Finite Automata

Part Three

From Last Time

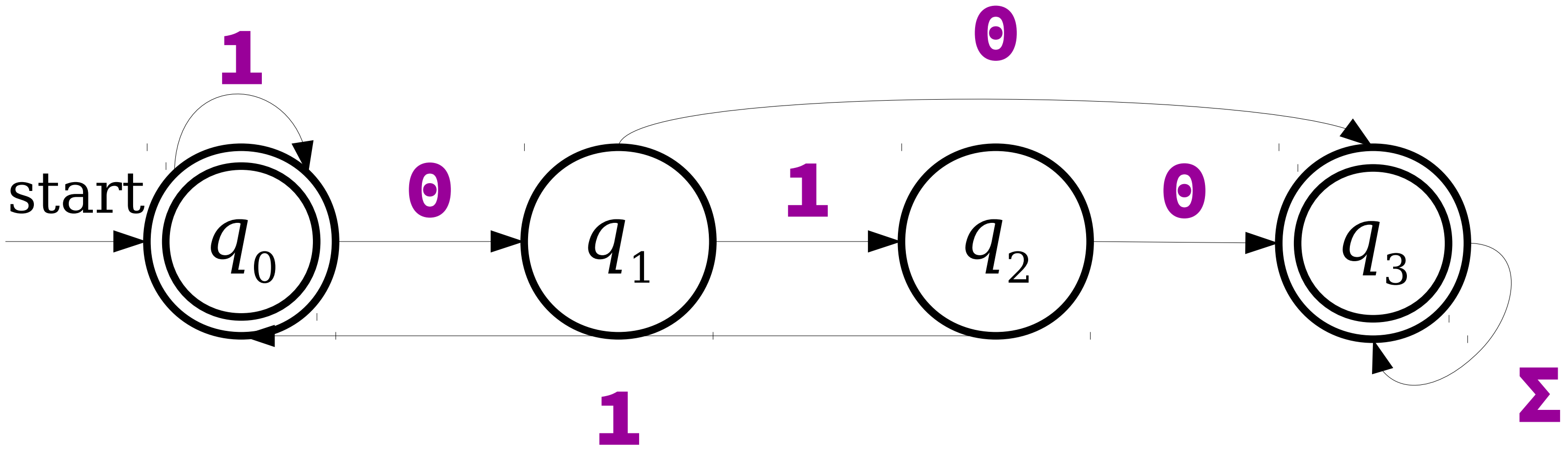
NFAs and DFAs

- **DFAs** (Deterministic Finite Automata)
  - are machines for accepting/rejecting strings.
  - The language of a DFA is the set of strings it accepts.
  - The set of languages for which there exists a DFA is called the **Regular Languages**.
- **NFAs** (Nondeterministic Finite Automata)
  - are DFAs but with some bonus superpowers of having more options for how we move from state to state.

New Stuff!

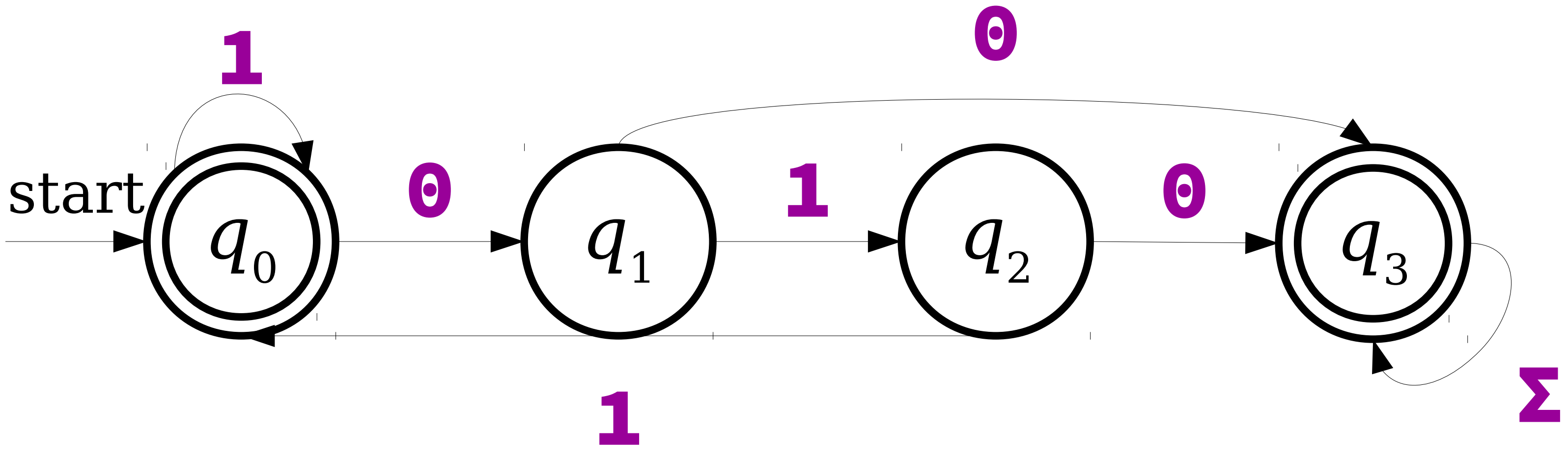
Table Representation of DFAs

Tabular DFAs



	0	1
q_0		
q_1		
q_2		
q_3		

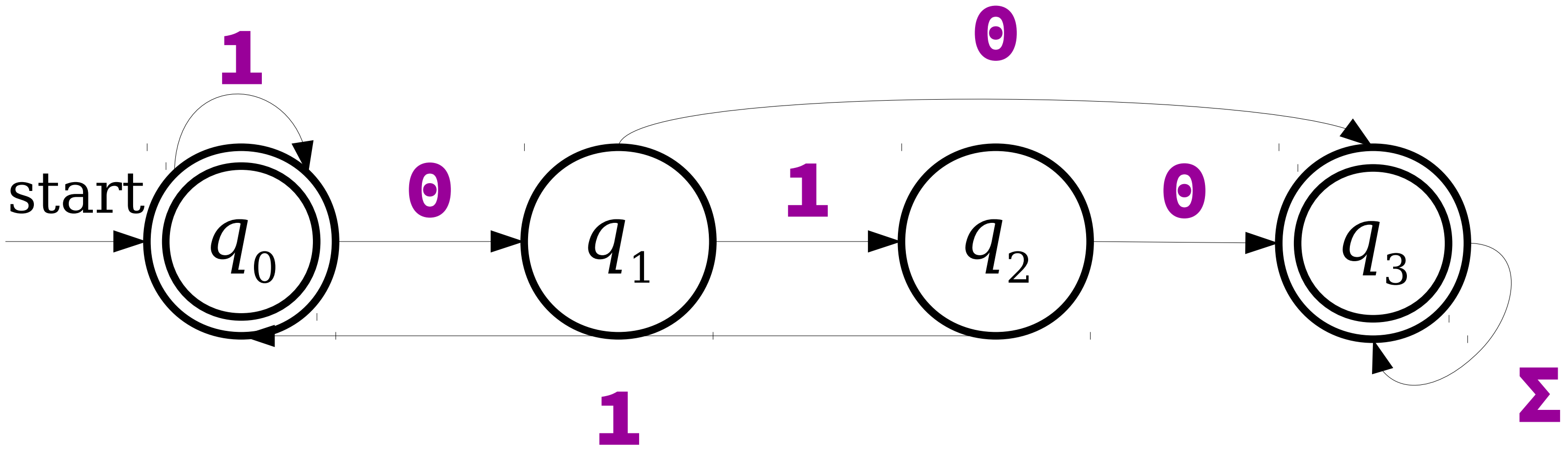
Tabular DFAs



	0	1
q_0	q_1	q_0
q_1		
q_2		
q_3		

Since this is the first row, it's the start state.

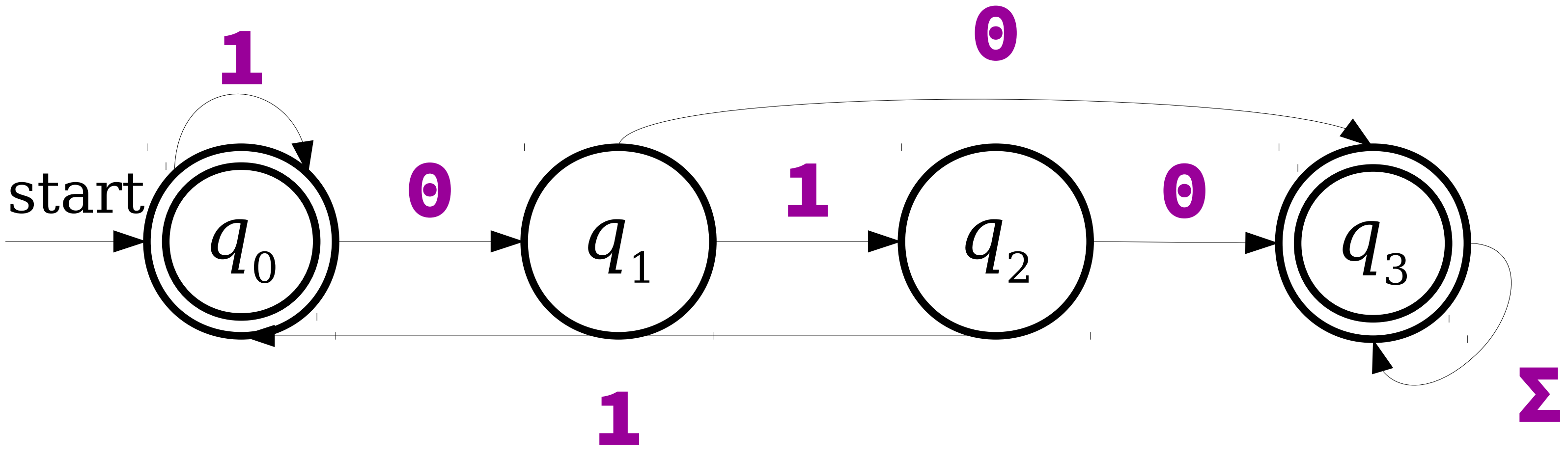
Tabular DFAs



	0	1
q_0	q_1	q_0
q_1		
q_2		
q_3		

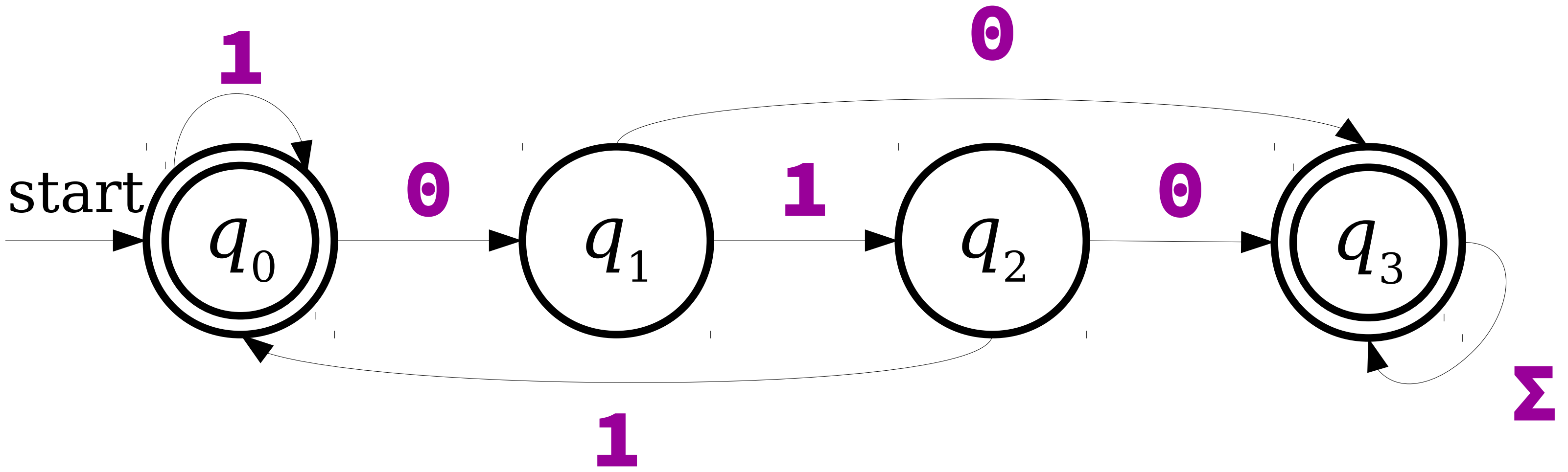
Question:
What goes in the q_1 row of the table?

Tabular DFAs



	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3		

Tabular DFAs



(A)

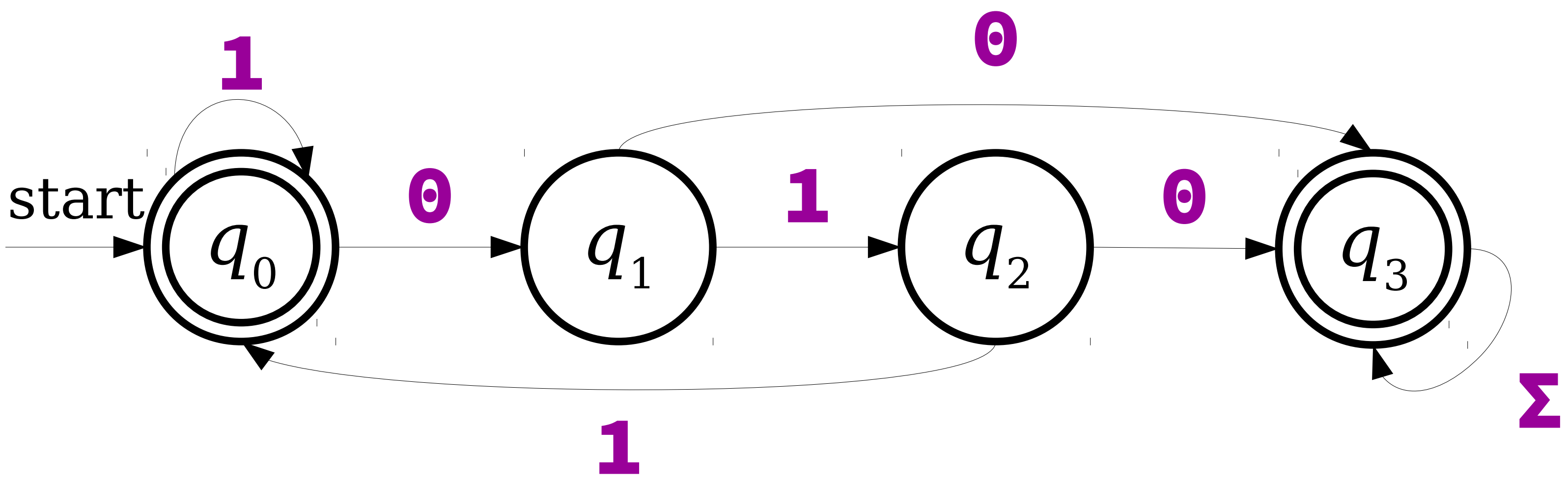
	0	1	Σ
q_0	q_1	q_0	-
q_1	q_3	q_2	-
q_2	q_3	q_0	-
q_3	-	-	q_3

(B)

	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3	q_3	q_3

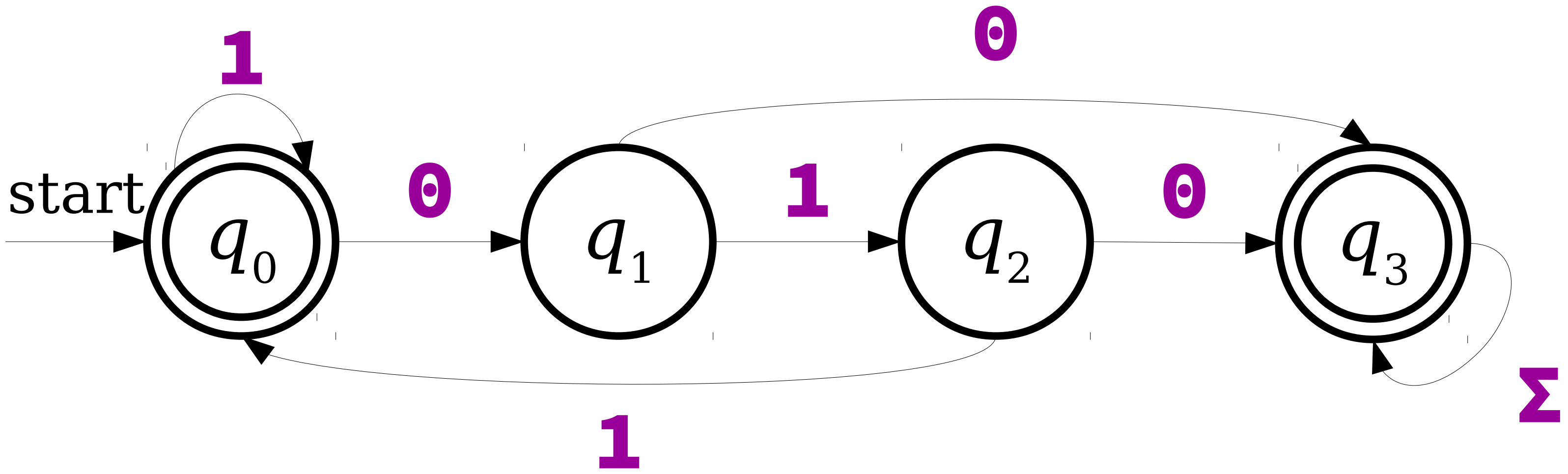
Question:
How should we complete the table?
Go to
pollev.com/cs103spr25

Tabular DFAs



	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3	q_3	q_3

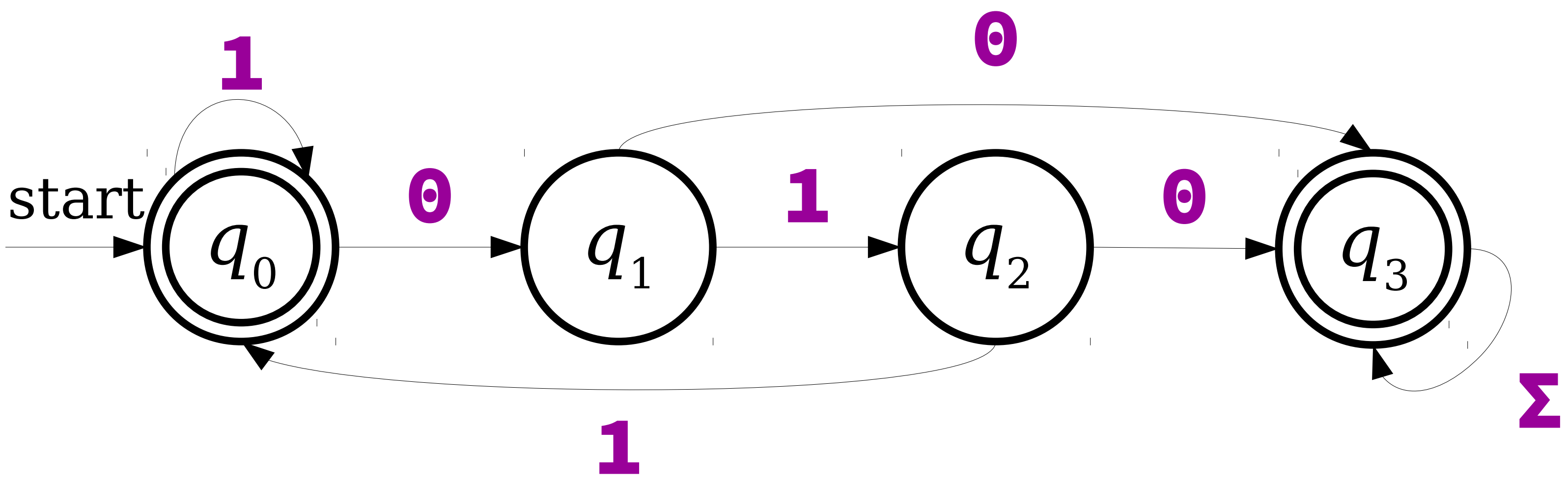
Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

These stars indicate accepting states.

Tabular DFAs



	0	**1**
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

NFAs and DFAs

- We know that any language for which there exists a DFA can also be recognized by an NFA.
- Why?
  - Every DFA is essentially already an NFA!
    - (No requirement that a given NFA use every NFA superpower available)

NFAs and DFAs

- We know that any language for which there exists a DFA can also be recognized by an NFA.
- Why?
  - Every DFA is essentially already an NFA!
    - (No requirement that a given NFA use every NFA superpower available)
- ***Question:*** Can any language recognized by an NFA also be recognized by a DFA?

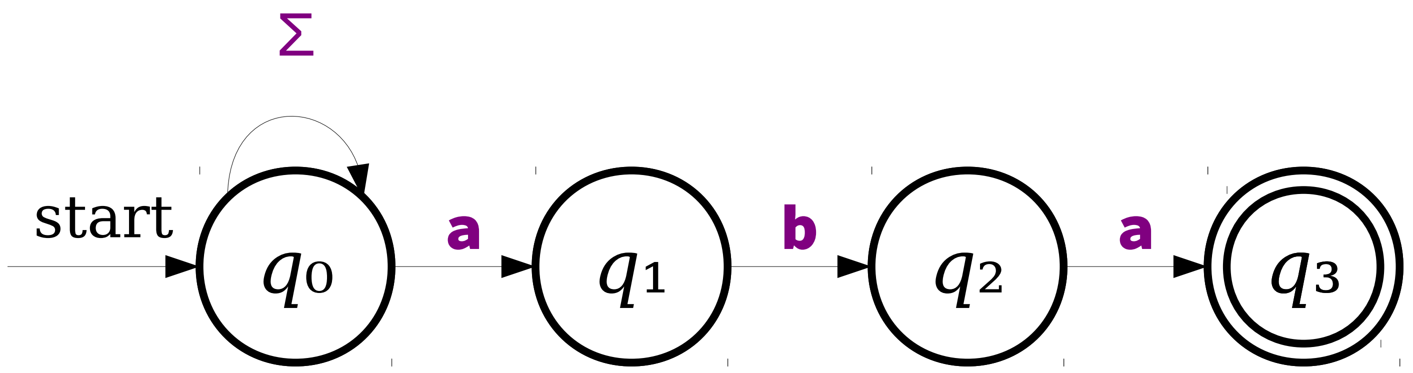
NFAs and DFAs

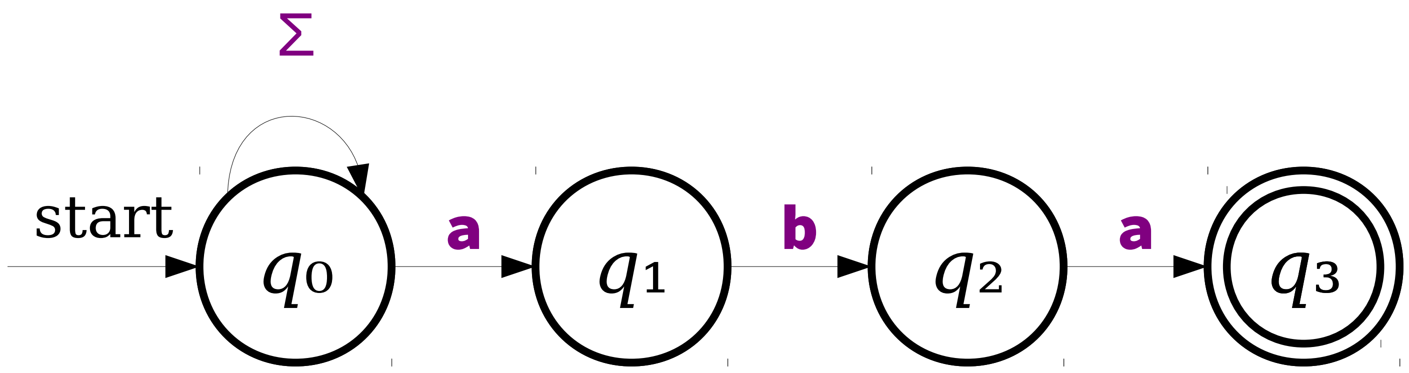
- We know that any language for which there exists a DFA can also be recognized by an NFA.
- Why?
  - Every DFA is essentially already an NFA!
    - (No requirement that a given NFA use every NFA superpower available)
- ***Question**:* Can any language recognized by an NFA also be recognized by a DFA?
- Surprisingly, the answer is ***yes***!

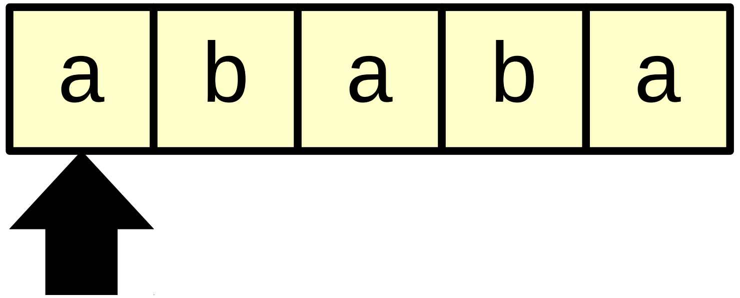
NFAs and DFAs

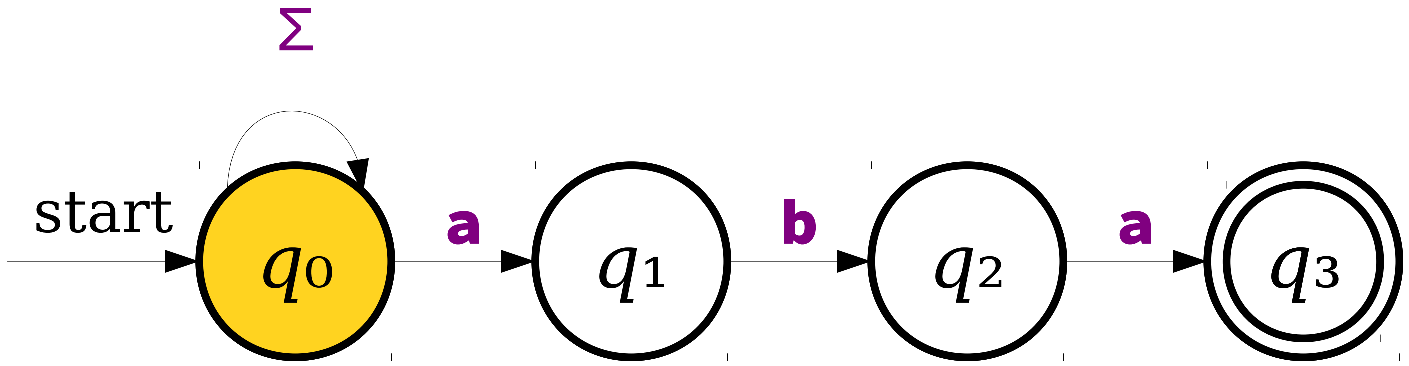
- ***Question:*** Can any language recognized by an NFA also be recognized by a DFA?
- Surprisingly, the answer is ***yes***!
  - Theorem: **For all languages L**, if L is recognized by an NFA, then there exists a DFA that also recognizes L.
  - To prove this, we need to:
    - **Pick an arbitrary language L**, assuming an NFA exists for L
    - **Want to show** there exists a DFA for L (describe how we would construct a DFA with the same language, in a generalizable way)
    - For the next few slides, we'll ponder how to approach that...

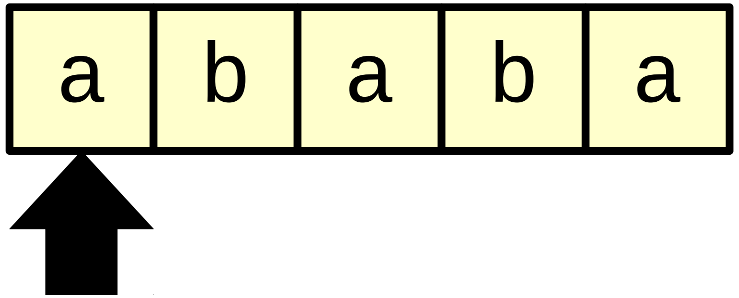
Thought Experiment:
How would you simulate an NFA in software?

