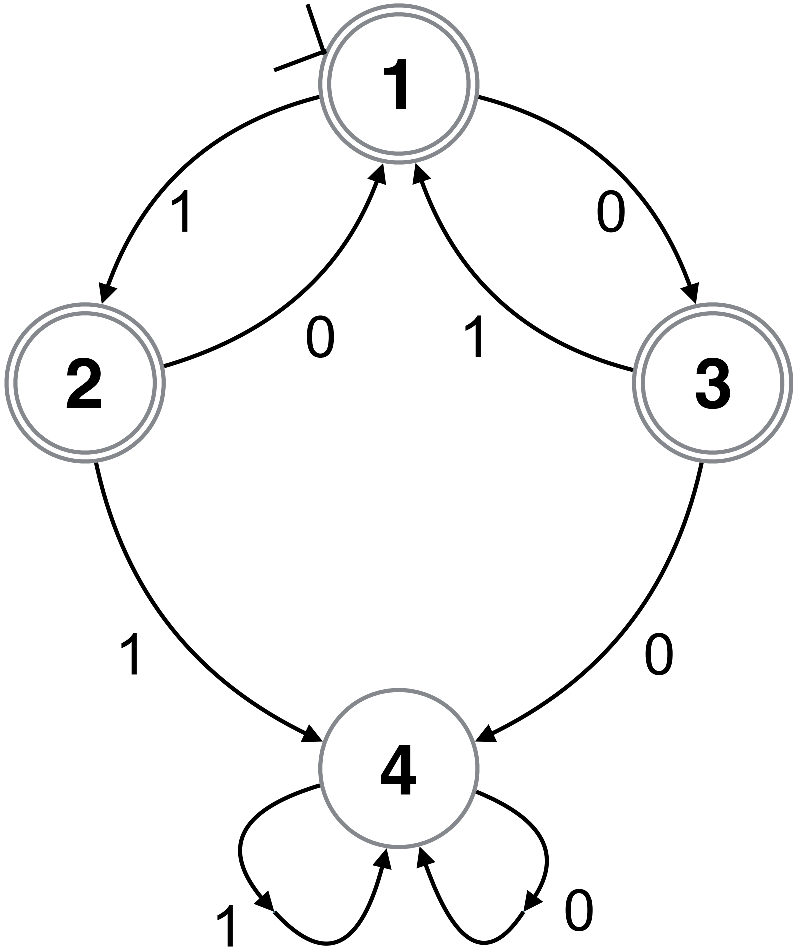
Regular Languages

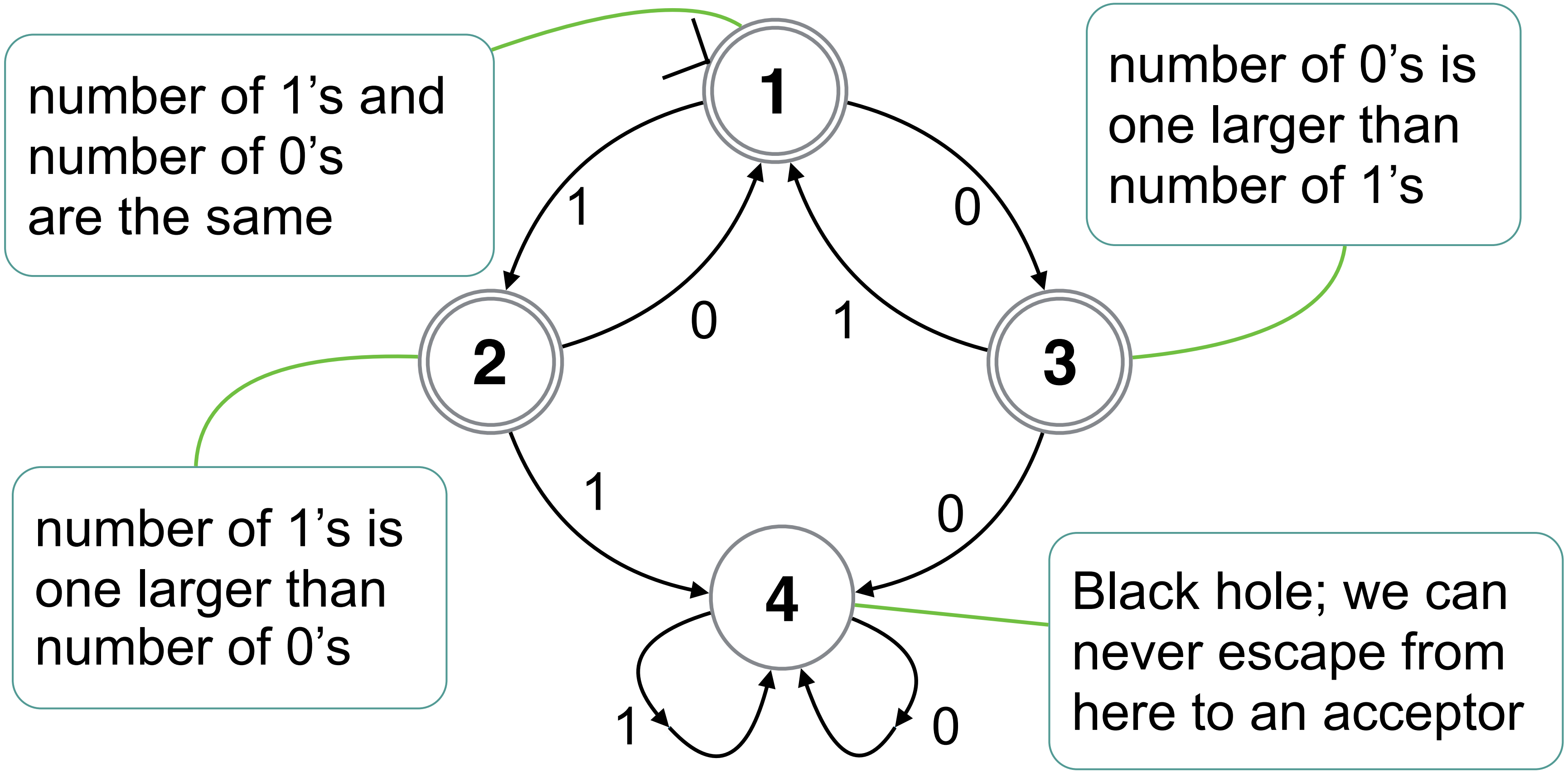
cl

- deterministic machines
- languages and machines
- the Boolean algebra of languages
- non-determinism

Acceptor Example



Acceptor Example



Accepts strings of 0's and 1's for which the difference between number of 0's and number of 1's in a subsequence is at most 1.

A finite-state automaton, or machine (FSM)
M consists of:

Q: the set of states,

Σ: the alphabet of the machine

- the tokens the machine can process,

B: the set of **beginning** or start states of the machine

A: the set of **accepting** states.

δ: the set of **transitions**

is a set of (state, symbol, state) triples

$$\delta \subseteq Q \times \Sigma \times Q.$$

An FSM is a **deterministic** automaton **DFA** if
it has a single start state $B = \{ s_0 \}$
and it has a next-state function

$$F : Q \times \Sigma \rightarrow Q$$

such that $\delta = \{ (q, s, F(q,s)) \mid (q,s) \in Q \times \Sigma \}$

Language

Σ: a finite alphabet

A **language** L is a set of finite strings

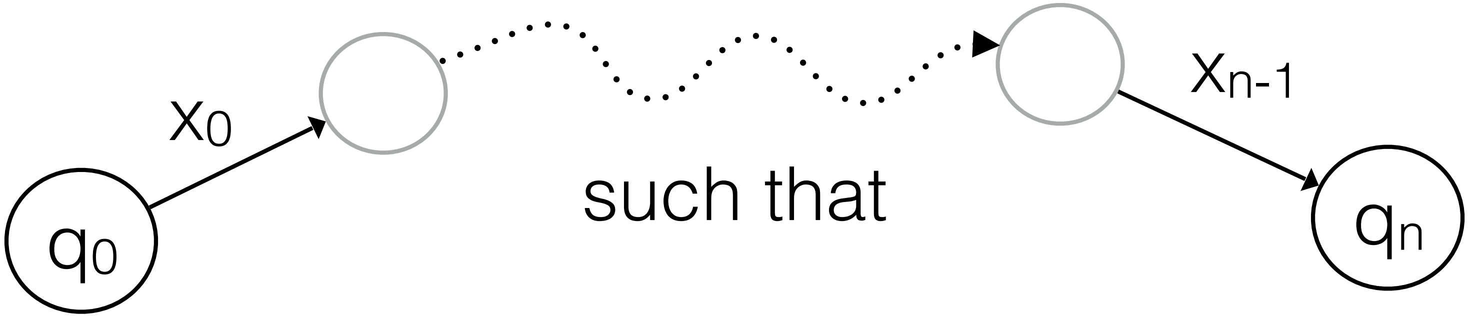
$$L \subseteq \Sigma^*$$

