Finite Automata

Part Two

Recap from Last Time

Strings

- An ***alphabet*** is a finite set of symbols called ***characters***.
  - Typically, we use the symbol **Σ** to refer to an alphabet.
- A ***string over an alphabet Σ*** is a finite sequence of characters drawn from Σ.
- Example: If Σ = {**a**, **b**}, some valid strings over Σ include

a

aabaaabbabaaabaaaabbb

abbababba

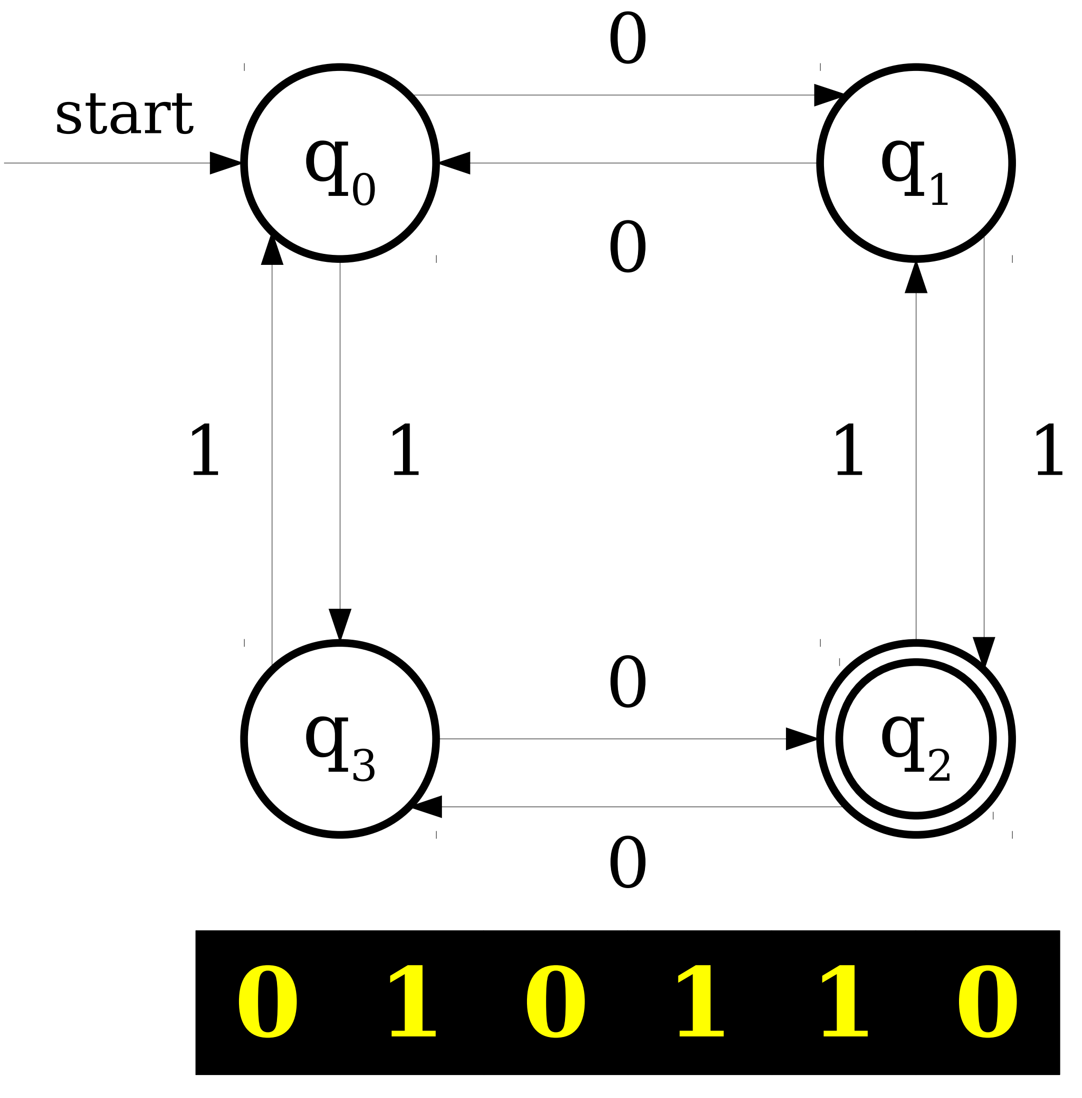
- The ***empty string*** contains no characters and is denoted **ε**.