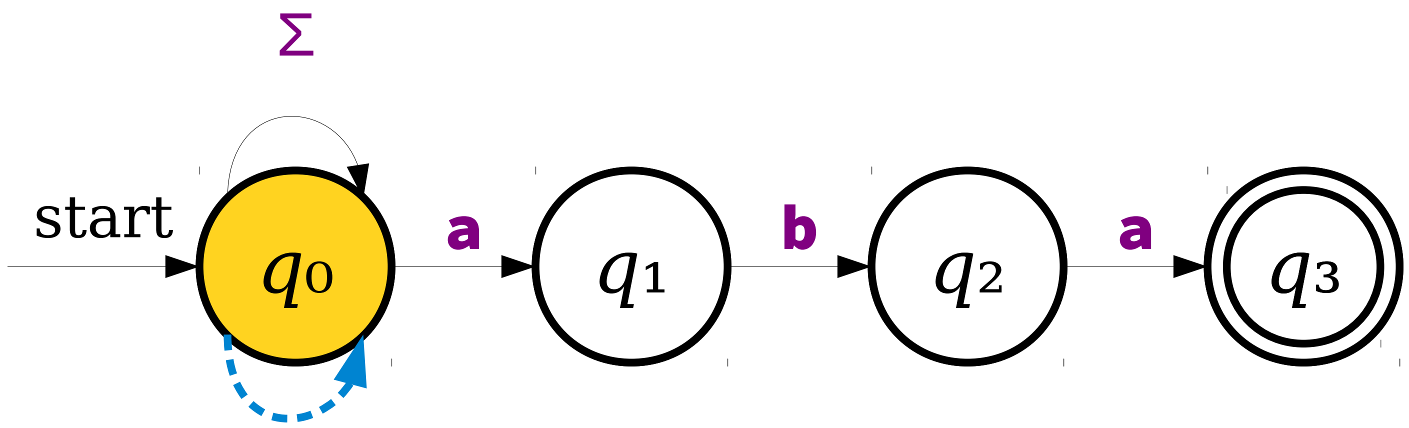

Σ
start
q_0
a
q_1
b
q_2
a
q_3

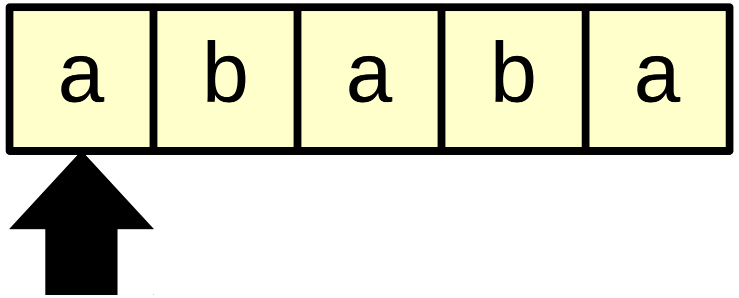

Σ
start
q_0
a
q_1
b
q_2
a
q_3

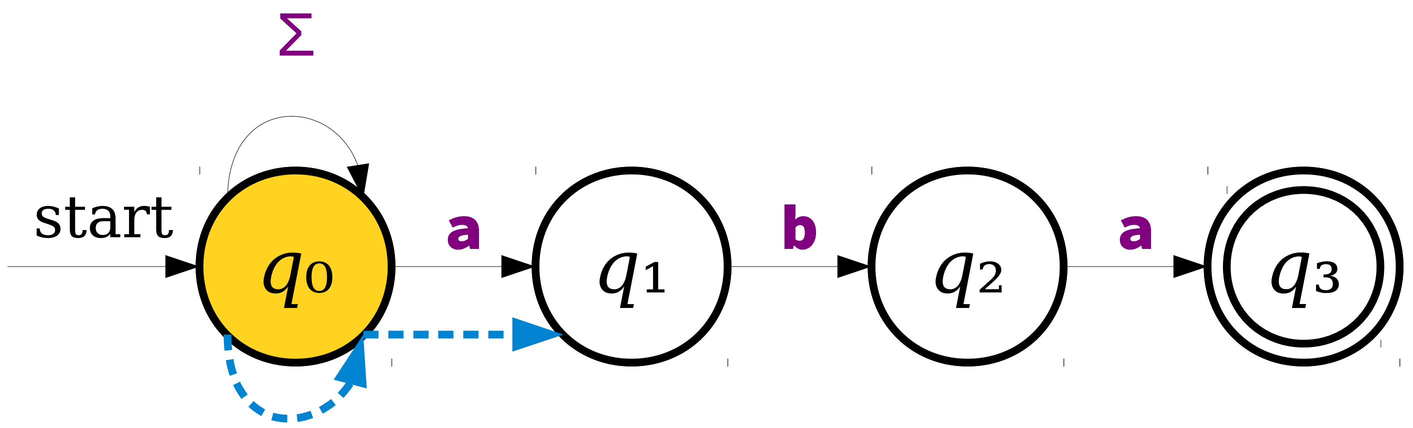

a
b
a
b
a

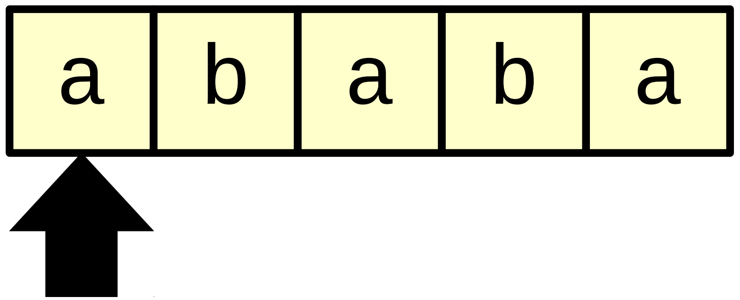

Σ
start
q_0
a
q_1
b
q_2
a
q_3

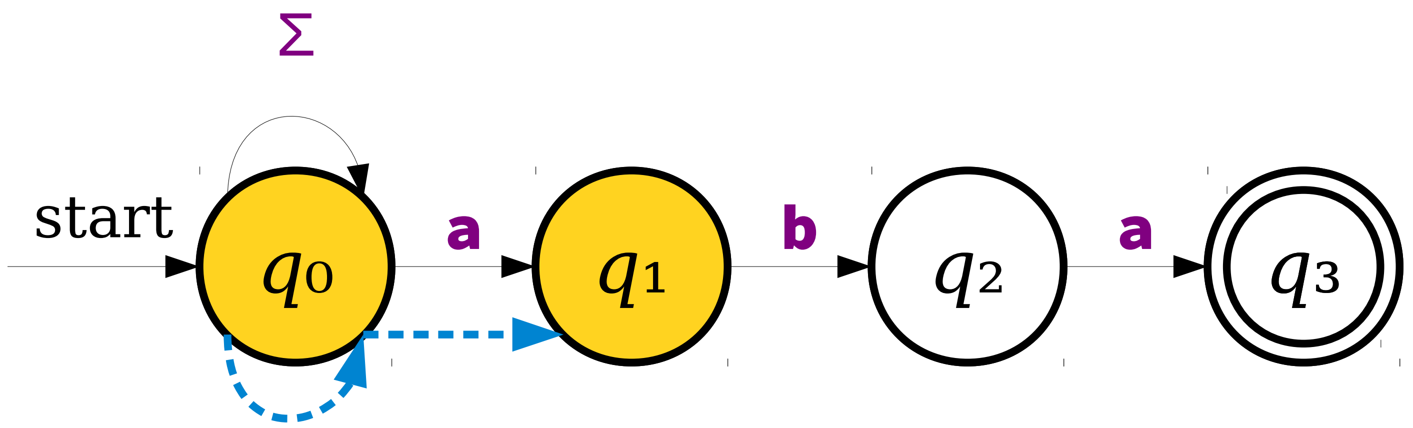

a
b
a
b
a


Σ
start
q_0
a
q_1
b
q_2
a
q_3

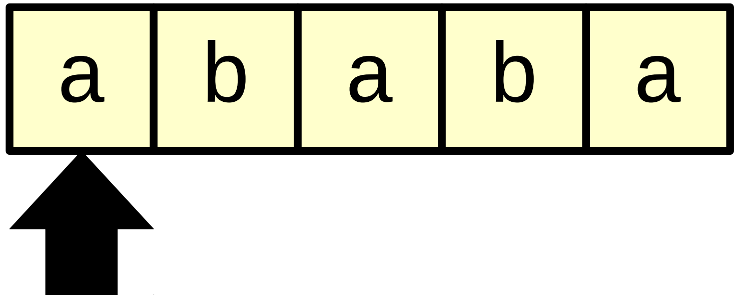

a b a b a


Σ
start
q0
a
q1
b
q2
a
q3

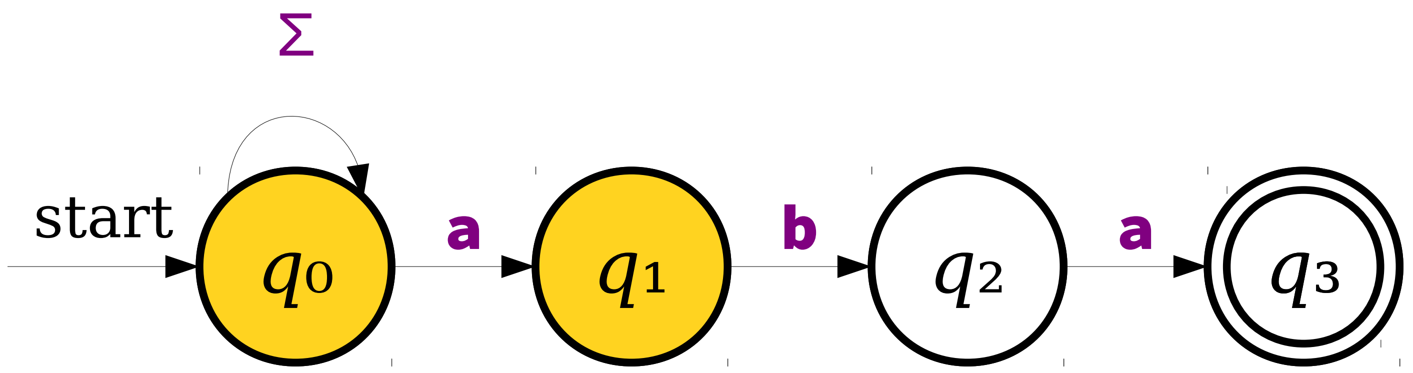

a b a b a



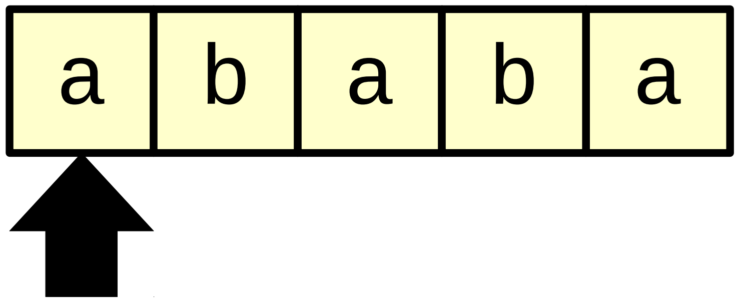
Σ
start
q_0
a
q_1
b
q_2
a
q_3



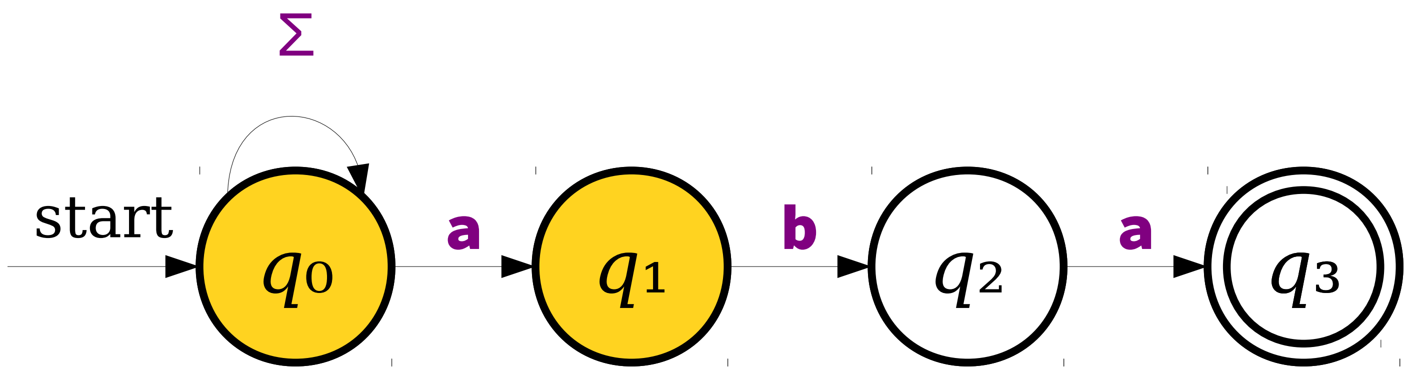
a
b
a
b
a

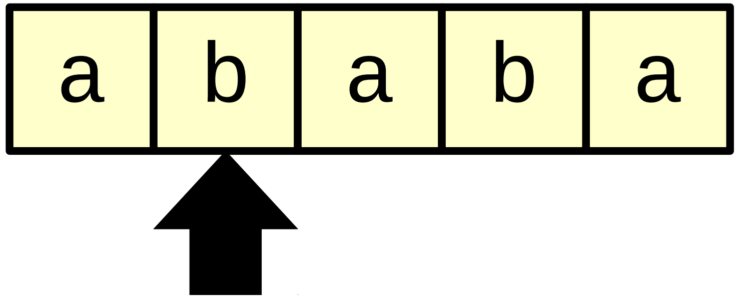


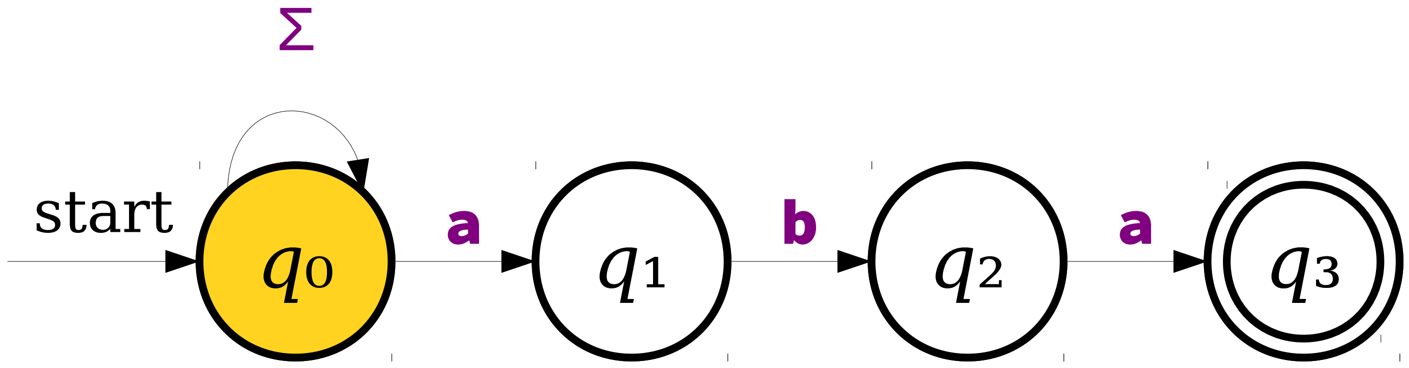
Σ
start
q0
a
q1
b
q2
a
q3

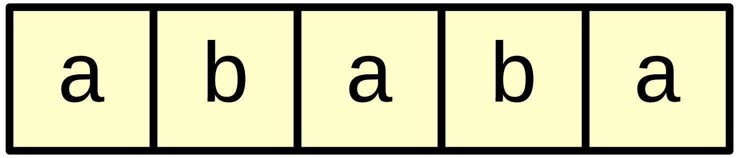


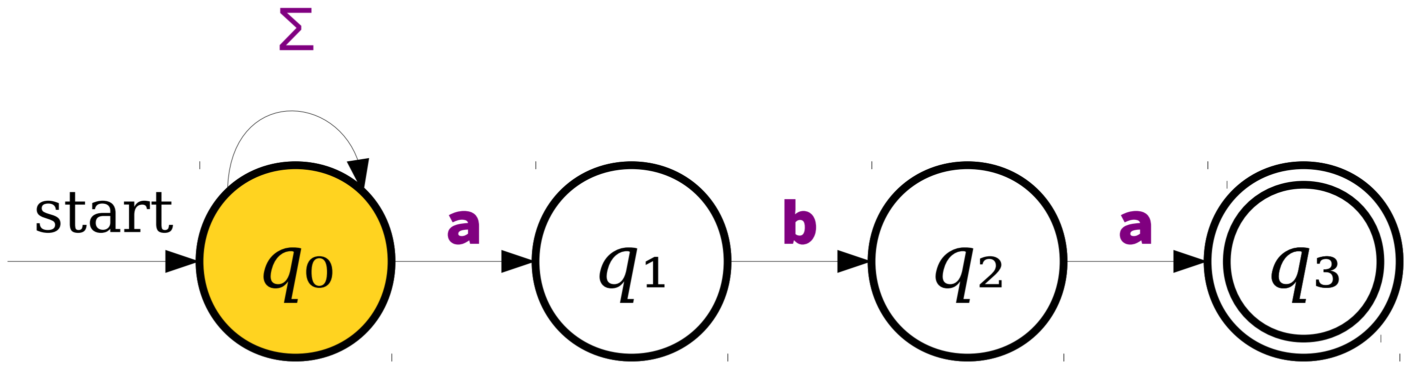
a
b
a
b
a

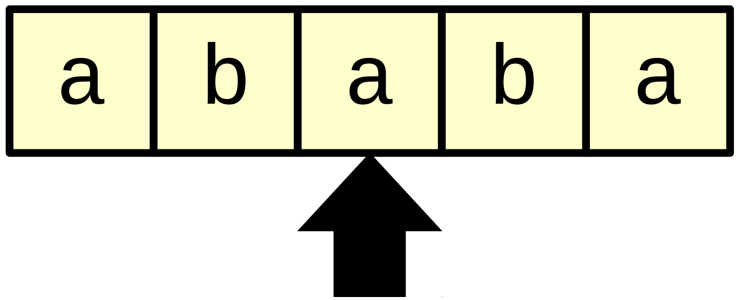

Σ
start
q0
a
q1
b
q2
a
q3

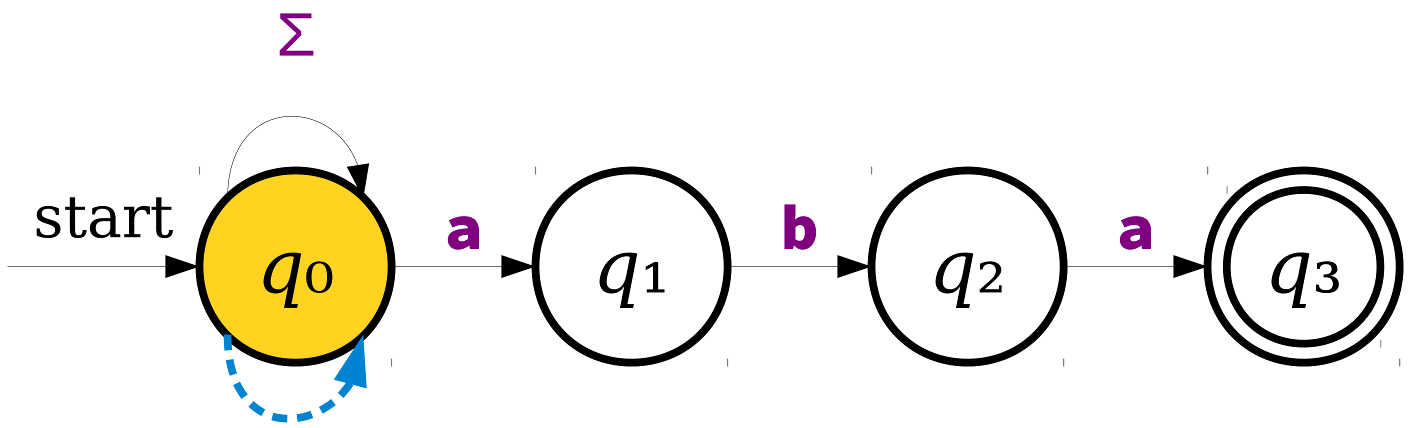

a
b
a
b
a

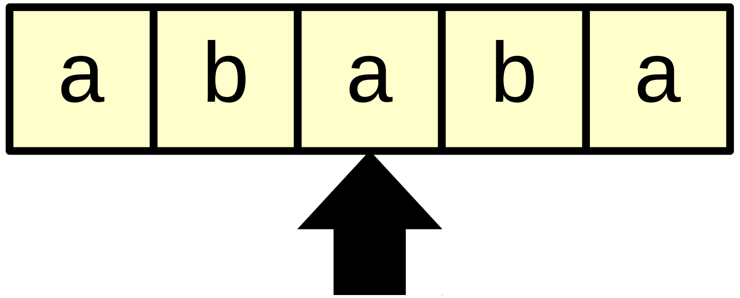

Σ
start
q0
a
q1
b
q2
a
q3

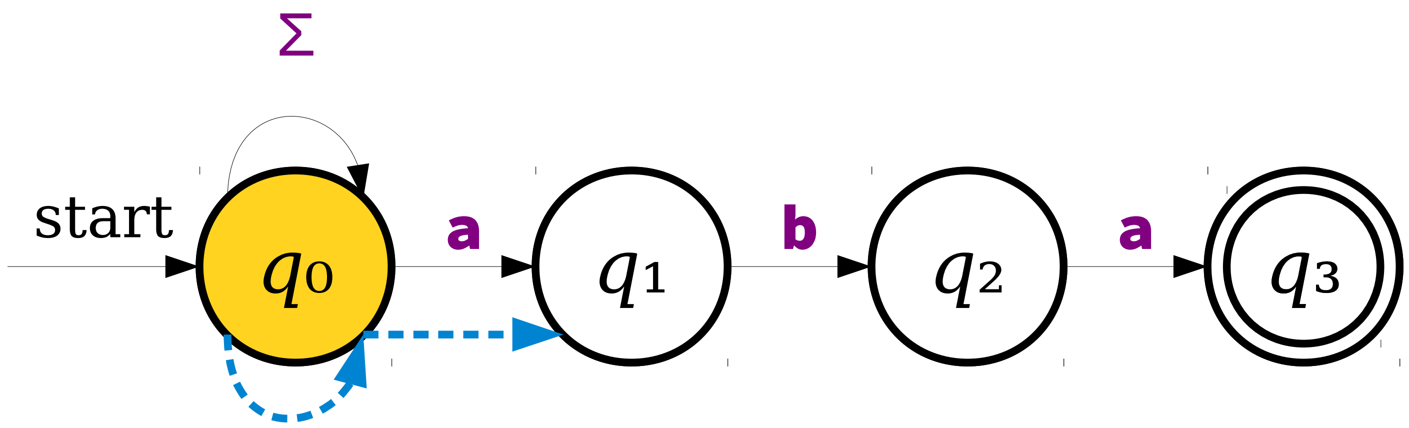

a b a b a

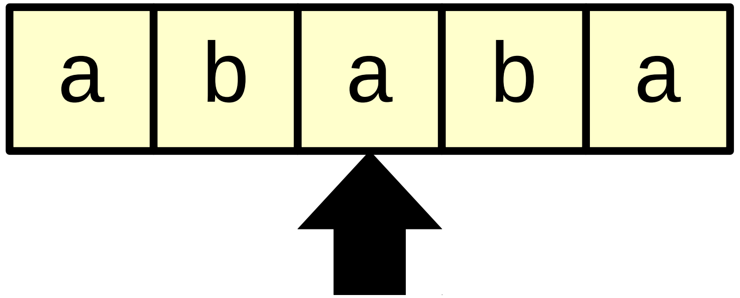

Σ
start
q0
a
q1
b
q2
a
q3

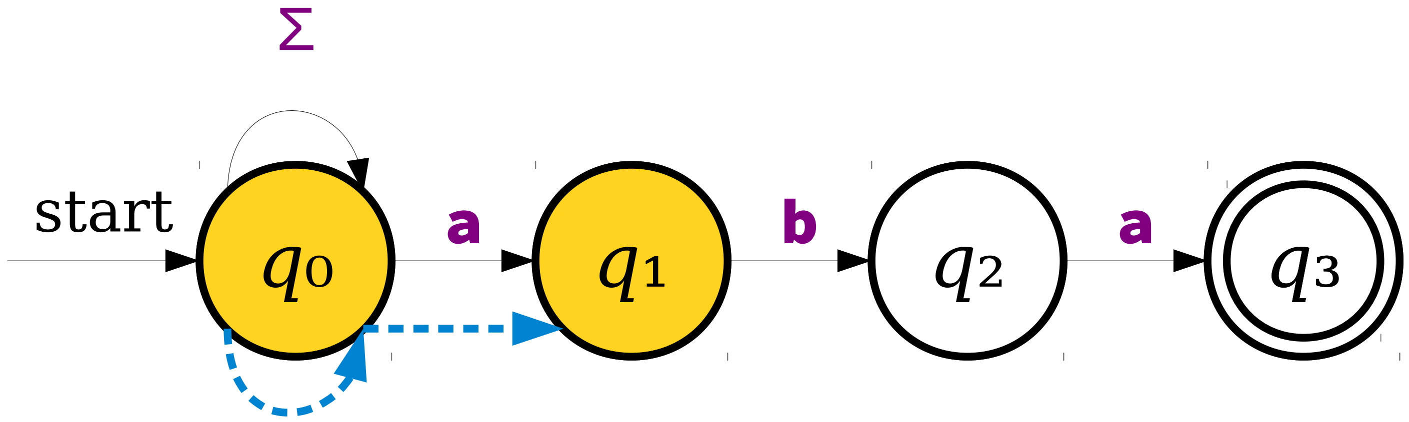

a
b
a
b
a

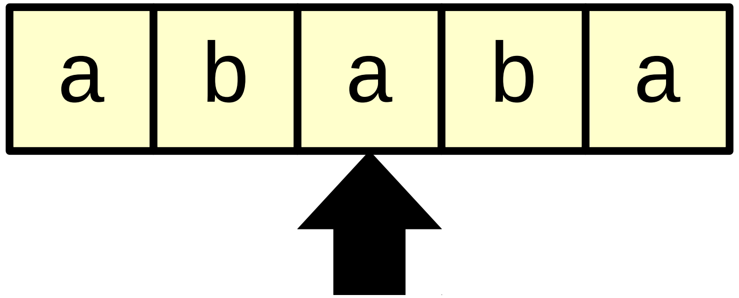

Σ
start
q0
a
q1
b
q2
a
q3

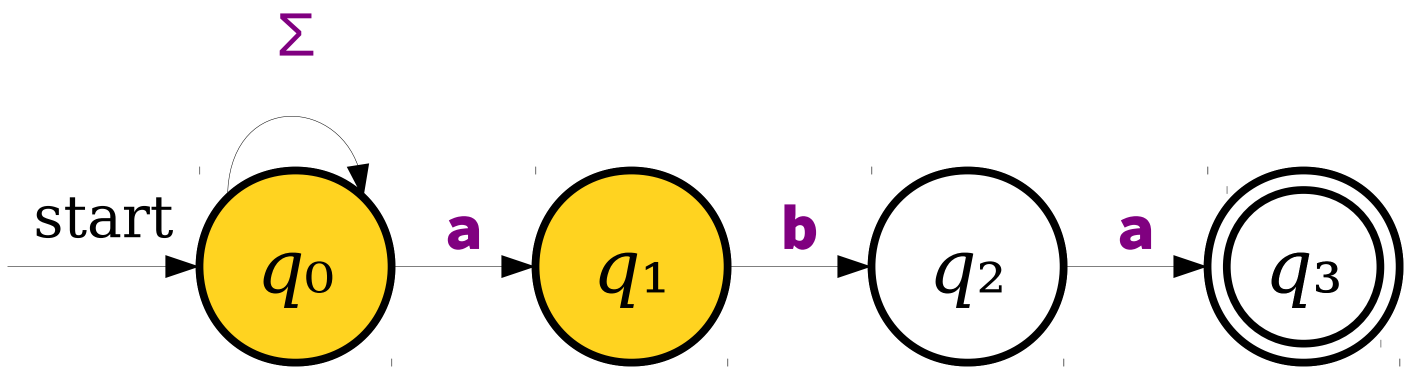

a b a b a

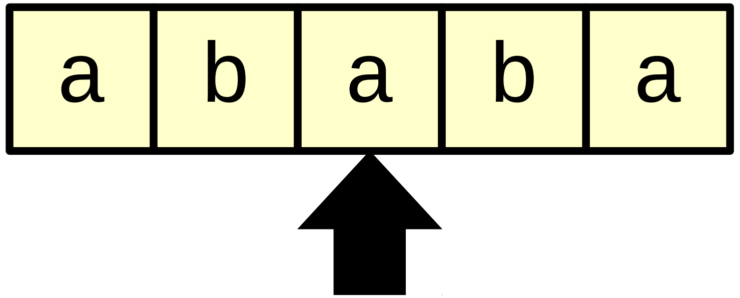

Σ
start
q_0
a
q_1
b
q_2
a
q_3

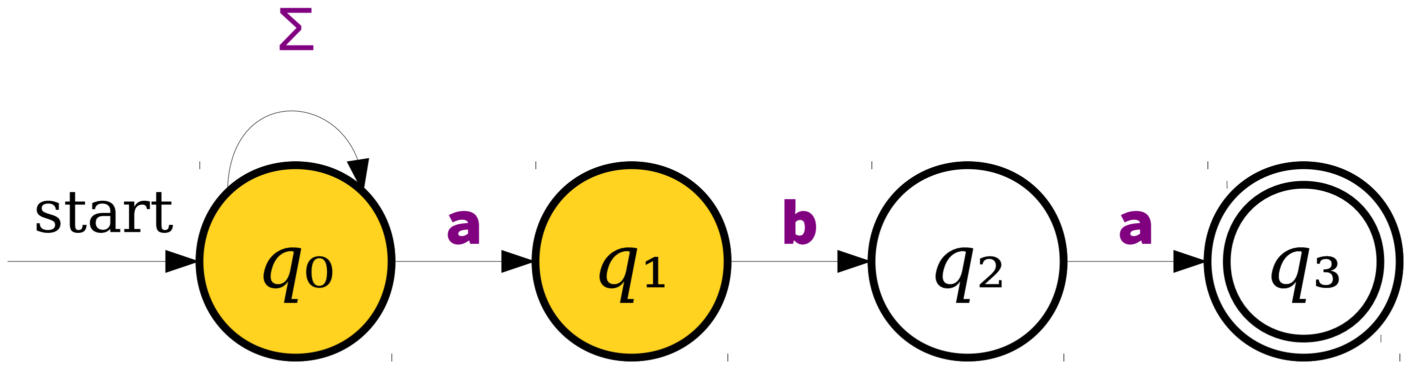

a
b
a
b
a

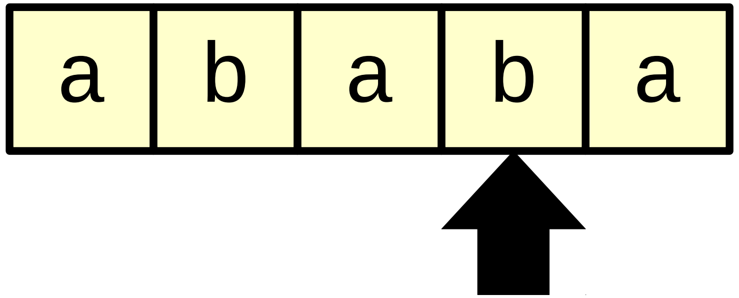

Σ
start
a
b
a
q0
q1
q2
q3

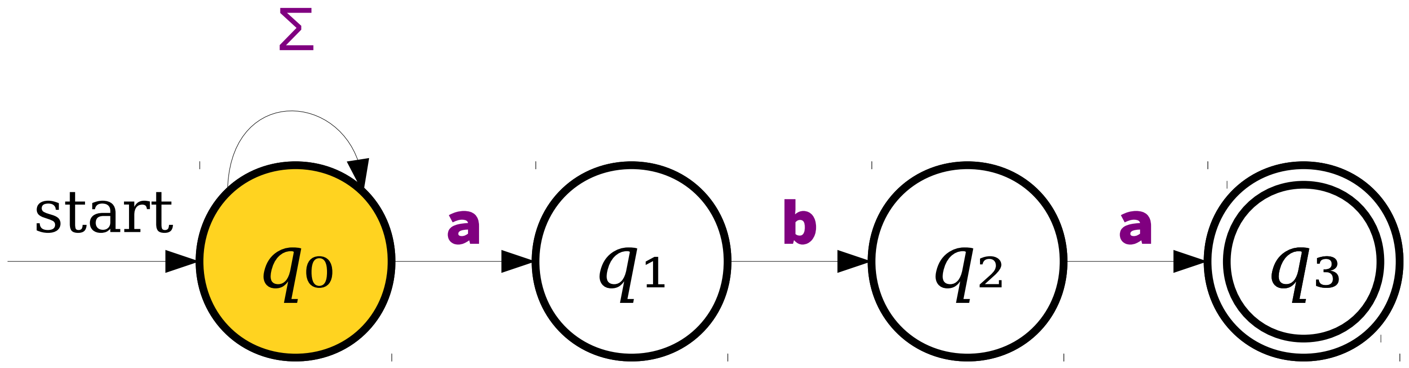

a
b
a
b
a

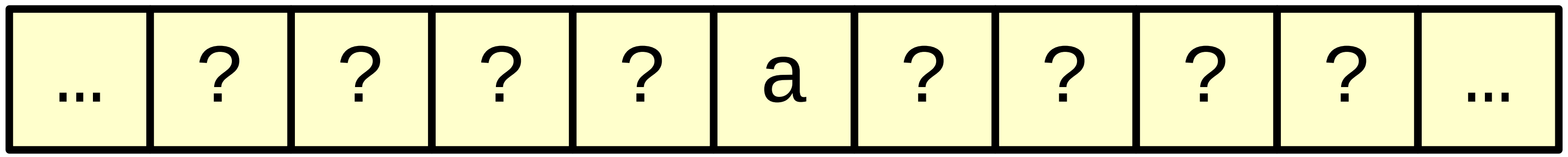

Σ
start
q_0
a
q_1
b
q_2
a
q_3

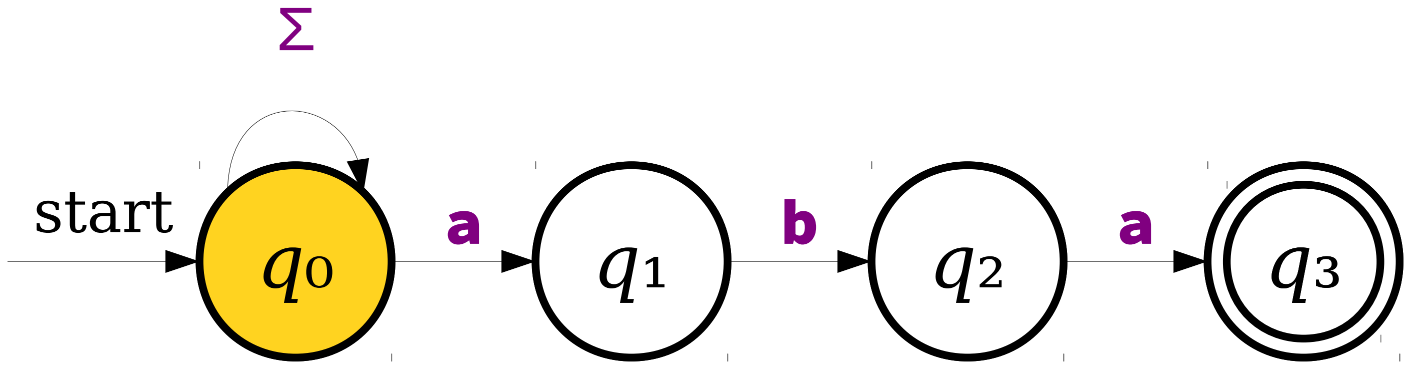

a b a b a

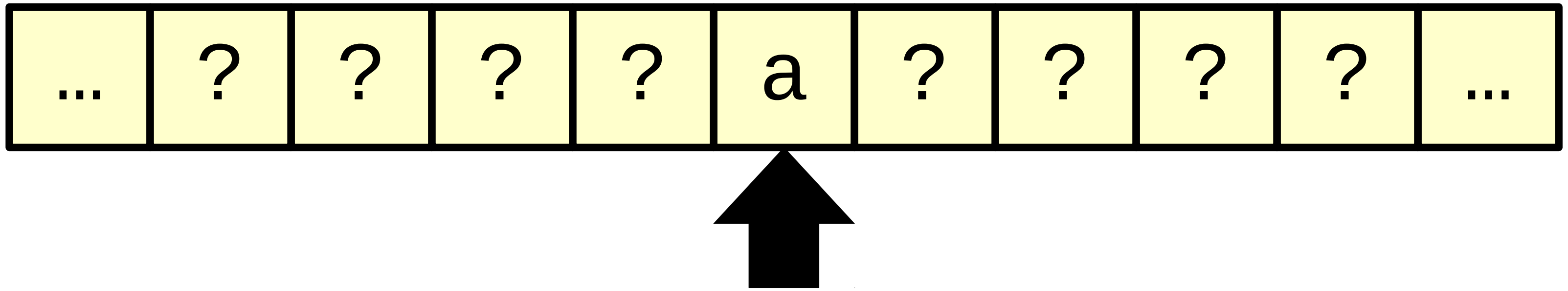

Σ
start
q_0
a
q_1
b
q_2
a
q_3

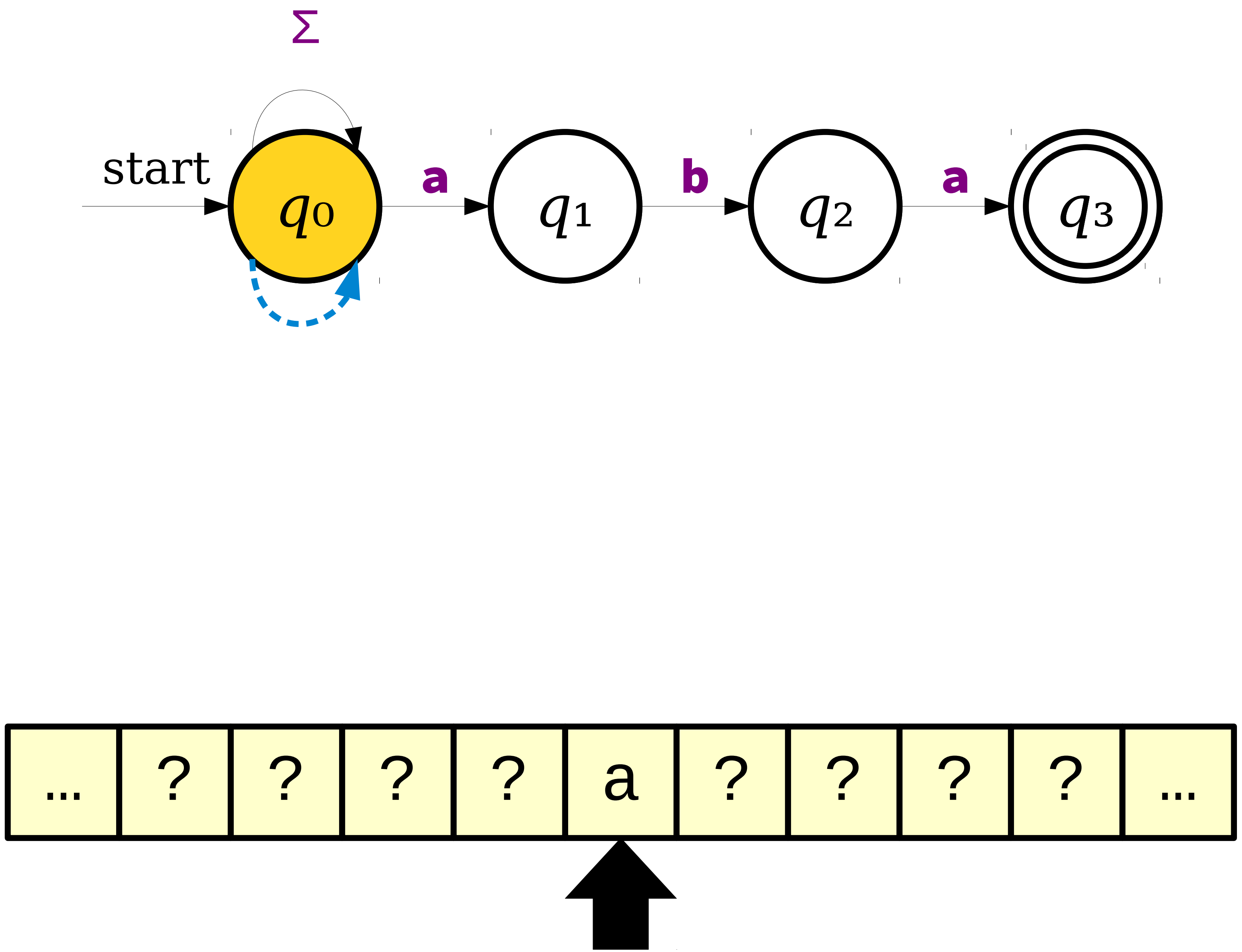

a
b
a
b
a

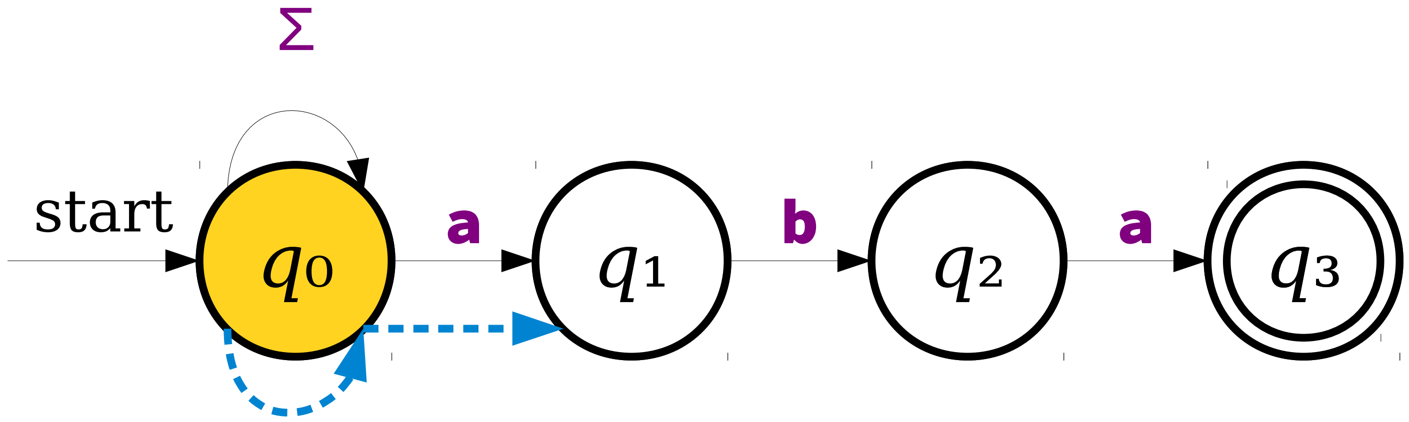

Σ
start
q0
a
q1
b
q2
a
q3


...
?
?
?
?
a
?
?
?
?
...

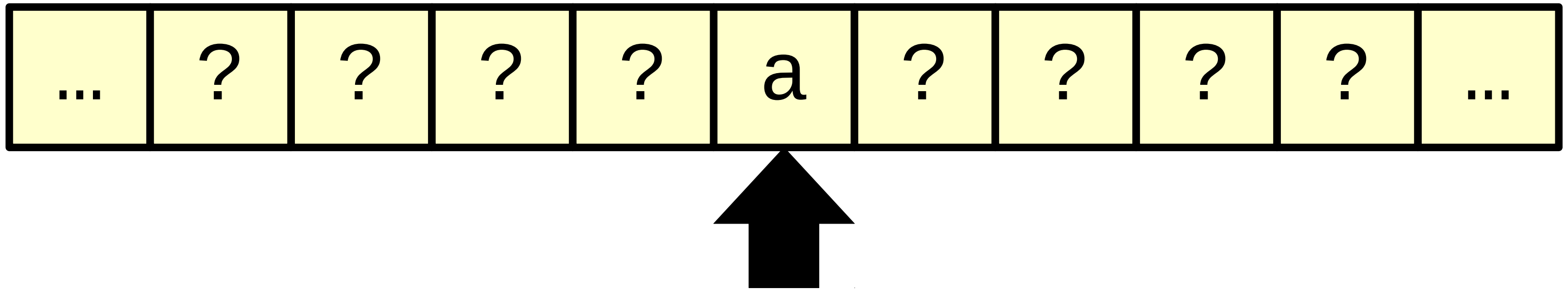

Σ
start
q0
a
q1
b
q2
a
q3


...
?
?
?
?
a
?
?
?
?
...

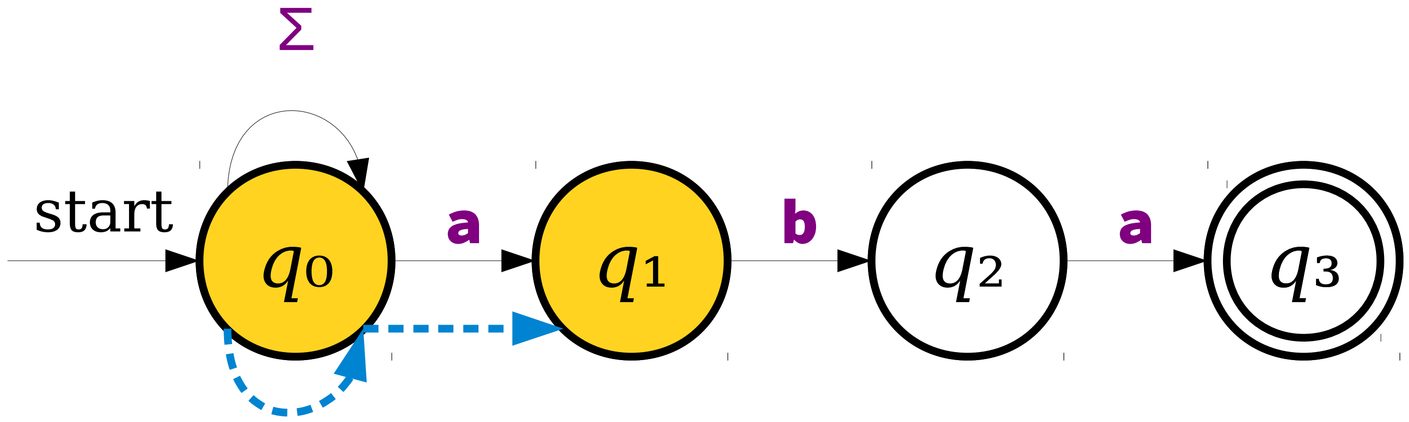

Σ
start
q_0
a
q_1
b
q_2
a
q_3
… ? ? ? ? a ? ? ? ? …

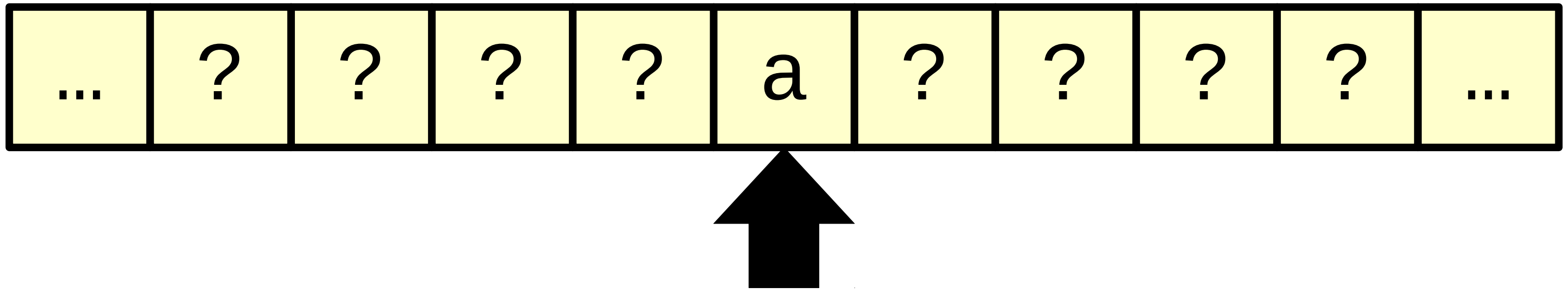


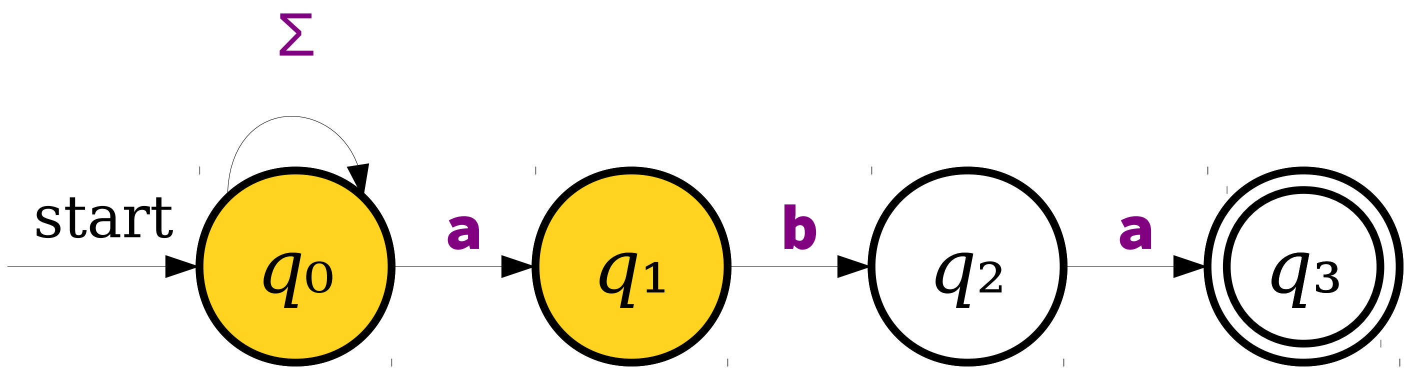
Σ
start
q0
a
q1
b
q2
a
q3

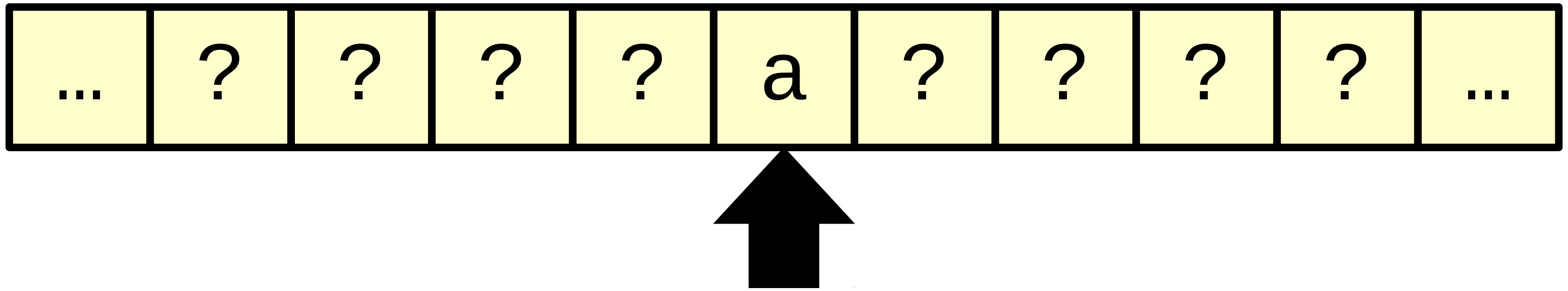


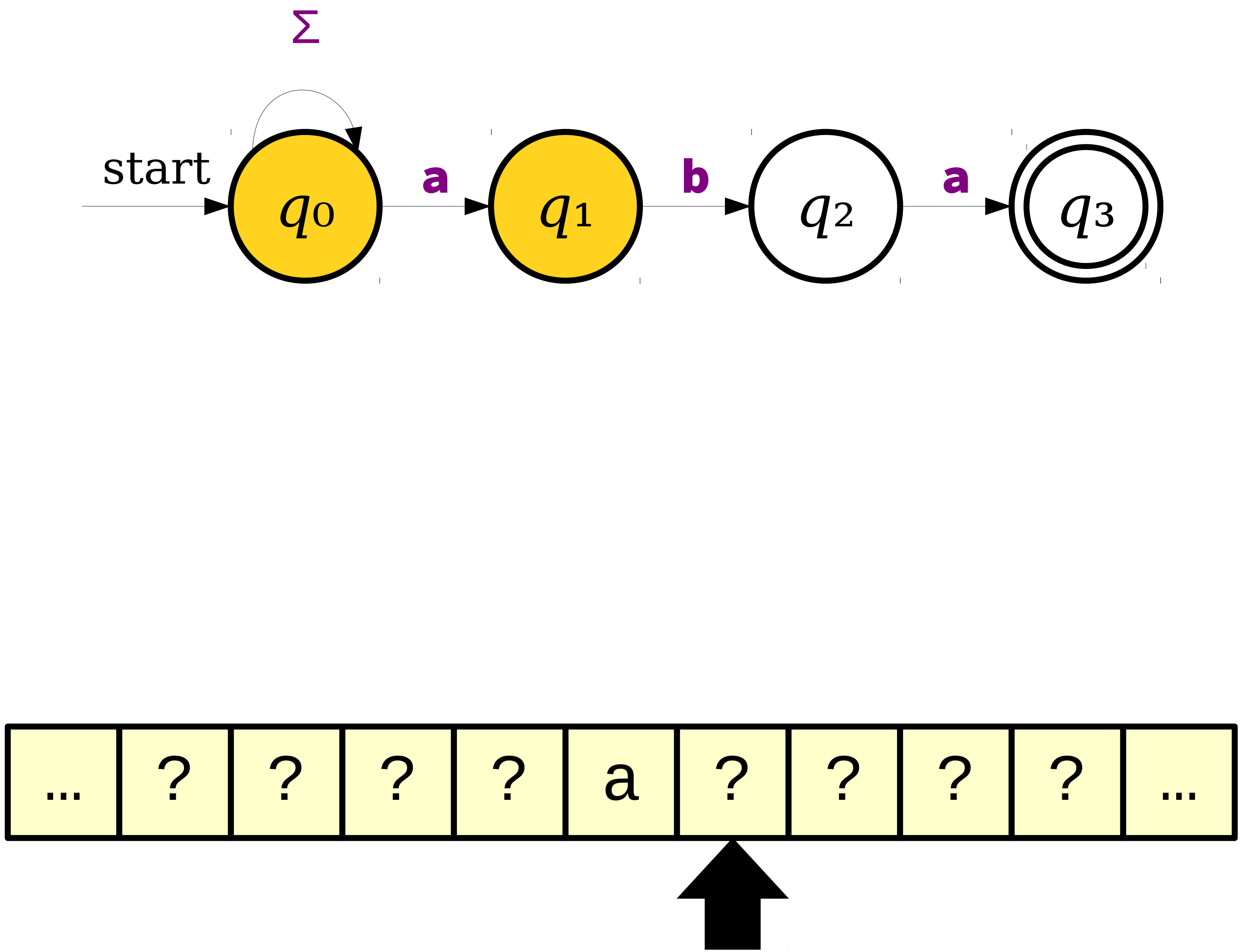
...
?
?
?
?
a
?
?
?
?
...

