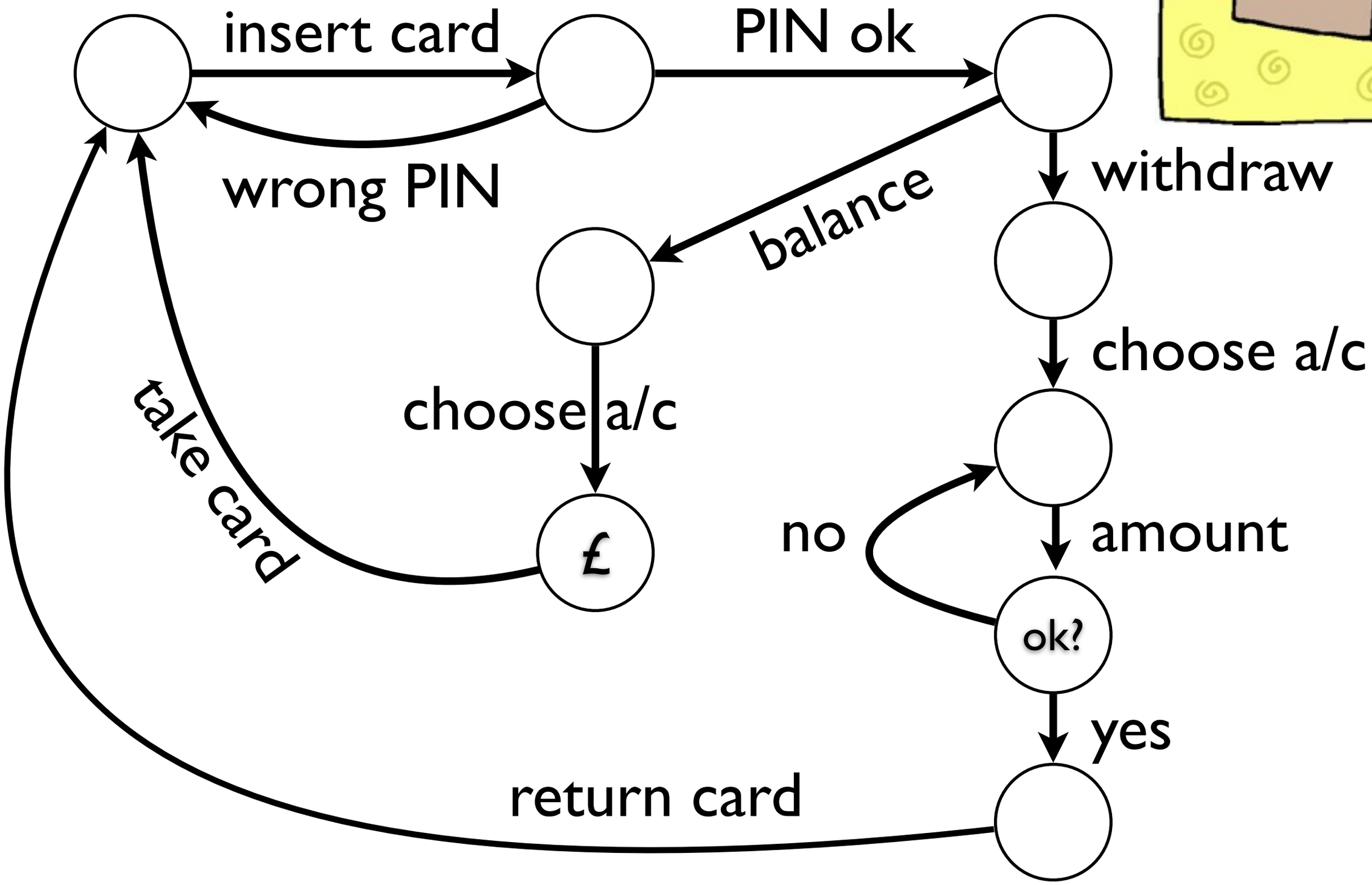
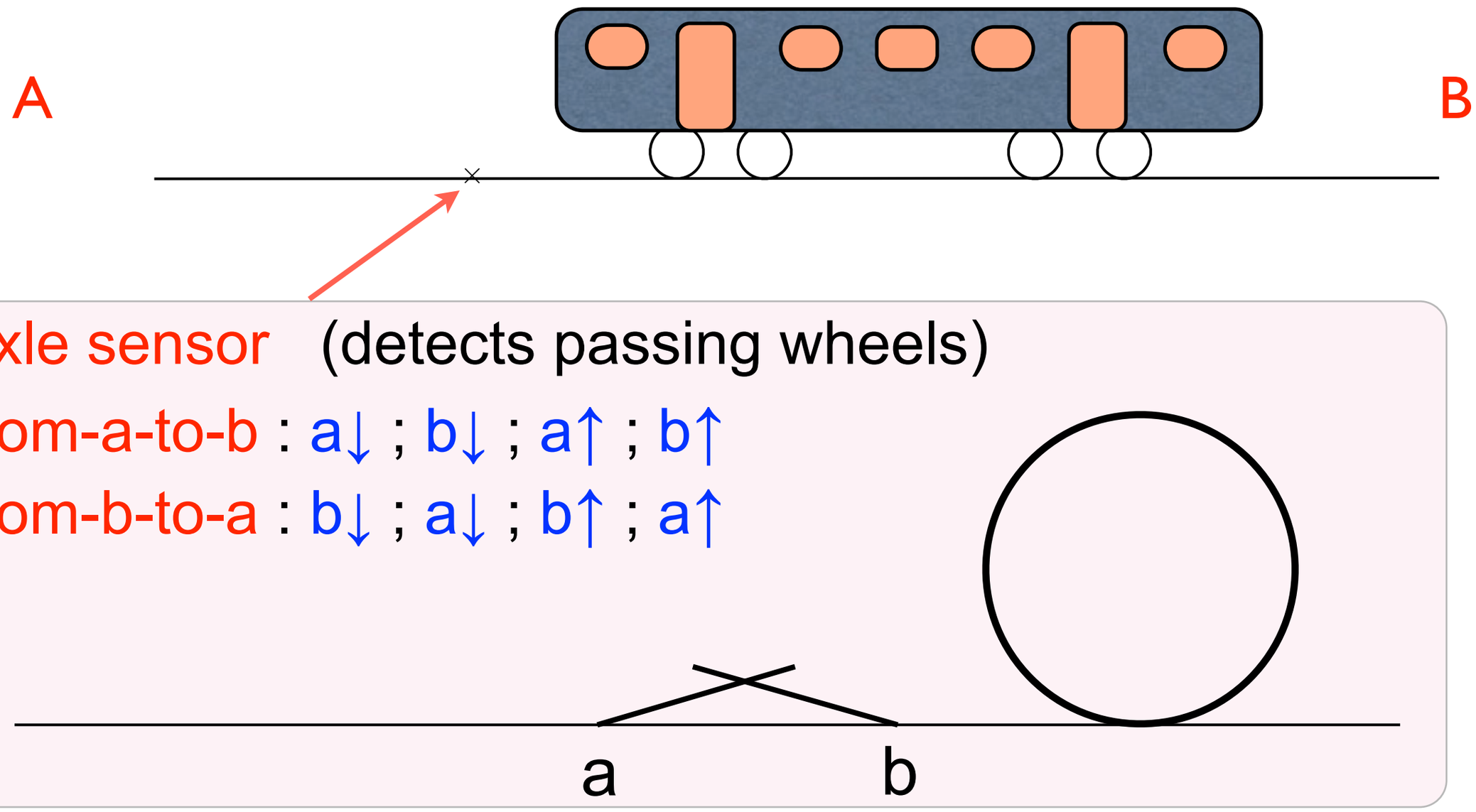
Finite-State Machines (Automata)

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

ATM


insert card
PIN ok
wrong PIN
withdraw
balance
choose a/c
choose a/c
take card
£
no
amount
ok?
yes
return card

Counting trains



Saturday, 3 December 2011

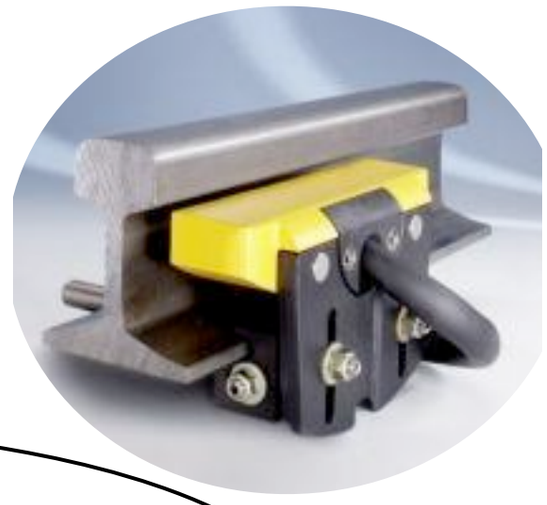
Consider the problem of keeping track of trains passing a level crossing.
For simplicity we consider a single-carriage train. The carriage has four axles. We look at each axle end-on, so we just see one wheel.