Languages

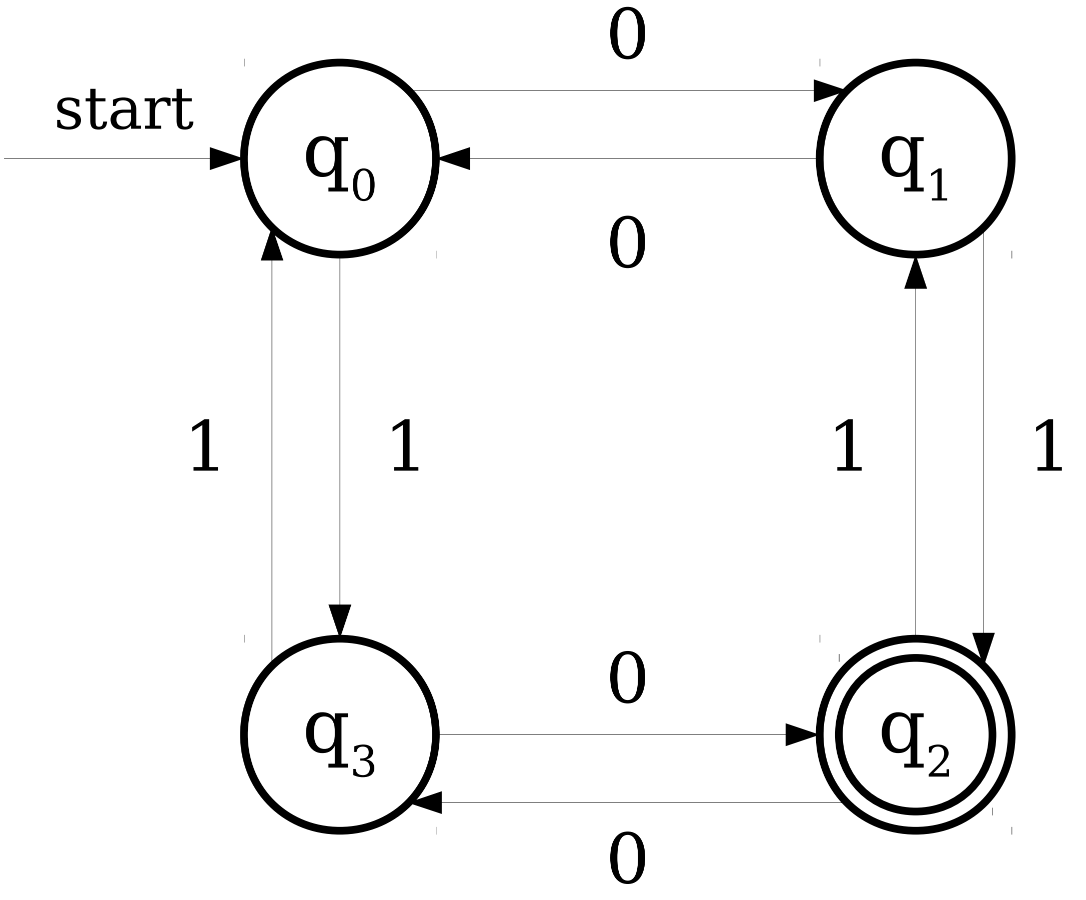
- A ***formal language*** is a set of strings.
- We say that L is a ***language over Σ*** if it is a set of strings over Σ.
- Example: The language of palindromes over $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ is the set

  $\{\varepsilon, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{aa}, \mathbf{bb}, \mathbf{cc}, \mathbf{aaa}, \mathbf{aba}, \mathbf{aca}, \mathbf{bab}, \dots\}$
- The set of all strings composed from letters in Σ is denoted **Σ***.
- Formally: L is a language over Σ iff $L \subseteq \Sigma^*$.