where the **strings** in Σ^* are of
finite sequences of tokens from Σ
the string $< x_0, \ldots, x_{n-1} >$ has length n
strings include the empty string
$\varepsilon = <>$ of length 0

Traces

Let $M = (\mathbf{Q}, \mathbf{\Sigma}, \mathbf{B}, \mathbf{A}, \boldsymbol{\delta})$ be a machine,
and $\mathbf{s} = \langle x_0, \ldots, x_{n-1} \rangle \in \mathbf{\Sigma}^*$

A **trace** for **s** in M is a
sequence $\langle q_0, \ldots, q_n \rangle \in \mathbf{Q}^*$ of states



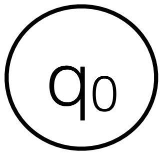
such that

$(q_i, x_i, q_{i+1}) \in \boldsymbol{\delta}$, for each $i < n$,

Traces

Let M = (**Q**, **Σ**, **B, A**, **δ)** be a machine,
and **s** = $< x_0, \ldots, x_{n-1} > \in \mathbf{\Sigma^*}$

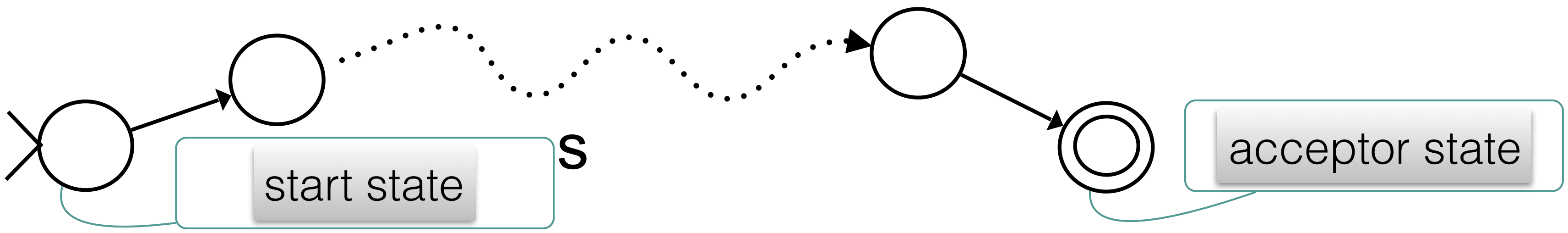
A **trace** for **s** in M is a
sequence $< q_0, \ldots, q_n > \in \mathbf{Q^*}$ of states
such that
$(q_i, x_i, q_{i+1}) \in \boldsymbol{\delta}$**,** for each $i < n$,



When n=0 $<>$

A single state path $< q_0 >$
is a trace for the empty string $<>$

The language accepted by a machine



A string s is accepted if there is a trace for s
from an initial state to an acceptor state.

