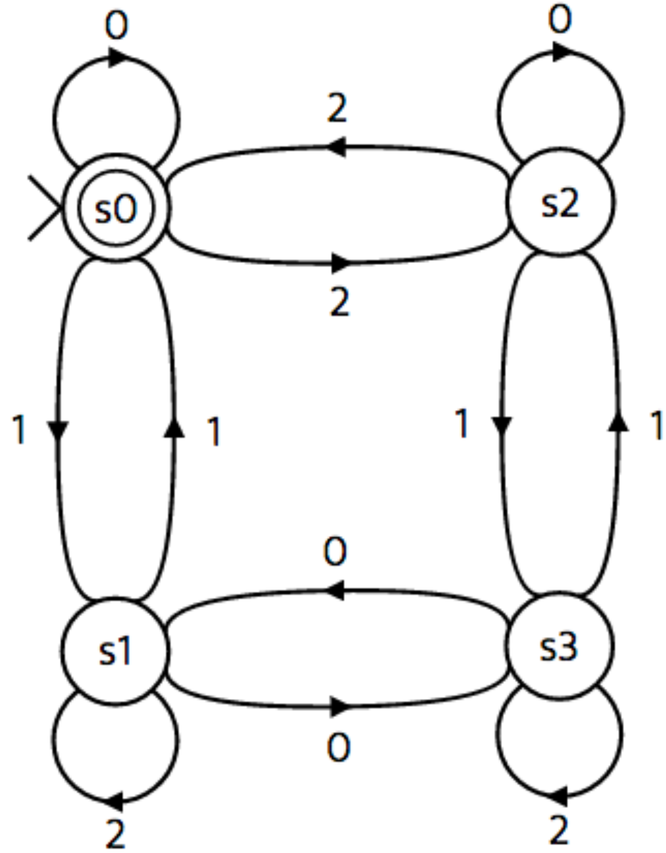
from **q** with label **s**



This means that

q′ is **determined** by **(q,s)**

It has only one start state, this means that any input string determines a sequence of states.



This is a **deterministic finite-state automaton**

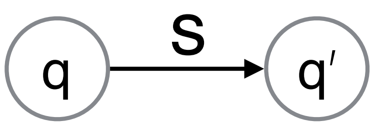
DFA

For each

(state, symbol) pair, **(q,s)**

there is **exactly one** transition

from **q** with label **s**



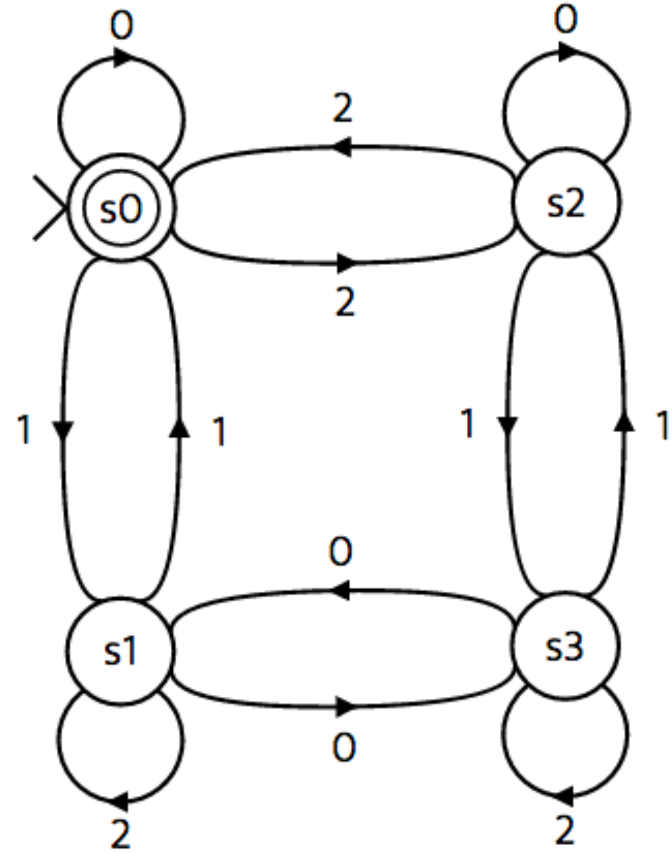
q′ is **determined** by **(q,s)**

It has only one start state

Every DFA has a

next-state function

q′ = F(q,s)