How can we check that the entire carriage has passed the sensor before we open the crossing.
How do we make a sensor that can check the direction in which the train is moving?

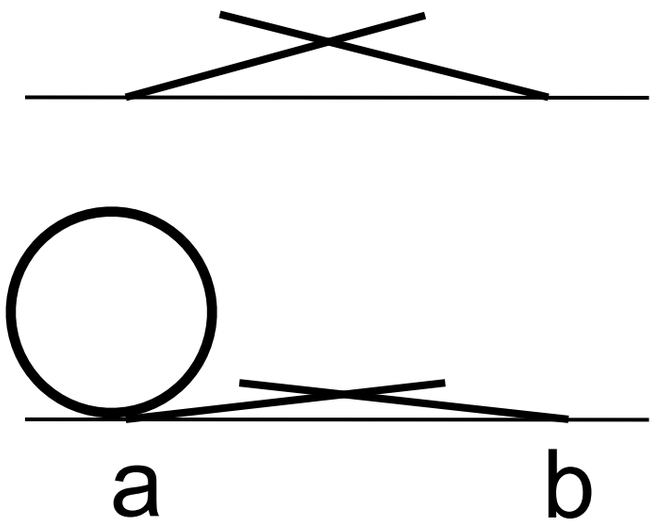
Our sensor has two overlapping arms. We get signals when they go down and up as a wheel rolls over them.
If a wheel rolls over the sensor, going from a to b the we have the sequence a↓ ; b↓ ; a↑ ; b↑
If a wheel rolls in the other direction we have the sequence b↓ ; a↓ ; b↑ ; a↑

We must be careful, because when a train stops and starts it may sometimes roll backwards and forwards – and not just keep moving in its direction of travel. We need a diagram to see what can happen.

Finite-state machines



axle sensor

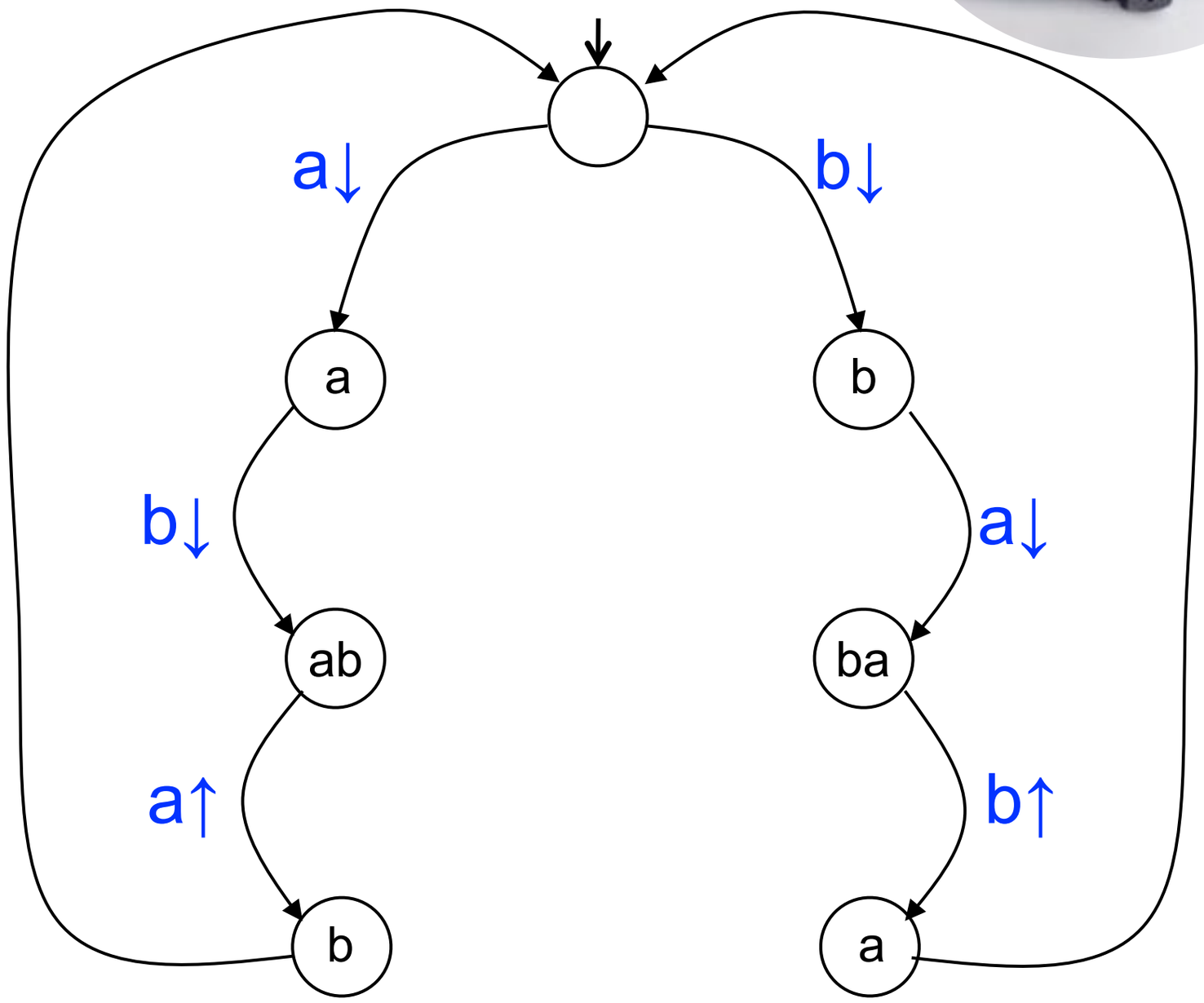


inputs :

a↑, a↓, b↑, b↓

outputs :

from-a-to-b,
from-b-to-a



b↑ / from-a-to-b

a↑ / from-b-to-a

Saturday, 3 December 2011

This is a finite state machine (or automaton).

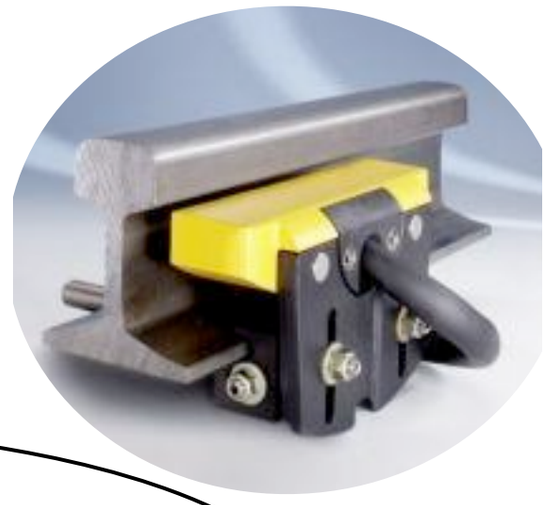
The nodes of this graph are states and the arcs tell us how the machine changes from one state to another in response to input events.

Here the events correspond to change of state. On the left-hand side of the diagram we keep track of a wheel that came from the left-hand a-direction; on the right-hand side, we keep track of a wheel that came from the right-hand b-direction.

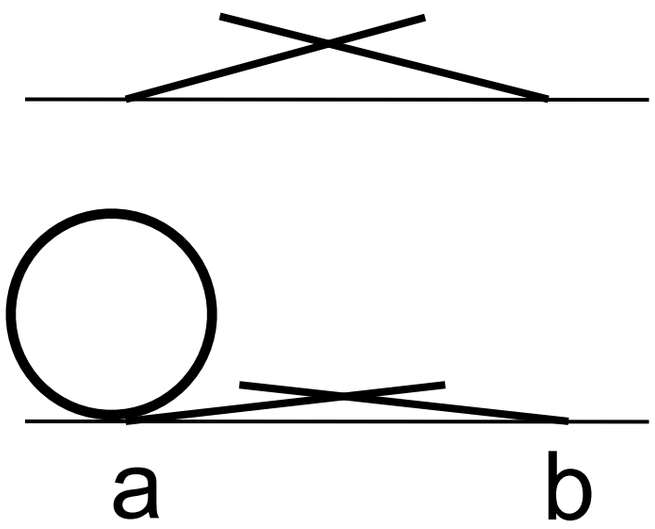
We keep track even if a wheel comes part-way over the sensor and then rolls back.