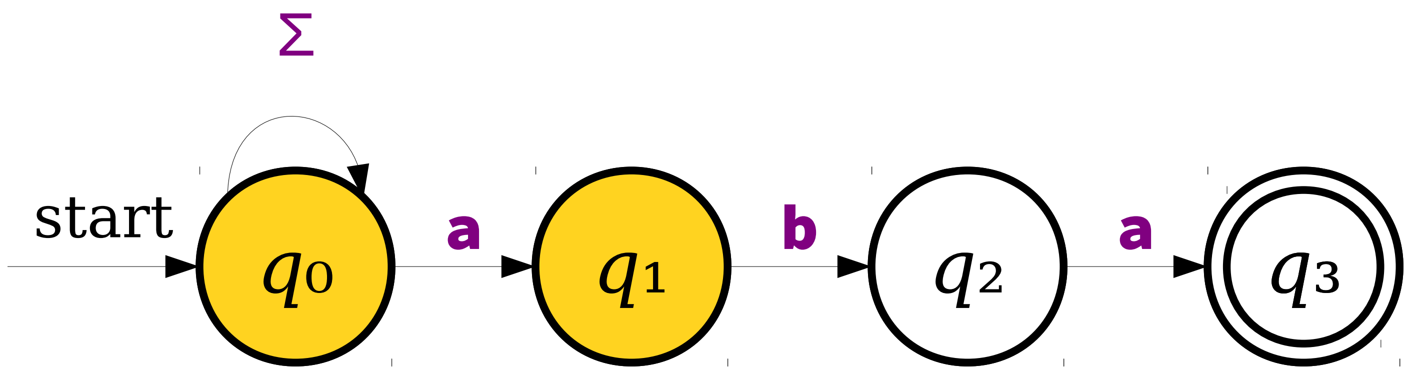

Σ
start
q0
a
q1
b
q2
a
q3


... ? ? ? ? a ? ? ? ? ...


Σ
start
a
b
a
q_0
q_1
q_2
q_3

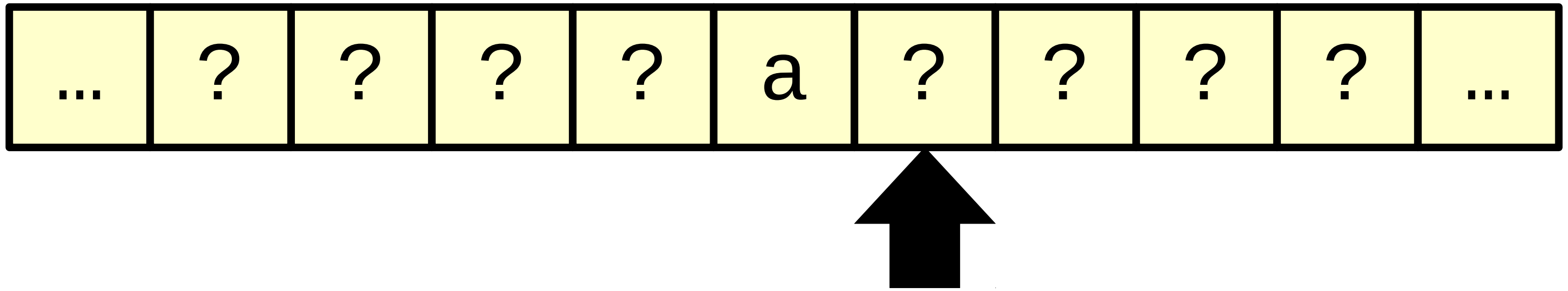

...
?
?
?
?
a
?
?
?
?
...


Σ
start
a
b
a
q0
q1
q2
q3
... ? ? ? ? a ? ? ? ? ...



Key insight:

NFA works like a DFA, but “which state am I in right now?” and “which states will I be in next given where I am right now and which character I’m reading?” are questions answered by a set of states, not a single state.



DFAs

- A DFA consists of:
  - A set of states
  - Exactly one element of the set of states designated as start state
    - (as a consequence, the set of states must be nonempty)
  - A subset of the states designated as accepting states
  - An alphabet Σ
  - A transition function that maps (state, character) ordered pairs to states
    - (i.e., for each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ)

Remember: For DFAs, we said we could formally define the transitions (arrows) as a function $f : S \times \Sigma \rightarrow S$. (in other words, an order pair of current state and input character (q, c) maps to a destination state q'.

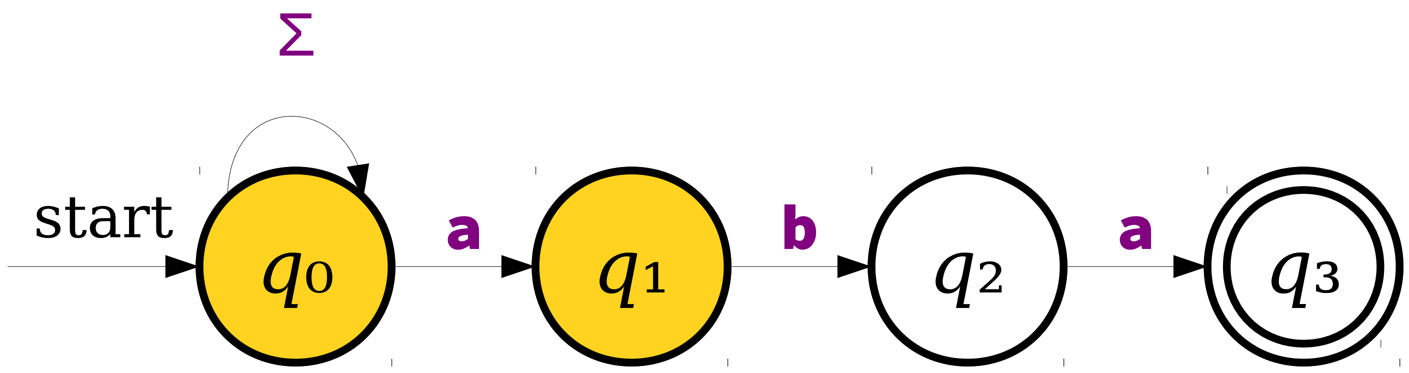
Math notation note: In $f : S \times \Sigma \rightarrow S$, the x (cross product) means "make ordered pairs out of these two sets." So if $S = \{a, b\}$ and $T = \{1, 2\}$, $S \times T = \ldots$?? Go to PollEv.com/cs103spr25

DFAs

- A DFA consists of:
  - A set of states
  - Exactly one element of the set of states designated as start state
    - (as a consequence, the set of states must be nonempty)
  - A subset of the states designated as accepting states
  - An alphabet Σ
  - A transition function that maps (state, character) ordered pairs to states
    - (i.e., for each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ)

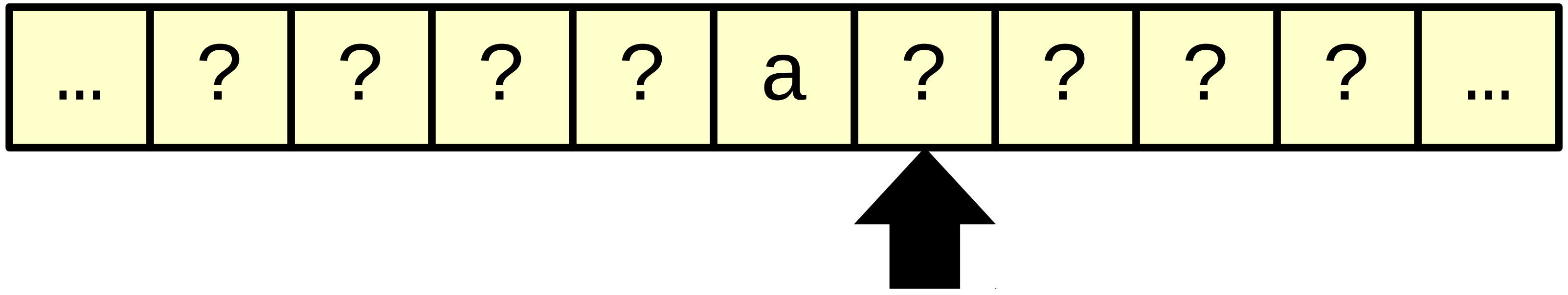
Remember: For DFAs, we said we could formally define the transitions (arrows) as a function $f : S \times \Sigma \rightarrow S$. (in other words, an order pair of current state and input character (q, c) maps to a destination state q'.

Math notation note: In $f : S \times \Sigma \rightarrow S$, the x (cross product) means "make ordered pairs out of these two sets." So if S = {a, b} and T = {1, 2}, $S \times T$ = {(a,1), (a,2), (b,1), (b,2)}



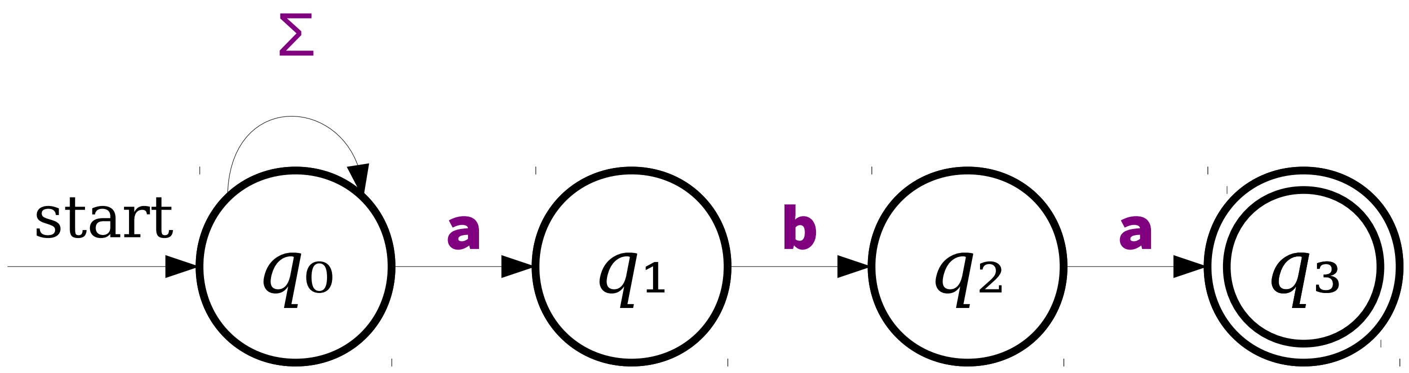
Key insight:

NFA works like a DFA, but “which state am I in right now?” and “which states will I be in next given where I am right now and which character I’m reading?” are questions answered by a set of states, not a single state.

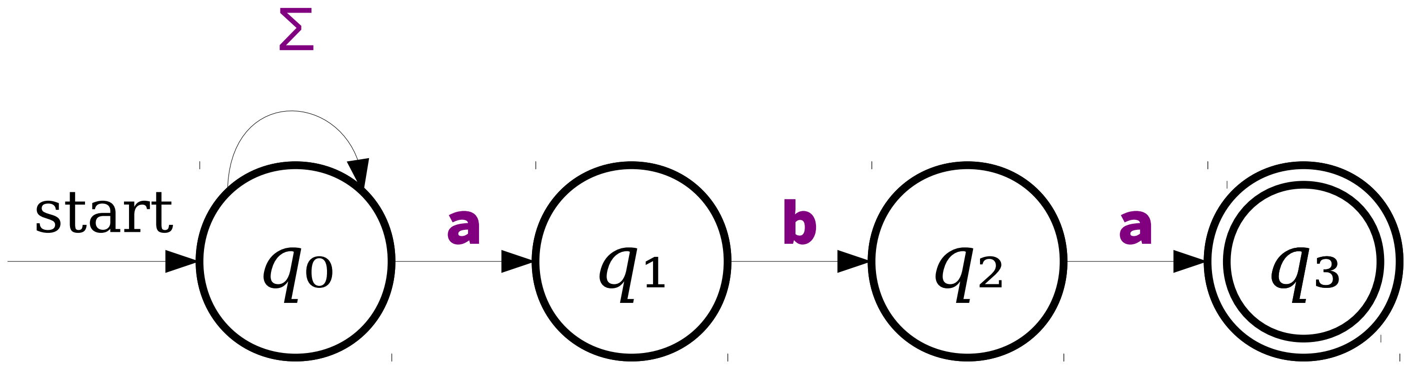


Remember: For DFAs, we said we could formally define the transitions (arrows) as a function $f : S \times \Sigma \rightarrow S$. (in other words, an order pair of current state and input character (q, c) maps to a destination state q'.

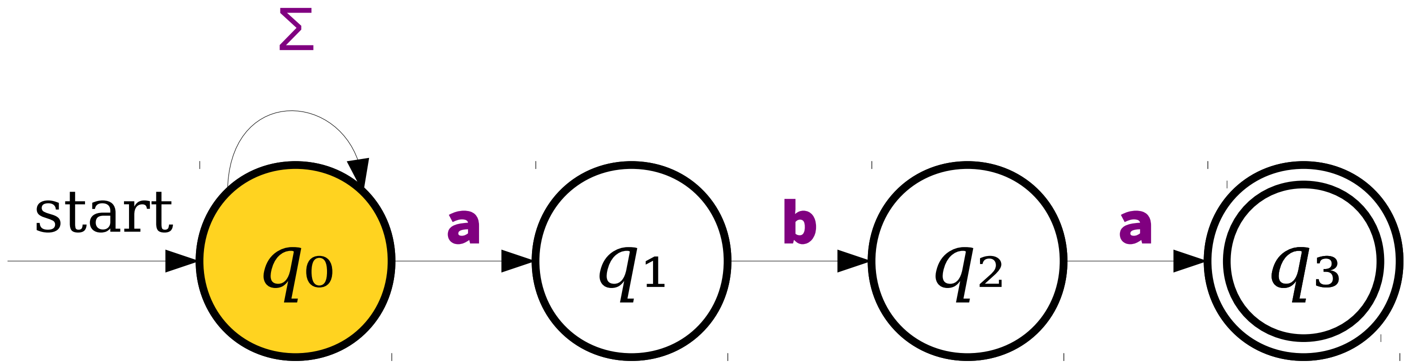
Question:
How would we formally define the transitions for an NFA as a function, given this insight?
Go to pollev.com/cs103spr25



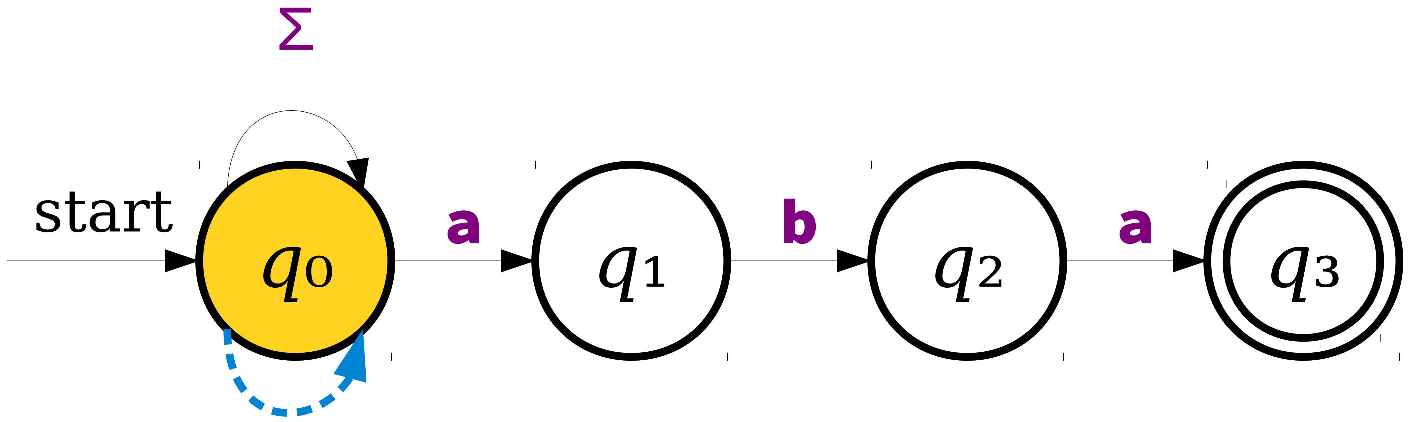
	a
$\{q_0\}$	$\{q_0, q_1\}$



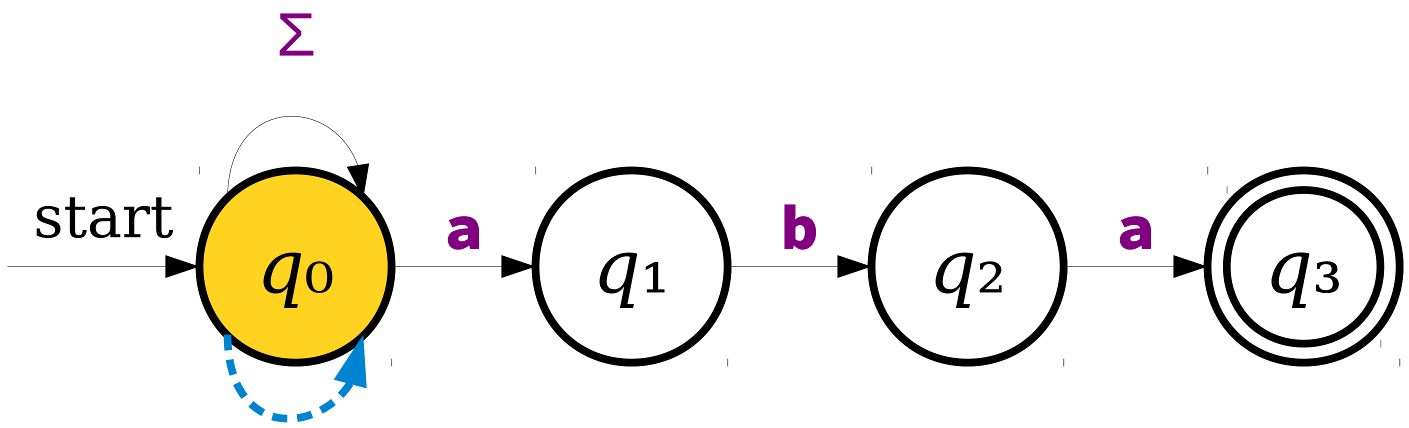
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	



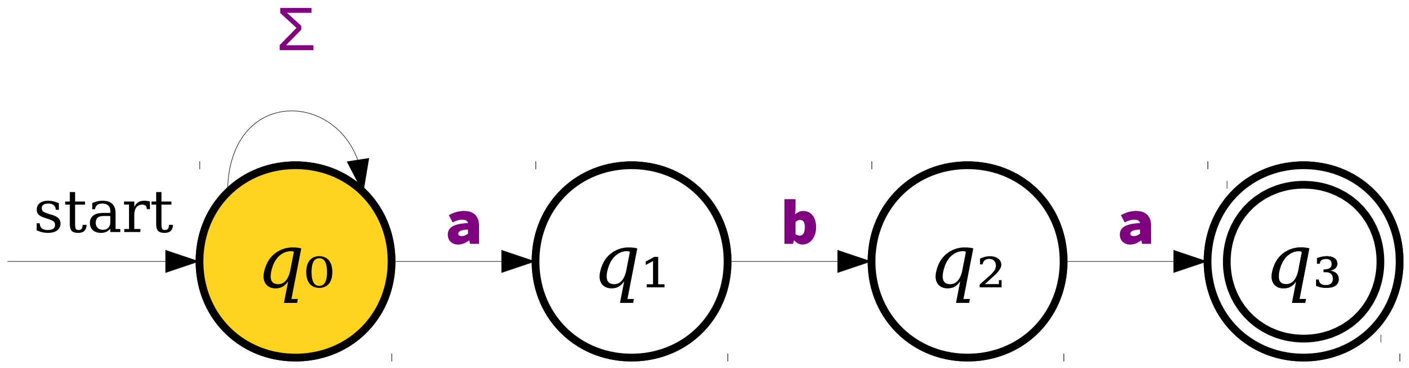
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	



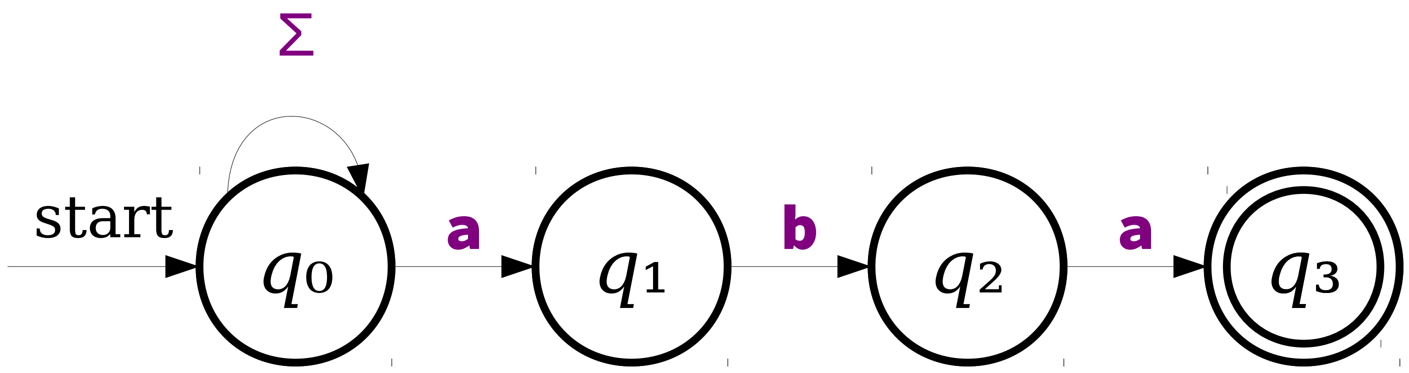
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	



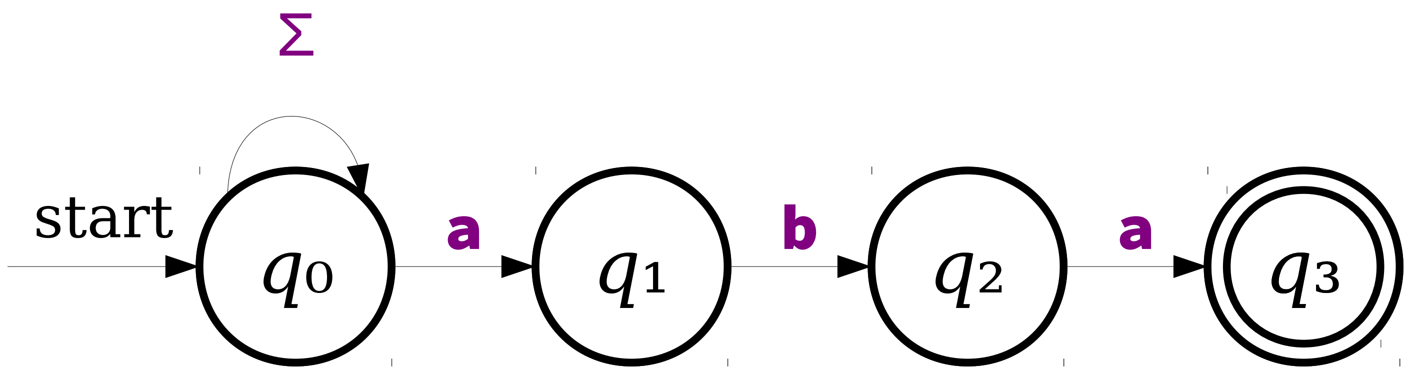
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



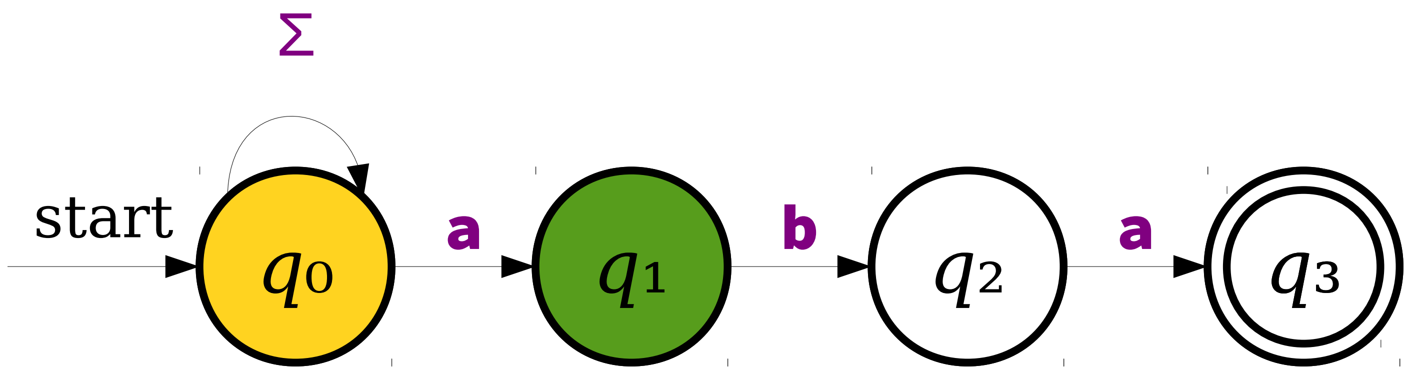
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



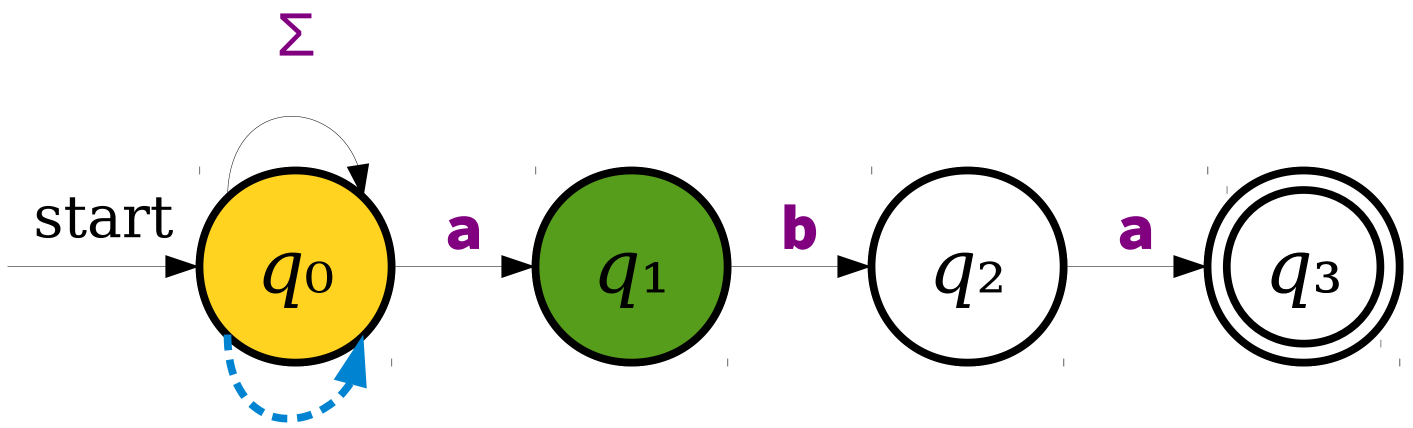
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



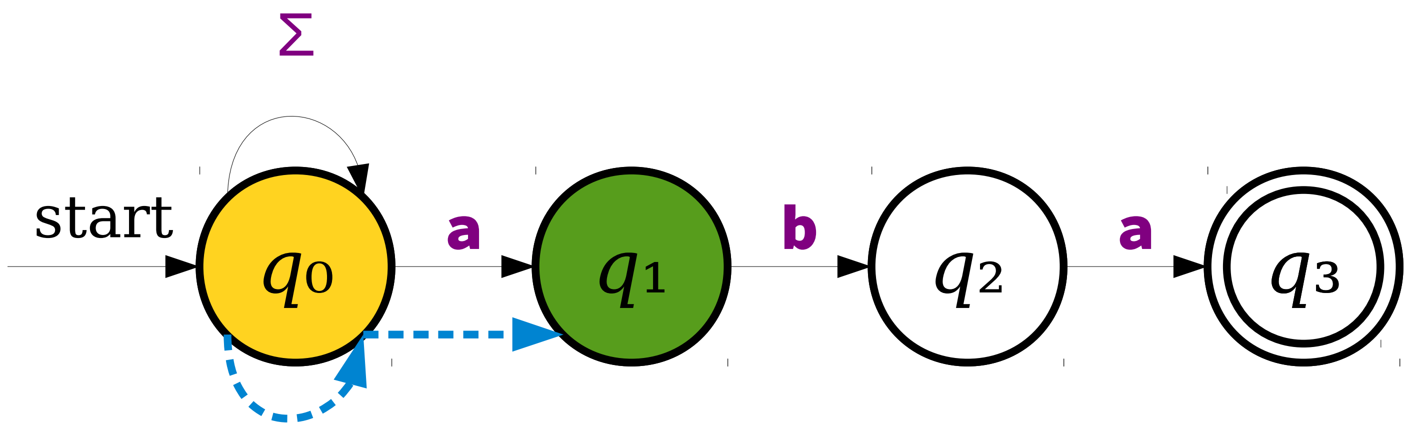
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



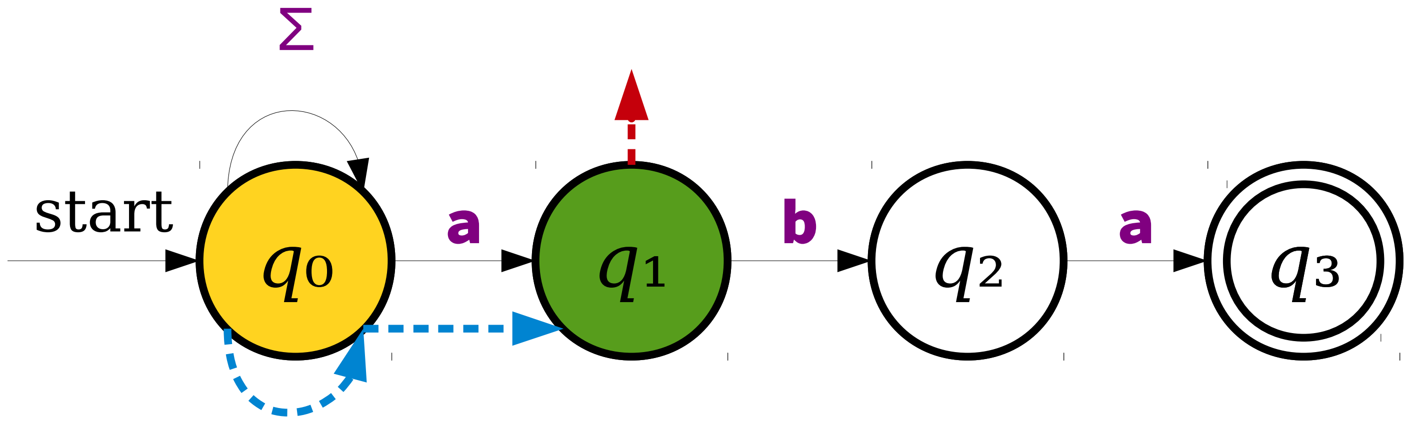
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



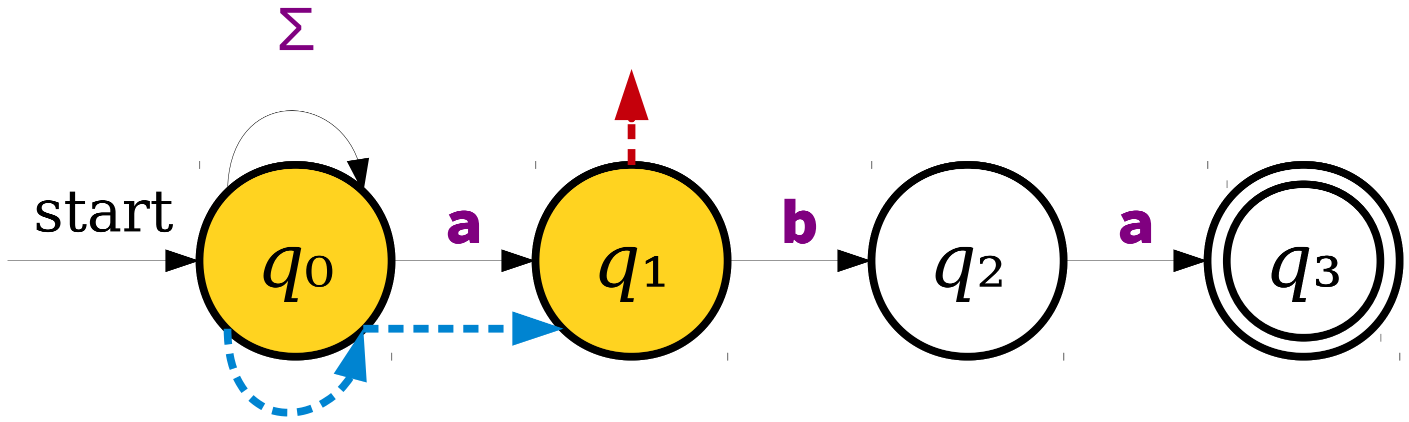
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



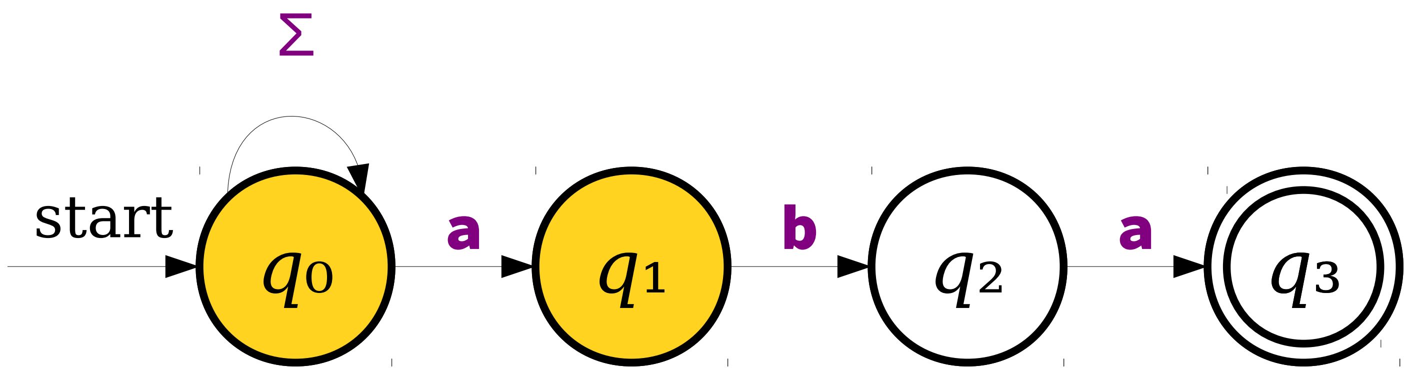
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



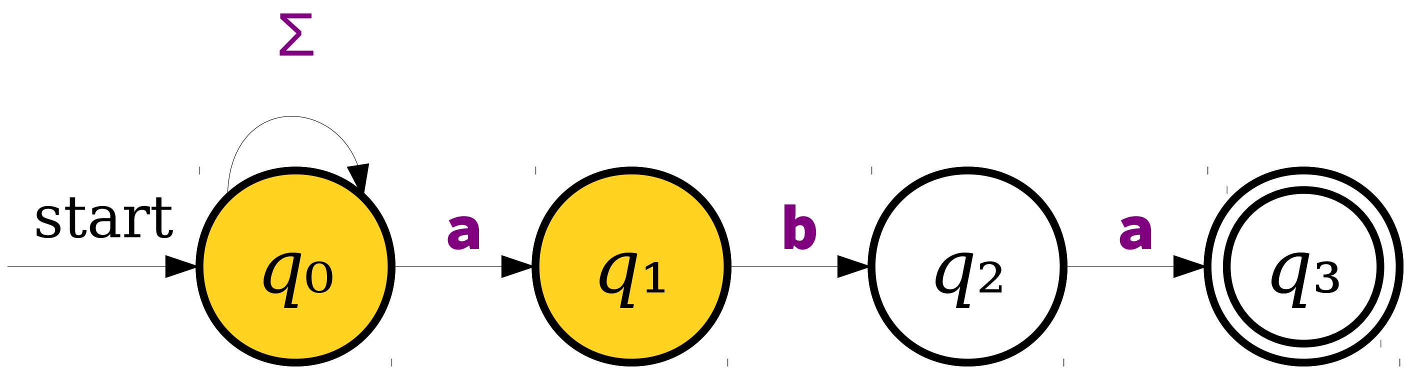
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



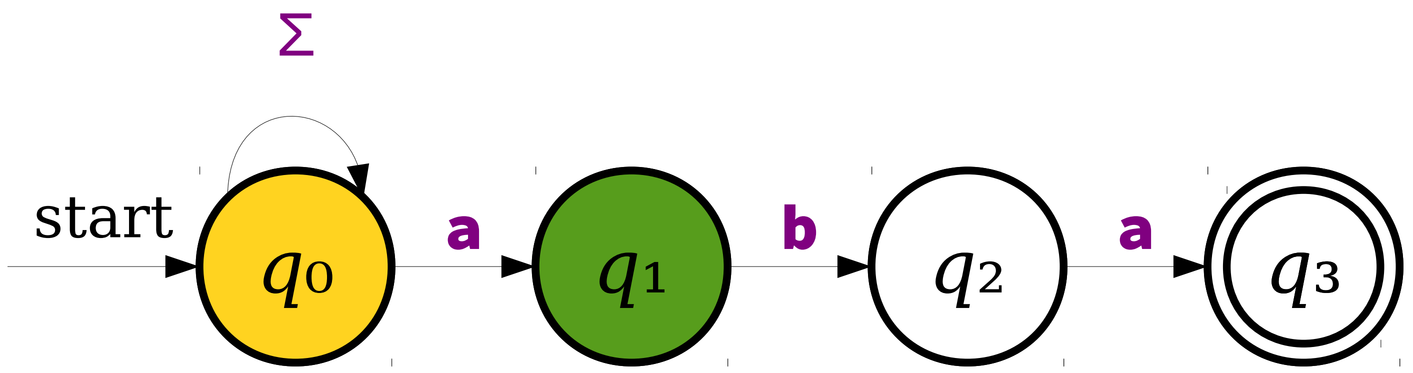
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



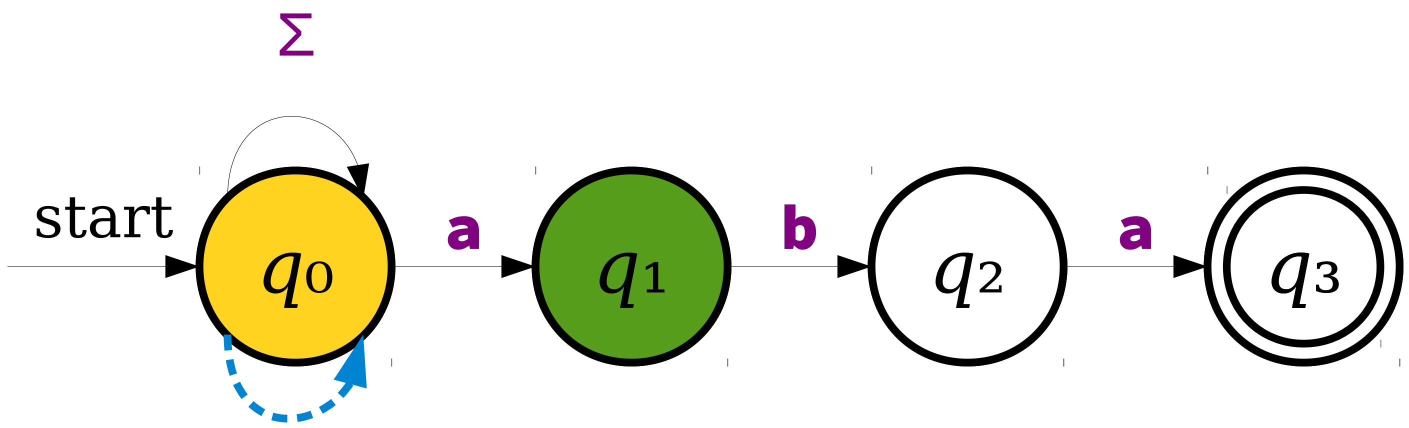
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