This is a

deterministic finite-state automaton, DFA

DFA ⊂ NFA

A DFA has only one start state

For each (state, symbol) pair, (q,s)

there is **exactly one** transition

from **q** with label **s**

q' is **determined** by **q** and **s**

Every DFA has a
next-state function
$q' = F(q,s)$

any input string
determines
a sequence of states.

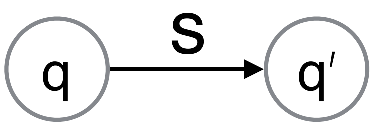
We will focus now on DFA and return later to NFA.

Then we will show that for any NFA there isa a DFA that defines the same language.

Every DFA has a

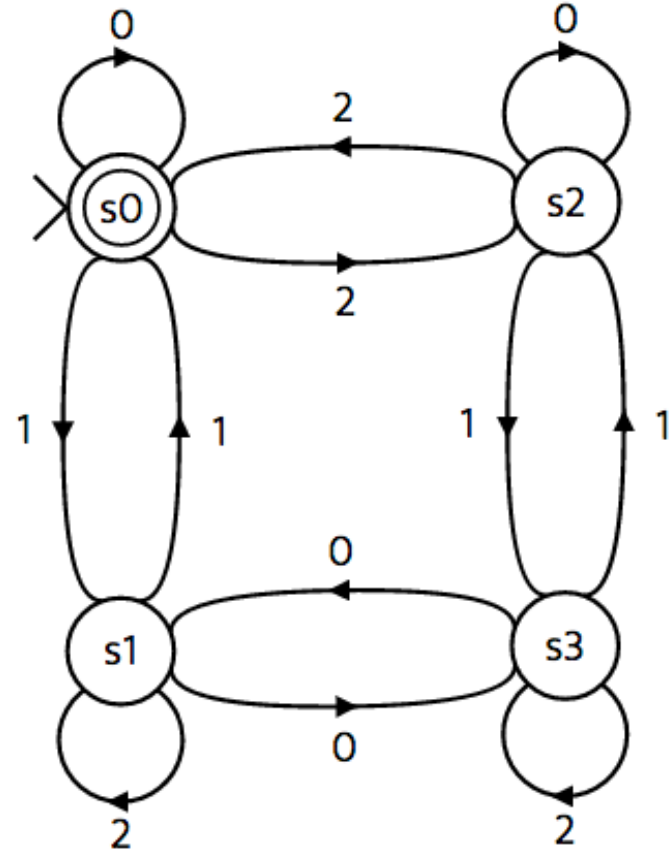
next-state function

$$q' = F(q,s)$$



Since we have
finitely many symbols and states
This can be given by a table