A Simple Finite Automaton



A Simple Finite Automaton



1	**0**	**1**	**0**	**0**	**0**

The ***language of an automaton*** is the set of strings that it accepts.

If D is an automaton, we denote the language of D as $\mathscr{L}(\boldsymbol{D})$.

$$\mathscr{L}(D) = \{ \ w \in \Sigma^* \mid D \text{ accepts } w \ \}$$

DFAs

- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- DFAs are the simplest type of automaton that we will see in this course.

DFAs, Informally

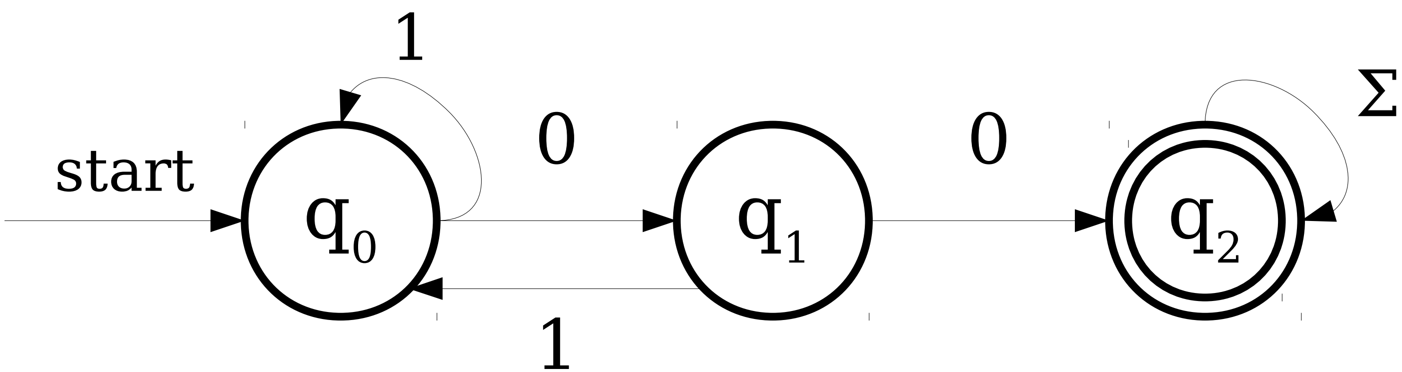
- A DFA is defined relative to some alphabet Σ.
- For each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ.
    - This is the "deterministic" part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

Designing DFAs

- At each point in its execution, the DFA can only remember what state it is in.
- **DFA Design Tip:** Build each state to correspond to some piece of information you need to remember.
  - Each state acts as a “memento” of what you're supposed to do next.
  - Only finitely many different states ≈ only finitely many different things the machine can remember.

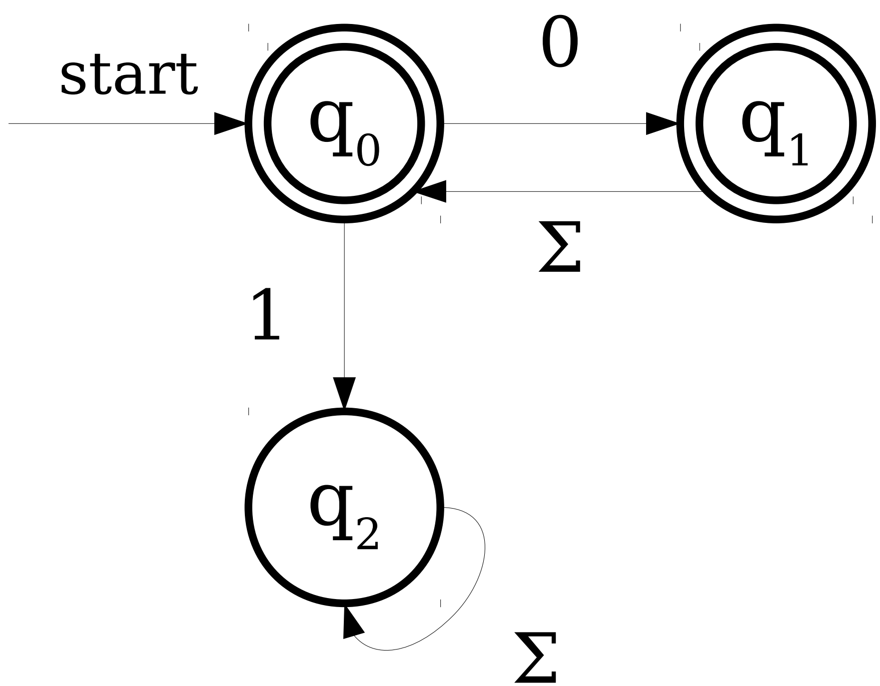
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ contains } \texttt{00} \text{ as a substring } \}$