Exercise for the students: Try drawing a state machine that takes the a→b and b→a events as inputs and out puts events A→B (when a carriage passes from left to right) and B→A (when a carriage passes from right to left). HINT: you can relabel the same diagram.

Finite-state machines



axle sensor

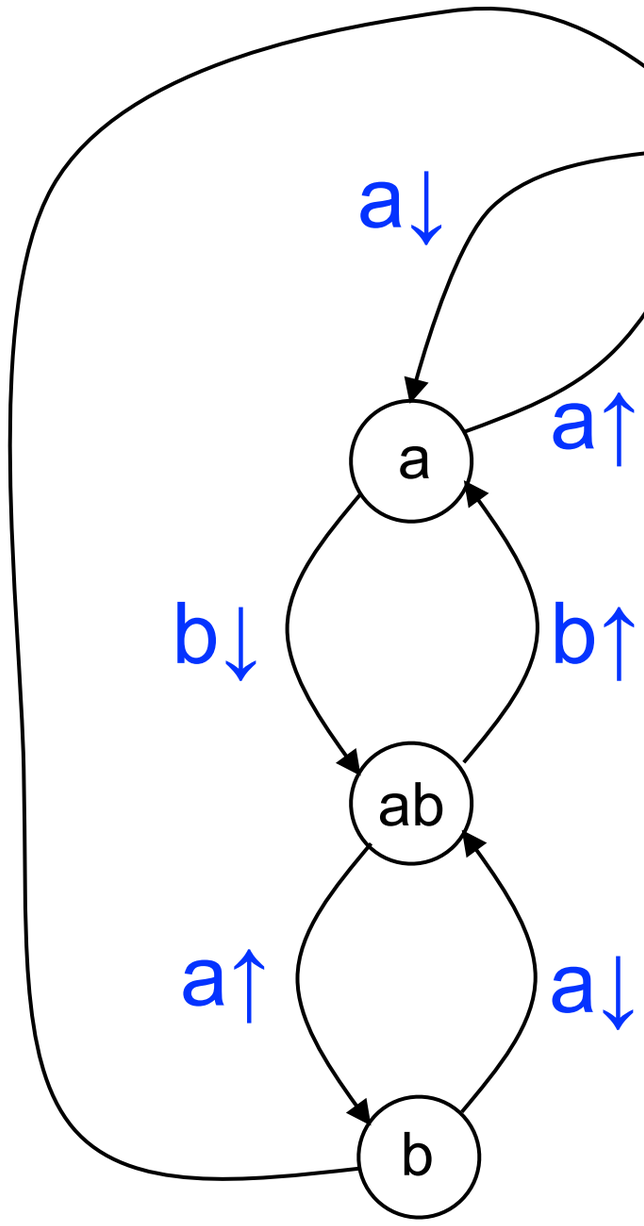


inputs :

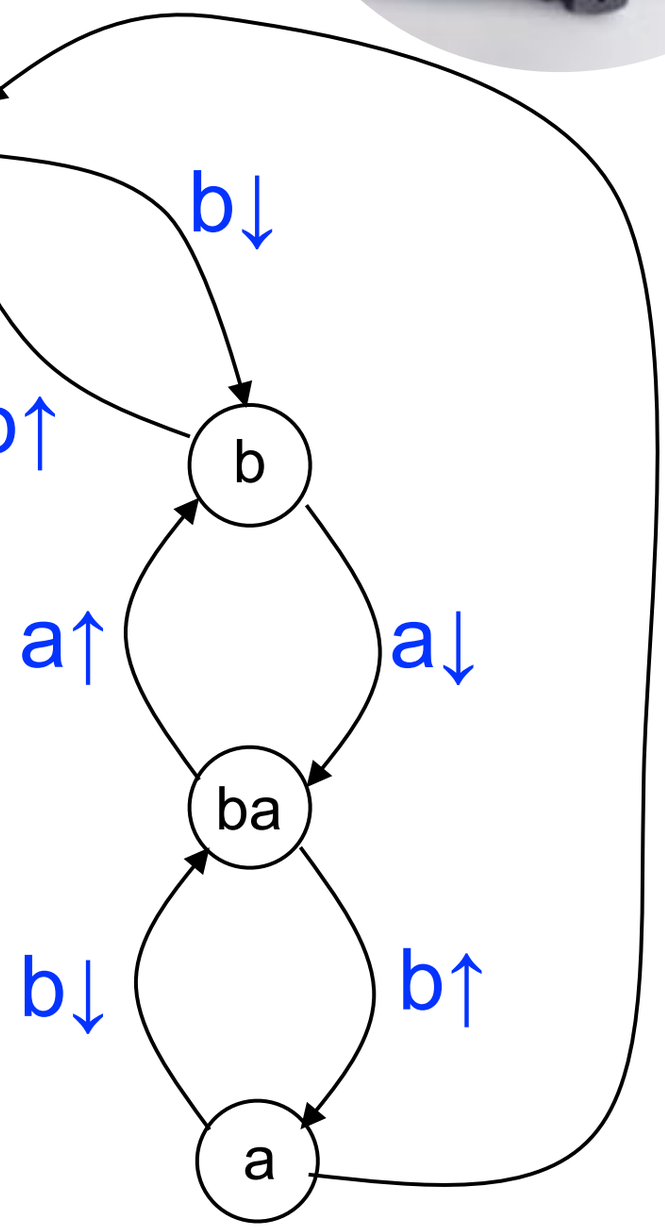
a↑, a↓, b↑, b↓

outputs :

from-a-to-b,
from-b-to-a



b↑ / from-a-to-b



a↑ / from-b-to-a

Saturday, 3 December 2011

This is a finite state machine (or automaton).

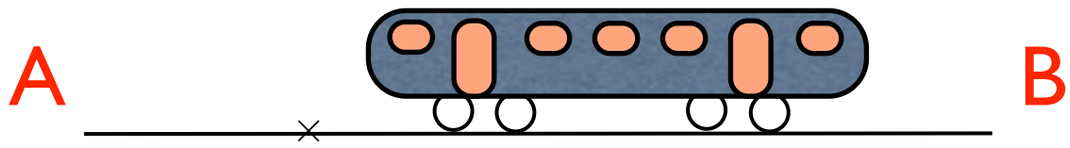
The nodes of this graph are states and the arcs tell us how the machine changes from one state to another in response to input events.

Here the events correspond to change of state. On the left-hand side of the diagram we keep track of a wheel that came from the left-hand a-direction; on the right-hand side, we keep track of a wheel that came from the right-hand b-direction.

We keep track even if a wheel comes part-way over the sensor and then rolls back.

Exercise for the students: Try drawing a state machine that takes the a→b and b→a events as inputs and out puts events A→B (when a carriage passes from left to right) and B→A (when a carriage passes from right to left). HINT: you can relabel the same diagram.