	0	**1**	**2**
s0	s0		
s1			s2
s2		s3	
s3	s1		

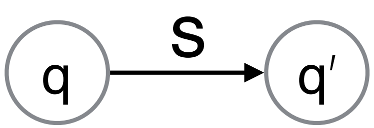


This is a
deterministic finite-state automaton, DFA

Every DFA has a

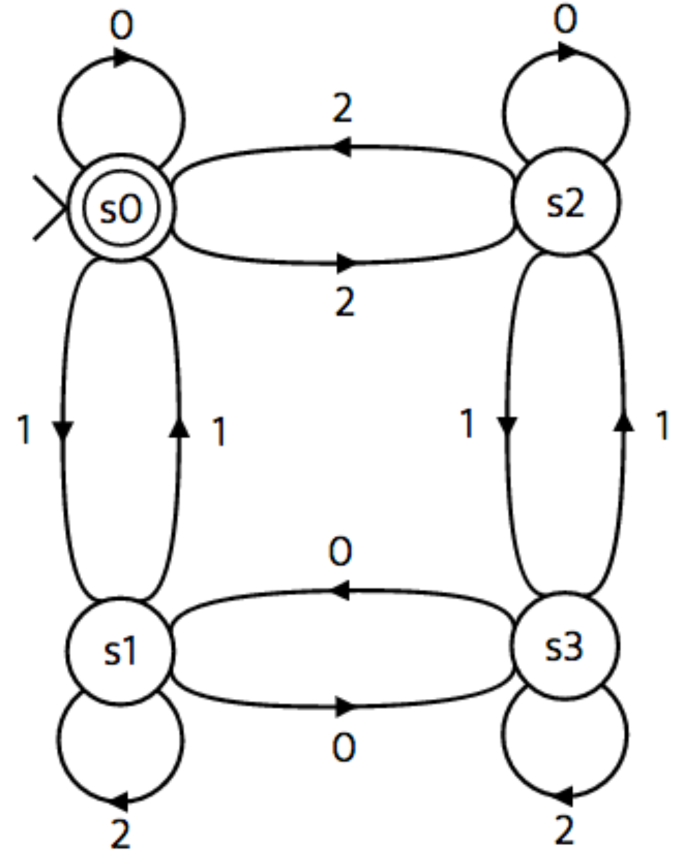
next-state function

$q' = F(q,s)$



Since we have
finitely many symbols and states
This can be given by a
next-state table

q\s	0	1	2
s0	s0	s1	s2
s1	s3	s0	s1
s2	s2	s3	s0
s3	s1	s2	s3

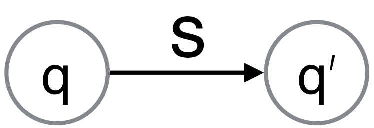


This is a
deterministic finite-state automaton, DFA

Every DFA has a

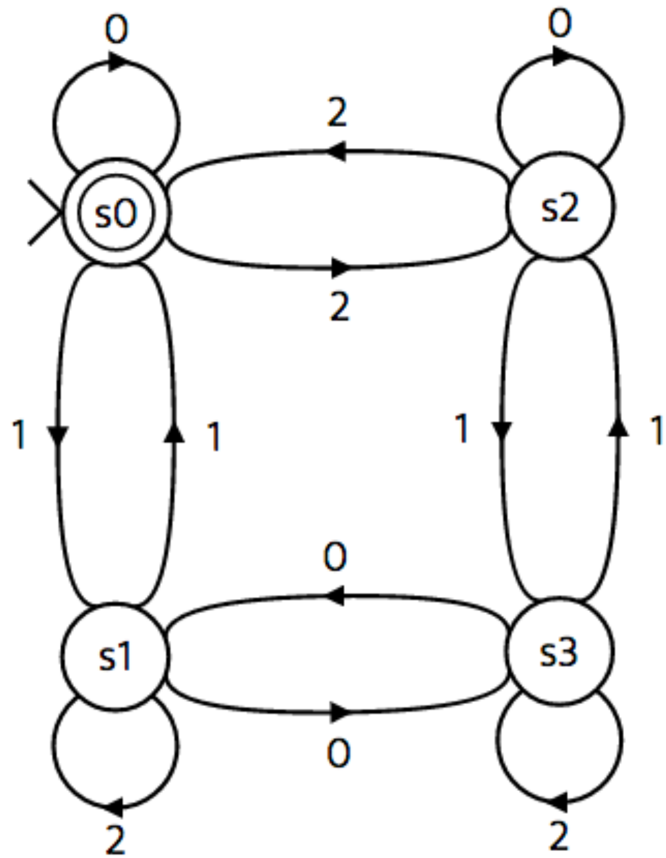
next-state function

$$q' = F(q,s)$$



We can also represent the transitions by a relation. Which symbol(s) take us from state **q** to state **r** ?