Recognizing Languages with DFAs

$L = \{\ w \in \{$`0`$,$ `1`$\}^* \mid$ every other character of w, starting with the first character, is `0` $\}$



More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$

Suppose the alphabet is

$\Sigma = \{$ **a, *, /** $\}$

Try designing a DFA for comments!

Some test cases:

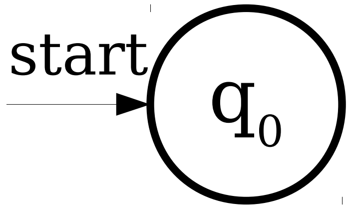
ACCEPTED	REJECTED
`/*a*/`	`/**`
`/**/`	`/**/a/*aa*/`
`/***/`	`aaa/**/`
`/*aaa*aaa*/`	`/*/`

More Elaborate DFAs

$L = \{ \ w \mid w \text{ is a C-style comment } \}$

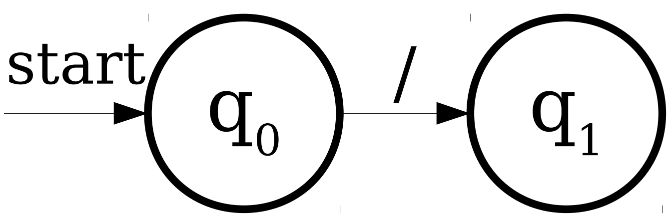
More Elaborate DFAs

$L = \{ \ w \mid w \text{ is a C-style comment } \}$



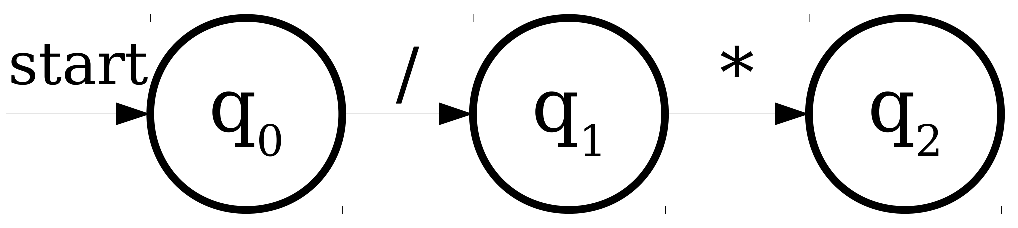
More Elaborate DFAs

$L = \{ \ w \mid w \text{ is a C-style comment } \}$



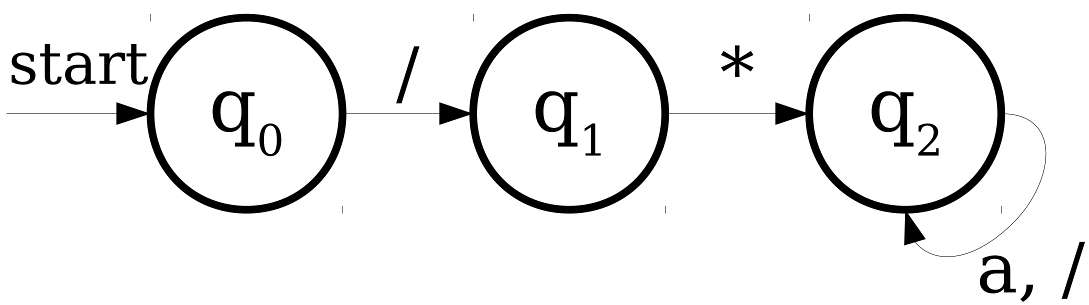
More Elaborate DFAs

$L = \{ \; w \mid w \text{ is a C-style comment } \}$



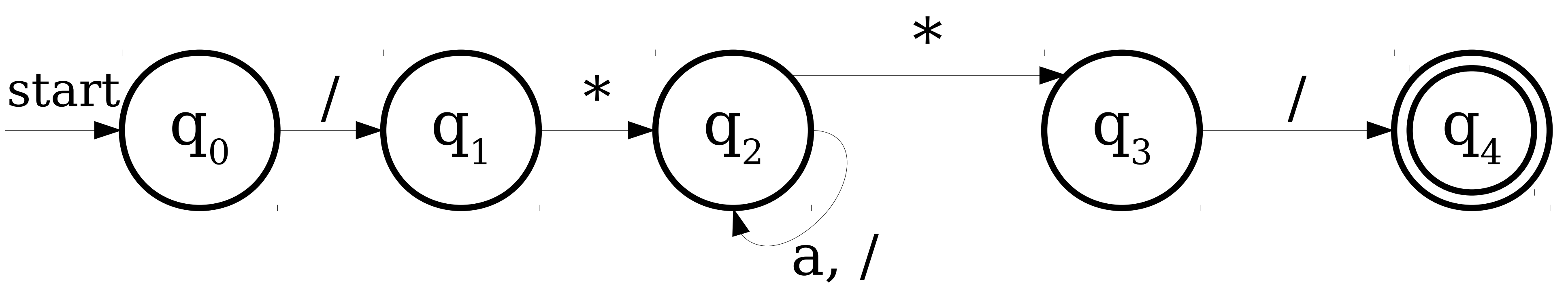
More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$



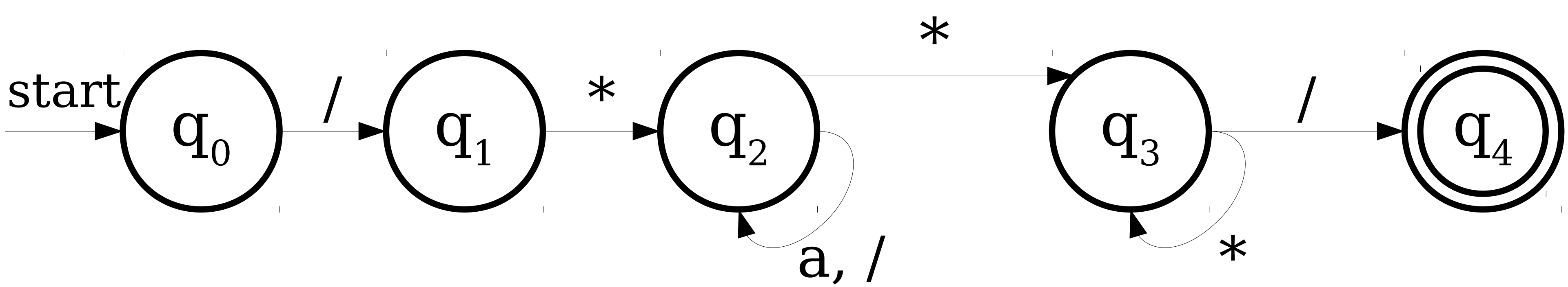
More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$