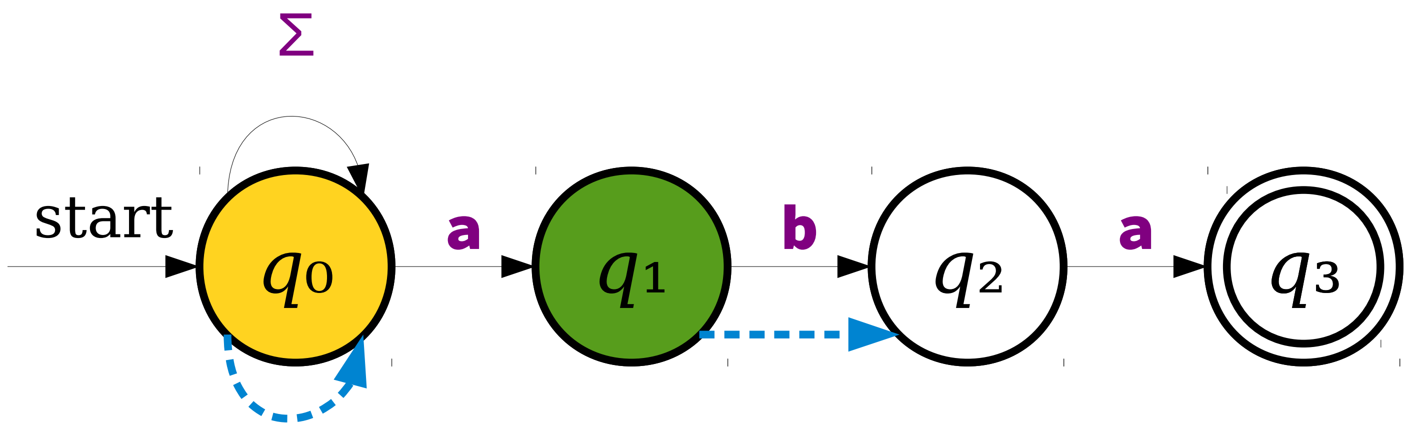
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



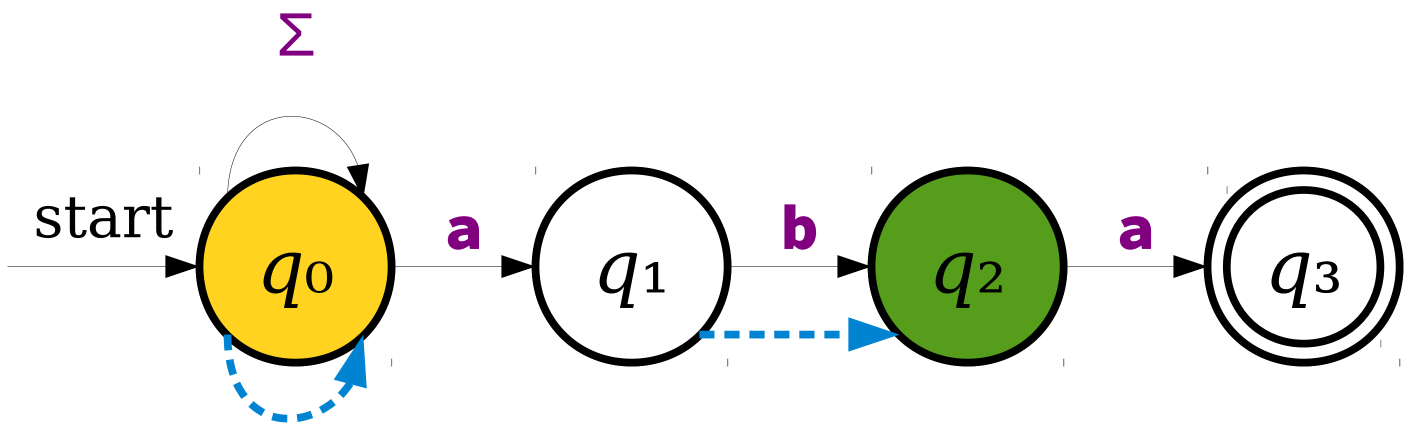
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



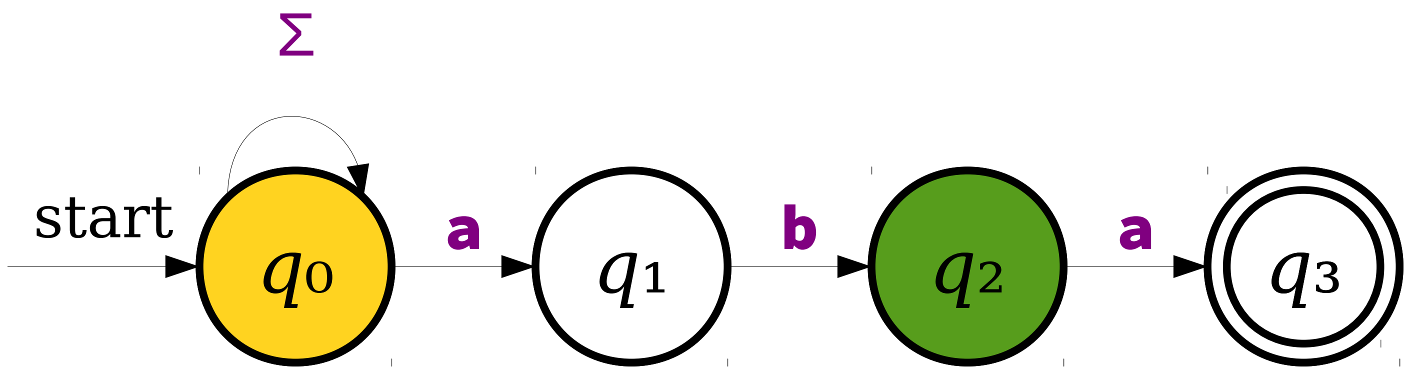
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



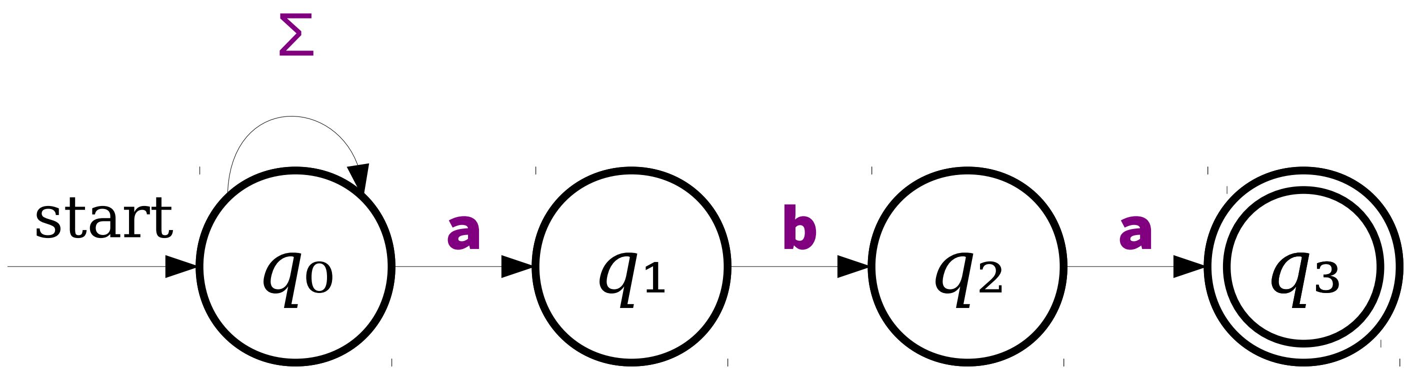
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



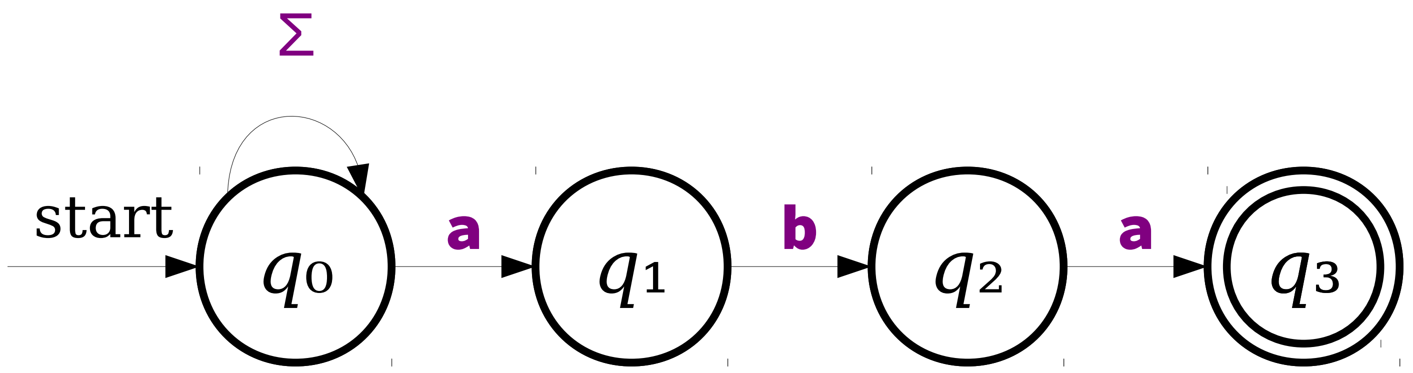
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



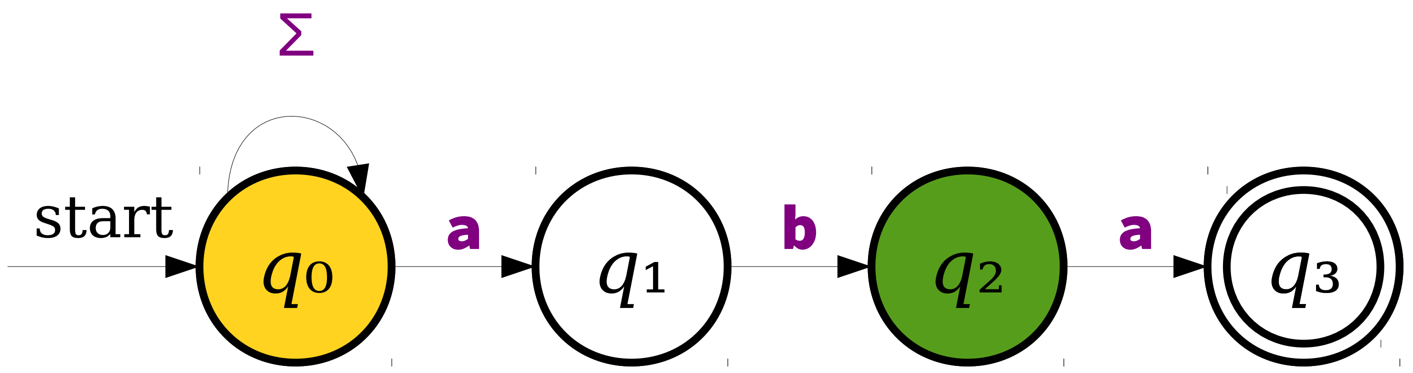
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



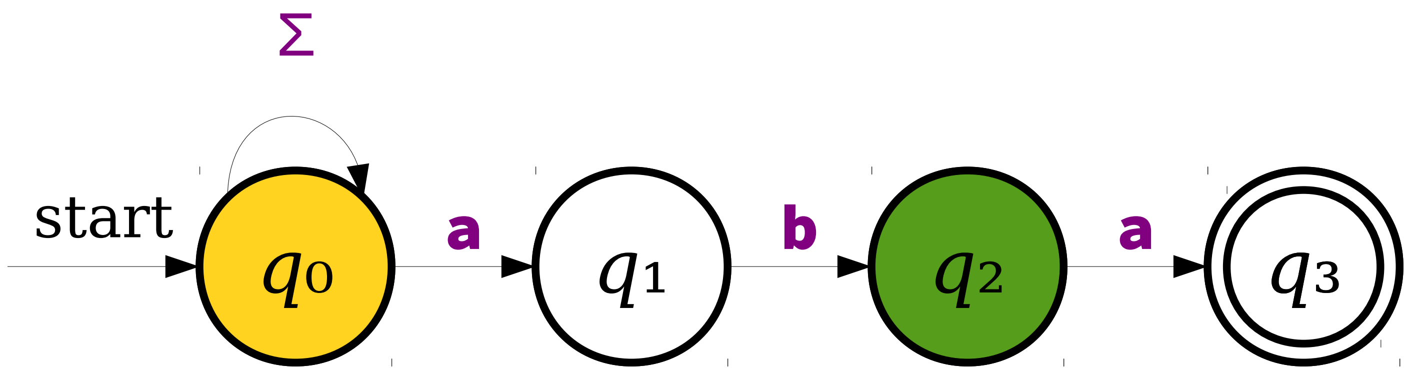
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		

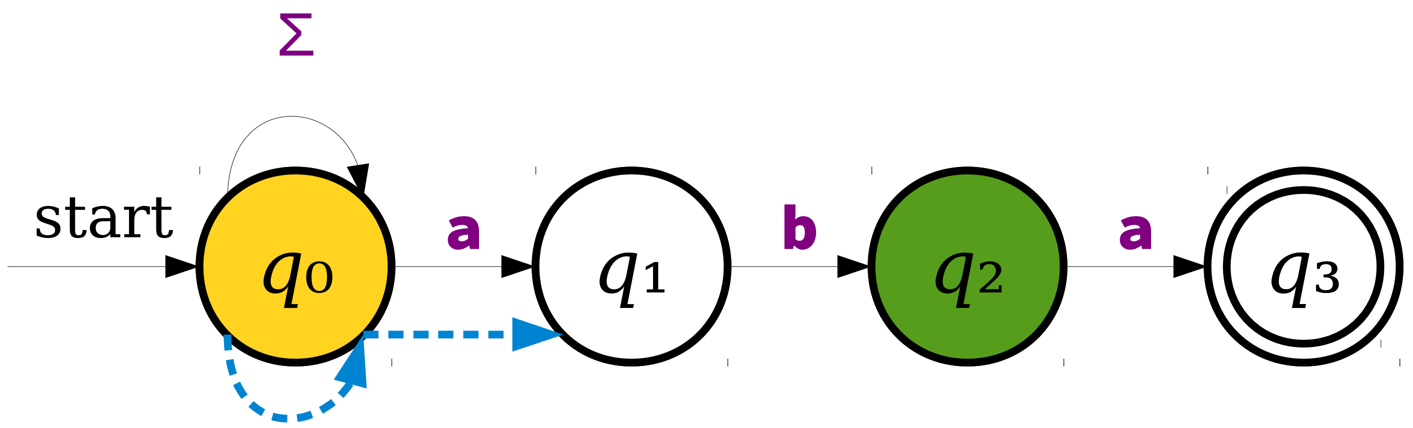


	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		

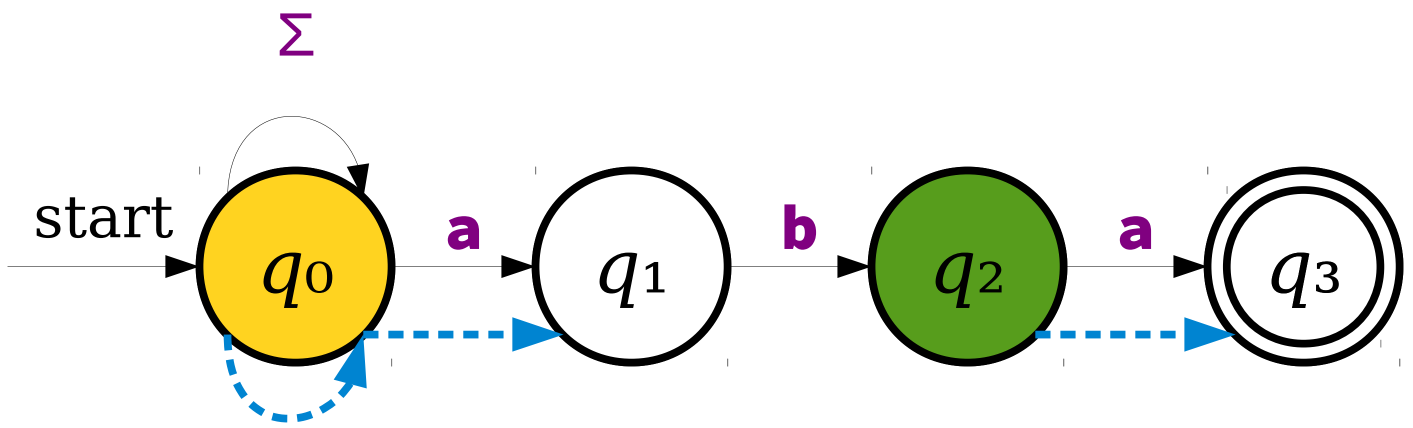


	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		

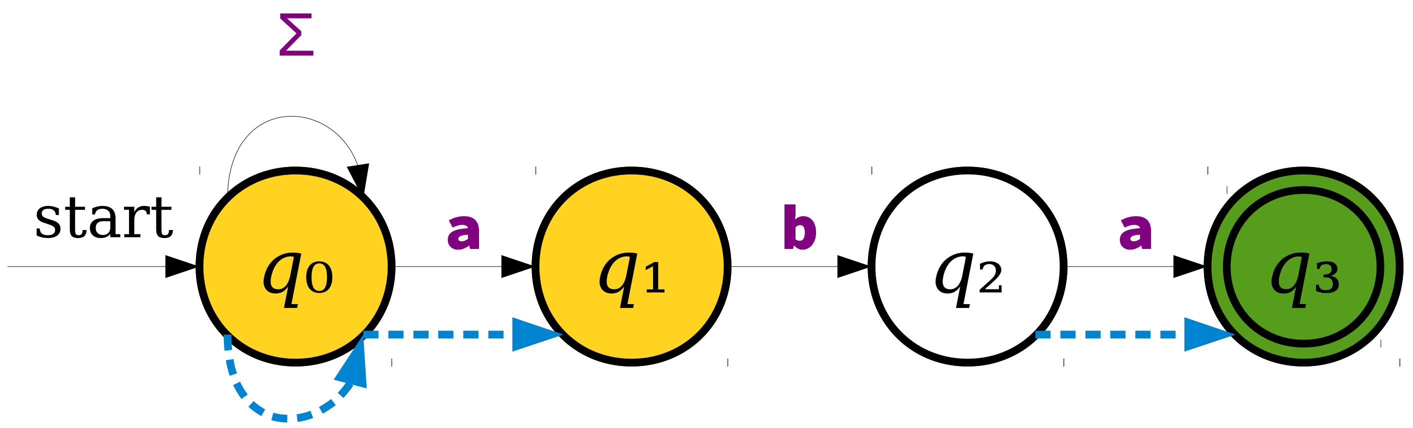
Quick check:
What are the contents of the next row?
Answer like
“a = {..}, b = {…}”
all in one response.
Go to
PollEv.com/cs103spr25



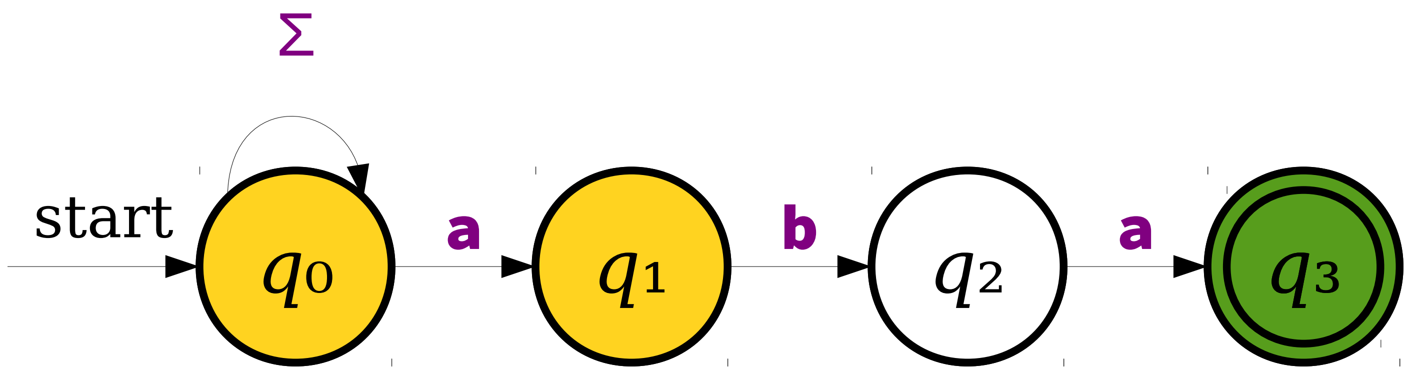
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



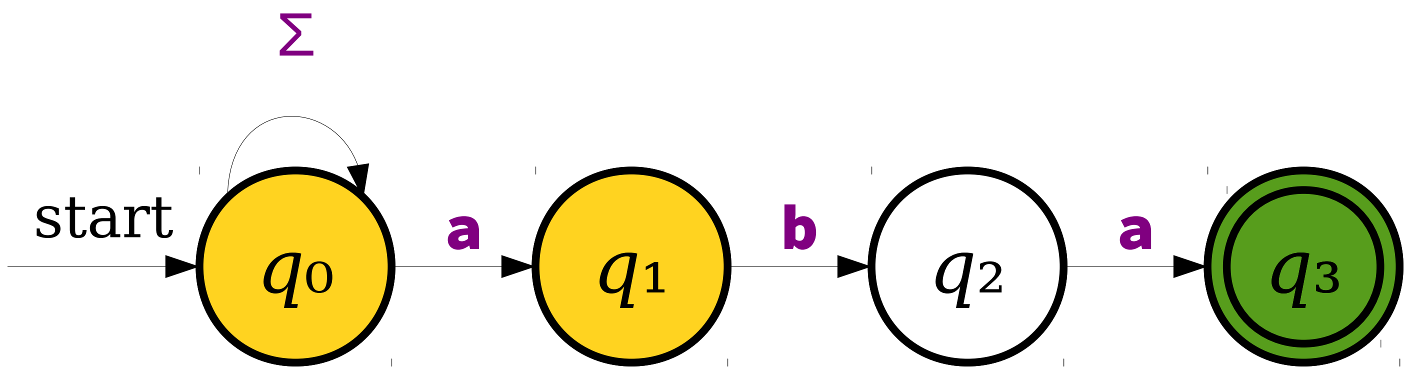
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



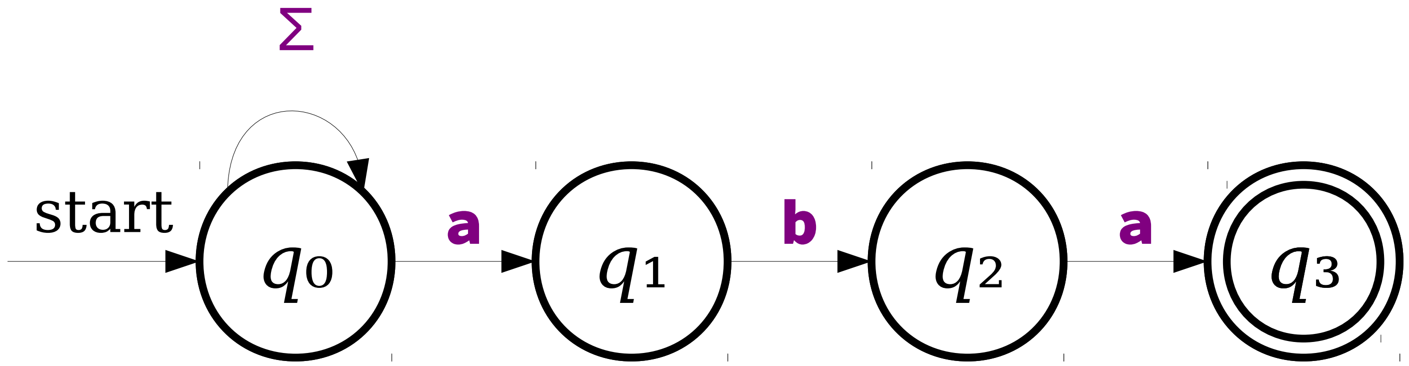
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



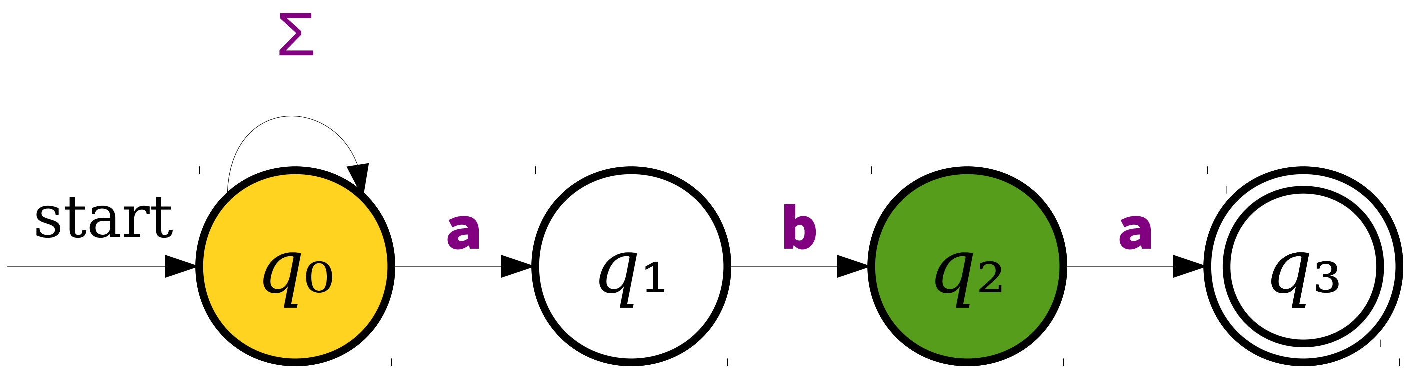
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



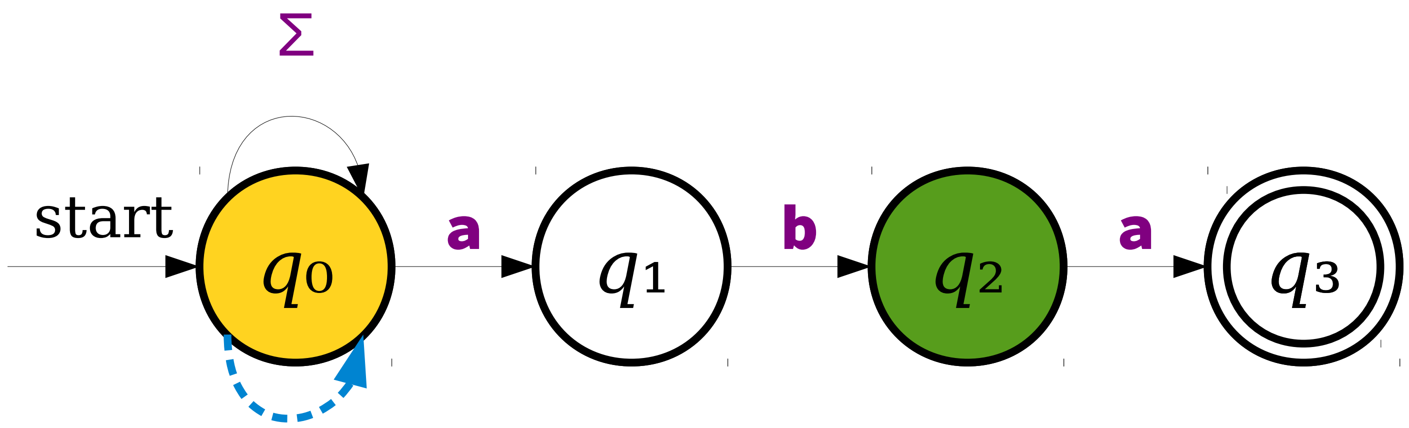
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



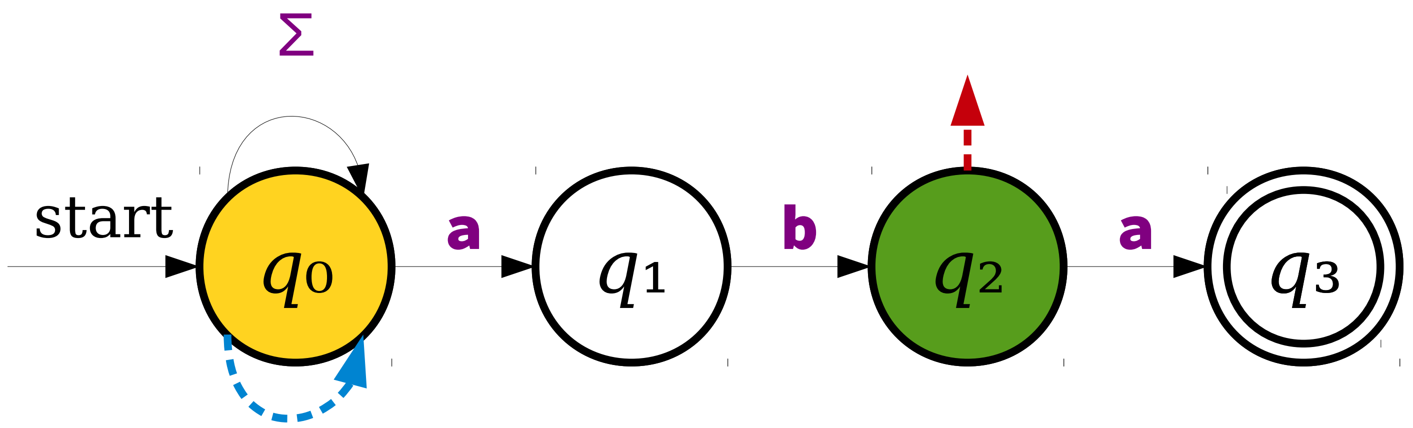
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



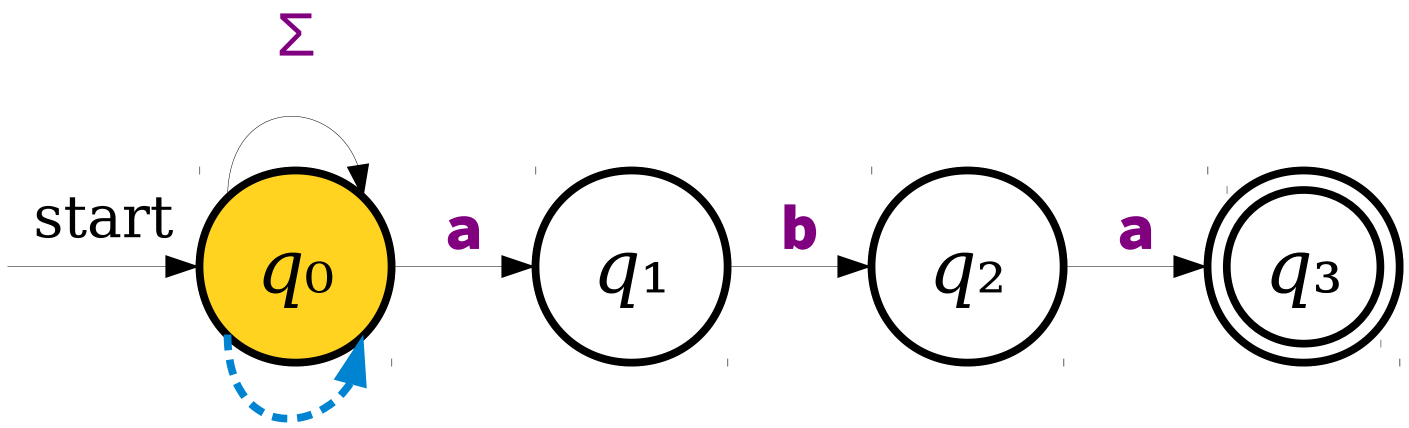
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



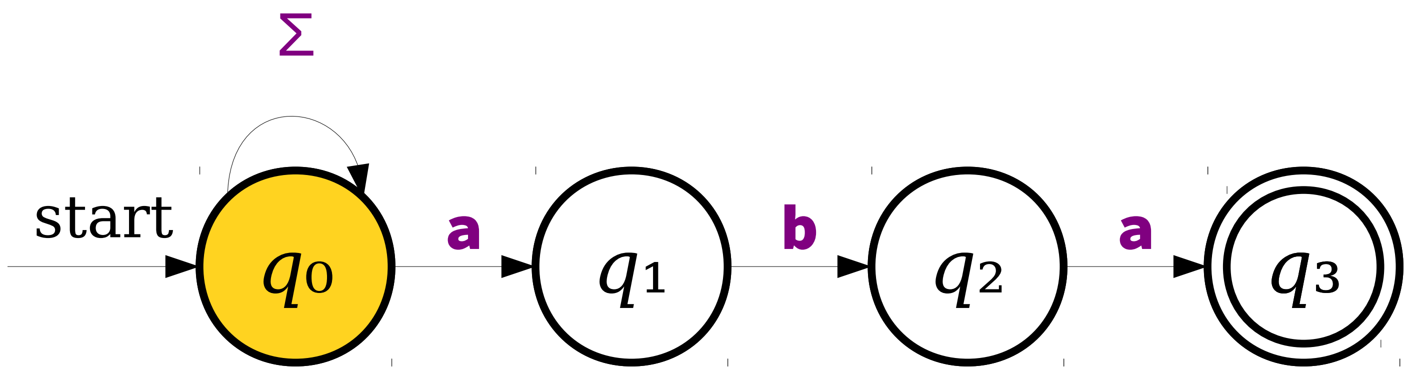
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



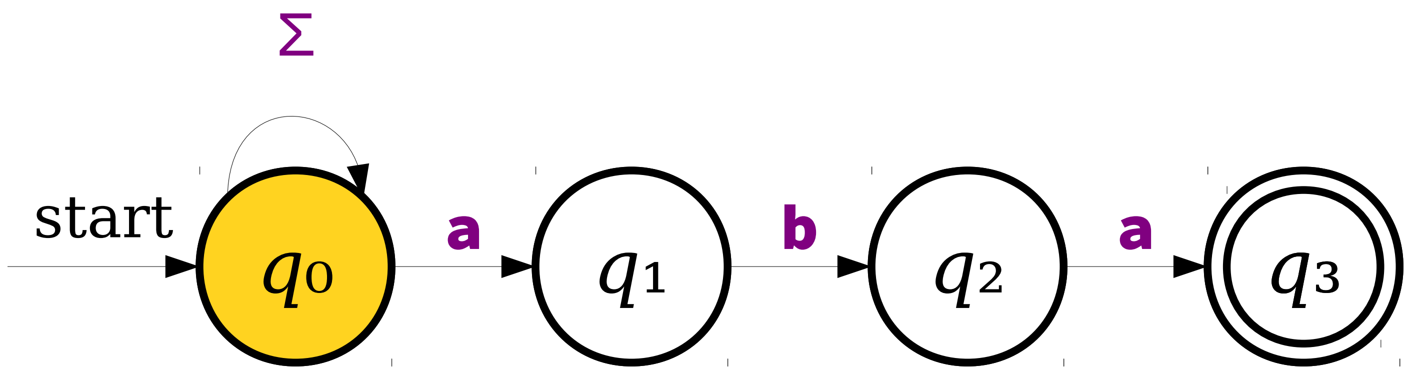
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



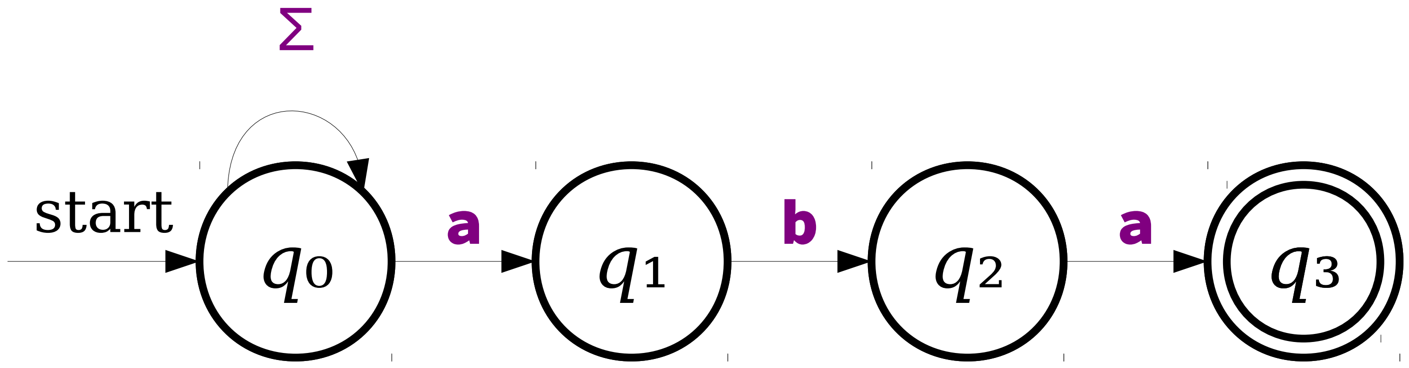
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



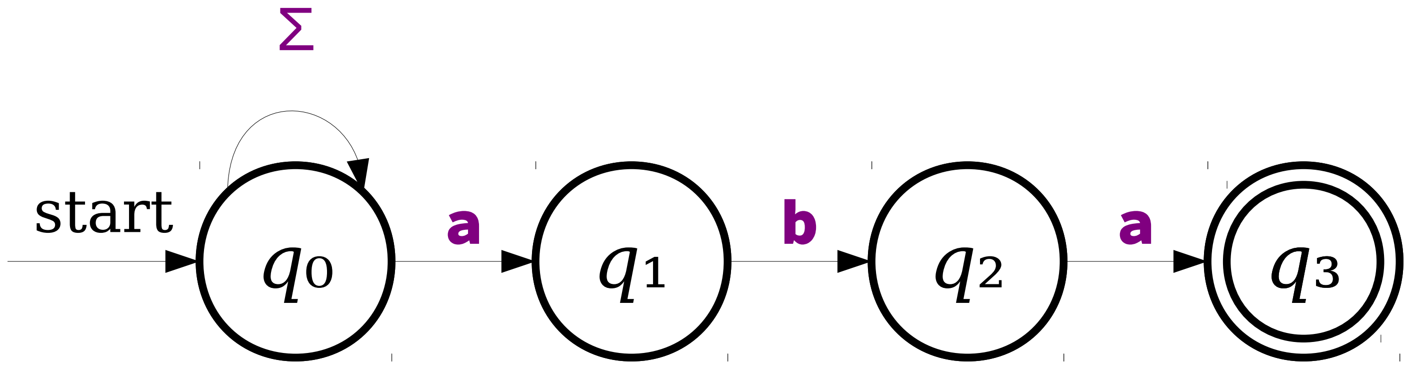
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



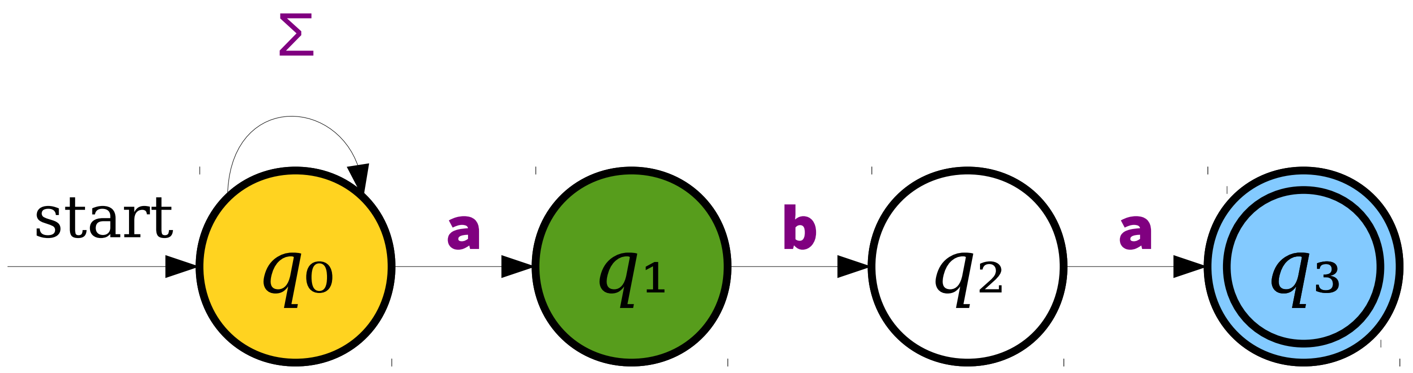
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$