q\r	s0	s1	s2	s3
s0	0	1	2	
s1				
s2				
s3		0	1	

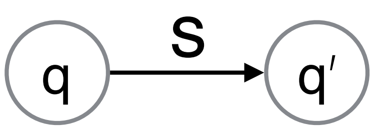


This is a **deterministic finite-state automaton, DFA**

Every DFA has a

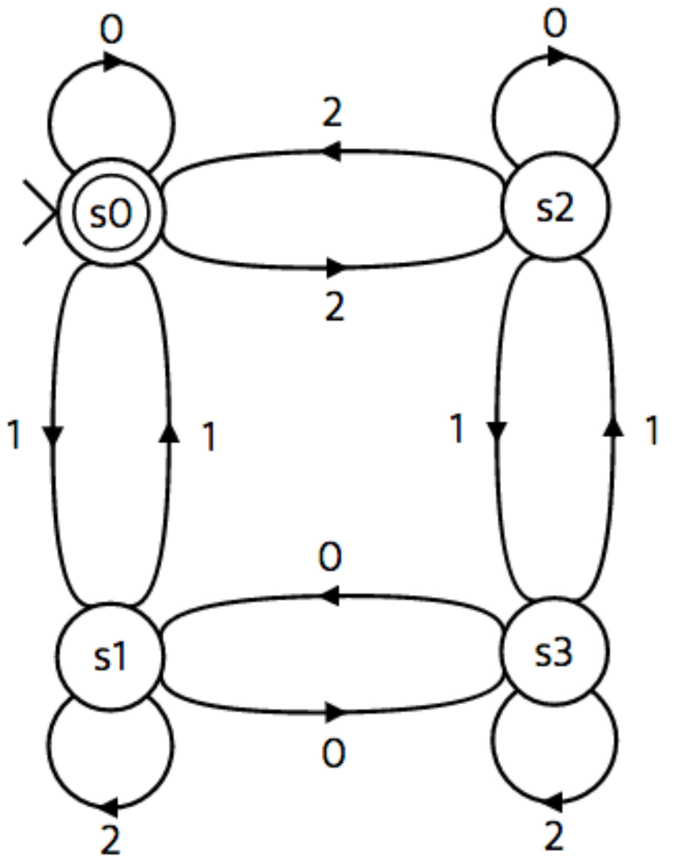
next-state function

$$q' = F(q,s)$$



We can also represent the transitions by a relation. Which symbol(s) take us from state **q** to state **r** ?

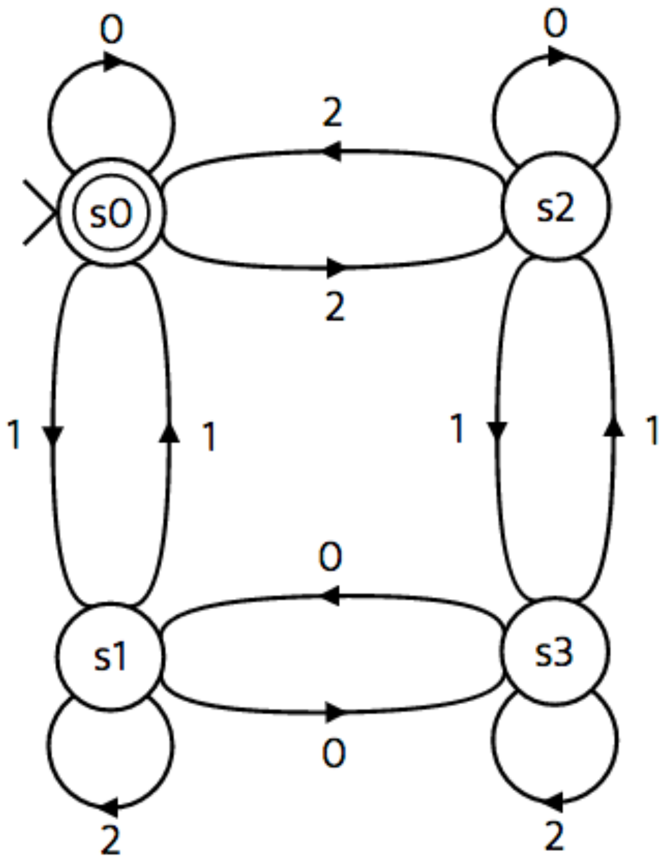
q\r	s0	s1	s2	s3
s0	0	1	2	
s1	1	2		0
s2	2		0	1
s3		0	1	2