The **language of M** is the set of sequences it accepts.

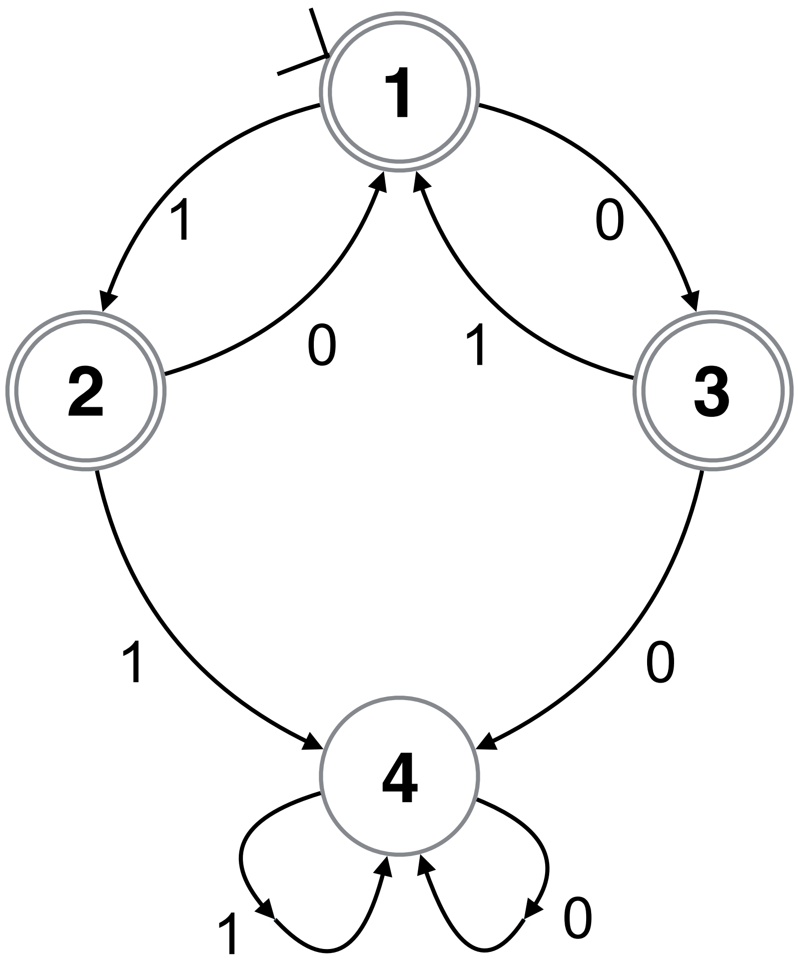
$$L(M) \subseteq \Sigma^*$$

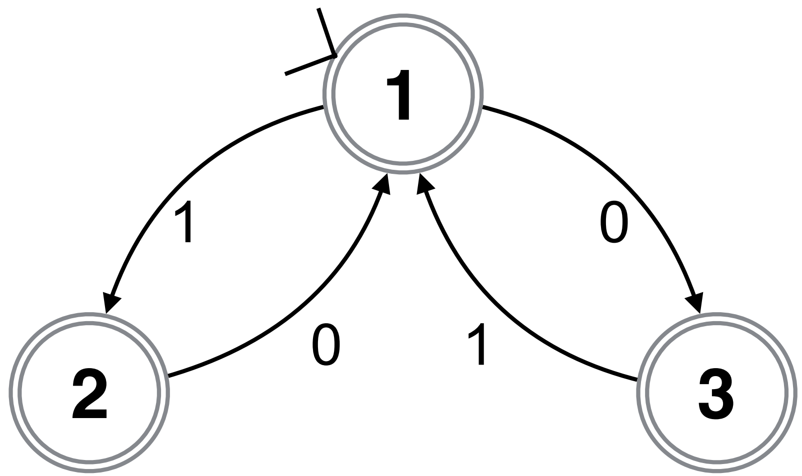
Two machines are **equivalent** if they define the same language.