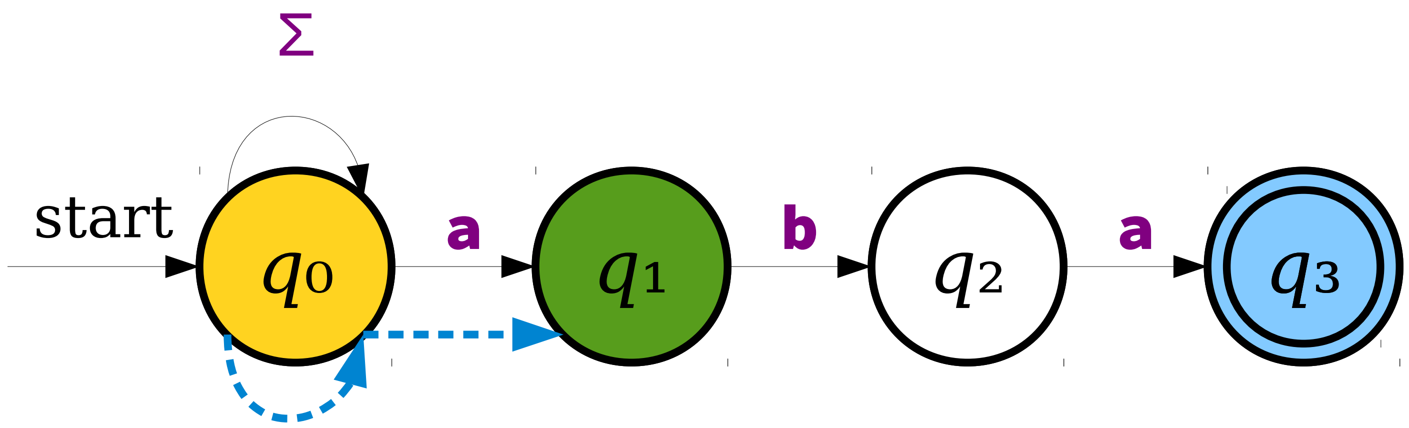
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$



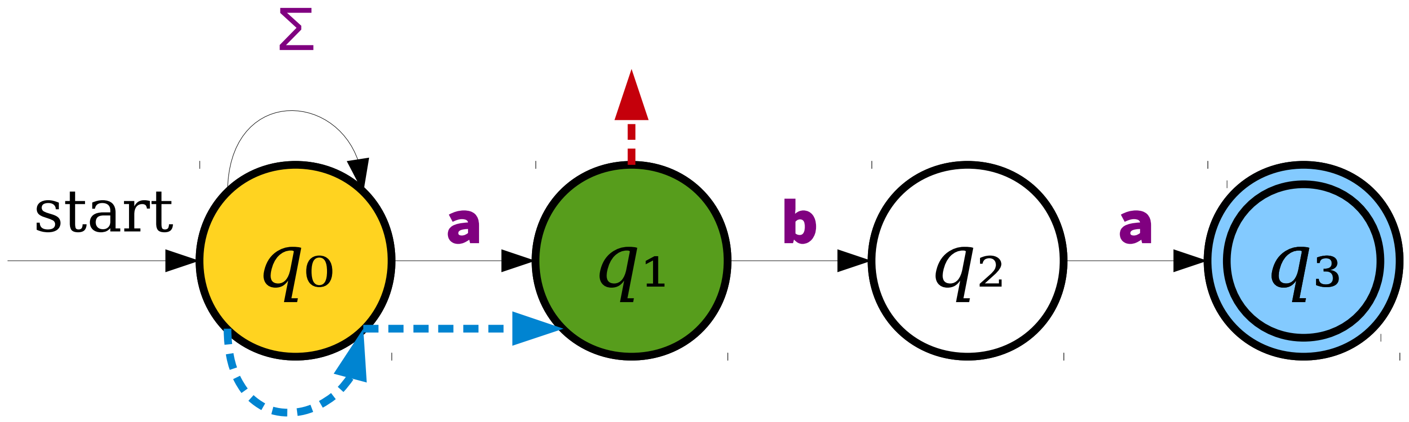
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



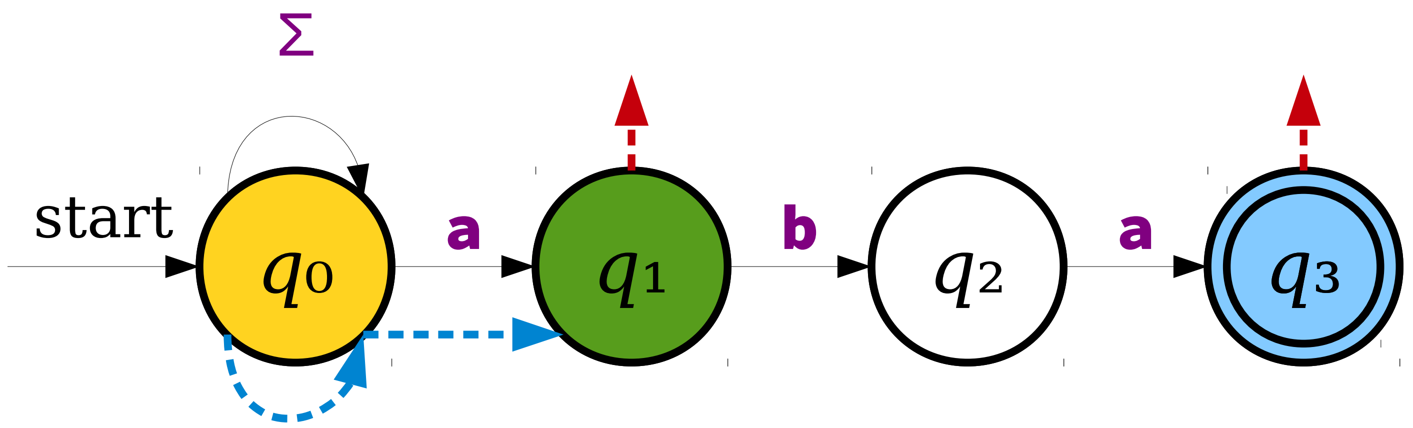
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



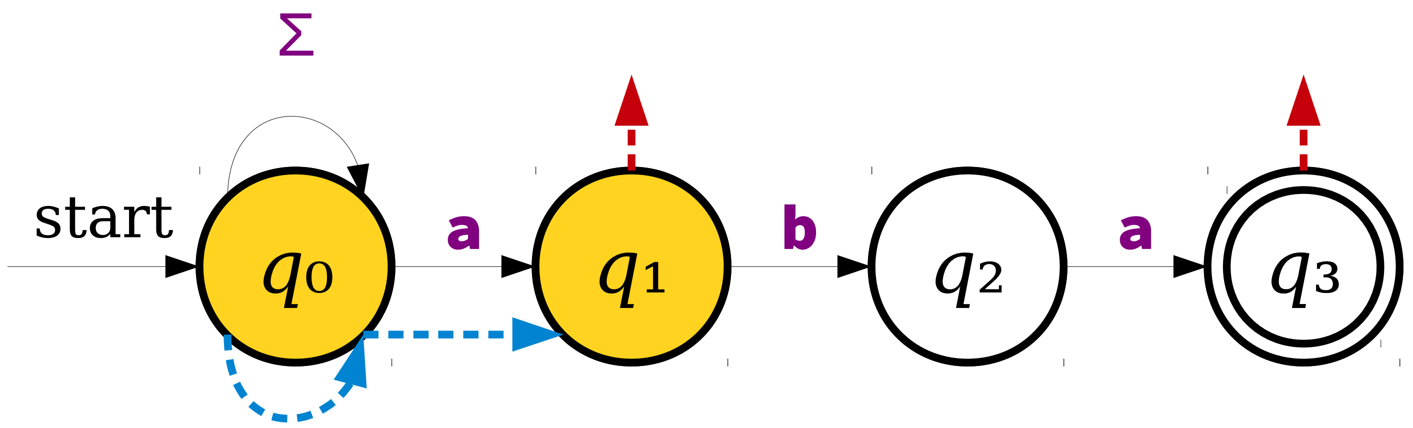
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



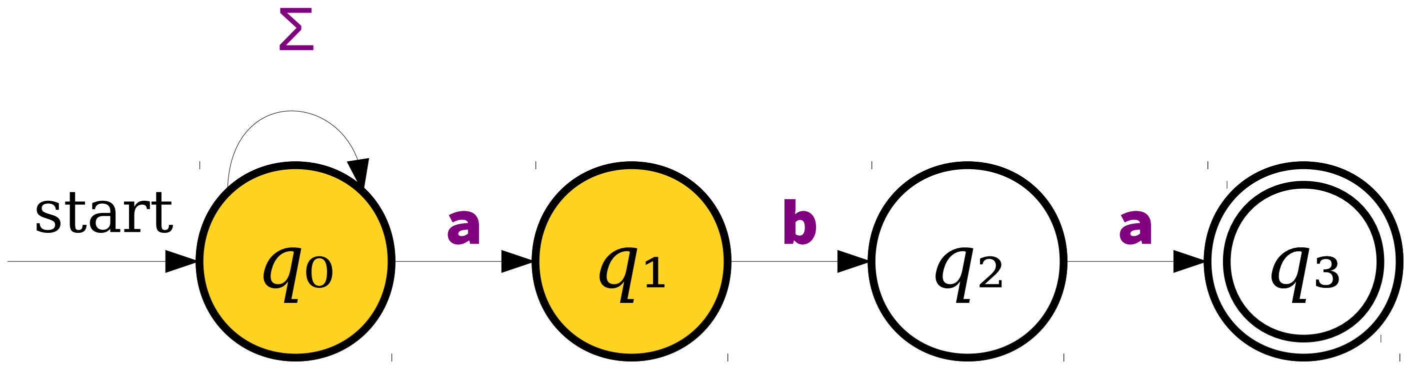
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



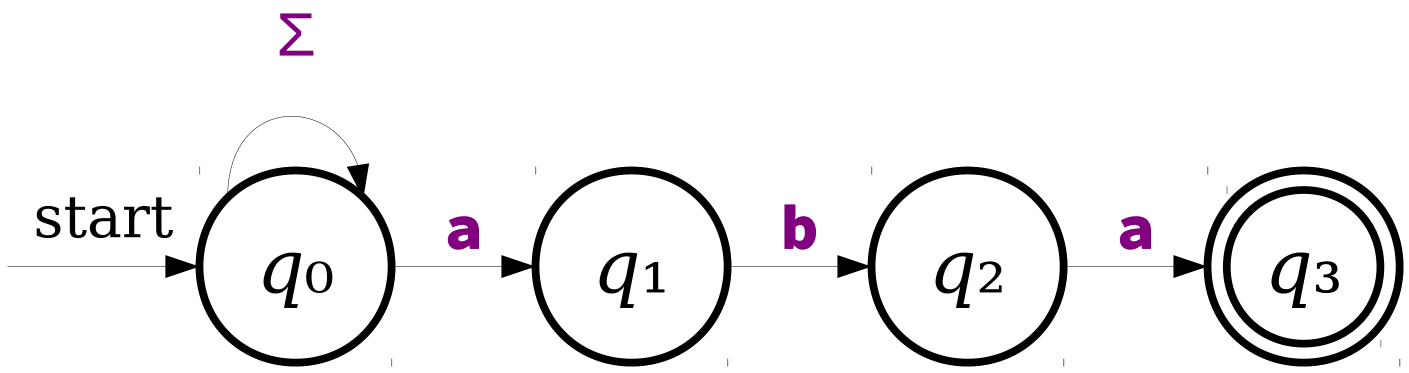
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



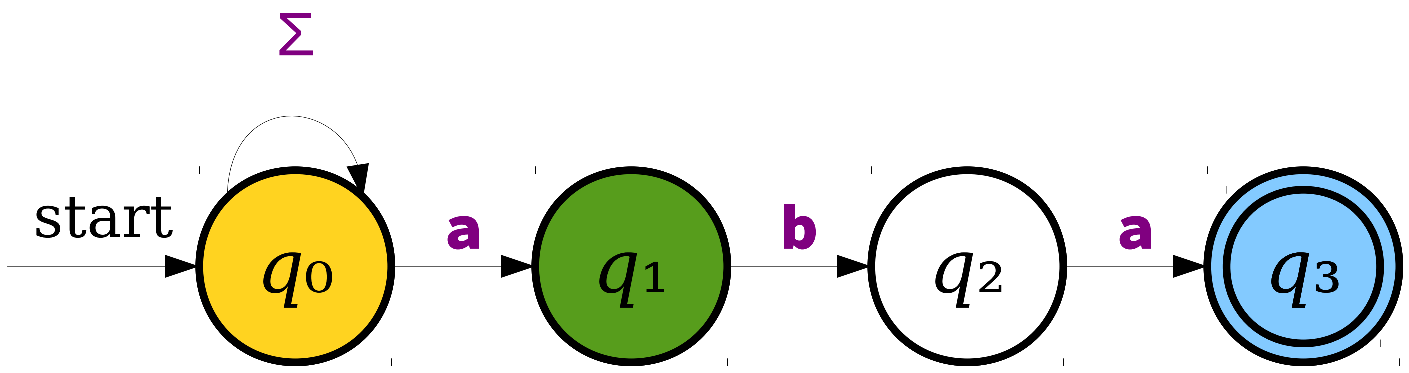
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



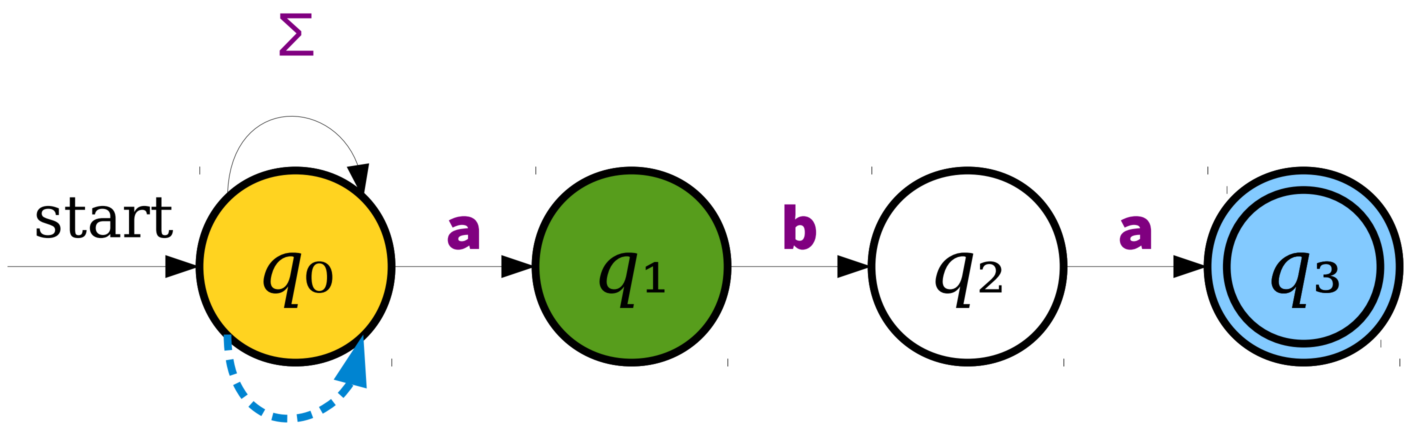
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



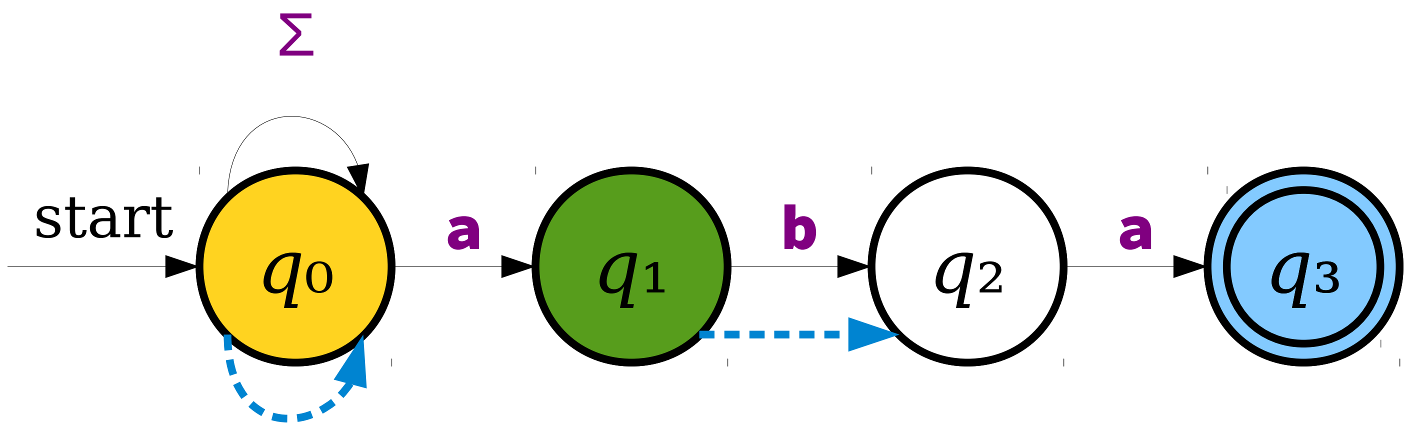
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



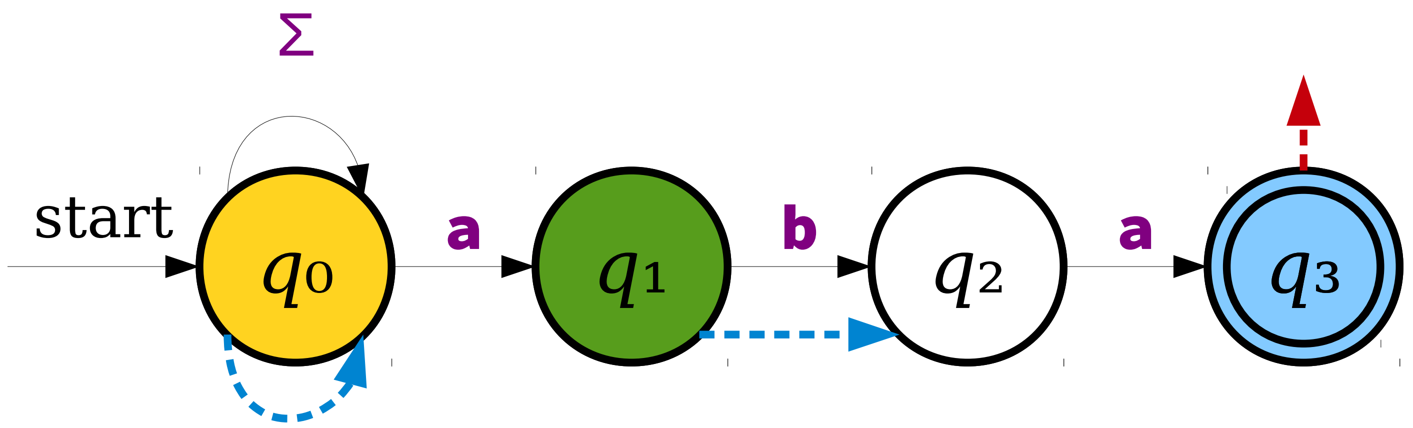
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



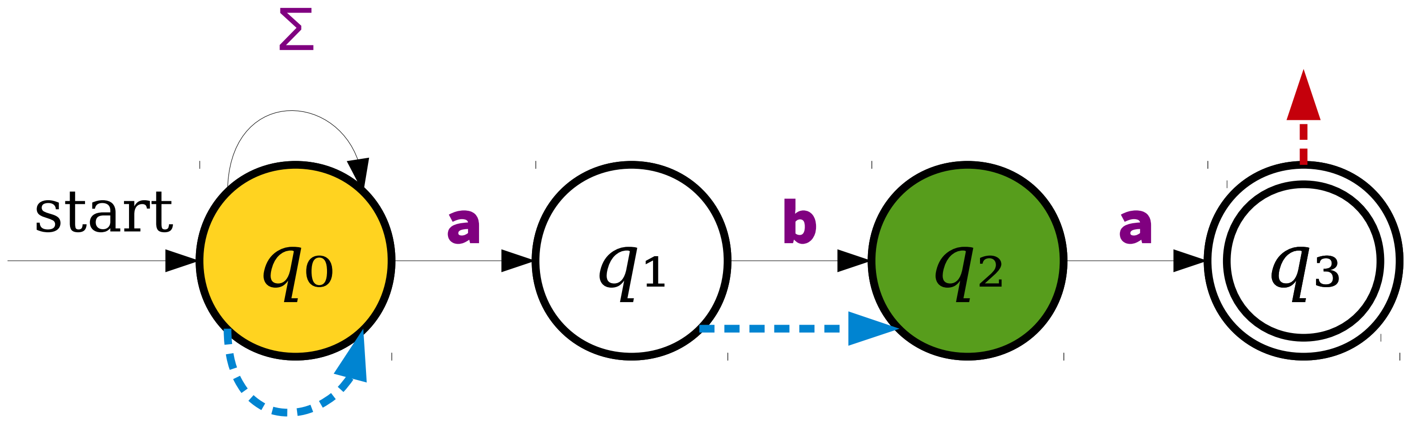
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



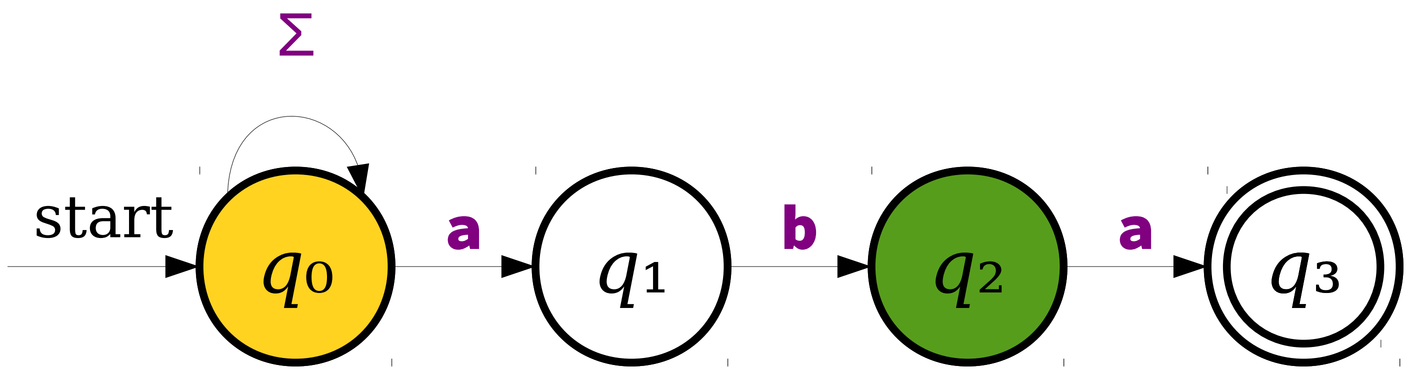
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



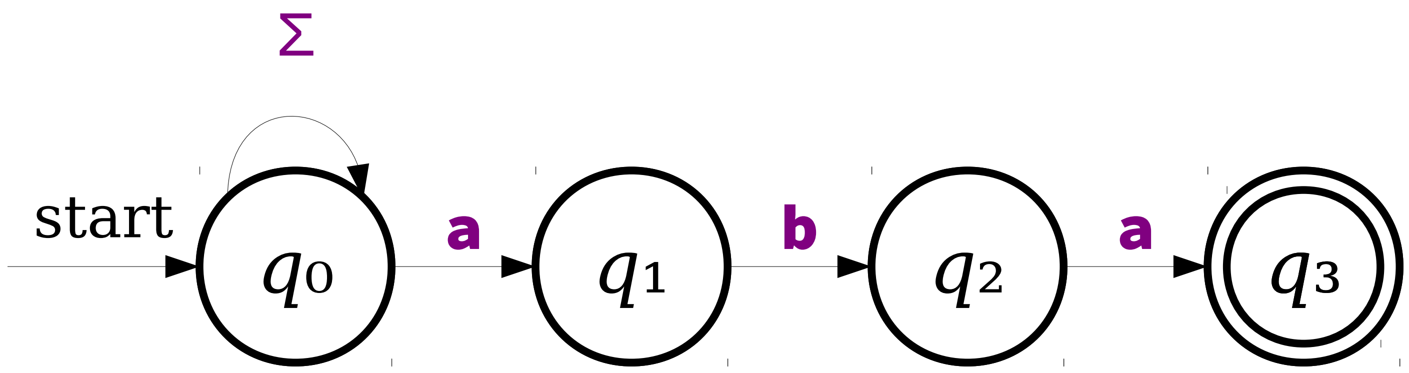
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



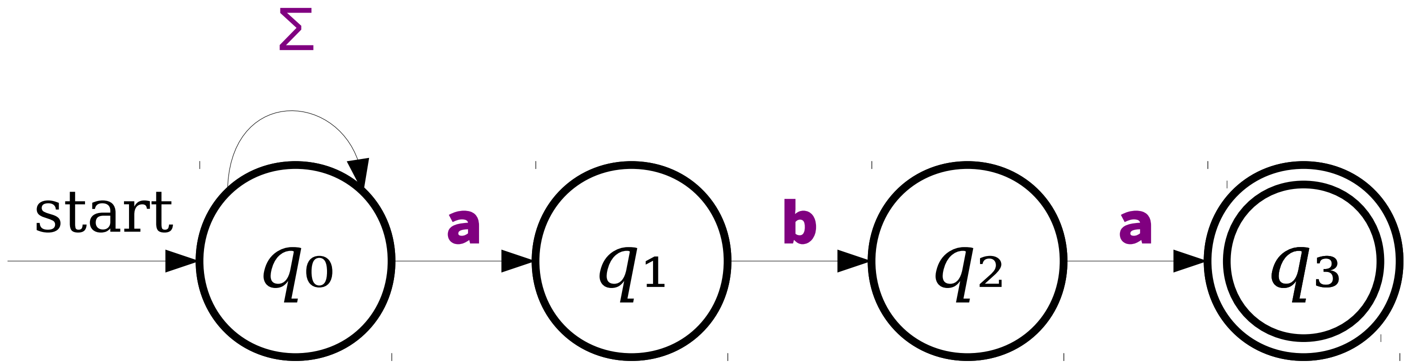
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



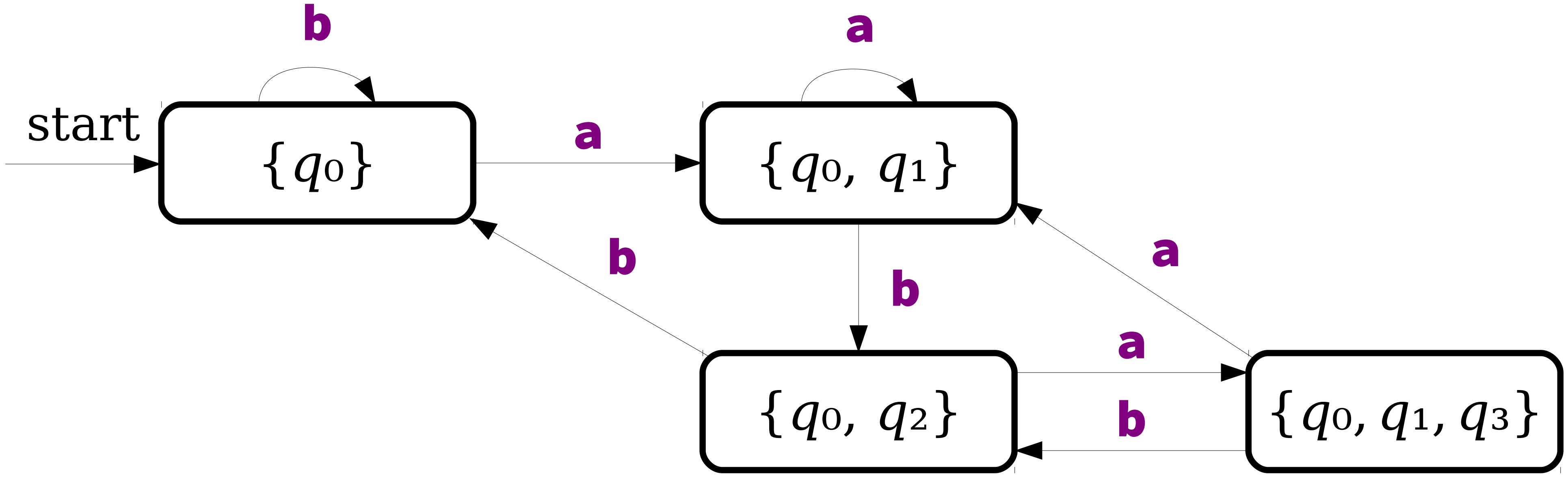
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

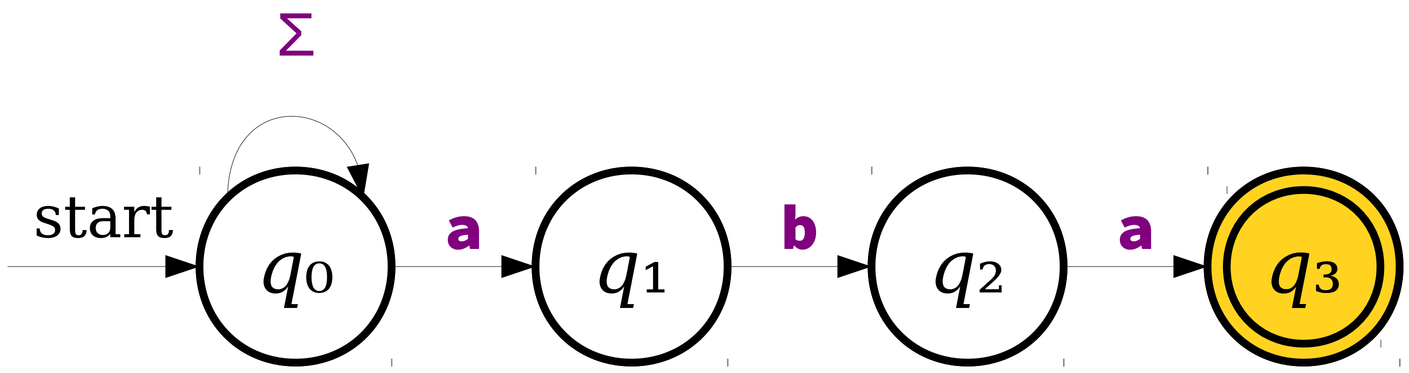


	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

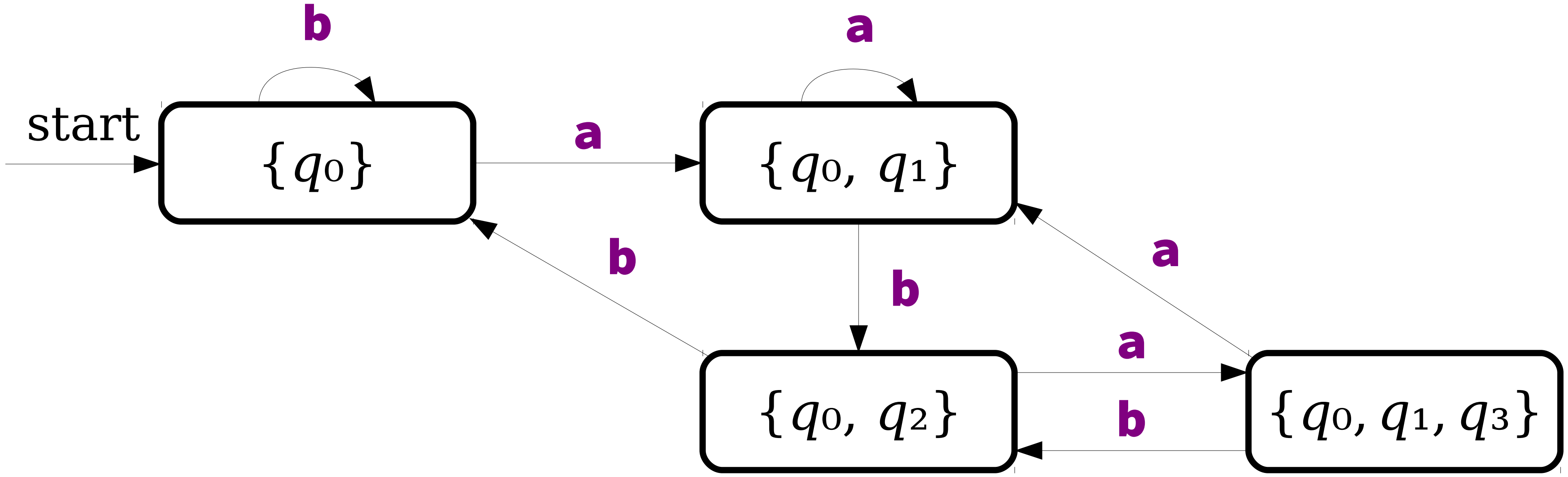


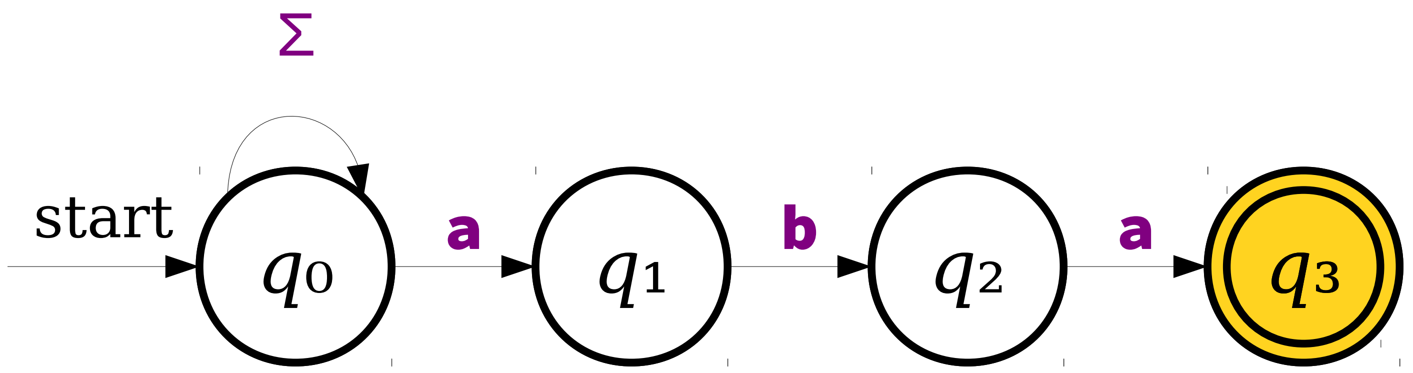
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



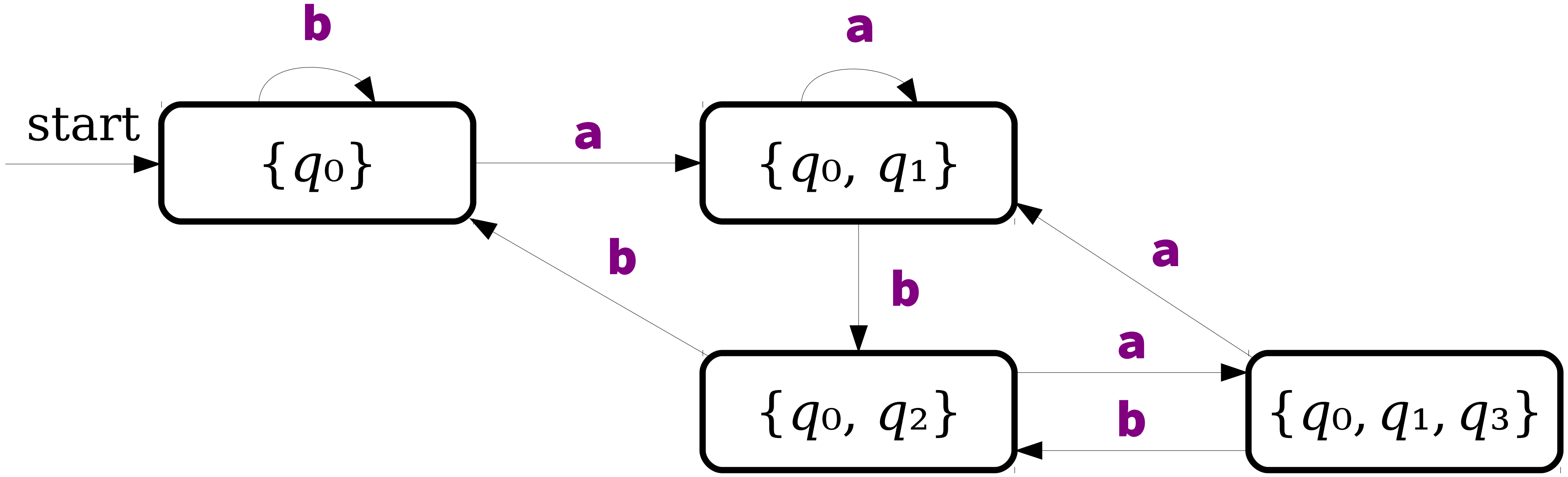


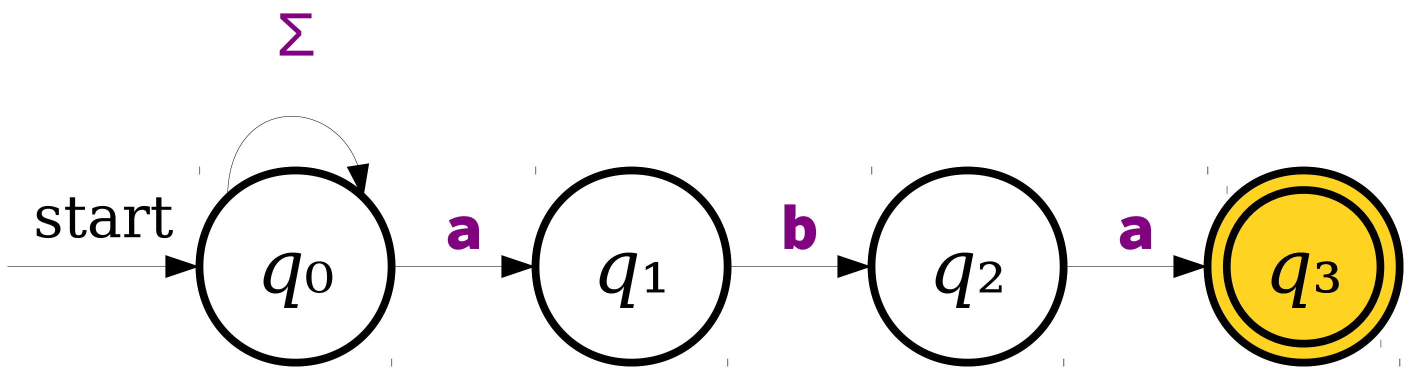
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



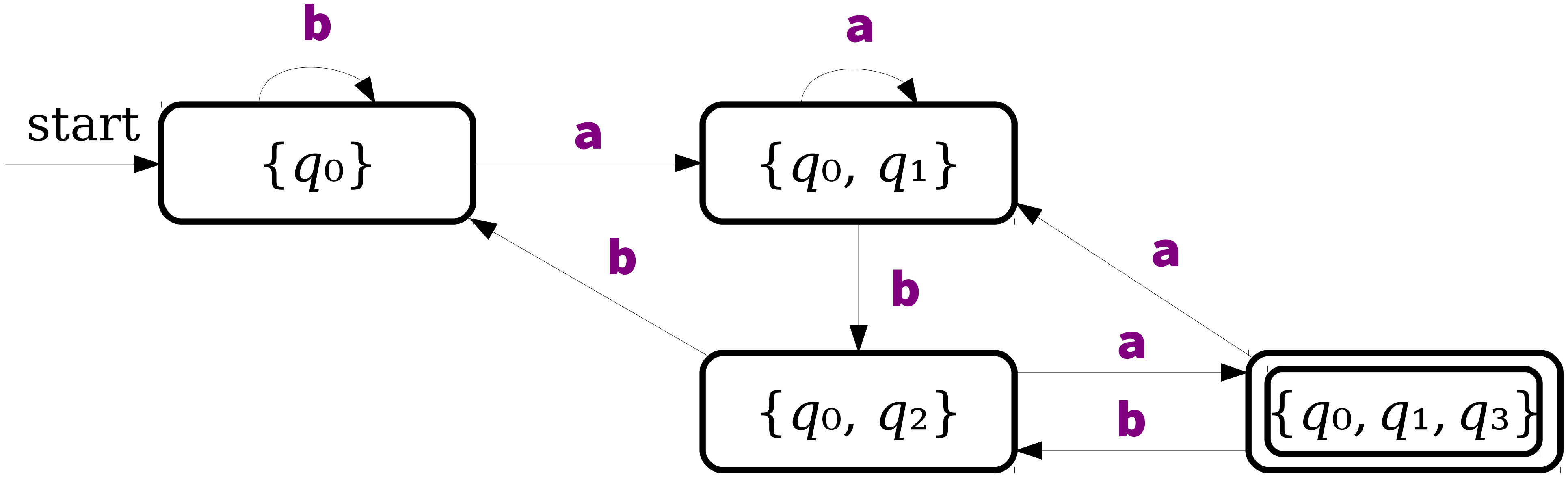


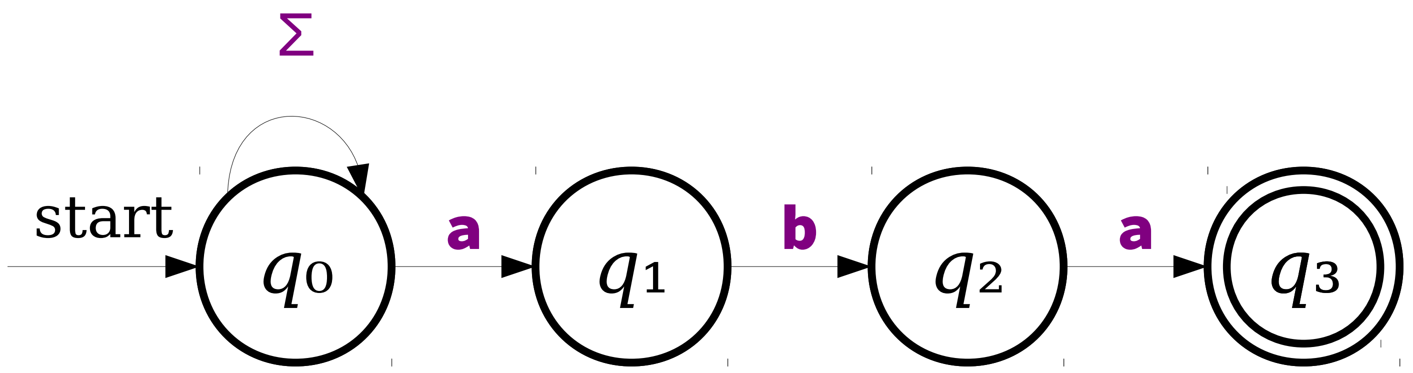
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