This is a **deterministic finite-state automaton, DFA**

For every DFA, any input string determines a path or **trace**

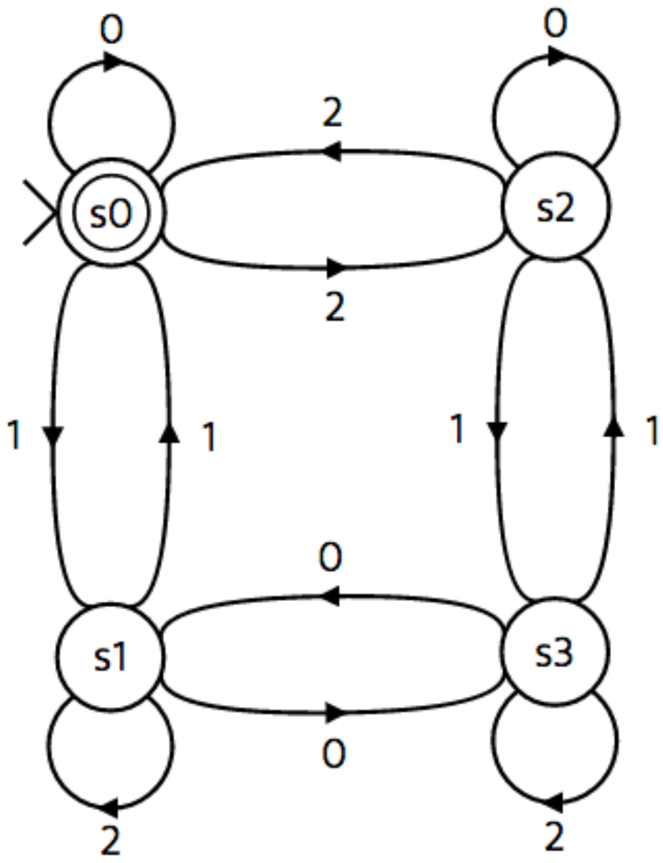
input	path
000	s0 s0 s0 s0
021	s0 s0 s2 s3
120	s0 s1 s1 s3
11	
12	
21	
22	
111	
110	
212	



This is a **deterministic finite-state automaton, DFA**

For every DFA, any input string determines a path

input	path
000	s0 s0 s0 s0
021	s0 s0 s2 s3
120	s0 s1 s1 s3
11	s0 s1 s0
12	s0 s1 s1
21	s0 s2 s3
22	s0 s2 s0
111	s0 s1 s0 s1
110	s0 s1 s0 s0
212	s0 s2 s3 s3