A **regular language** is

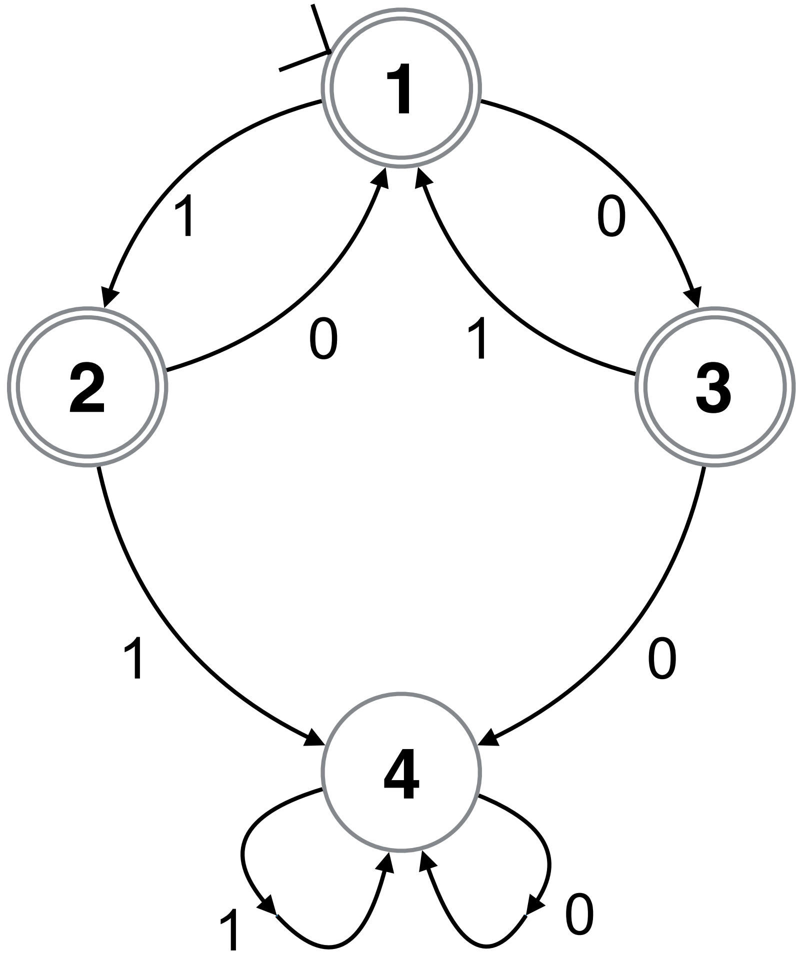
a language represented by some DFA

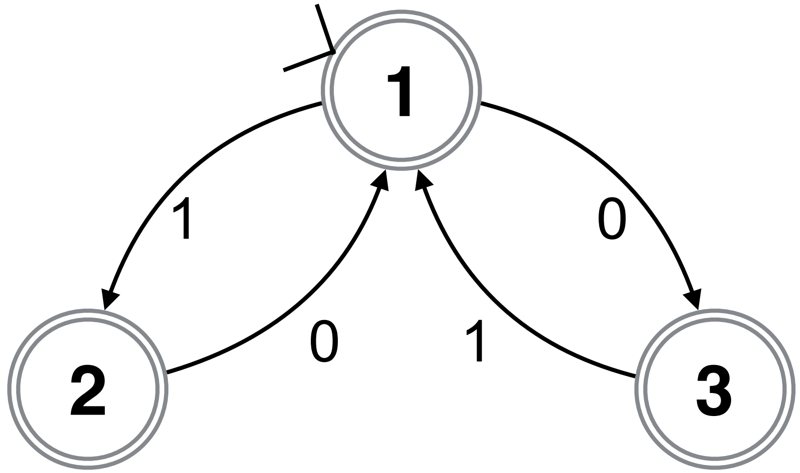
Are these two machines equivalent?





Are these two machines equivalent?

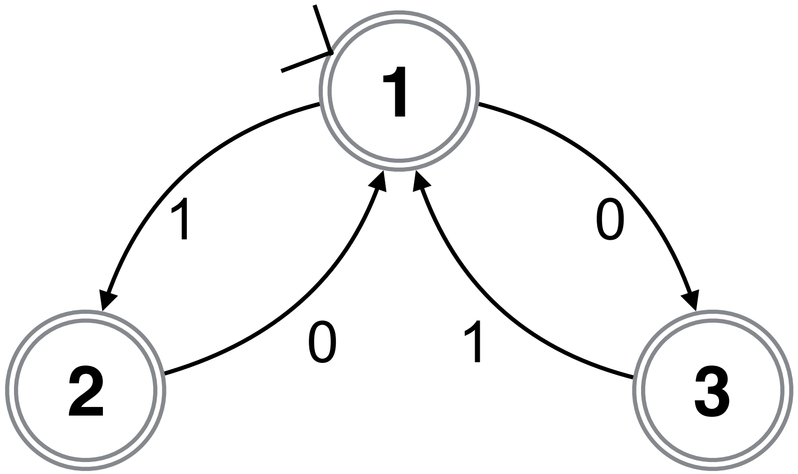




Yes
If there is a path from
the start state to an
accepting state then
it only uses states
1, 2, 3

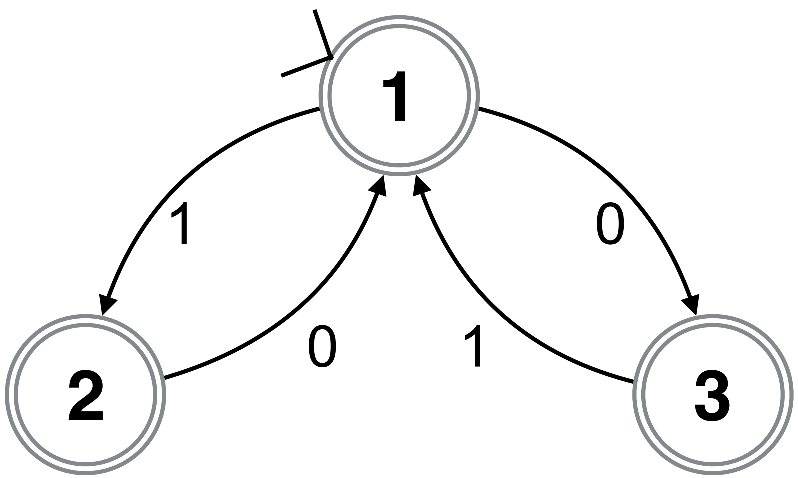
The two machines accept the same strings

A machine with **at most one** transition with a given label from a given state



This machine is not a DFA but it is equivalent to a DFA

A machine with **at most one** transition with a given label from a given state



This machine is not a DFA but it is equivalent to a DFA

If a machine has at most one transition with a given label from any state

Then we can construct an equivalent DFA by adding a new *black-hole* state, which is not an accepting state and making the missing transitions go there.