Hierarchical FSMs



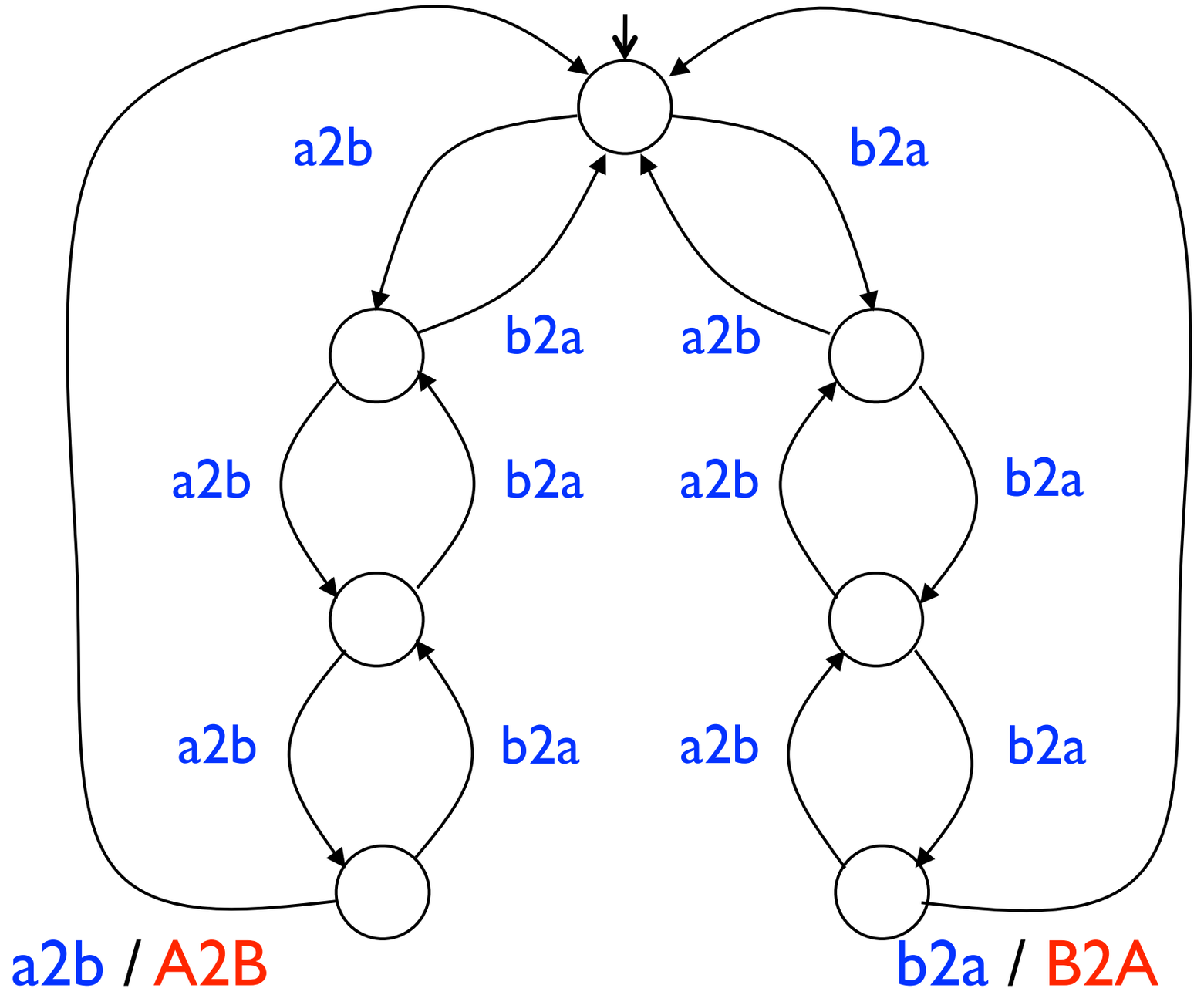
carriage counter

inputs :

a2b, b2a

outputs :

A2B, B2A



a2b = from-a-to-b

Saturday, 3 December 2011

Here is the solution. We just keep track of how many axles have passed.

Here we see an example of hierarchy – the outputs of one machine form the inputs of another.

Many machines are constructed in this way – by connecting together simple finite-state machines.

We'll come back to finite-state machines in a few minutes.

Application Fields

Industry

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

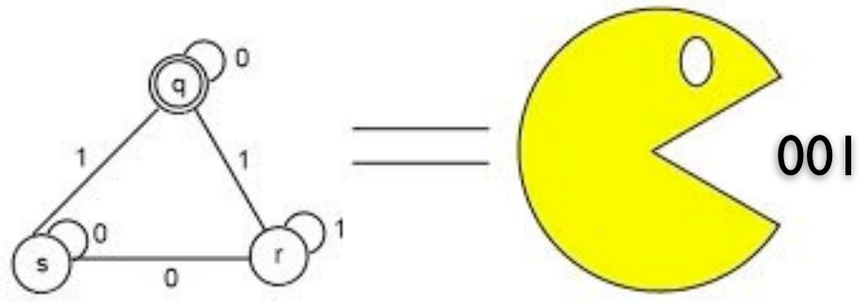
Electronic circuits

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

Communication protocols

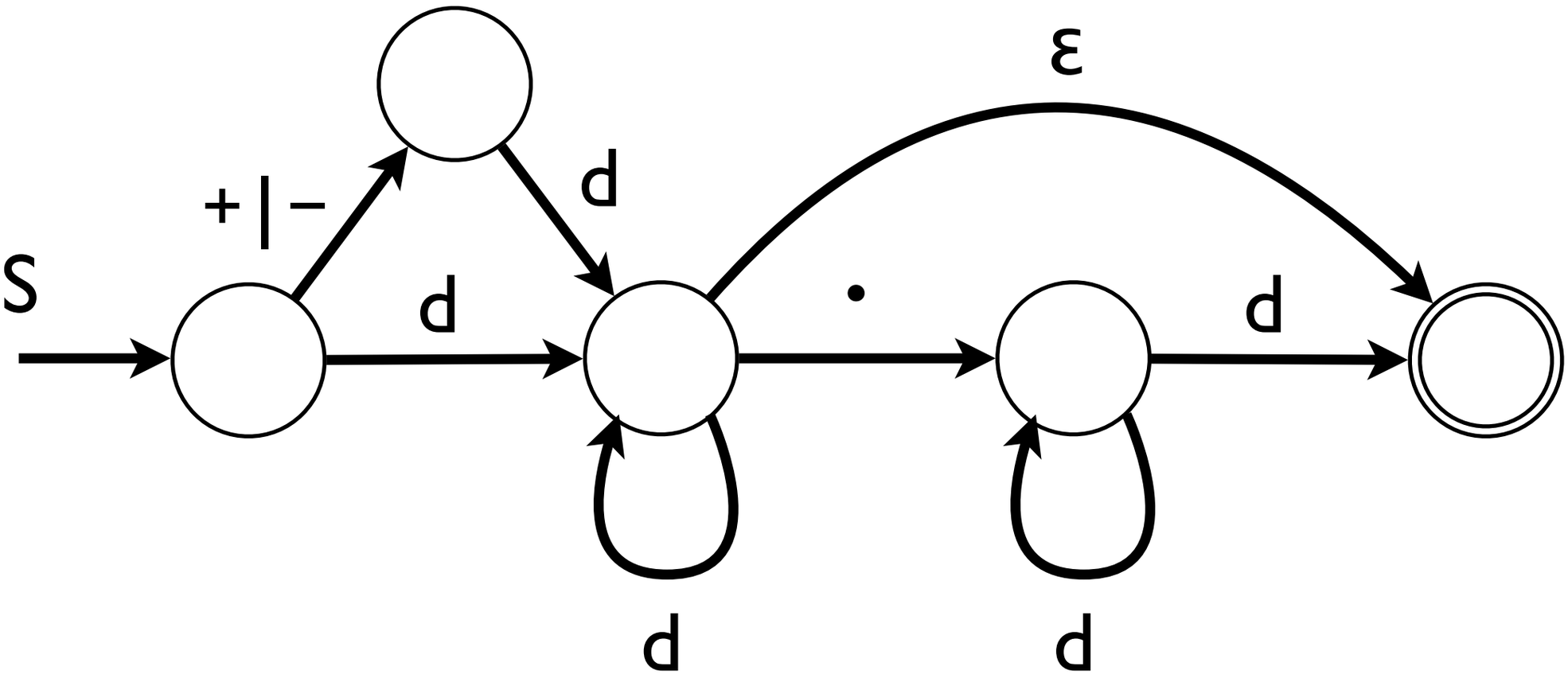
- initiation and maintenance of communication links
- error detection and handling, packet retransmission