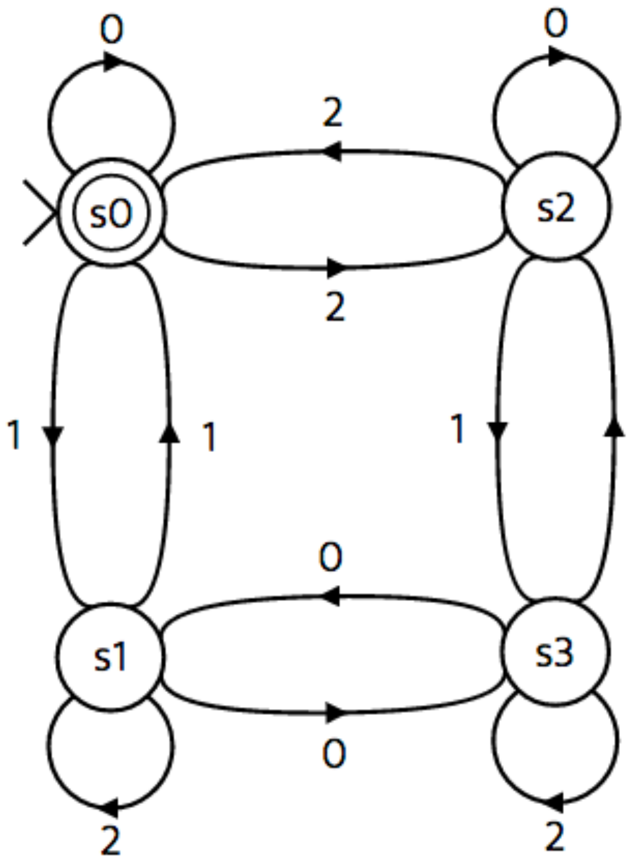


This is a deterministic finite-state automaton, DFA

Which paths

end in an accepting state?

input	path
000	**s0 s0 s0 s0**
021	s0 s0 s2 s3
120	s0 s1 s1 s3
11	**s0 s1 s0**
12	s0 s1 s1
21	s0 s2 s3
22	**s0 s2 s0**
111	s0 s1 s0 s1
110	**s0 s1 s0 s0**
212	s0 s2 s3 s3