Determinism

If we have a machine with at most one transition for each (q,s) pair, we can always convert to an equivalent DFA for which every state has exactly one transition leaving the state for each input symbol.

For this machine there is exactly one trace for each input string

- Proof

  Add a new "black hole" state, ●

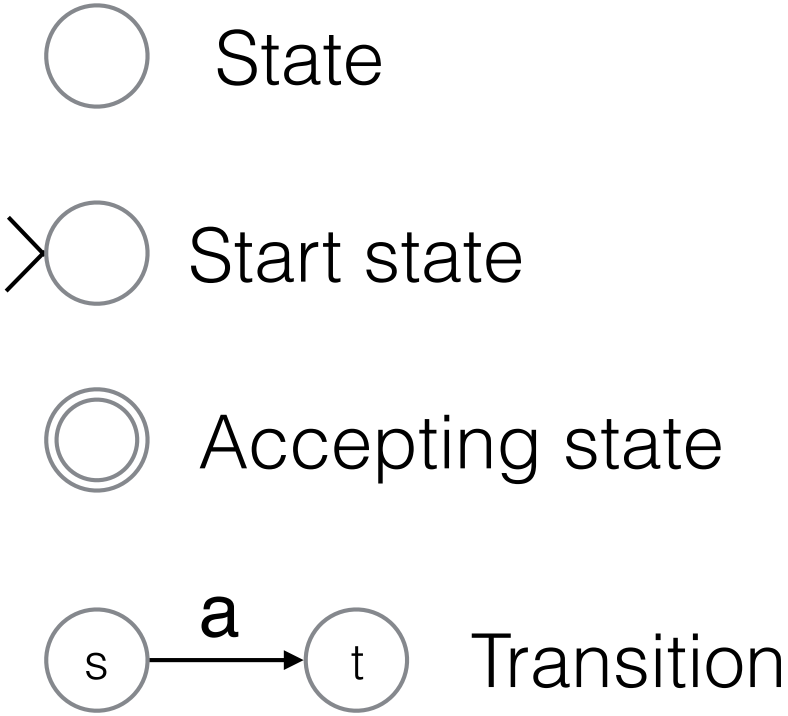
  For every pair (q, s) for which there is no state r with a transition T(q, s, r), add a transition T(q, s, ●).

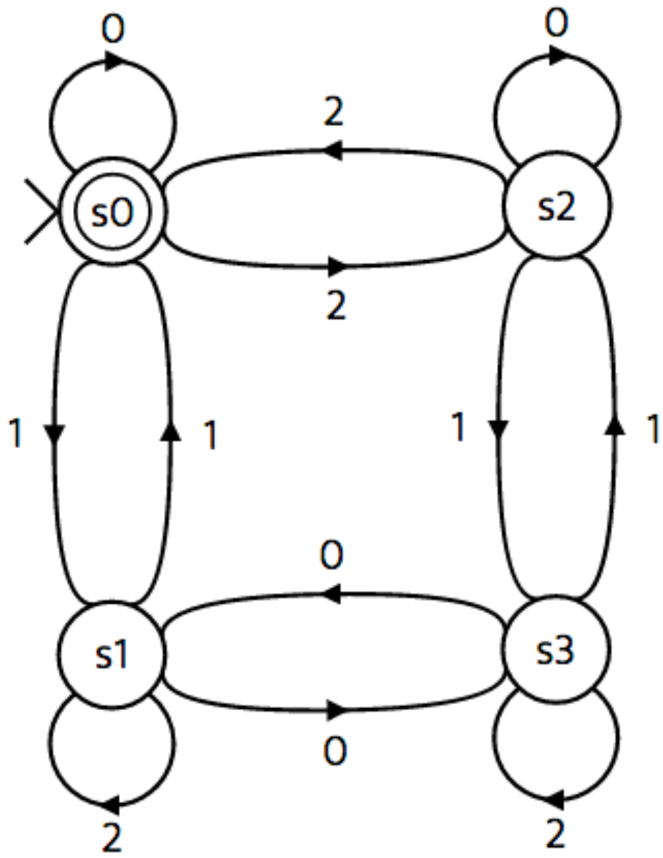
  This includes a transition T(●, a, ●) for each a $\in \Sigma$. You cannot escape from the black hole.

  The black hole ● is not an accepting state.

This machine accepts the same language as the original.

Any language recognised by
a machine with **at most one** transition
with a given label from a given state
is regular.





depending on the application,
a is a letter, symbol, token, action, …

Abstraction

What happens if we make some transitions invisible?

if $\Sigma \subseteq \Sigma'$, say $\Sigma' = \Sigma \cup \{\varepsilon\}$
then
from every string in Σ'^*
we can get a string in Σ^*
by erasing all occurrences of ε

If L' is a regular language with alphabet Σ'
the language L is obtained from L'
by deleting all occurrences of ε
is L a regular language

non-determinism

- many arguments are easier for NFA
- we will see that NFA with an invisible ε define the same languages as NFA without ε
- we will see that NFA the languages defined by NFA form a Boolean Algebra of subsets of Σ^*
- we will see that NFA define the same languages as DFA, so any language defined by an NFA is regular

Abstraction

What happens if we make some transitions invisible?