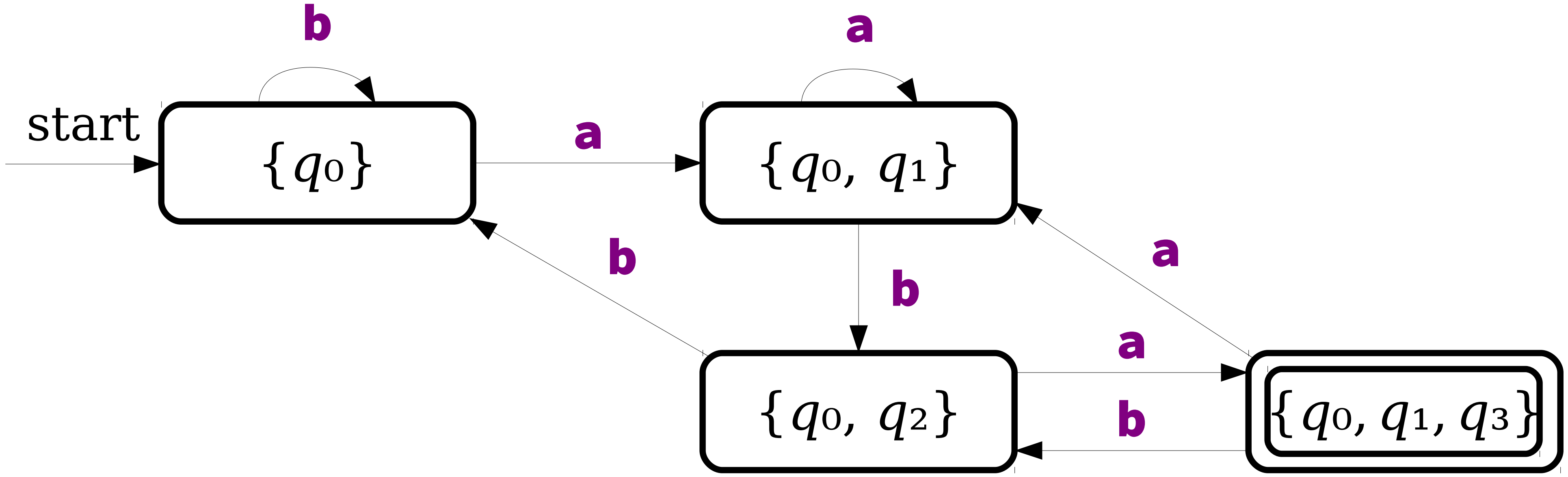


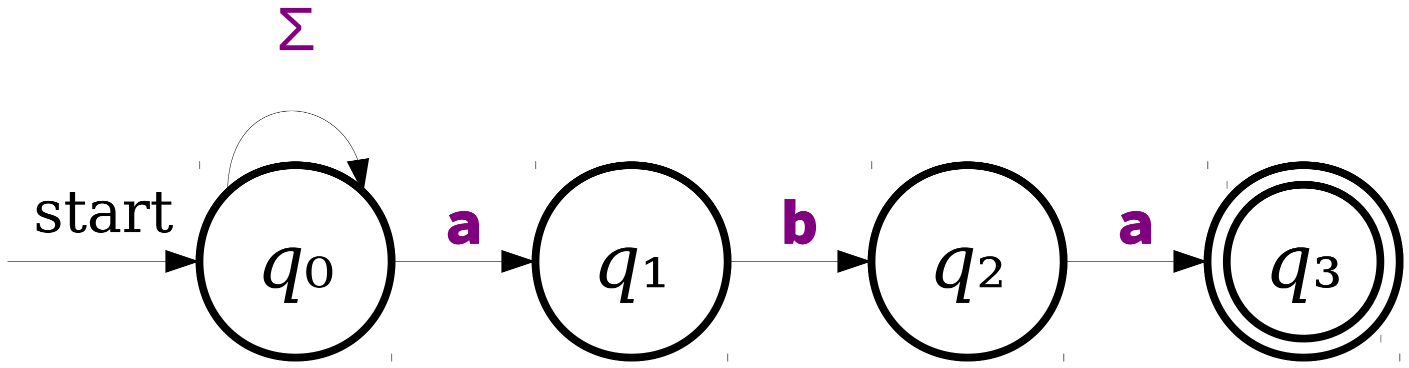
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



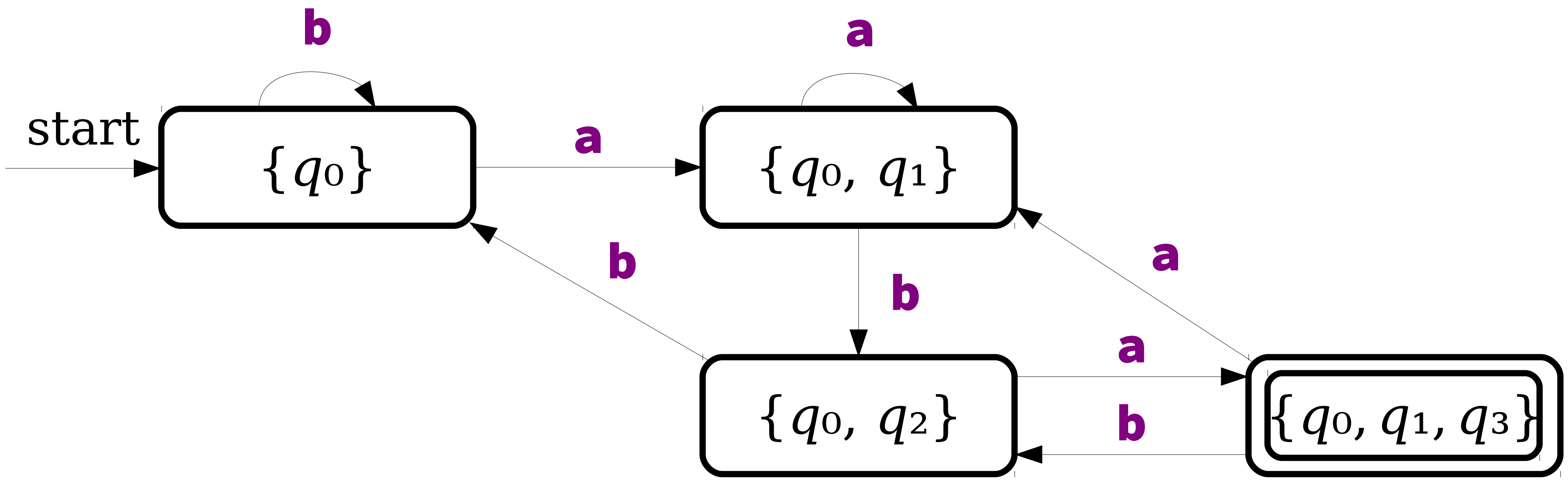


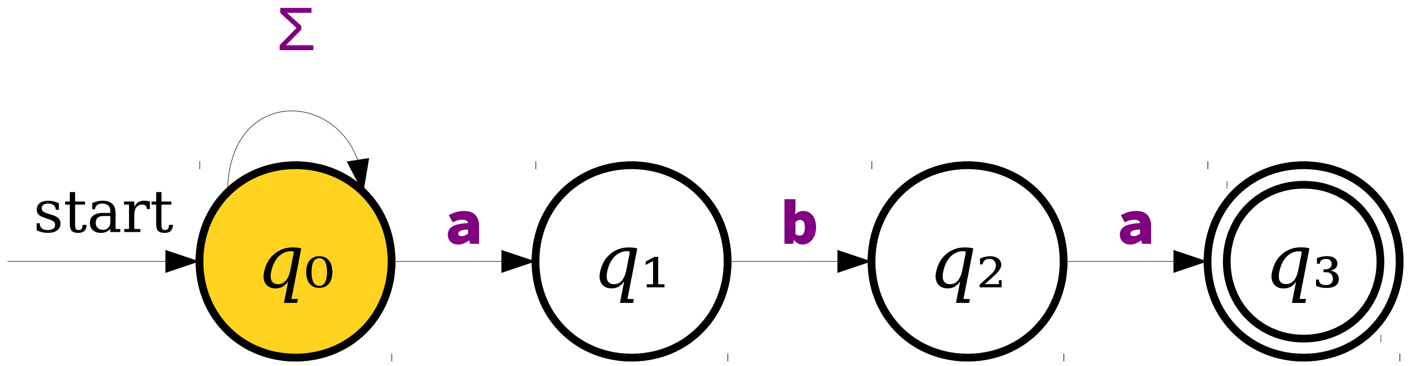
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



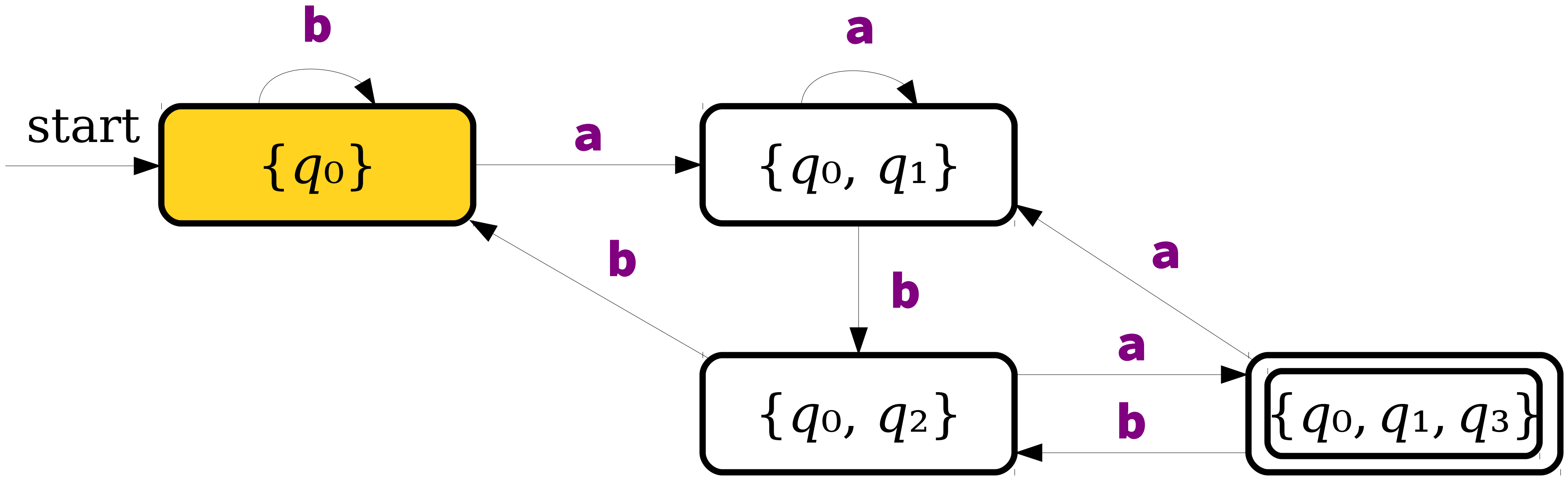


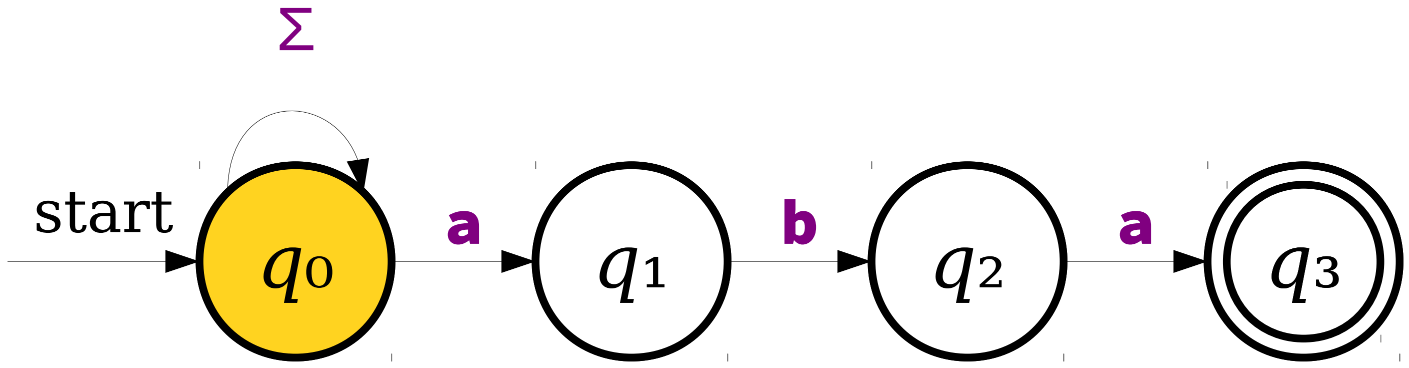
a	b	a	a	b	a

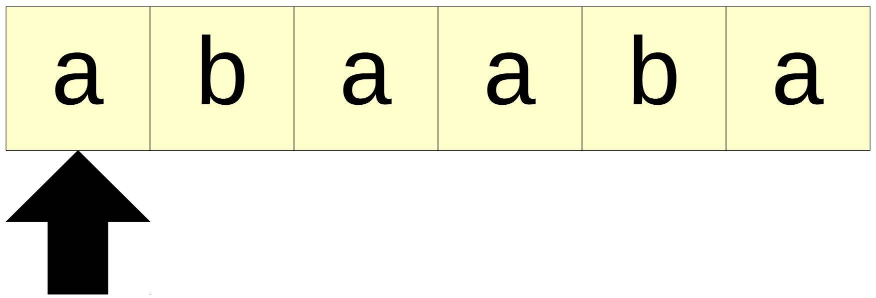


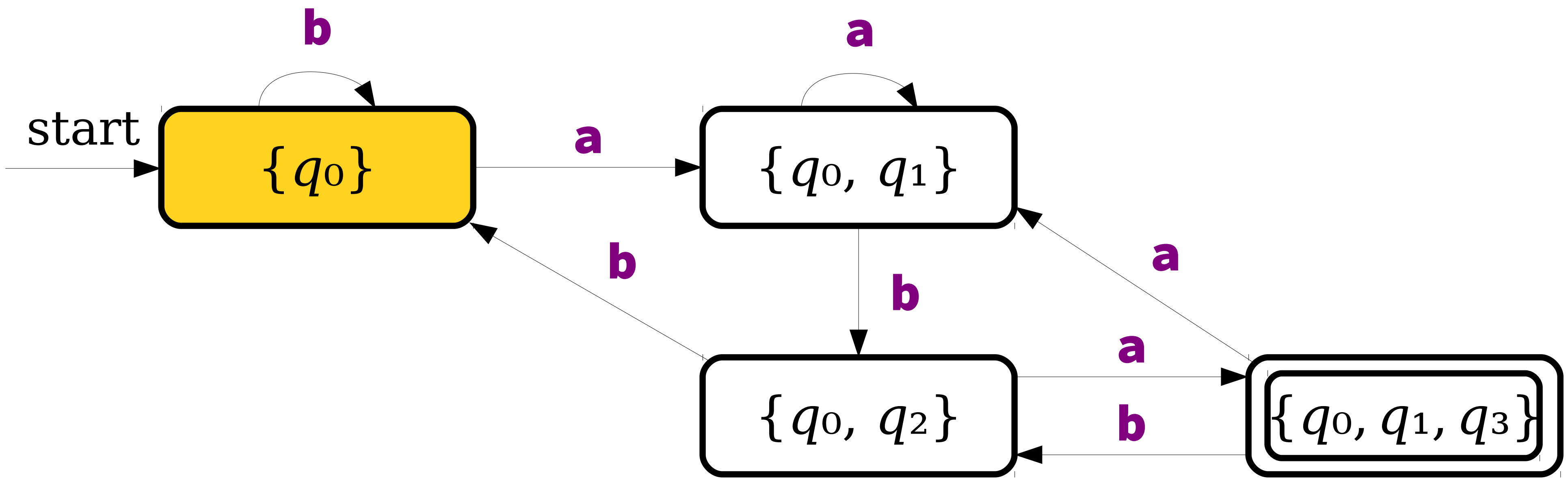


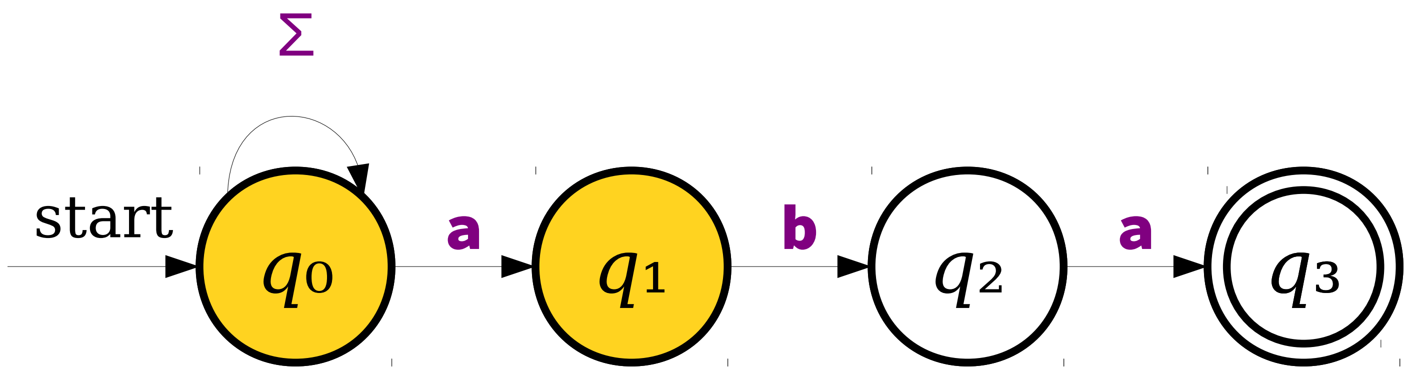
a	b	a	a	b	a

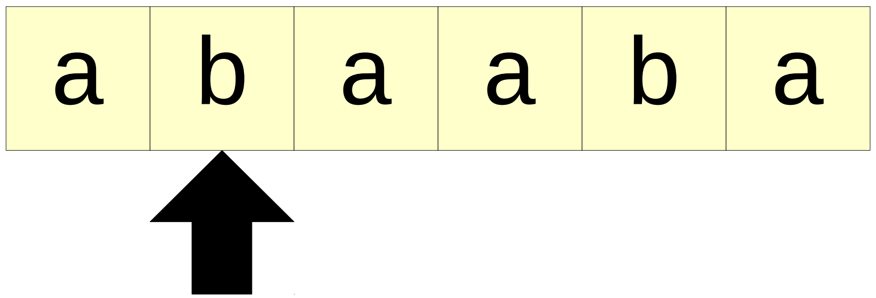


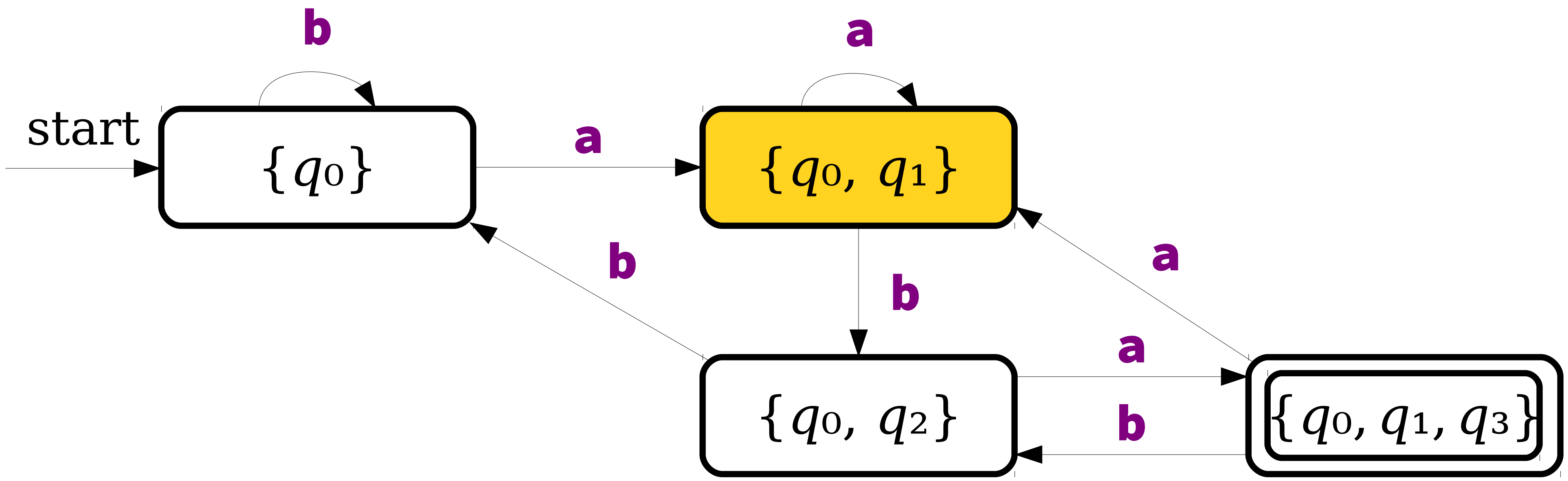

Σ
start
q_0
a
q_1
b
q_2
a
q_3

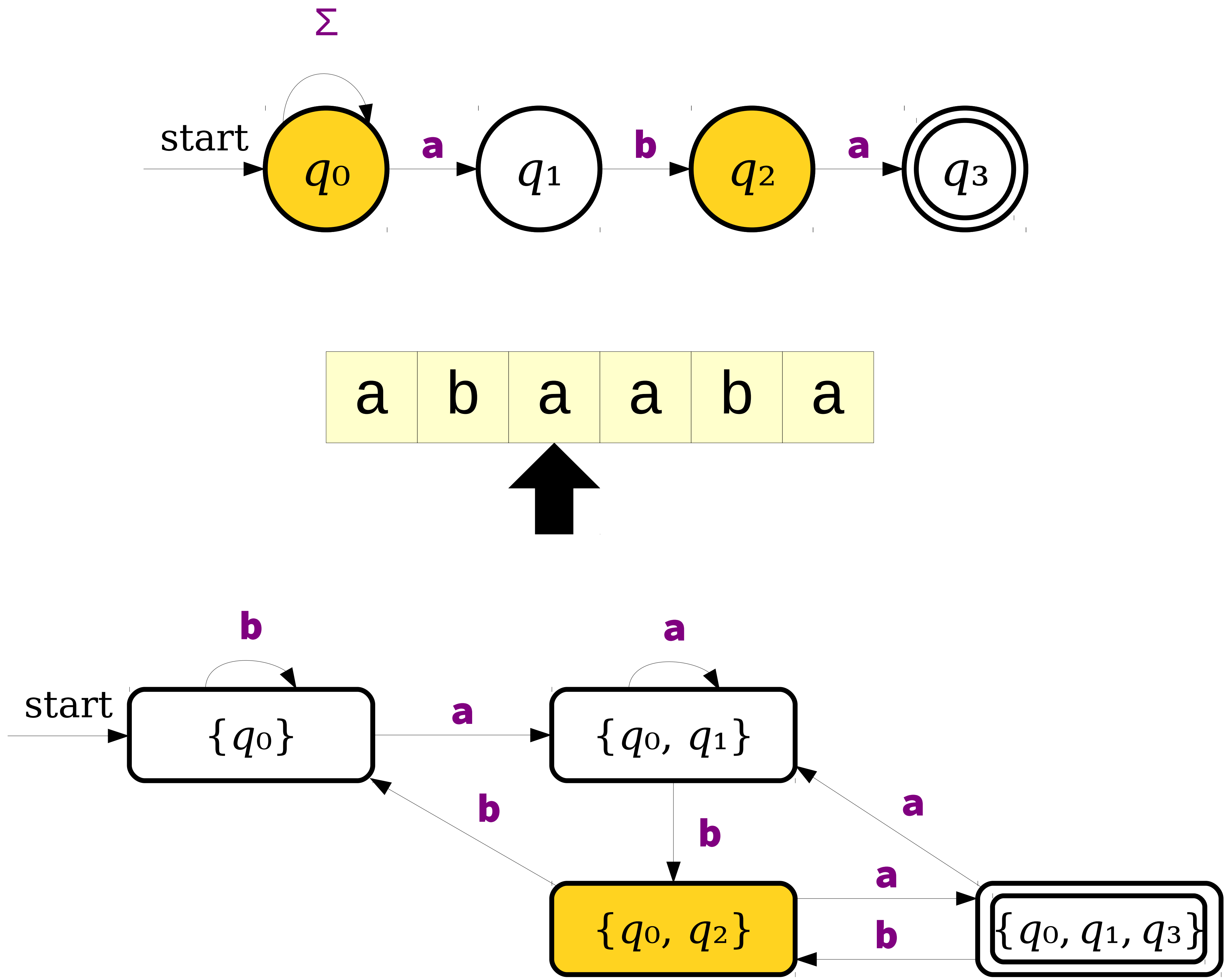

a b a a b a

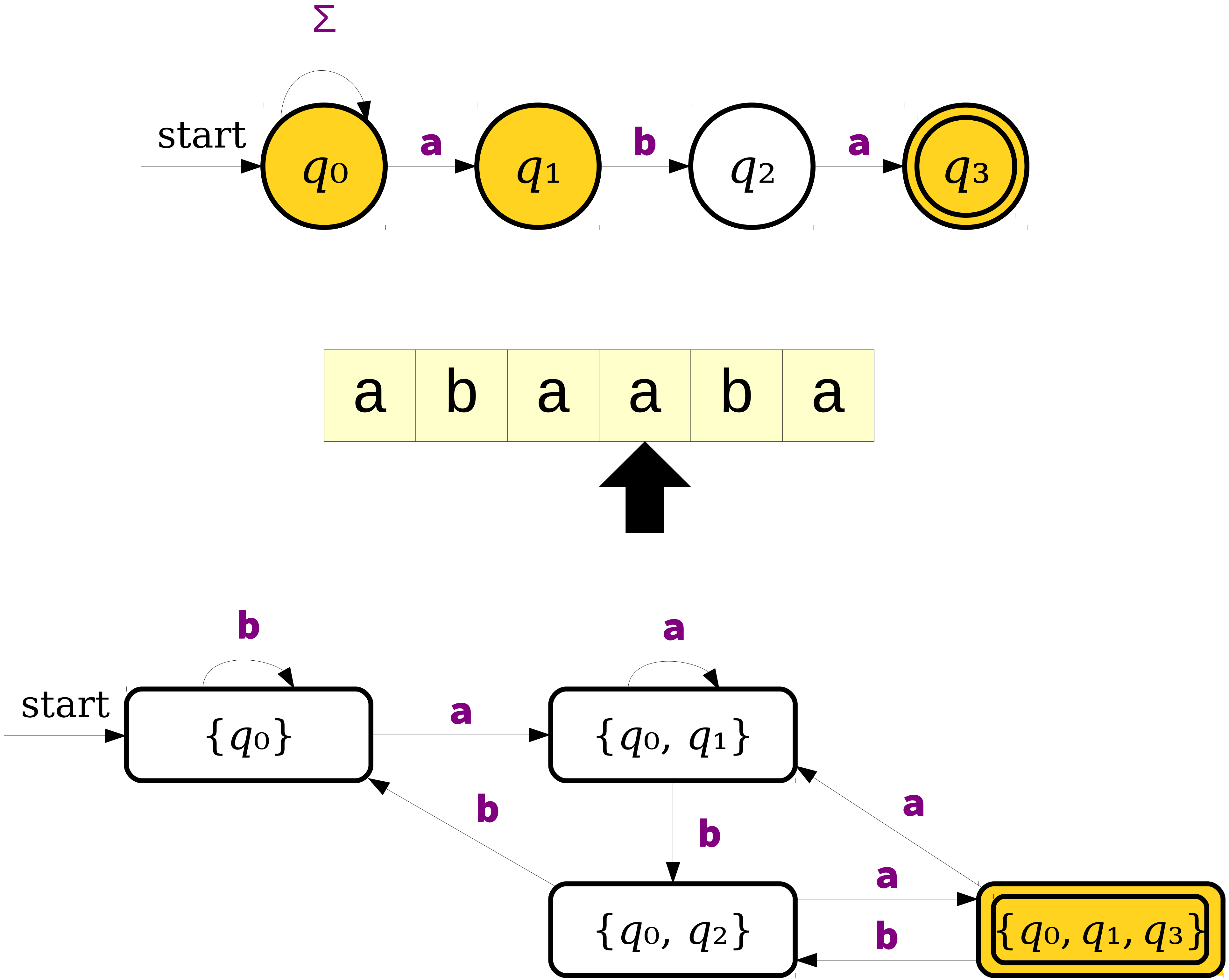

b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
a
b
a
$\{q_0, q_2\}$
b
$\{q_0, q_1, q_3\}$

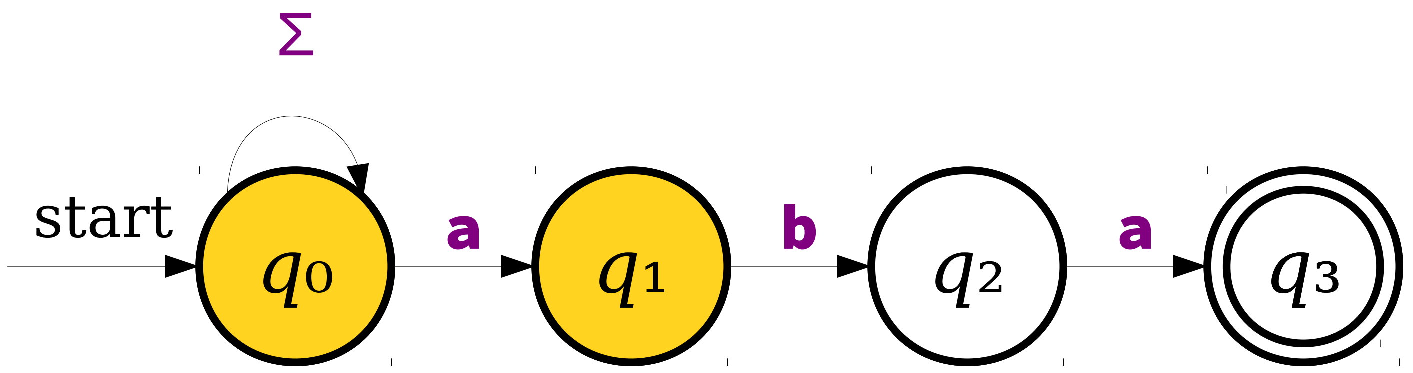

Σ
start
q_0
a
q_1
b
q_2
a
q_3

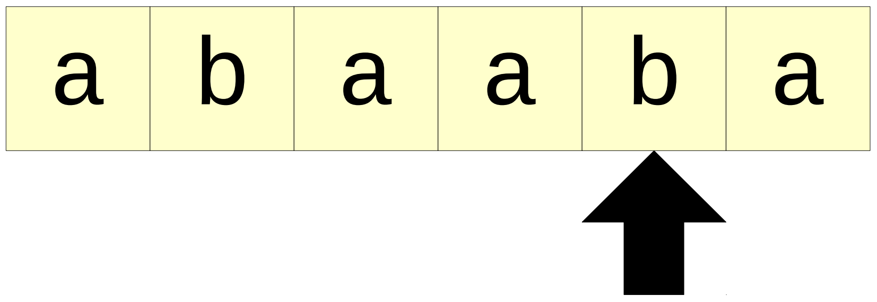

a b a a b a

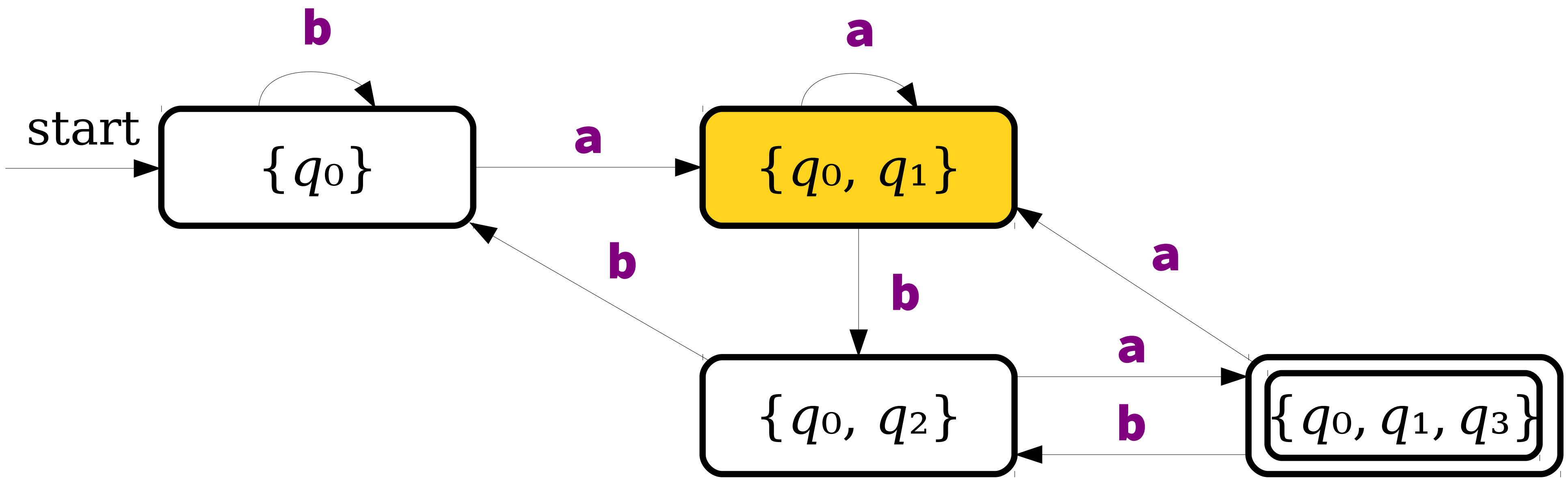

b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
b
a
a
$\{q_0, q_2\}$
b
$\{q_0, q_1, q_3\}$

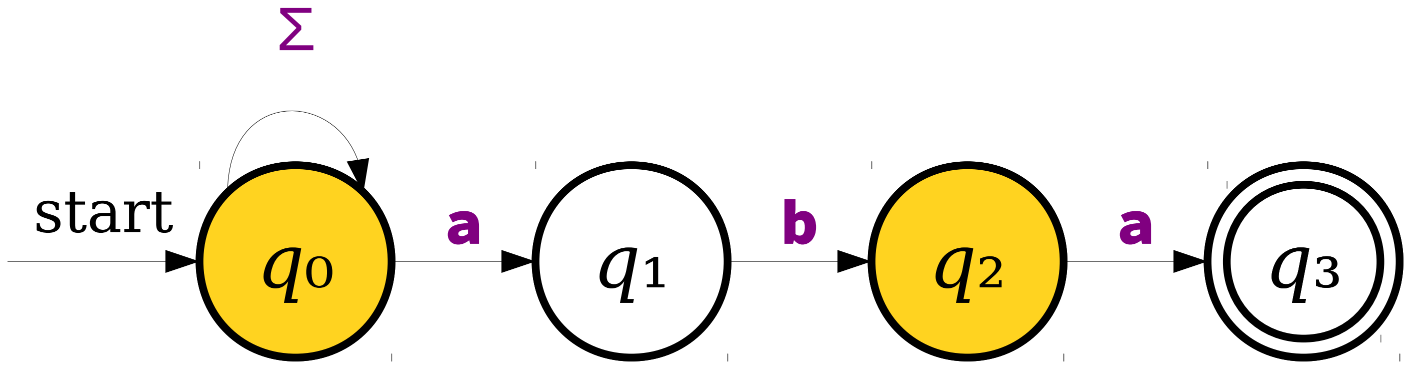

Σ
start
q0
a
q1
b
q2
a
q3
a b a a b a
b
a
start
{q0}
a
{q0, q1}
b
b
a
a
{q0, q2}
b
{q0, q1, q3}

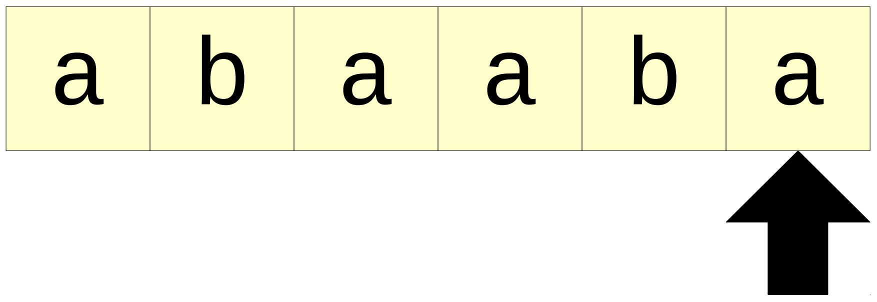

Σ
start
q0
a
q1
b
q2
a
q3
a b a a b a
b
a
start
{q0}
a
{q0, q1}
b
b
a
{q0, q2}
a
b
{q0, q1, q3}

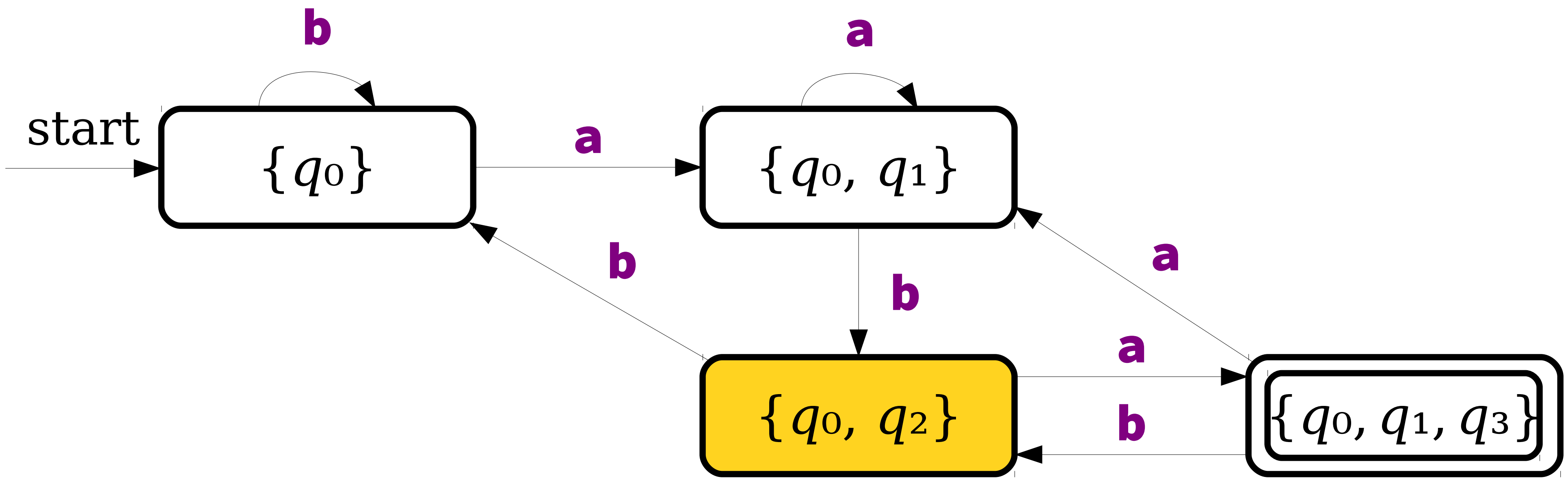

Σ
start
q_0
a
q_1
b
q_2
a
q_3

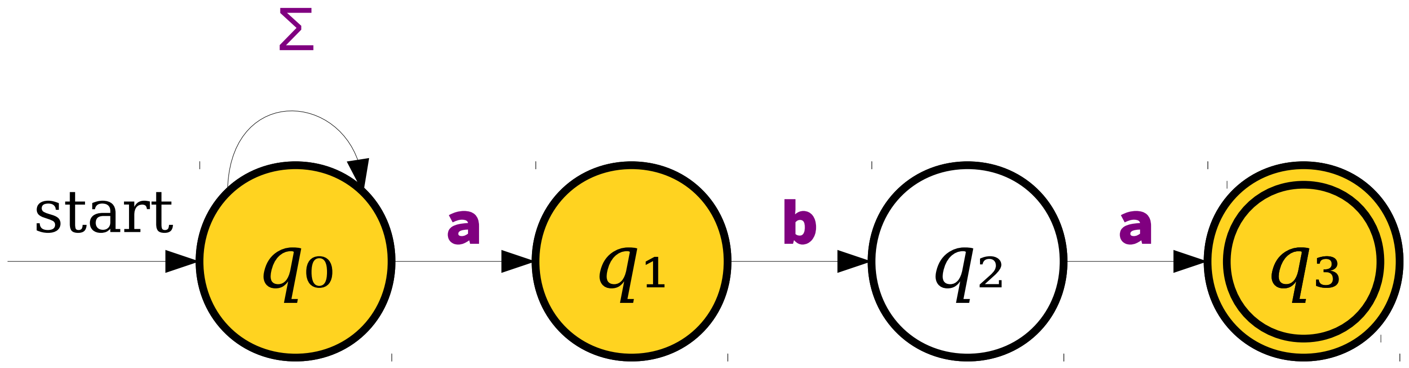

a b a a b a


start
$\{q_0\}$
b
a
$\{q_0, q_1\}$
a
b
b
a
$\{q_0, q_2\}$
a
b
$\{q_0, q_1, q_3\}$

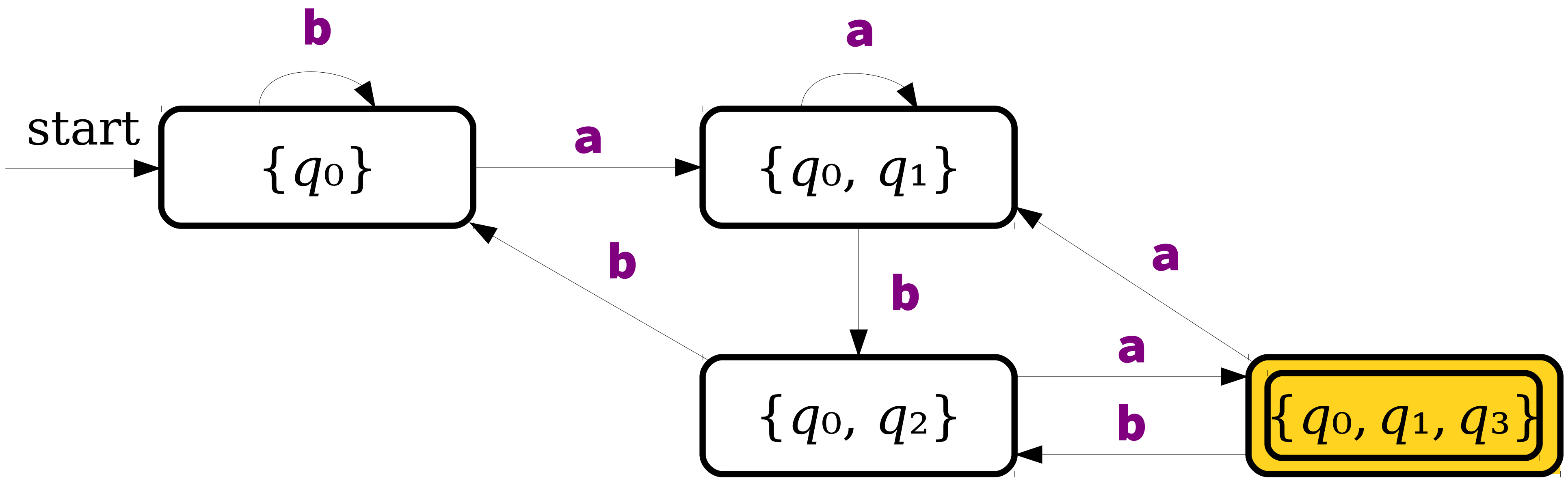

Σ
start
q_0
a
q_1
b
q_2
a
q_3


a b a a b a


b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
b
a
a
$\{q_0, q_2\}$
b
$\{q_0, q_1, q_3\}$



a	b	a	a	b	a



The Subset Construction

- This procedure for turning an NFA for a language L into a DFA for a language L is called the ***subset construction***.
  - It's sometimes called the ***powerset construction***; it's different names for the same thing!
- Intuitively:
  - Each state in the DFA corresponds to a set of states from the NFA.
  - Each transition in the DFA corresponds to what transitions would be taken in the NFA when using the massive parallel intuition.
  - The accepting states in the DFA correspond to which sets of states would be considered accepting in the NFA when using the massive parallel intuition.
- There's an online ***Guide to the Subset Construction*** with a more elaborate example involving ε-transitions and cases where the NFA dies; check that for more details.