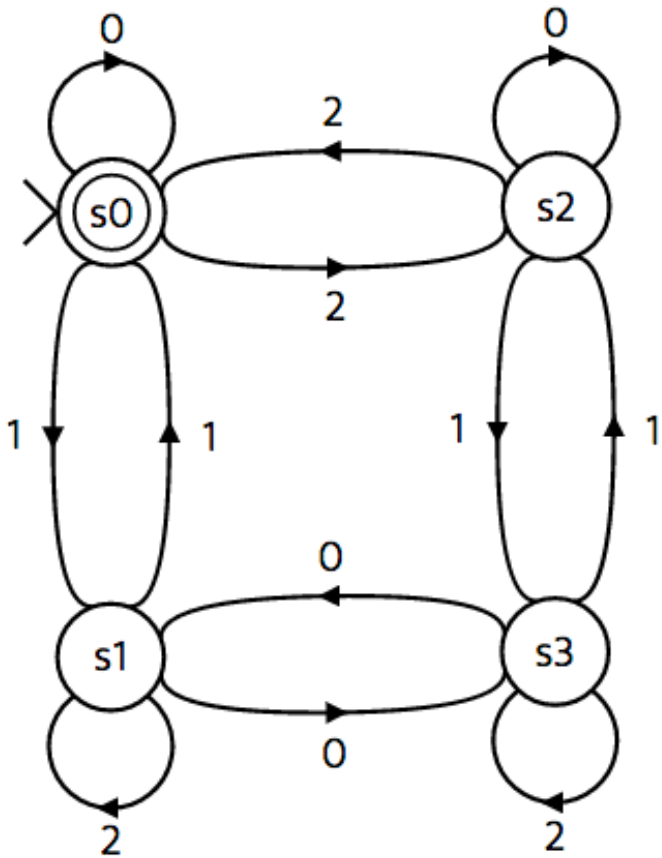


This is a deterministic finite-state automaton, DFA

Which paths

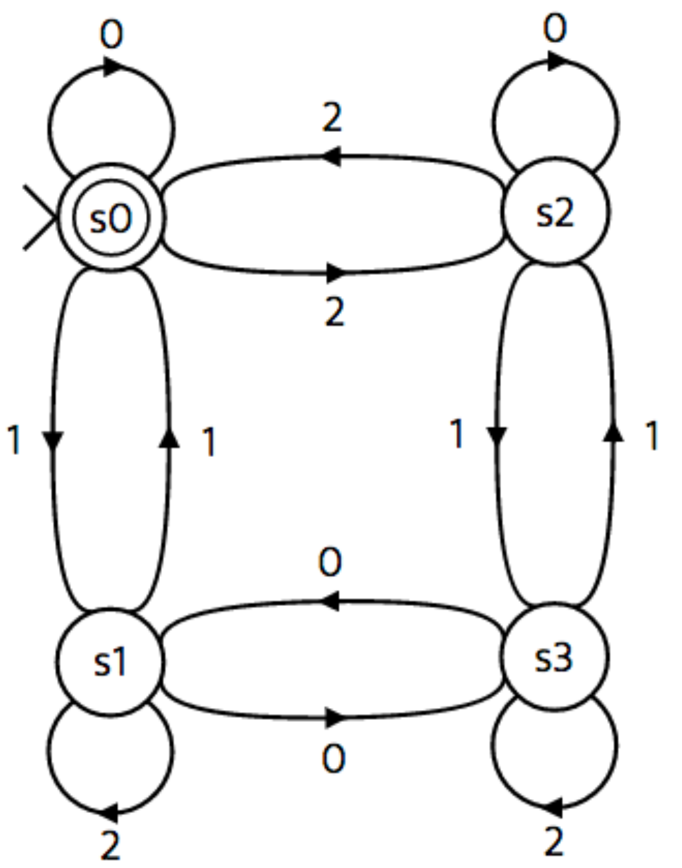
end in an accepting state?

	input	path
0	**000**	**s0 s0 s0 s0**
7	021	s0 s0 s2 s3
15	120	s0 s1 s1 s3
4	**11**	**s0 s1 s0**
5	12	s0 s1 s1
7	21	s0 s2 s3
8	**22**	**s0 s2 s0**
13	111	s0 s1 s0 s1
12	**110**	**s0 s1 s0 s0**
23	212	s0 s2 s3 s3



This is a
deterministic finite-
state automaton,
DFA

What is the meaning of each state?

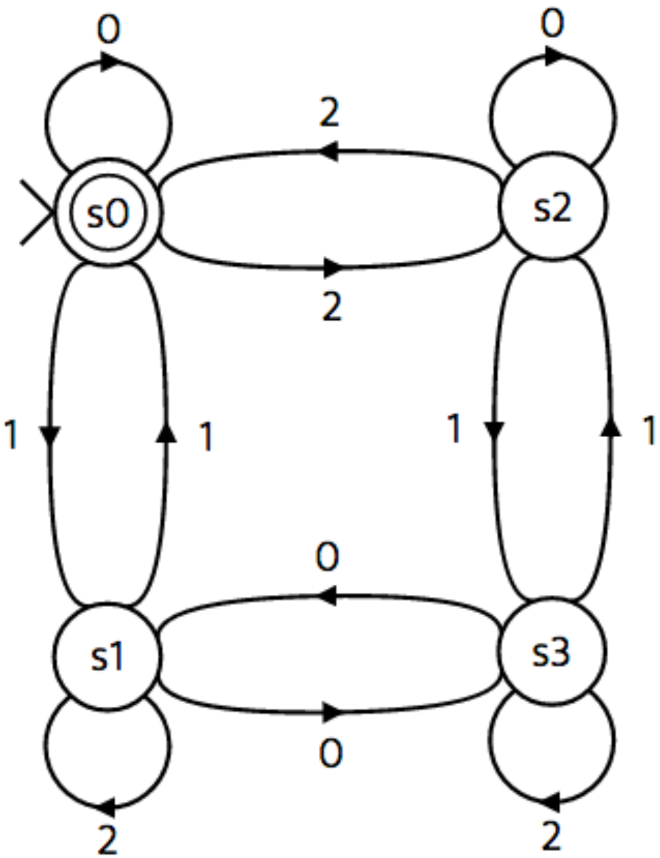


0
1
2
3
4
5
6
7
8
9

.Express each of these decimal numbers in ternary notation, and find out what state is reached by the trace they determine

This is a deterministic finite-state automaton, DFA

What is the meaning of each state?



0
1
2
3
4
5
6
7
8
9

This machine counts
ternary
mod 4

F(q, s) = 3 * q + s (mod 4)

This is a
deterministic finite-
state automaton,
DFA