Language analysis

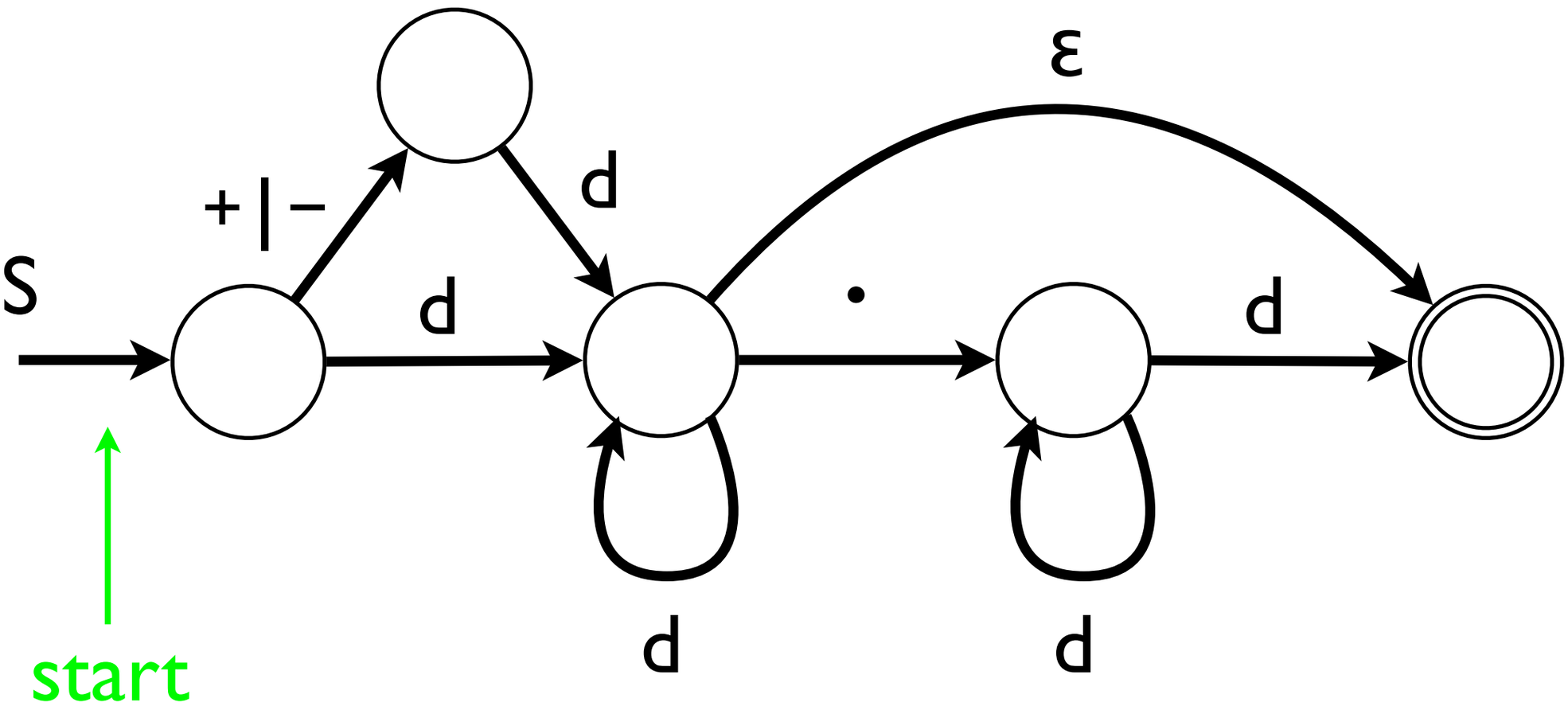


- natural languages
- programming languages
- search engines

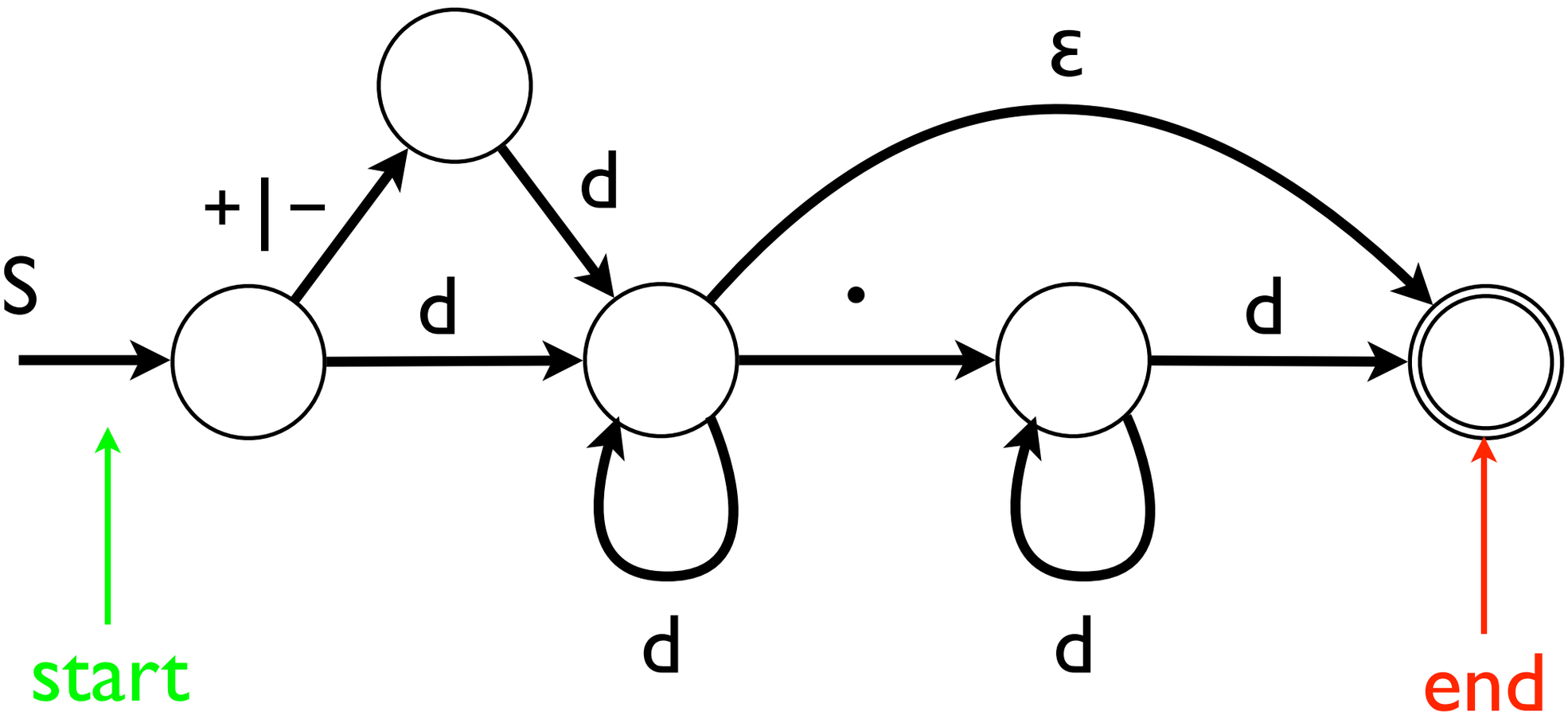
A Decimal Number



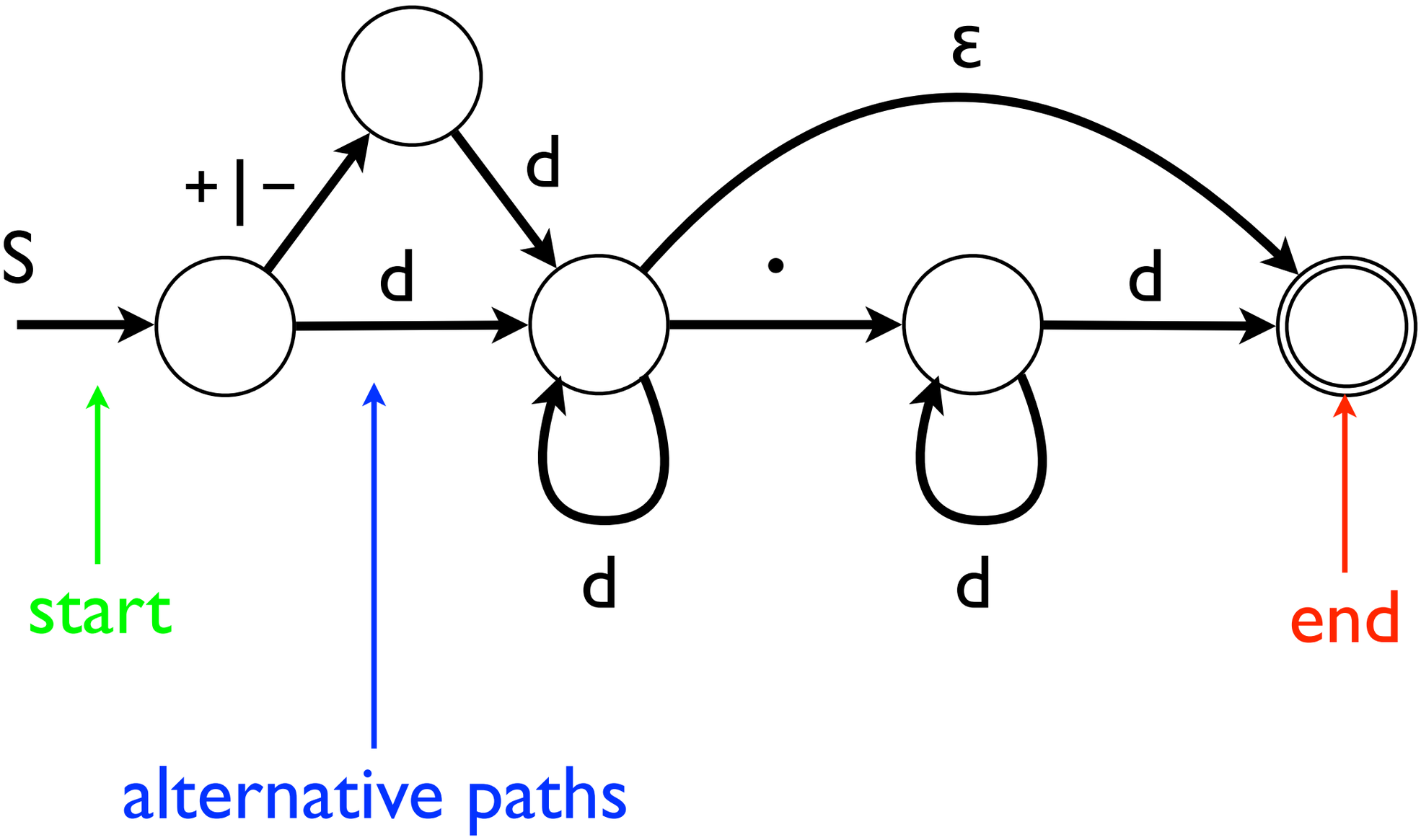