if $\Sigma \subseteq \Sigma'$, say $\Sigma' = \Sigma \cup \{\varepsilon\}$

then

from every string in Σ'^*

we can get a string in Σ^*

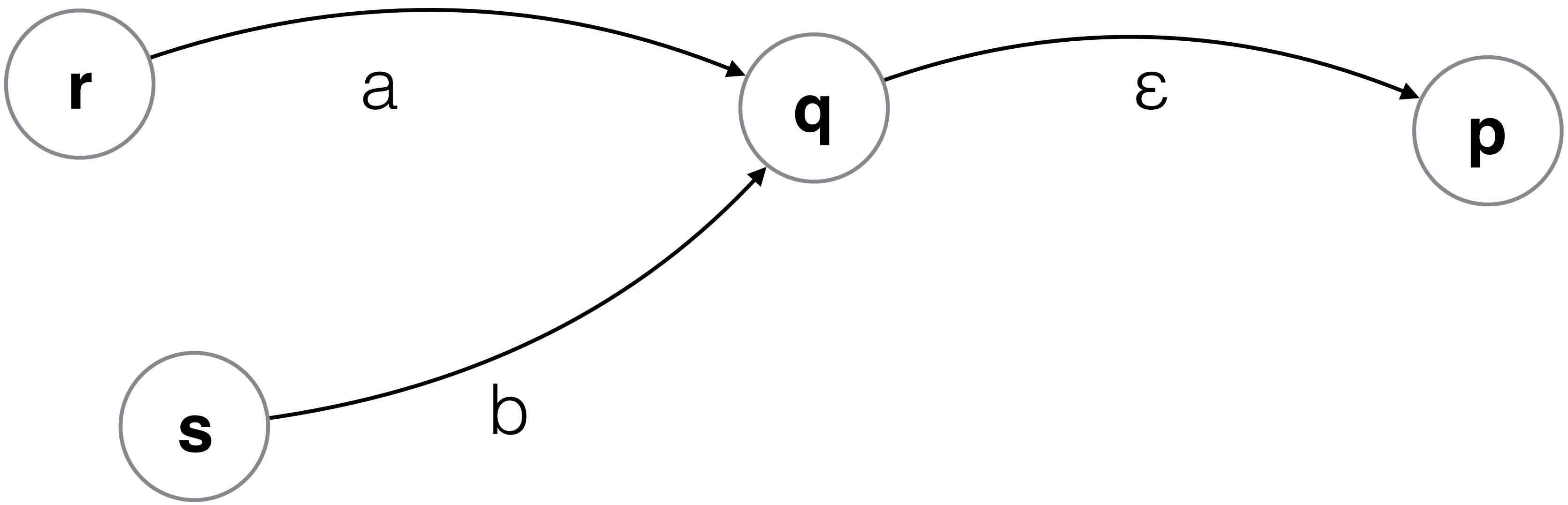
by erasing all occurrences of ε

If L' is a regular language with alphabet Σ'

the language L is obtained from L'

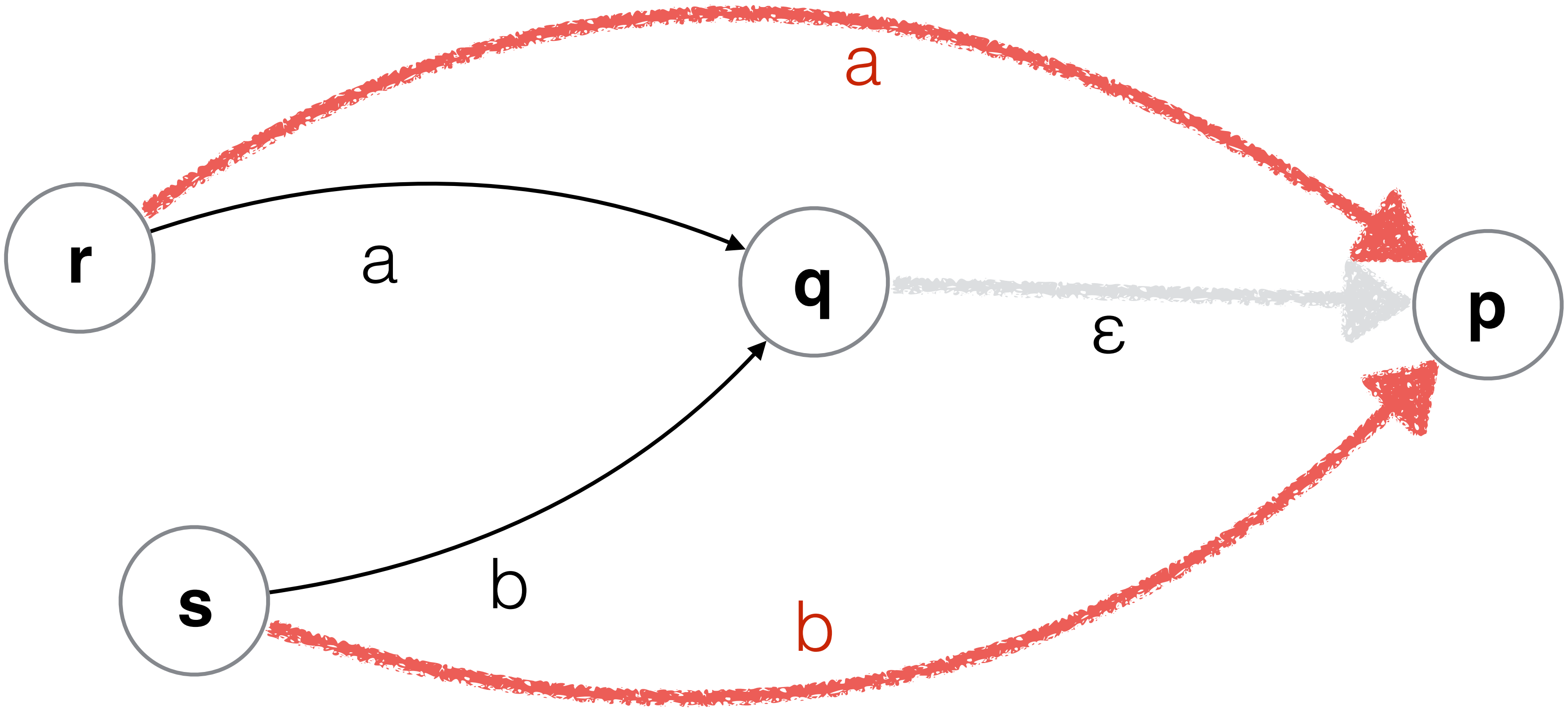
by deleting all occurrences of ε

is L a regular language



Suppose we have a machine with transitions as shown, that recognises L′.