The Subset Construction

- In converting an NFA to a DFA, the DFA's states correspond to sets of NFA states.
  - ***Useful fact:*** $|\wp(S)| = 2^{|S|}$ for any finite set S.
- So, in the worst-case, the construction can result in a DFA that is *exponentially larger* than the original NFA.
- ***Question to ponder:*** Can you find a family of languages that have NFAs of size n, but no DFAs of size less than 2^n?

Why This Matters

- We now have two perspectives on regular languages:
  - Regular languages are languages accepted by DFAs.
  - Regular languages are languages accepted by NFAs.
- We can now reason about the regular languages in two different ways.

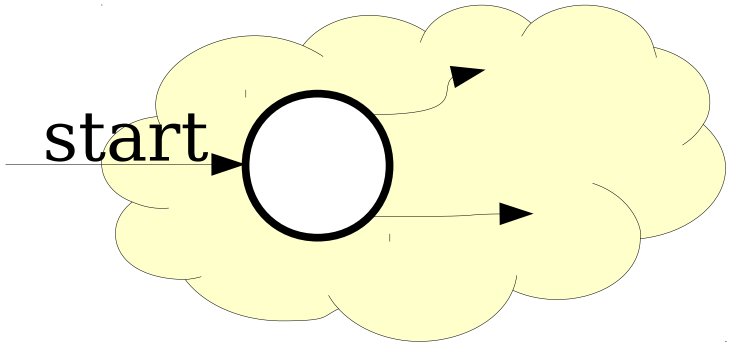
Properties of Regular Languages

The Union of Two Languages

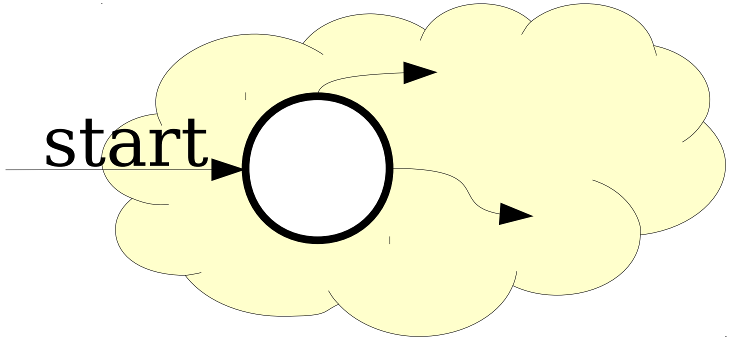
- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?

The Union of Two Languages

- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?



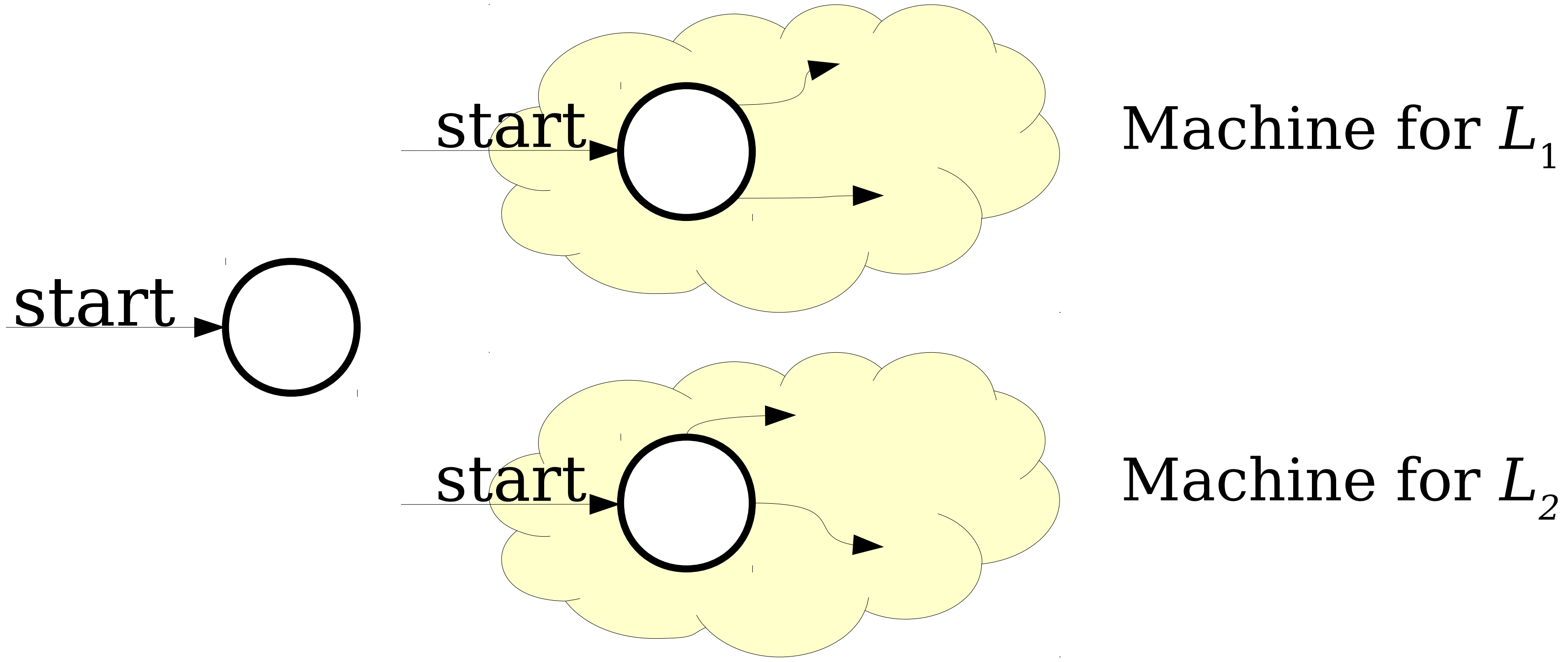
Machine for L_1



Machine for L_2

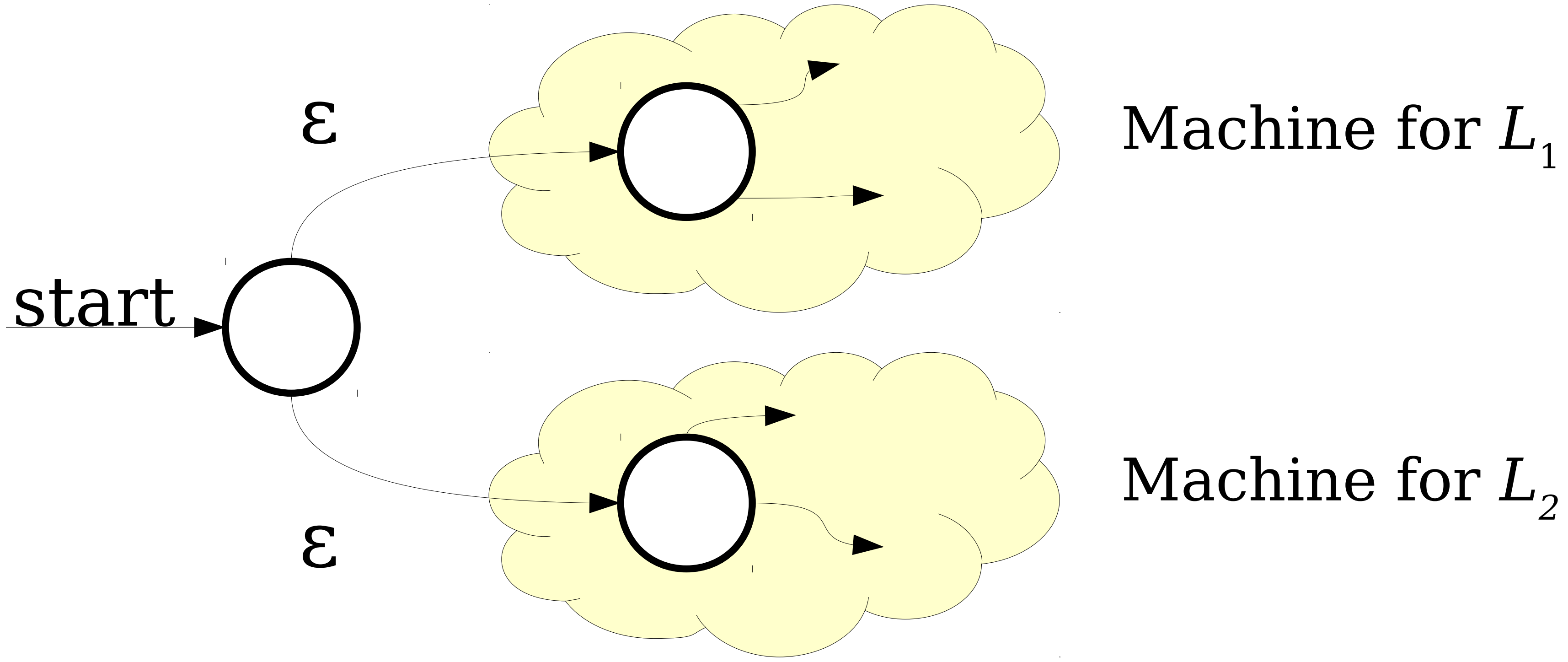
The Union of Two Languages

- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?



The Union of Two Languages

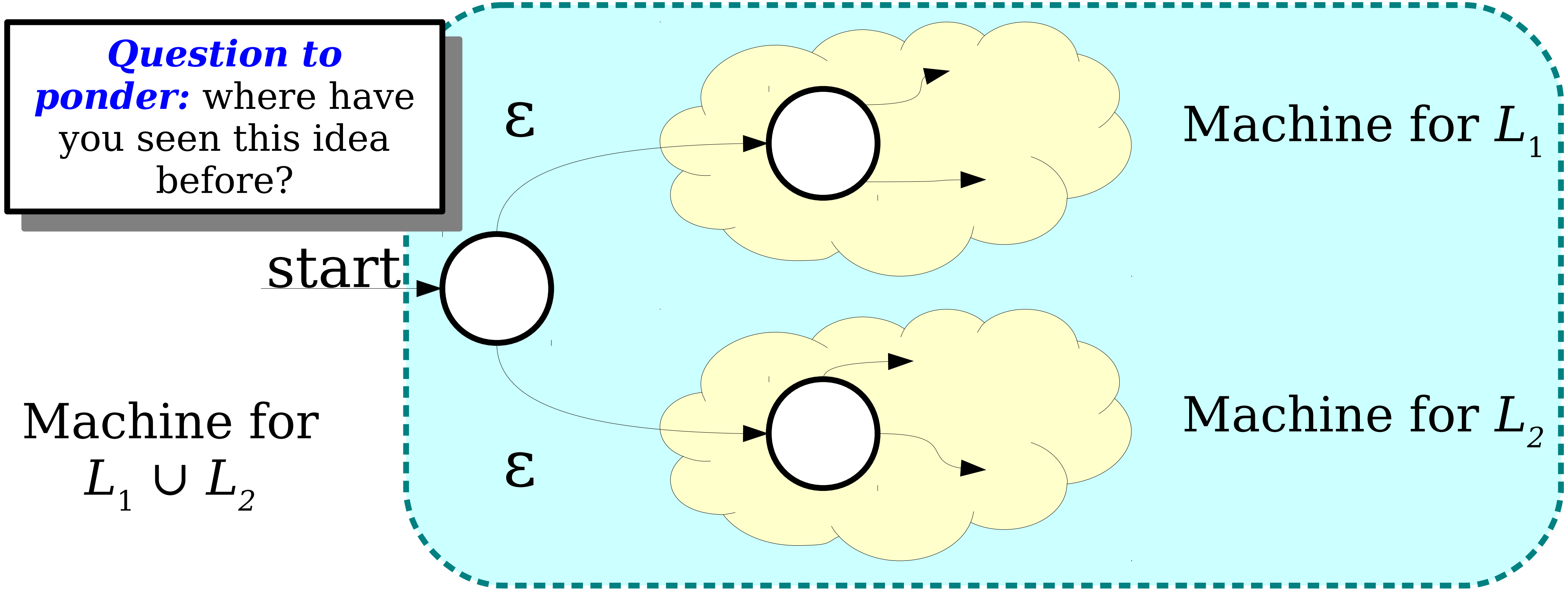
- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?



The Union of Two Languages

- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?

Question to ponder: where have you seen this idea before?

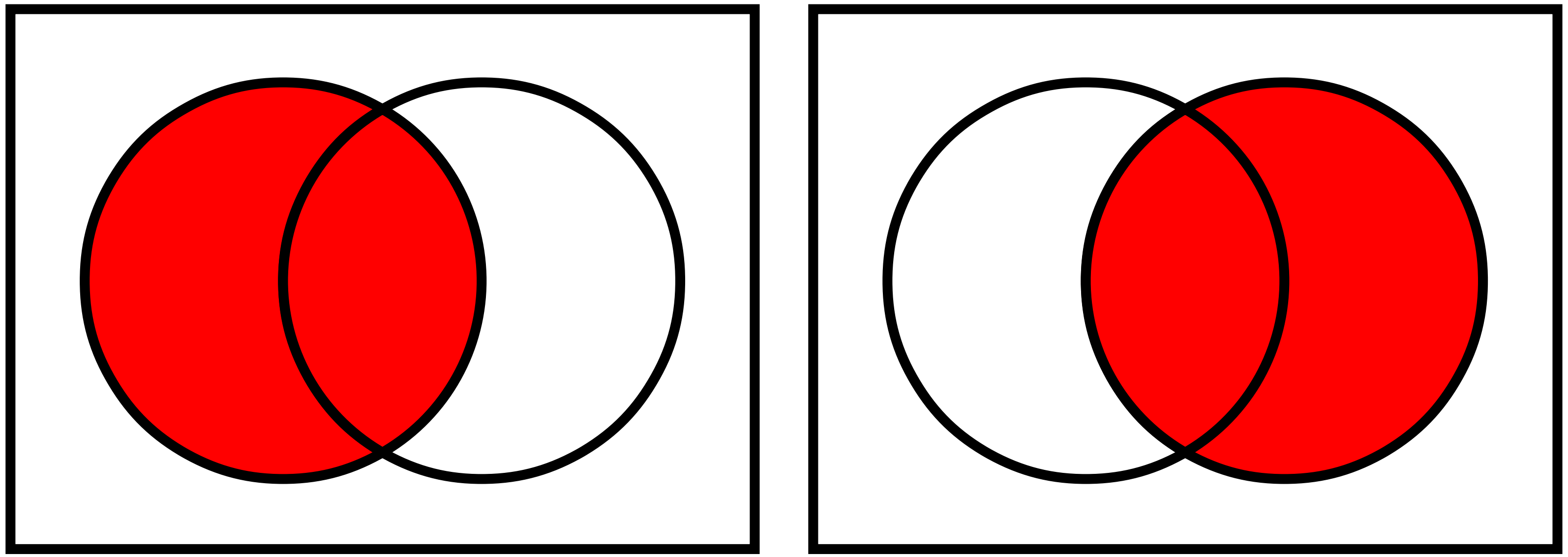


The Intersection of Two Languages

- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?

The Intersection of Two Languages

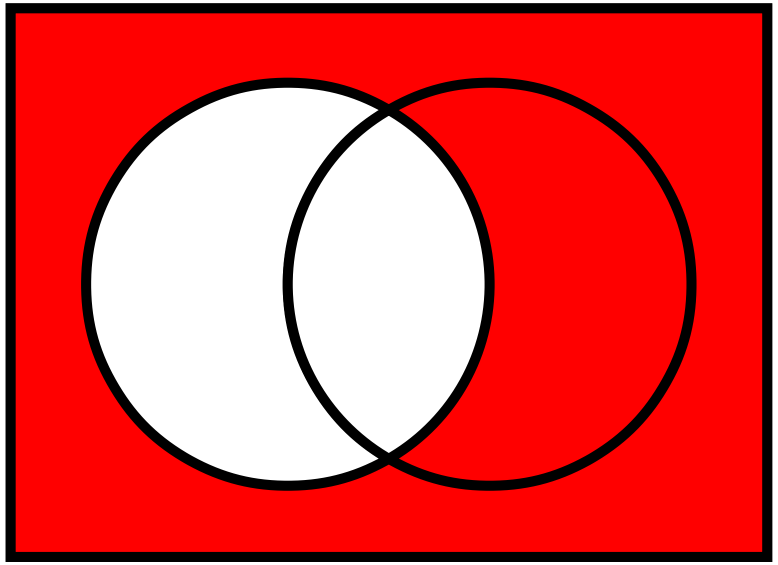
- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?



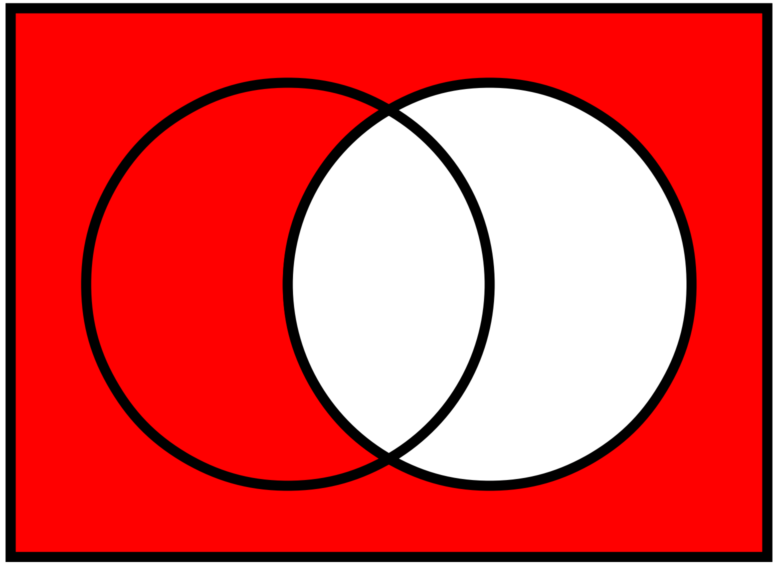
L_1 L_2

The Intersection of Two Languages

- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?



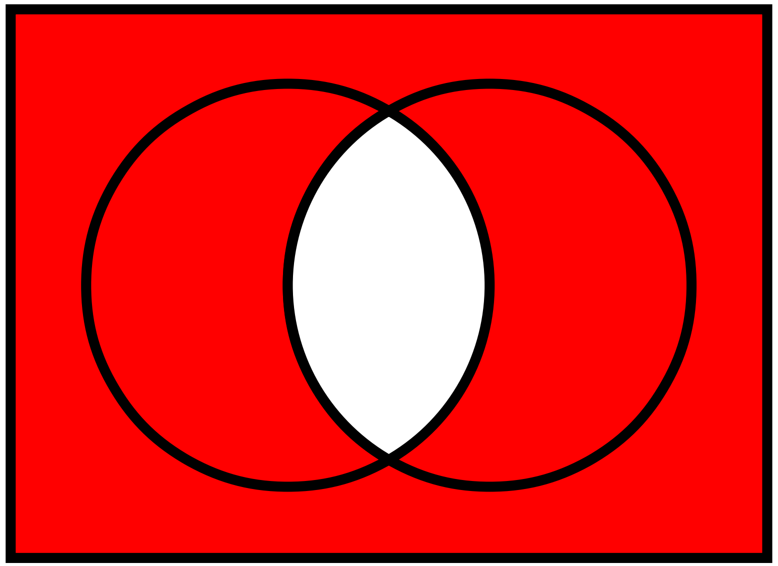
$\overline{L_1}$



$\overline{L_2}$

The Intersection of Two Languages

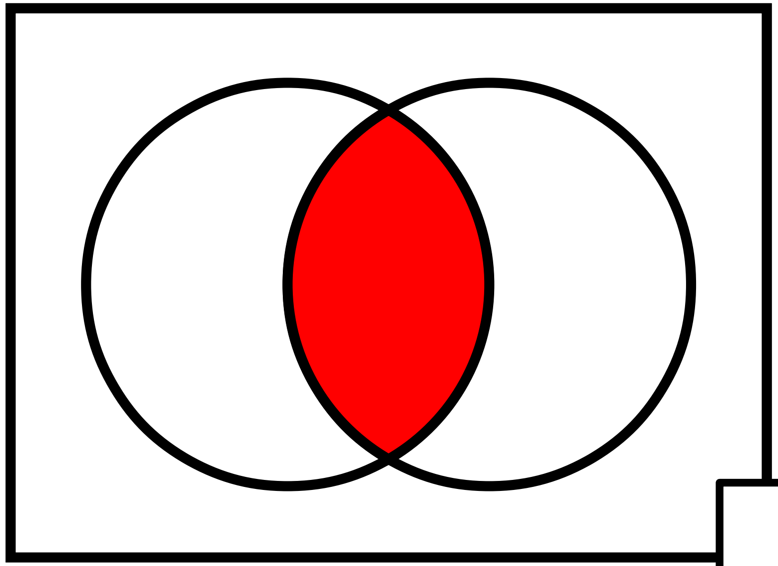
- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?



$$\overline{L}_1 \cup \overline{L}_2$$

The Intersection of Two Languages

- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?



$$\overline{\overline{L_1} \cup \overline{L_2}}$$

Hey, it's De Morgan's laws!

Concatenation

String Concatenation

- If $w \in \Sigma^*$ and $x \in \Sigma^*$, the ***concatenation*** of w and x, denoted $\boldsymbol{wx}$, is the string formed by tacking all the characters of x onto the end of w.
- Example: if w = **quo** and x = **kka**, the concatenation wx = **quokka**.
- This is analogous to the + operator for strings in many programming languages.
- Some facts about concatenation:
  - The empty string ε is the ***identity element*** for concatenation:

    $$w\varepsilon = \varepsilon w = w$$

  - Concatenation is ***associative***:

    $$wxy = w(xy) = (wx)y$$

Concatenation

- The ***concatenation*** of two languages L_1 and L_2 over the alphabet Σ is the language

$$\boldsymbol{L_1 L_2 = \{\, wx \in \Sigma^* \mid w \in L_1 \land x \in L_2 \,\}}$$

Concatenation Example

- Let Σ = { **a**, **b**, …, **z**, **A**, **B**, …, **Z** } and consider these languages over Σ:
  - ***Noun*** = { **Puppy**, **Rainbow**, **Whale**, … }
  - ***Verb*** = { **Hugs**, **Juggles**, **Loves**, … }
  - ***The*** = { **The** }
- The language ***TheNounVerbTheNoun*** is
  - { **ThePuppyHugsTheWhale**,
    TheWhaleLovesTheRainbow,
    TheRainbowJugglesTheRainbow, … }

Concatenation

- The ***concatenation*** of two languages L_1 and L_2 over the alphabet Σ is the language

$$L_1L_2 = \{ wx \in \Sigma^* \mid w \in L_1 \land x \in L_2 \}$$

- Two views of L_1L_2:
  - The set of all strings that can be made by concatenating a string in L_1 with a string in L_2.
  - The set of strings that can be split into two pieces: a piece from L_1 and a piece from L_2.

This is closely related to, but different than, the Cartesian product.

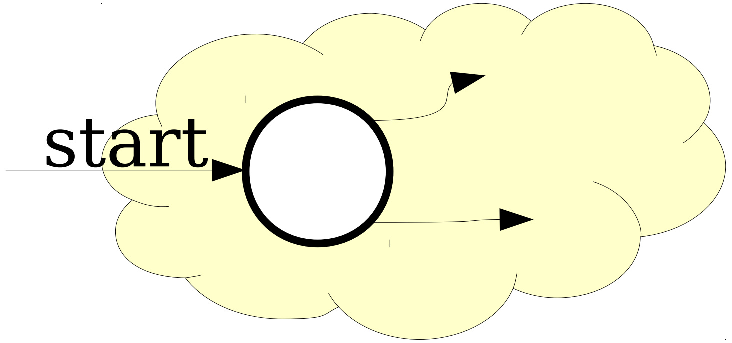
Question to ponder: In what ways are concatenations similar to Cartesian products? In what ways are they different?

Concatenating Regular Languages

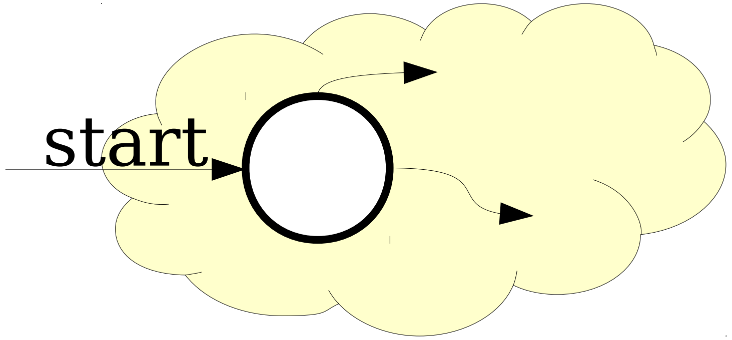
- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea*:**

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea:***



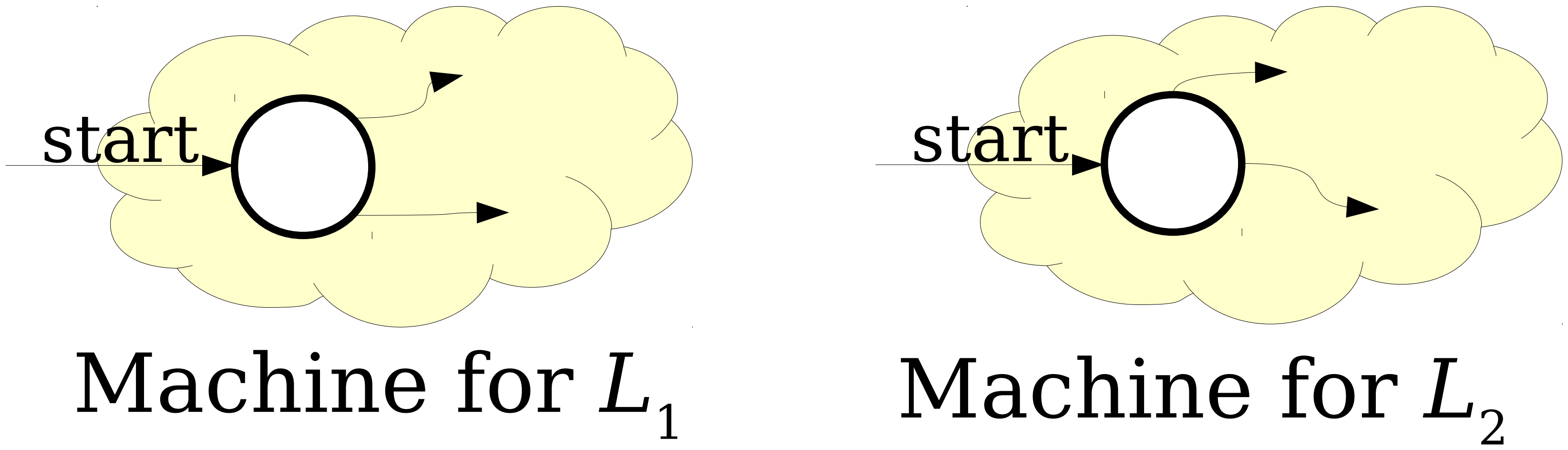
Machine for L_1



Machine for L_2

Concatenating Regular Languages

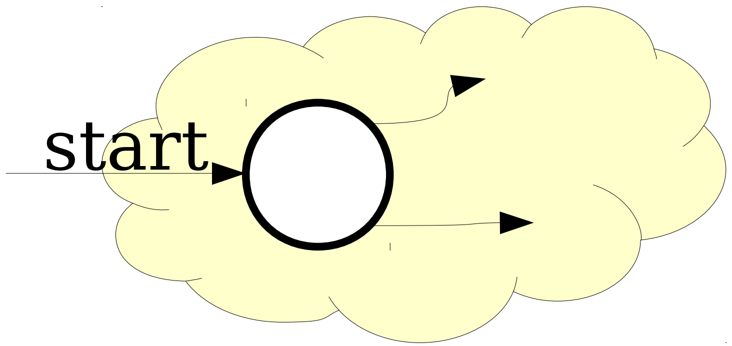
- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea:***



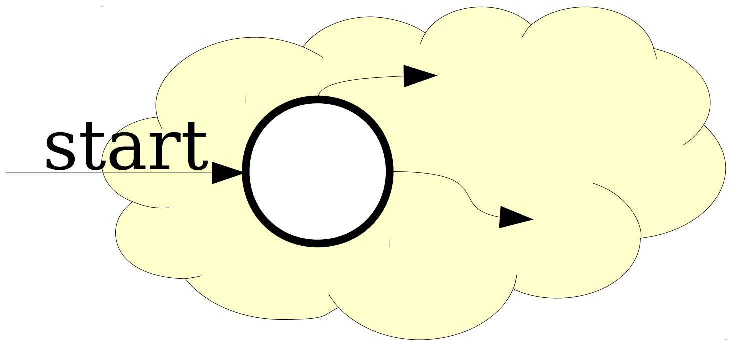
b	**o**	**o**	**k**	**k**	**e**	**e**	**p**	**e**	**r**

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea:***



Machine for L_1

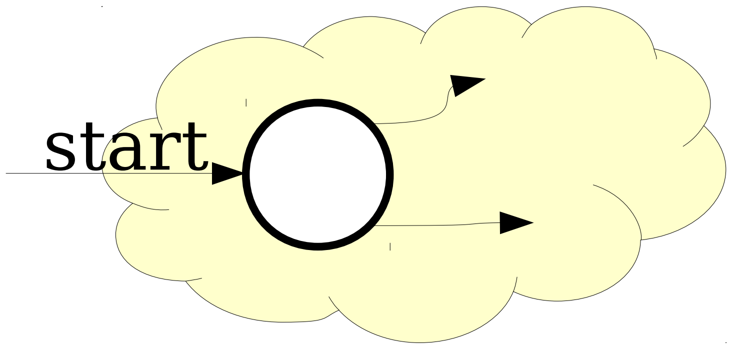


Machine for L_2

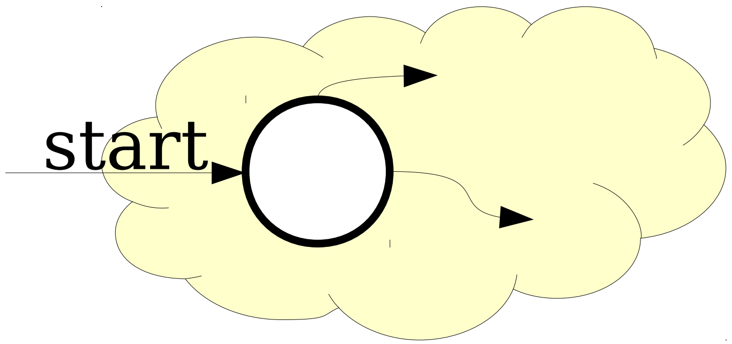
b	**o**	**o**	**k**	**k**	**e**	**e**	**p**	**e**	**r**

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea:***



Machine for L_1



Machine for L_2

b	**o**	**o**	**k**

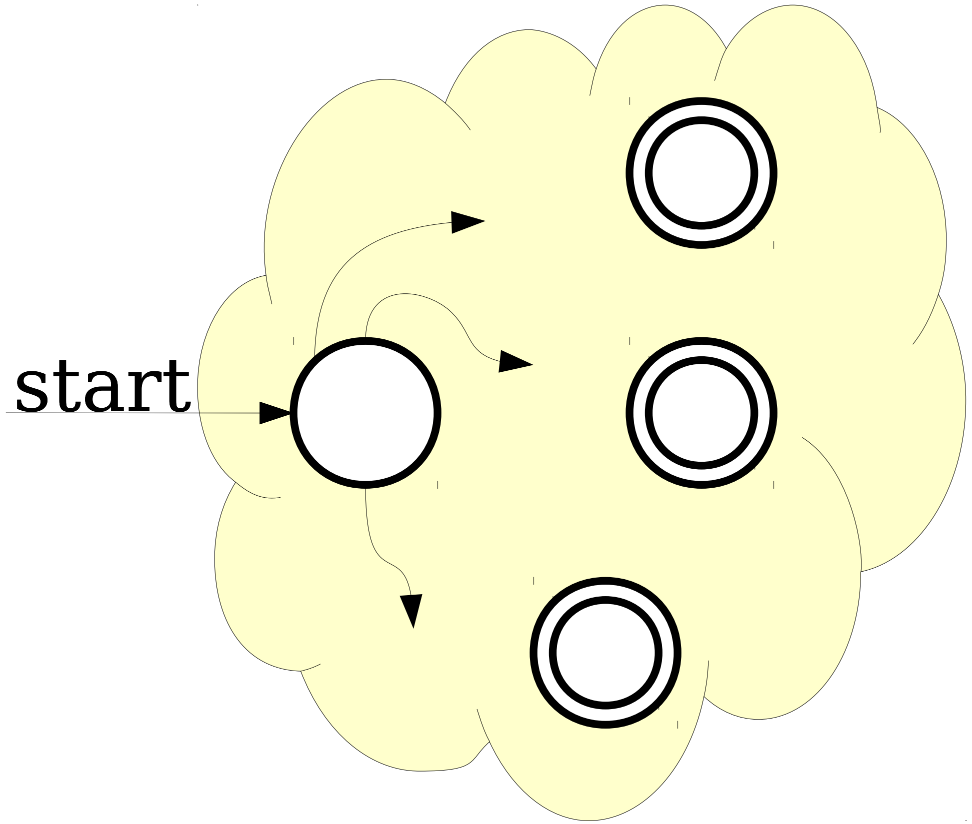
k	**e**	**e**	**p**	**e**	**r**

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea***:
  - Run a DFA/NFA for L_1 on w.
  - Whenever it reaches an accepting state, optionally hand the rest of w to a DFA/NFA for L_2.
  - If the automaton for L_2 accepts the rest, $w \in L_1L_2$.
  - If the automaton for L_2 rejects the remainder, the split was incorrect.