A Decimal Number



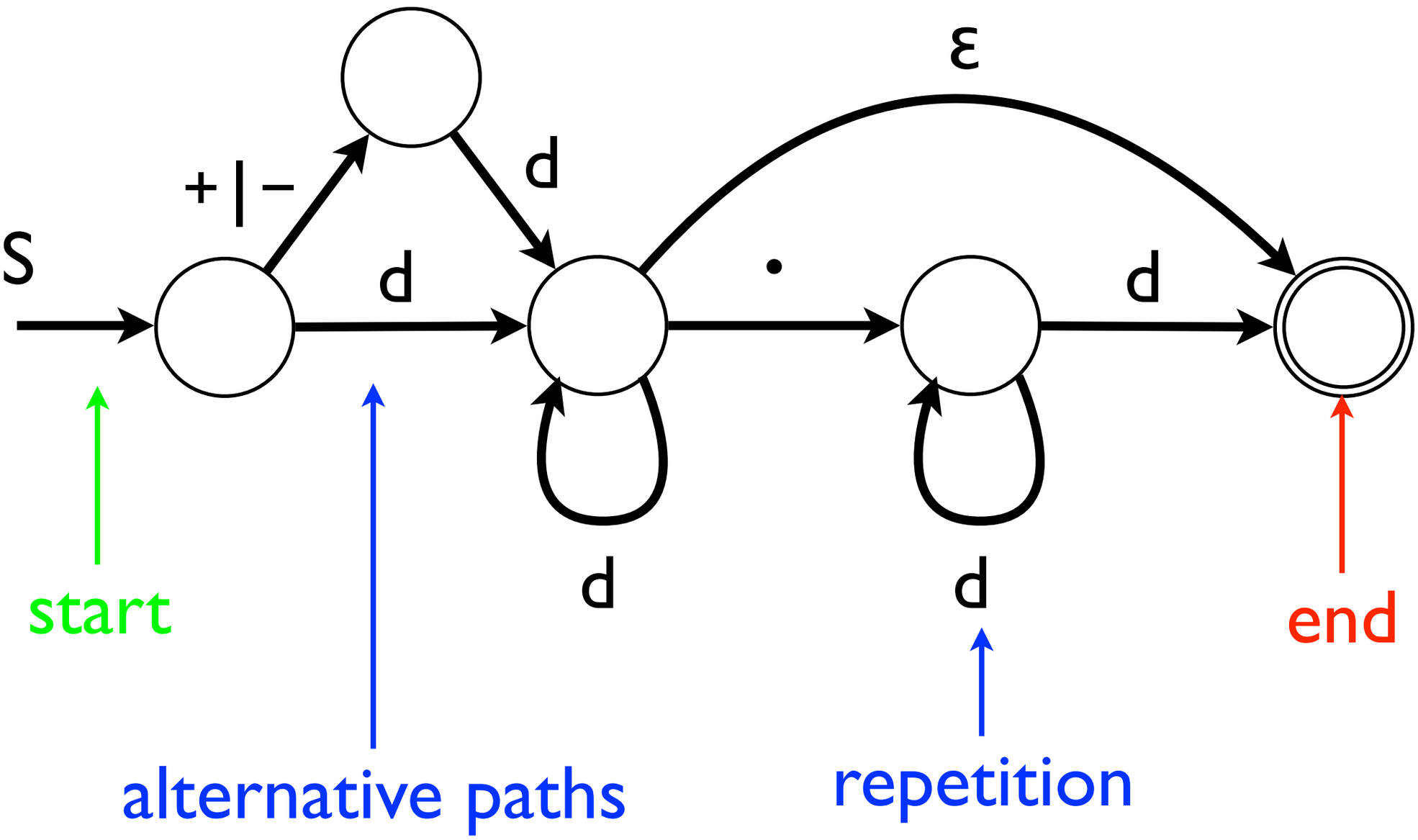
A Decimal Number



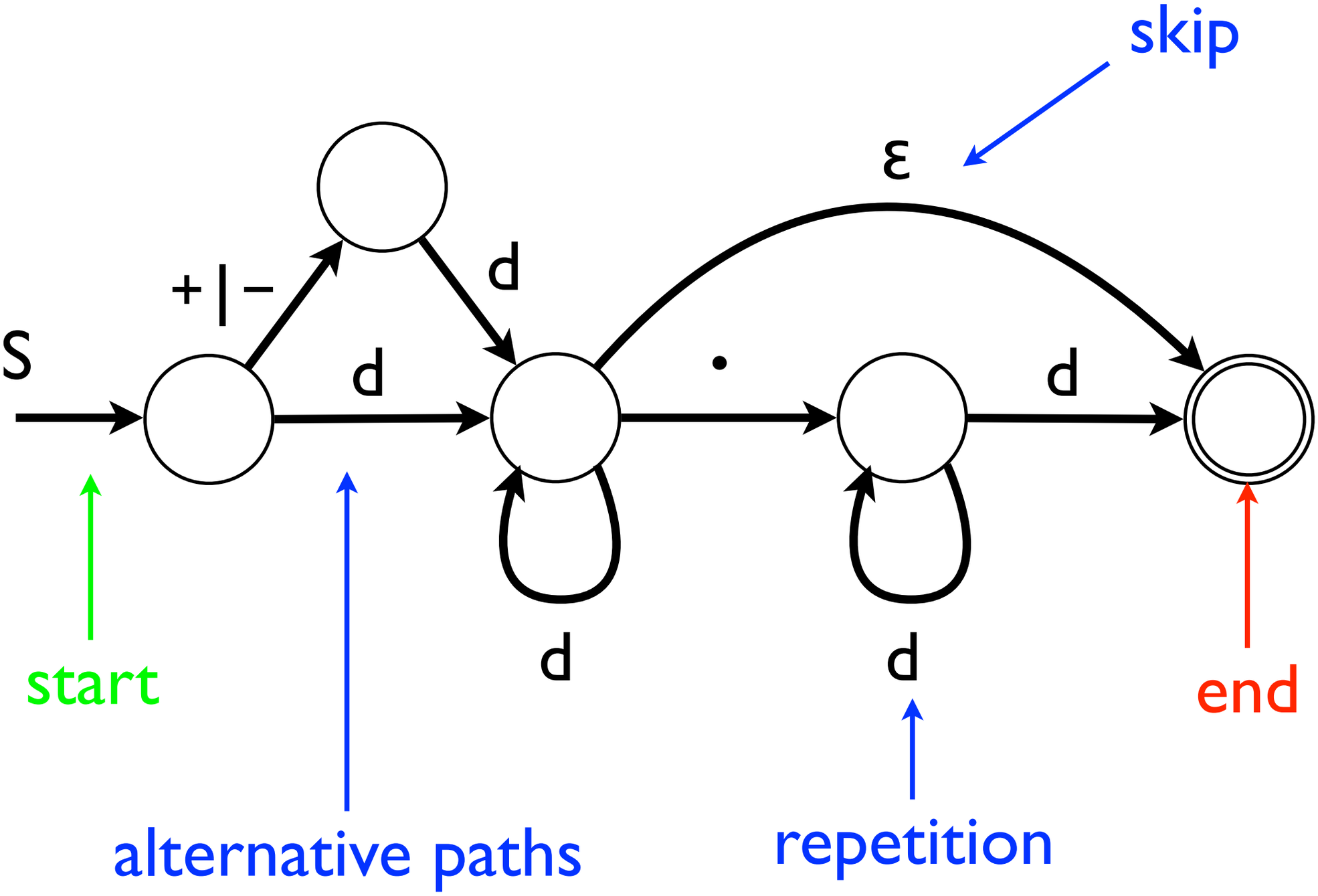
A Decimal Number



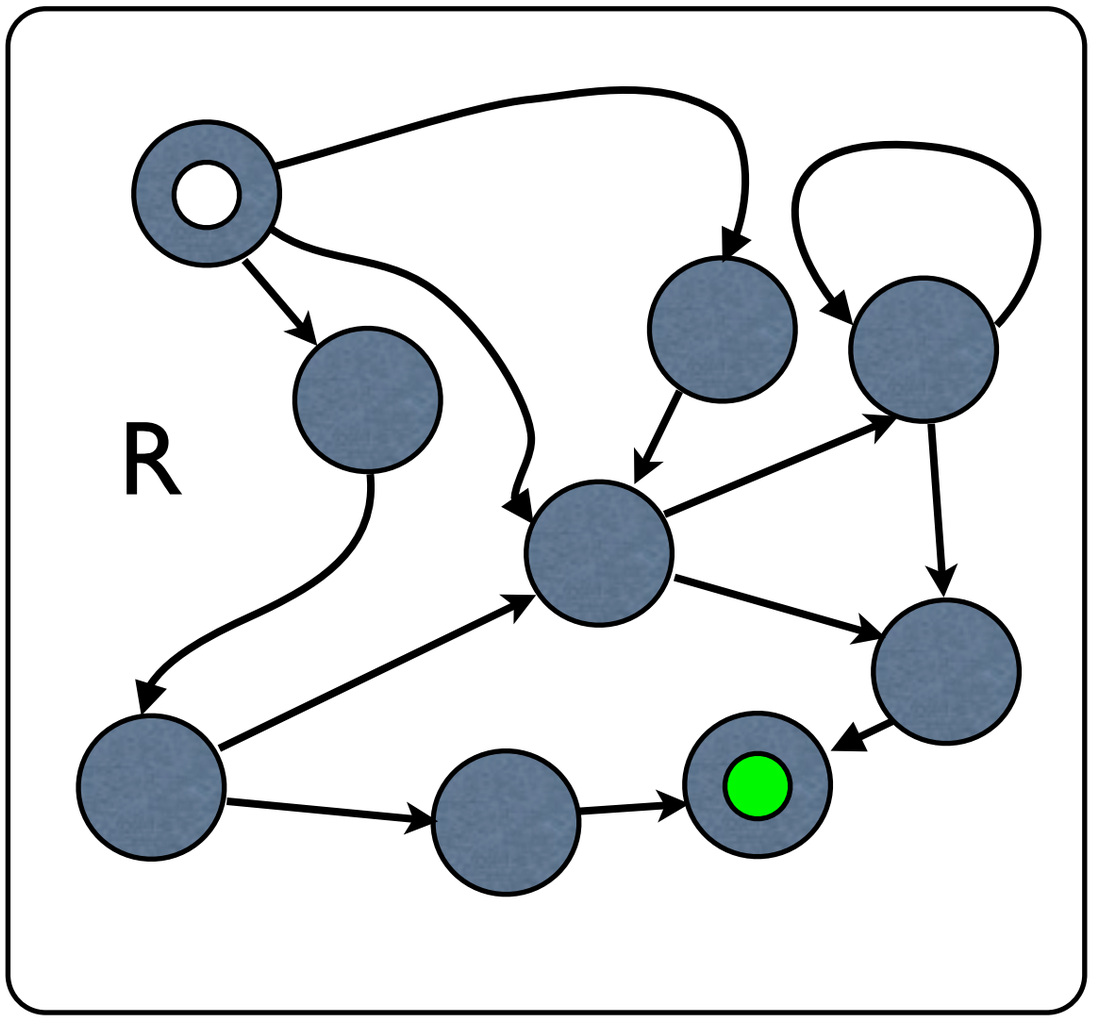
A Decimal Number