Can we create a machine without ε that recognises L?



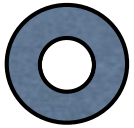
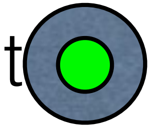
Whenever **(q, ε, p)** ∈ **δ**

remove this transition and for every

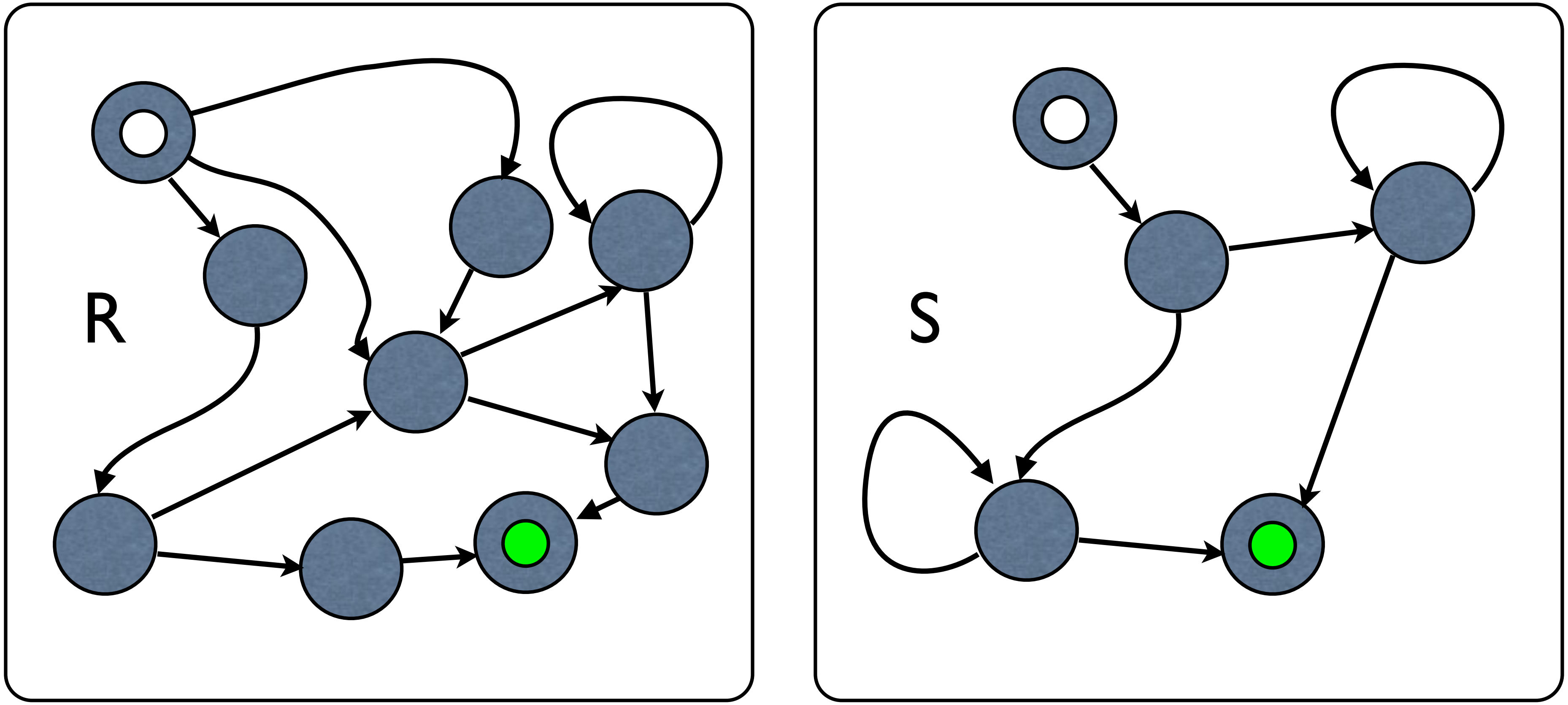
transition (x, s, **q**) ∈ **δ**

add a transition (x, s, **p**);