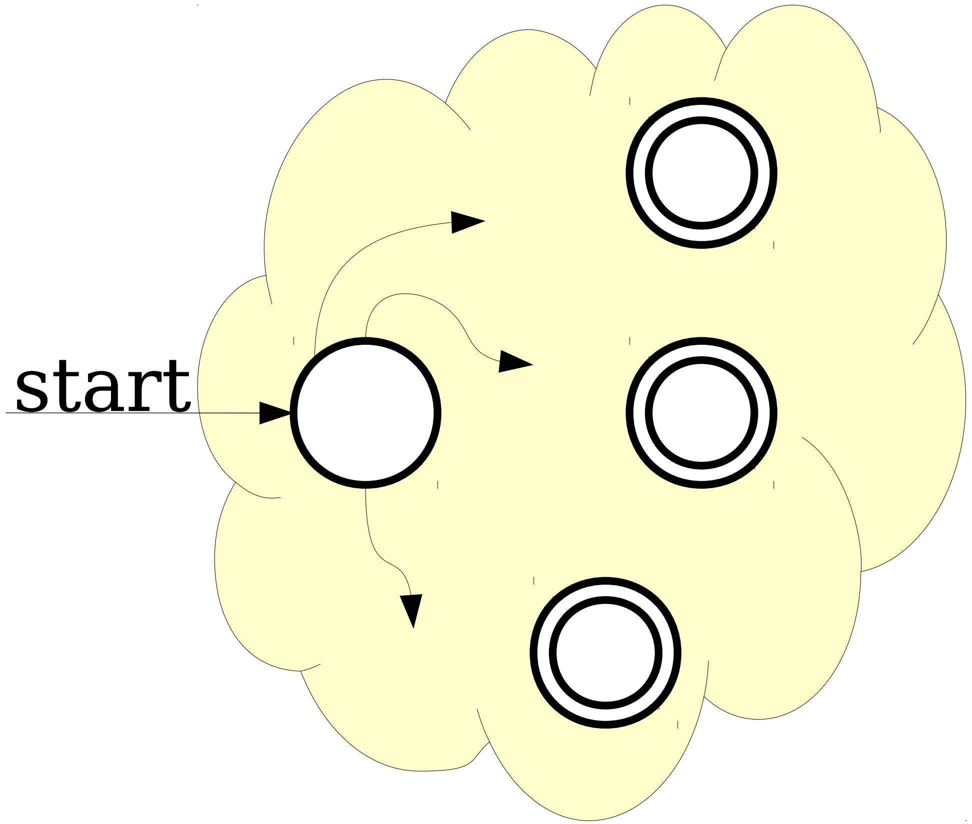
Concatenating Regular Languages

Concatenating Regular Languages

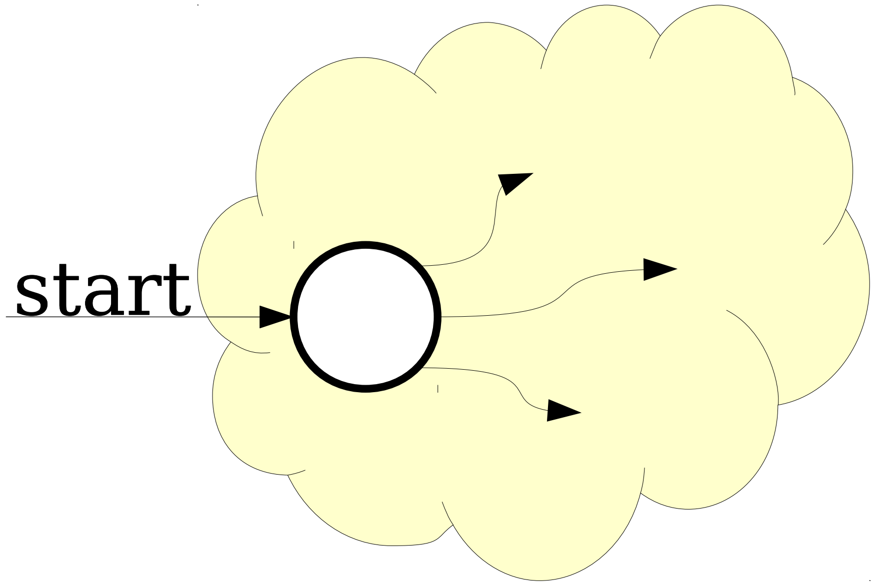


Machine for L_1

Concatenating Regular Languages

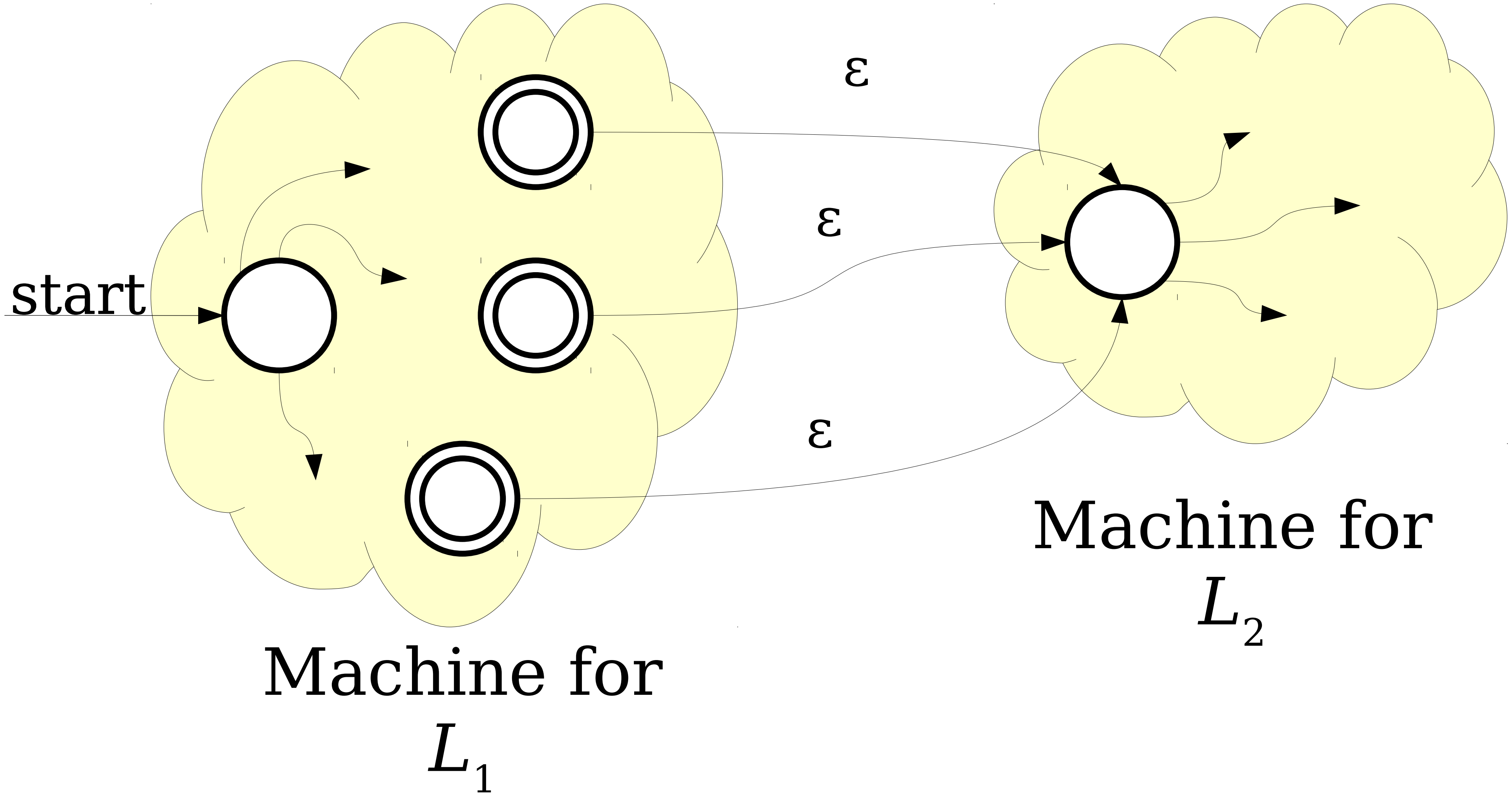


Machine for
L_1

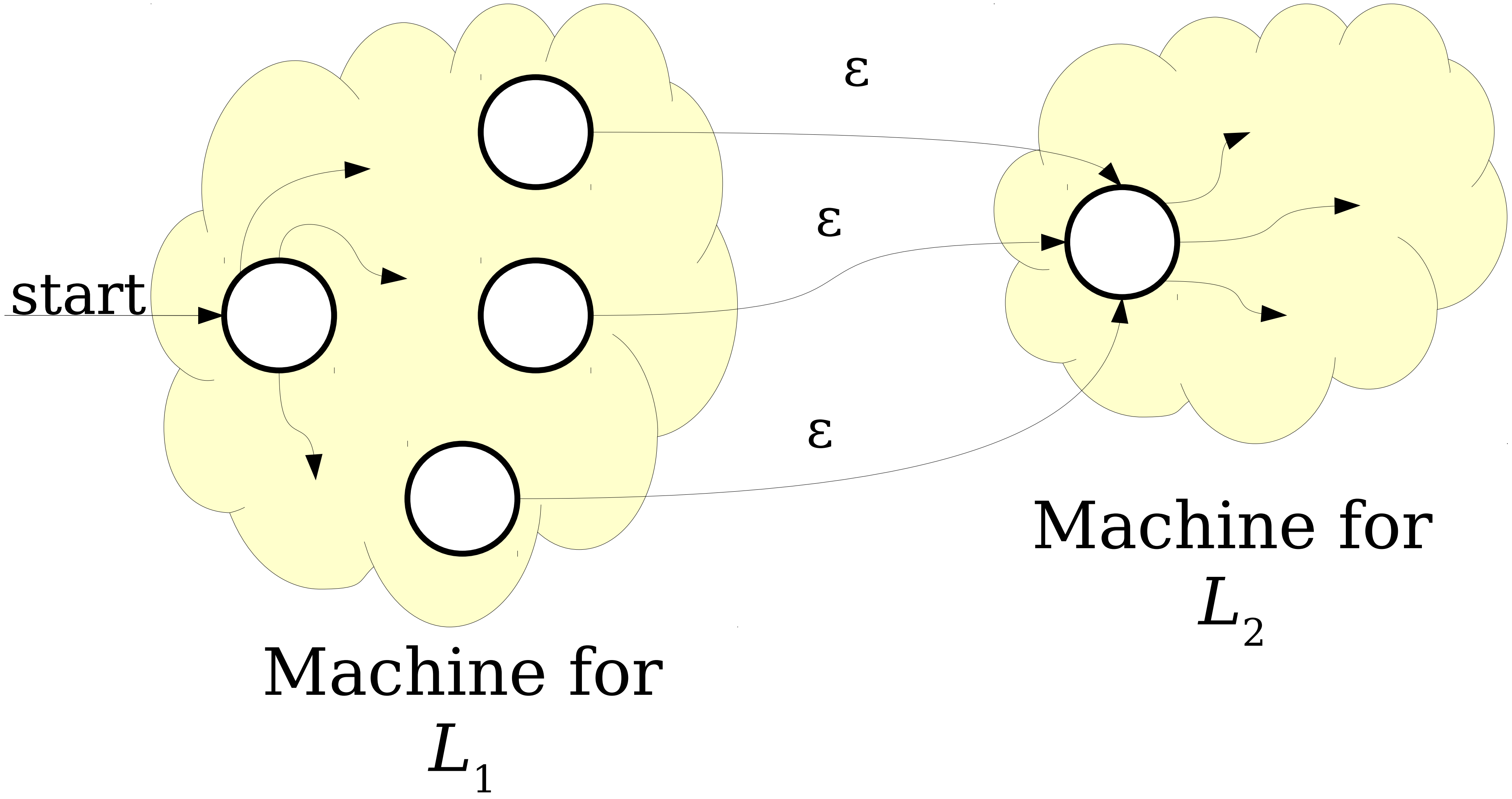


Machine for
L_2

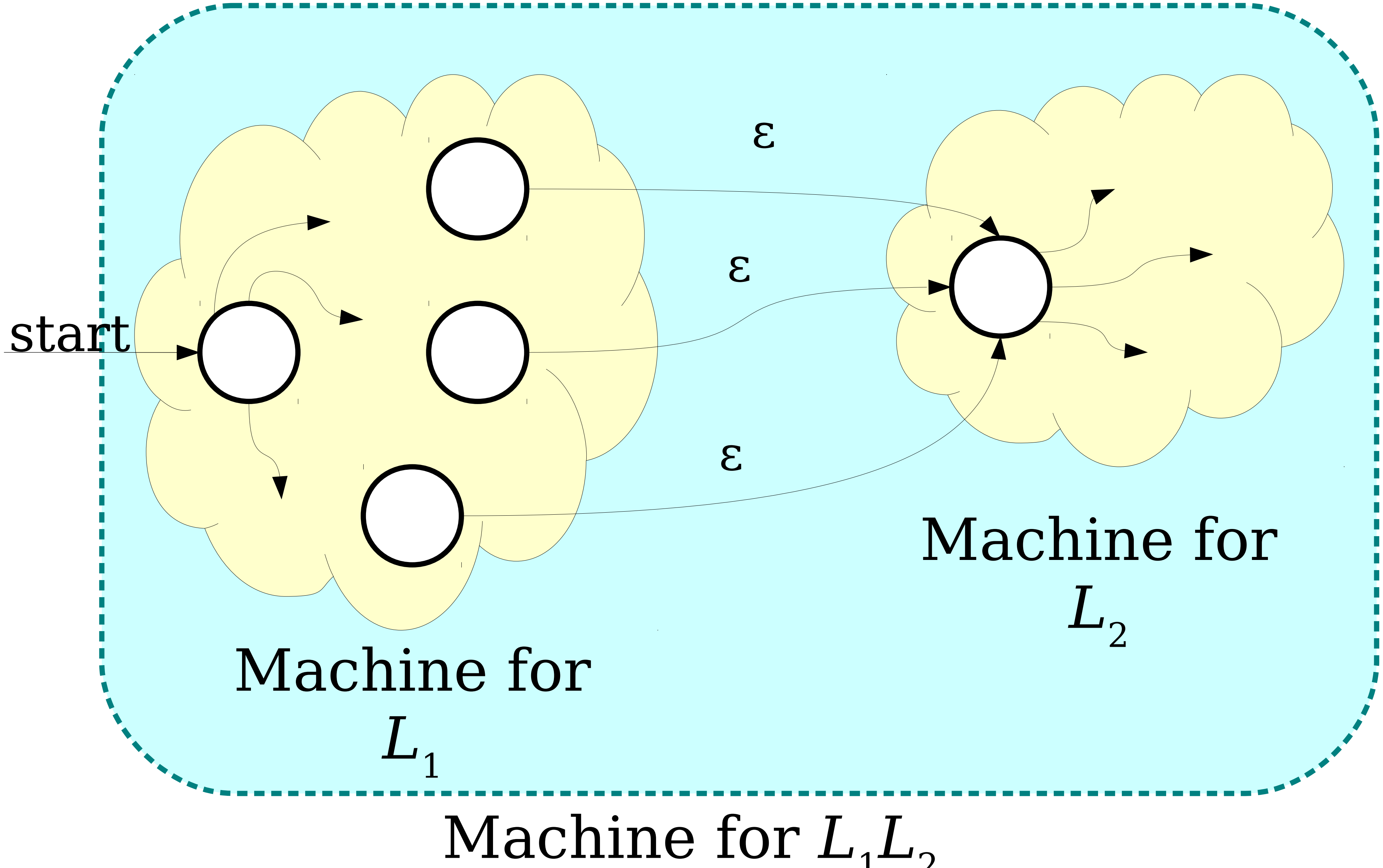
Concatenating Regular Languages



Concatenating Regular Languages



Concatenating Regular Languages



Lots and Lots of Concatenation

- Consider the language L = { **aa**, **b** }
- LL is the set of strings formed by concatenating pairs of strings in L.

{ **aaaa**, **aab**, **baa**, **bb** }

- LLL is the set of strings formed by concatenating triples of strings in L.

{ **aaaaaa**, **aaaab**, **aabaa**, **aabb**, **baaaa**, **baab**, **bbaa**, **bbb**}

- $LLLL$ is the set of strings formed by concatenating quadruples of strings in L.

{ **aaaaaaaa**, **aaaaaab**, **aaaabaa**, **aaaabb**, **aabaaaa**, **aabaab**, **aabbaa**, **aabbb**, **baaaaaa**, **baaaab**, **baabaa**, **baabb**, **bbaaaa**, **bbaab**, **bbbaa**, **bbbb**}

Language Exponentiation

- We can define what it means to "exponentiate" a language as follows:
- $L^0 = \{\varepsilon\}$
  - Intuition: The only string you can form by gluing no strings together is the empty string.
  - Notice that $\{\varepsilon\} \neq \emptyset$. Can you explain why?
- $L^{n+1} = LL^n$
  - Idea: Concatenating (n+1) strings together works by concatenating n strings, then concatenating one more.
- ***Question to ponder:*** Why define $L^0 = \{\varepsilon\}$?
- ***Question to ponder:*** What is $\emptyset^0$?

The Kleene Star

The Kleene Closure

- An important operation on languages is the ***Kleene Closure***, which is defined as

$$L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ w \in L^n \}$$

- Mathematically:

$$w \in L^* \quad \leftrightarrow \quad \exists n \in \mathbb{N}.\ w \in L^n$$

- Intuitively, L^* is the language all possible ways of concatenating zero or more strings in L together, possibly with repetition.
- ***Question to ponder:*** What is $\varnothing^*$?

The Kleene Closure

If L = { **a**, **bb** }, then L^* = {

ε,

a, **bb**,

aa, **abb**, **bba**, **bbbb**,

aaa, **aabb**, **abba**, **abbbb**, **bbaa**, **bbabb**, **bbbba**, **bbbbbb**,

…

}

Think of L^* as the set of strings you can make if you have a collection of stamps – one for each string in *L* – and you form every possible string that can be made from those stamps.

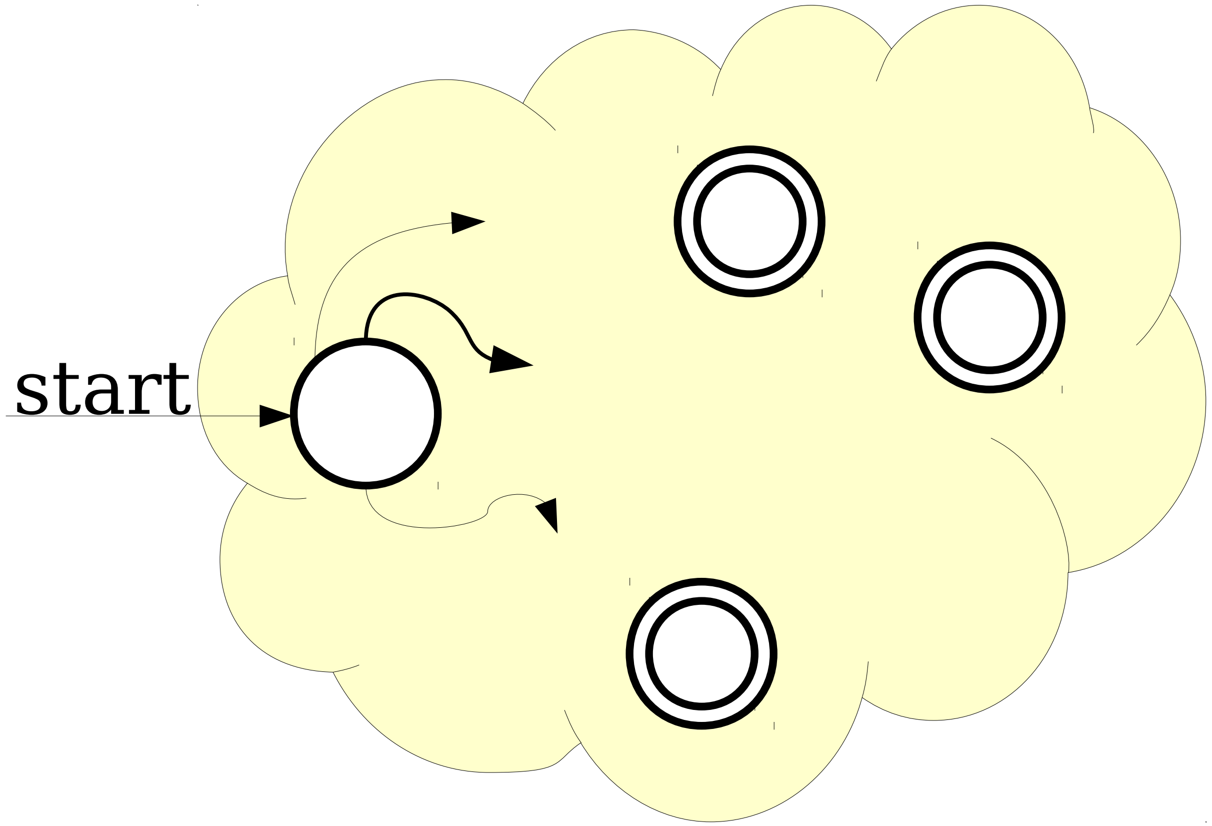
Reasoning about Infinity

- If L is regular, is L^* necessarily regular?
- **⚠ A Bad Line of Reasoning: ⚠**
  - $L^0 = \{ \varepsilon \}$ is regular.
  - $L^1 = L$ is regular.
  - $L^2 = LL$ is regular
  - $L^3 = L(LL)$ is regular
  - …
  - Regular languages are closed under union.
  - So the union of all these languages is regular.

We won't get into the reasons why, but infinity just doesn't work this way where it neatly extends from the combination of a bunch of finite things.

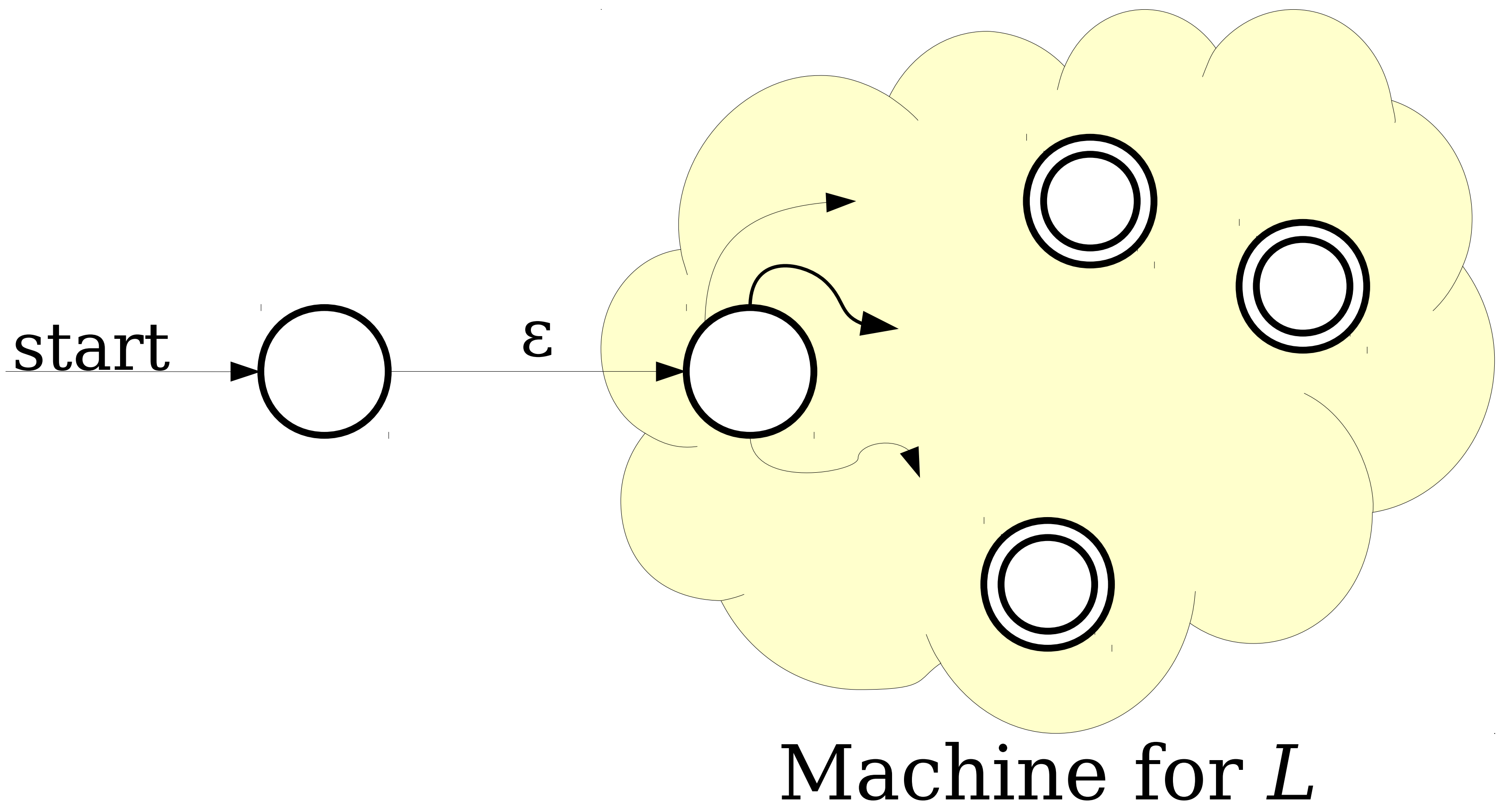
Idea: Can we directly convert an NFA for language L to an NFA for language L^*?

The Kleene Star

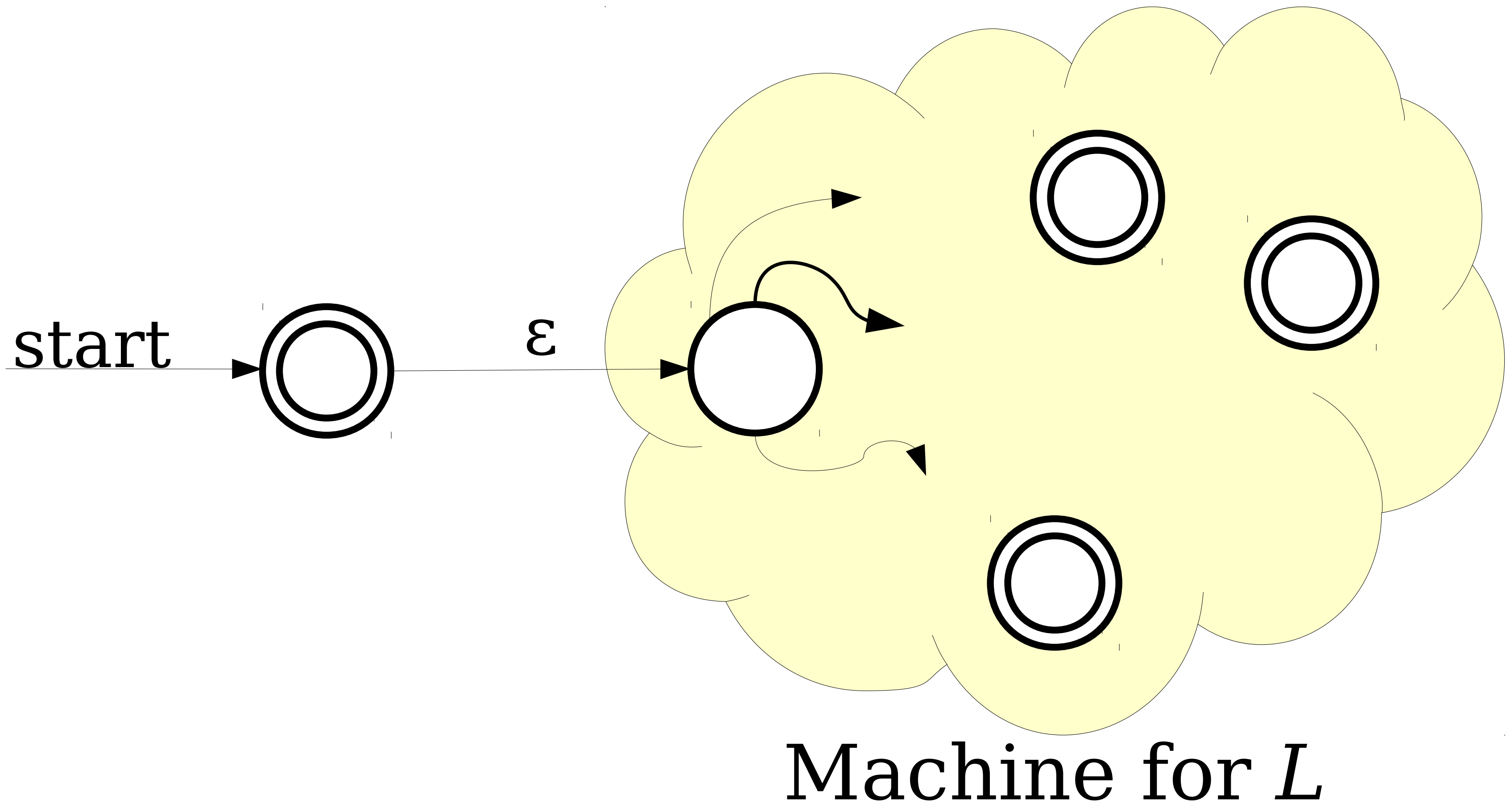


Machine for L

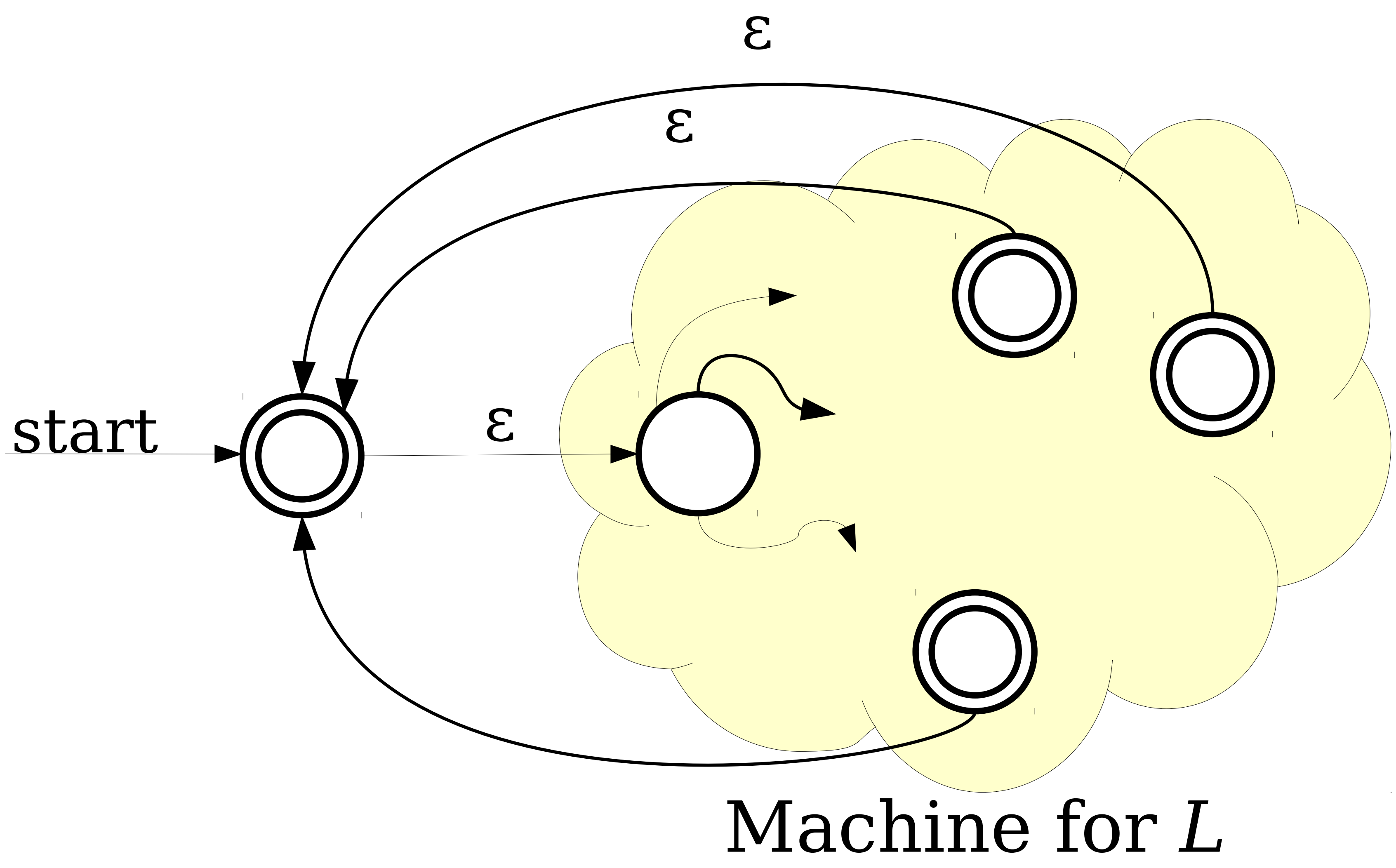
The Kleene Star



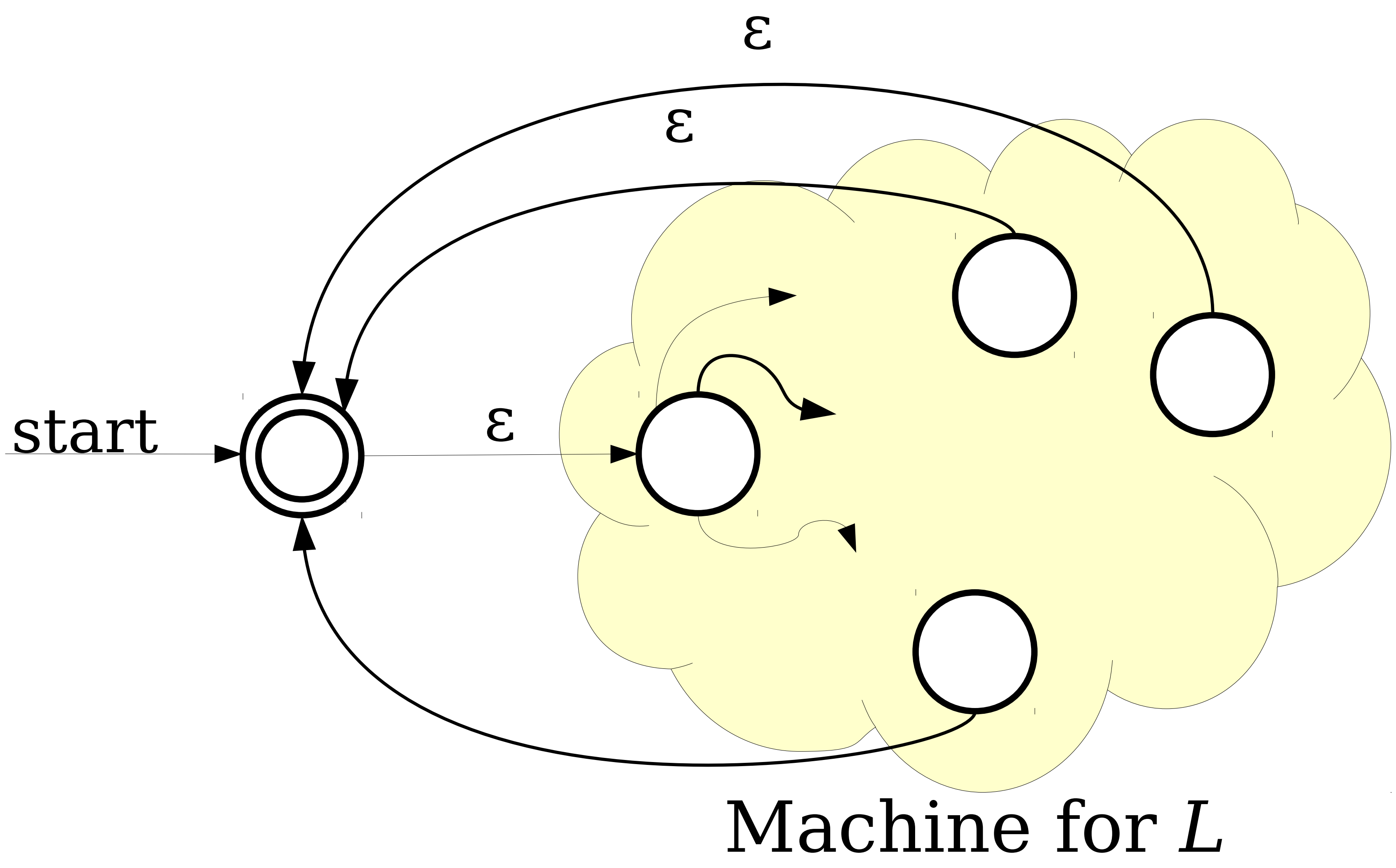
The Kleene Star



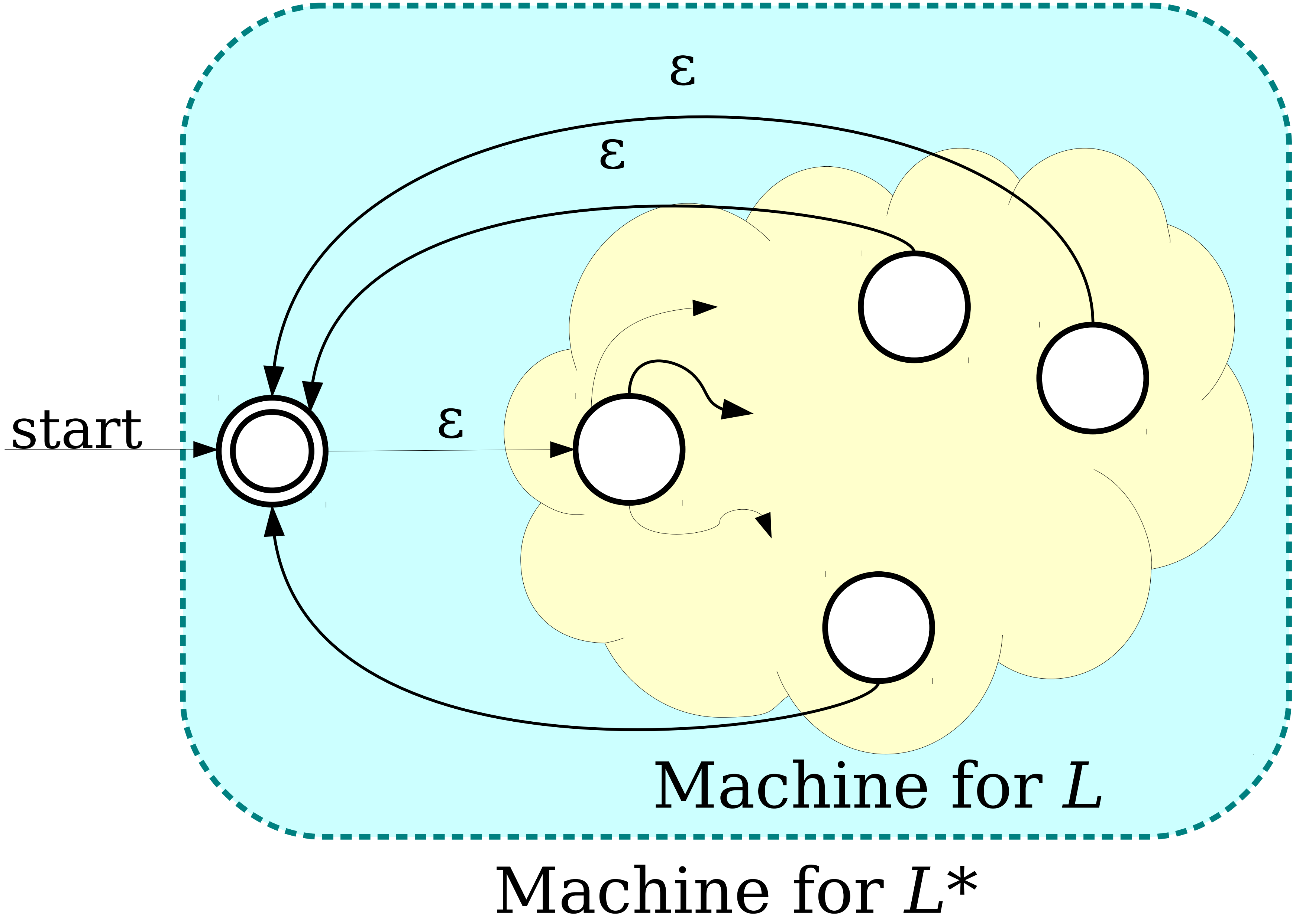
The Kleene Star



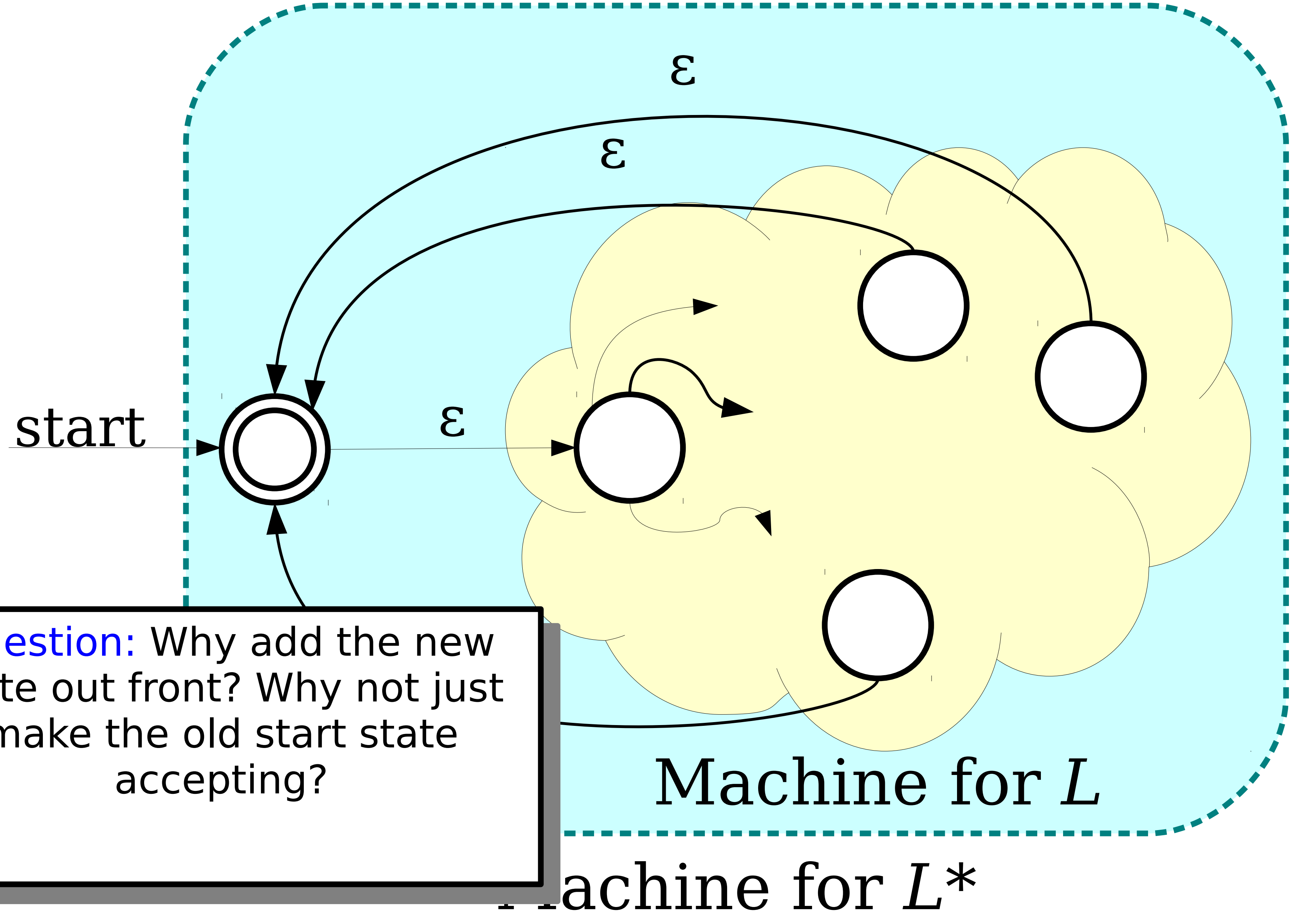
The Kleene Star



The Kleene Star



The Kleene Star



Question: Why add the new state out front? Why not just make the old start state accepting?

Closure Properties

- ***Theorem:*** If L_1 and L_2 are regular languages over an alphabet Σ, then so are the following languages:
  - $\overline{L_1}$
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - L_1L_2
  - L_1^*
- These properties are called ***closure properties of the regular languages***.

Next Time

- ***Regular Expressions***
  - Building languages from the ground up!
- ***Thompson's Algorithm***
  - A UNIX Programmer in Theoryland.
- ***Kleene's Theorem***
  - From machines to programs!