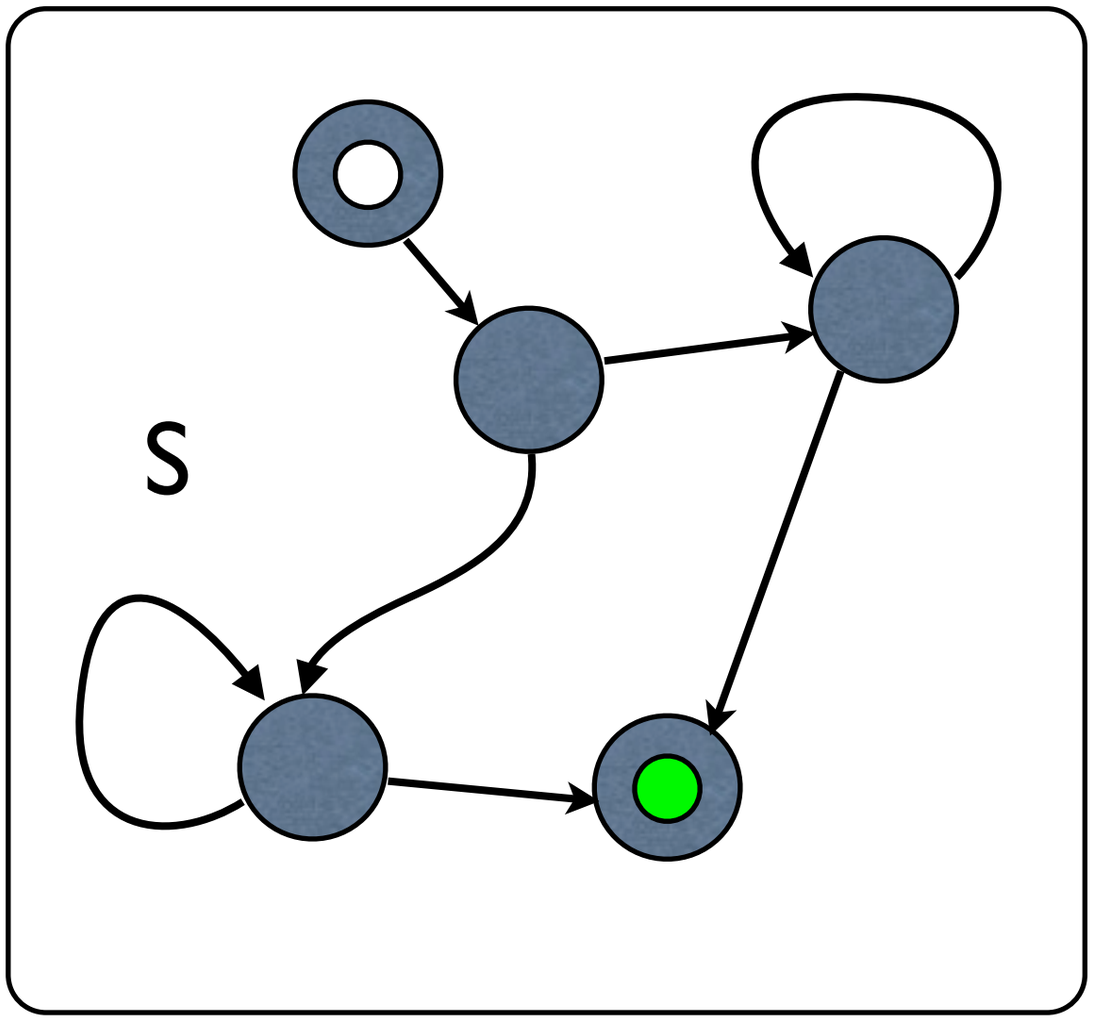


A Decimal Number

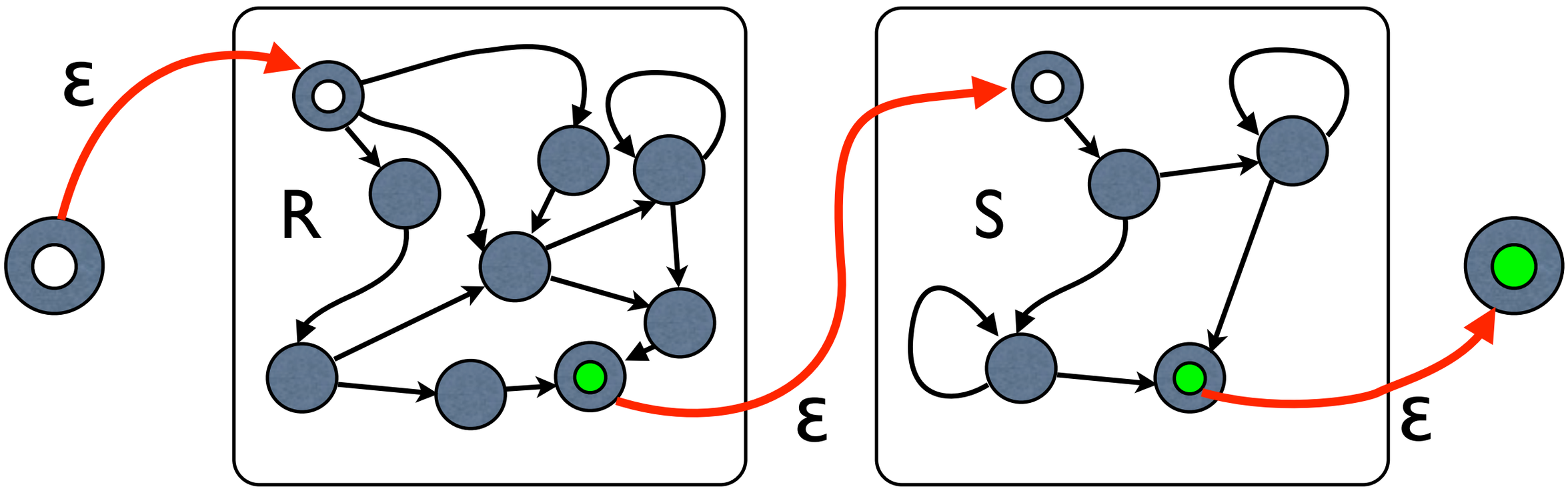


finite state machines





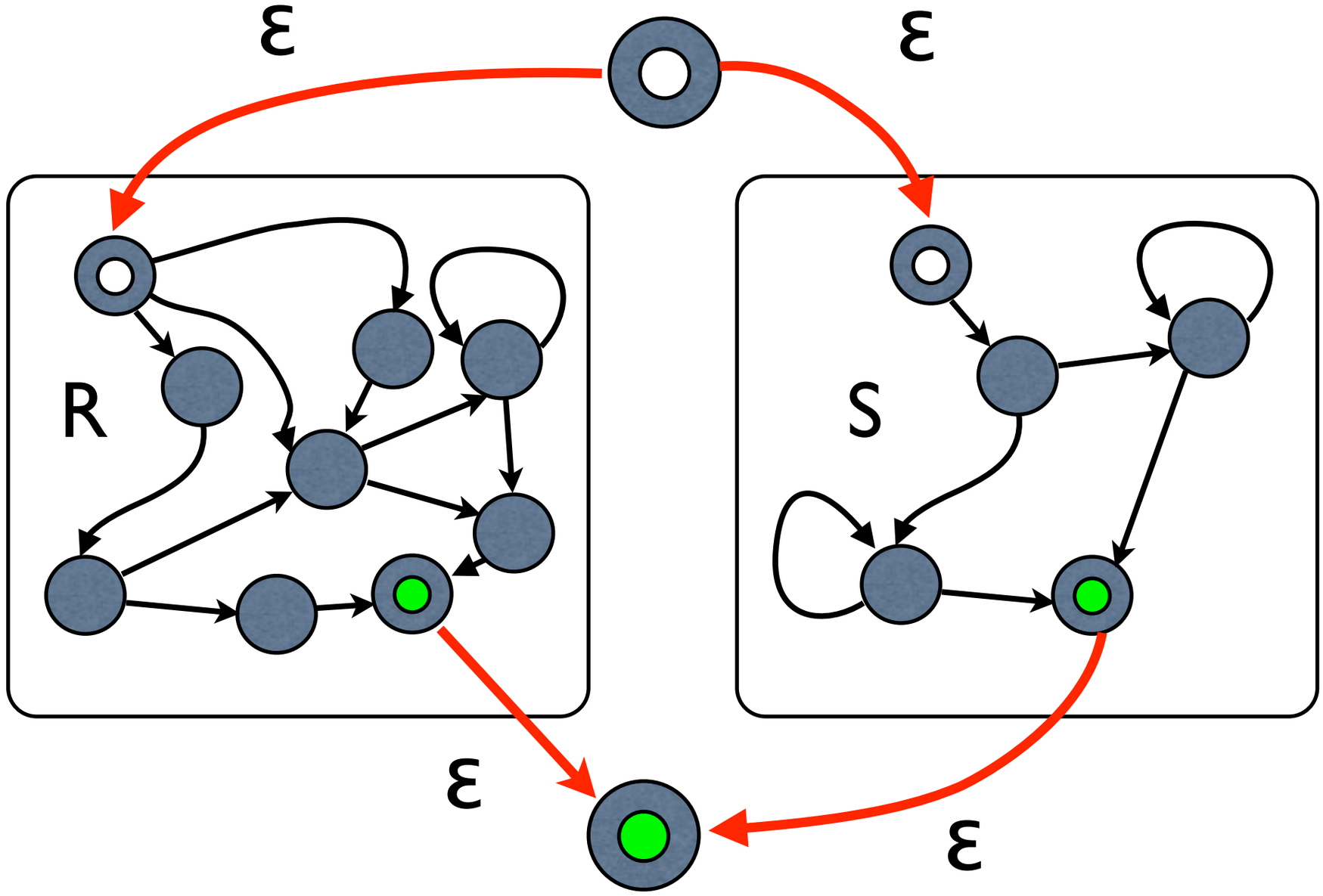
sequence
RS



The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε

alternation

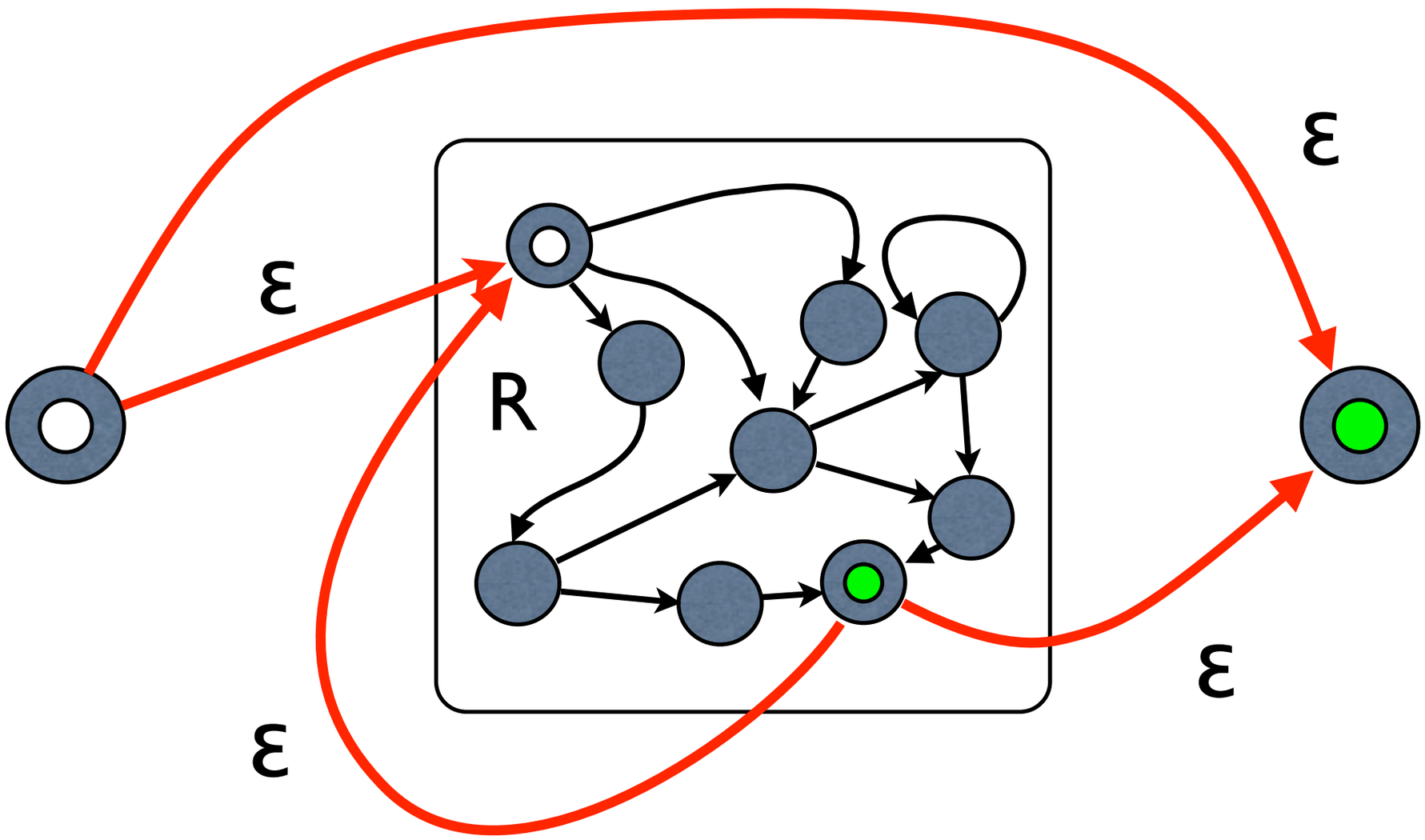
R|S



The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε

iteration

R*



The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε

finite state spaghetti