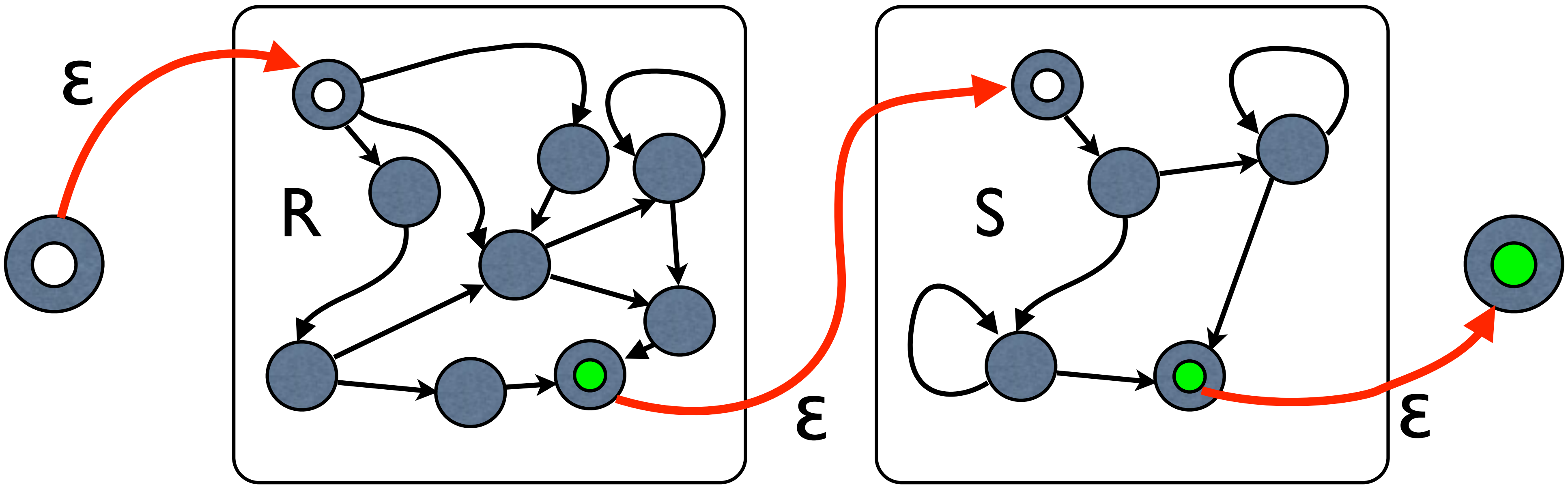
if **q** is a start state, make **p** a start state

The new machine has alphabet Σ and recognises L start  accept 

finite state machines

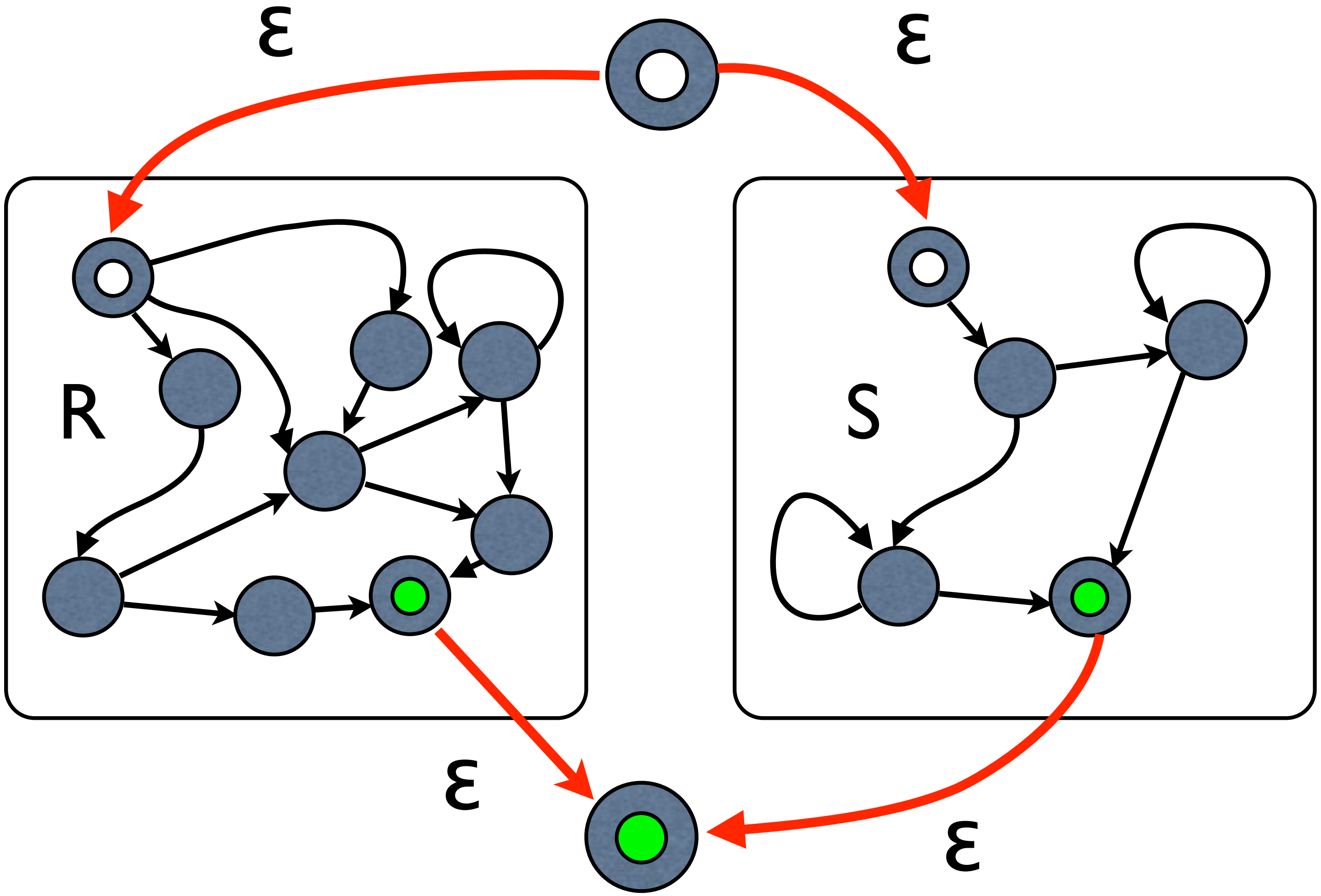


sequence
RS



alternation

R|S



iteration

R*