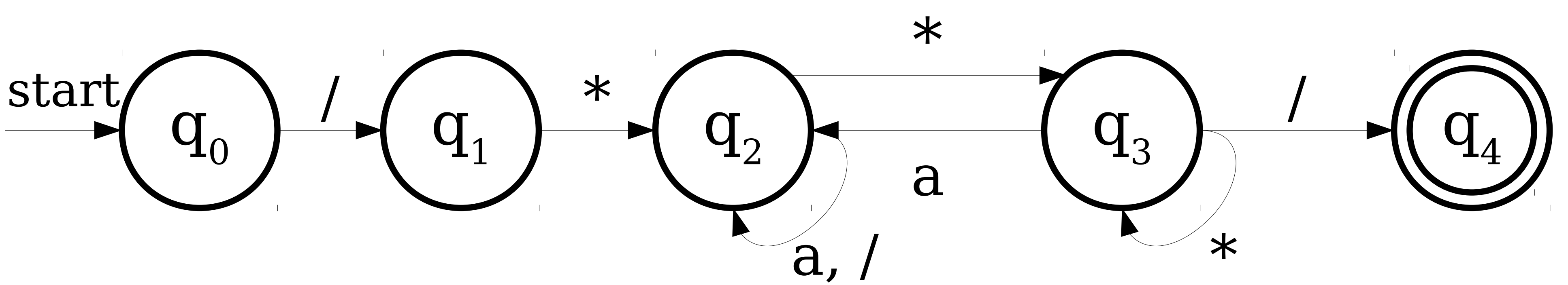
More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$



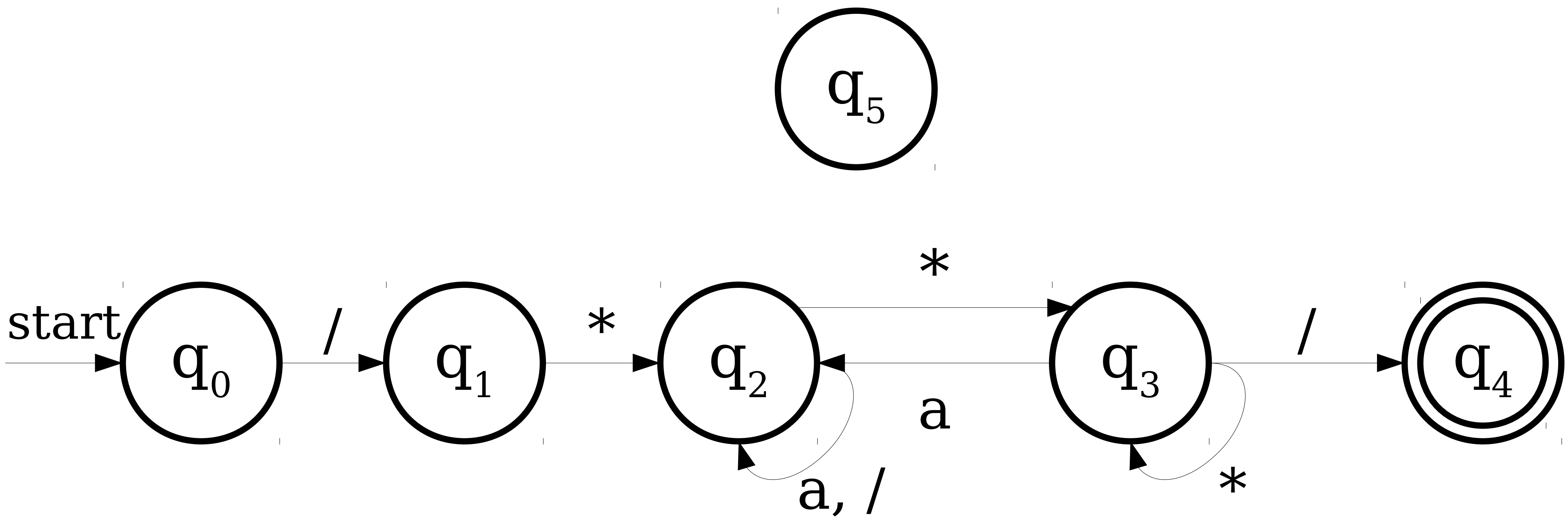
More Elaborate DFAs

$L = \{\ w \mid w \text{ is a C-style comment }\}$



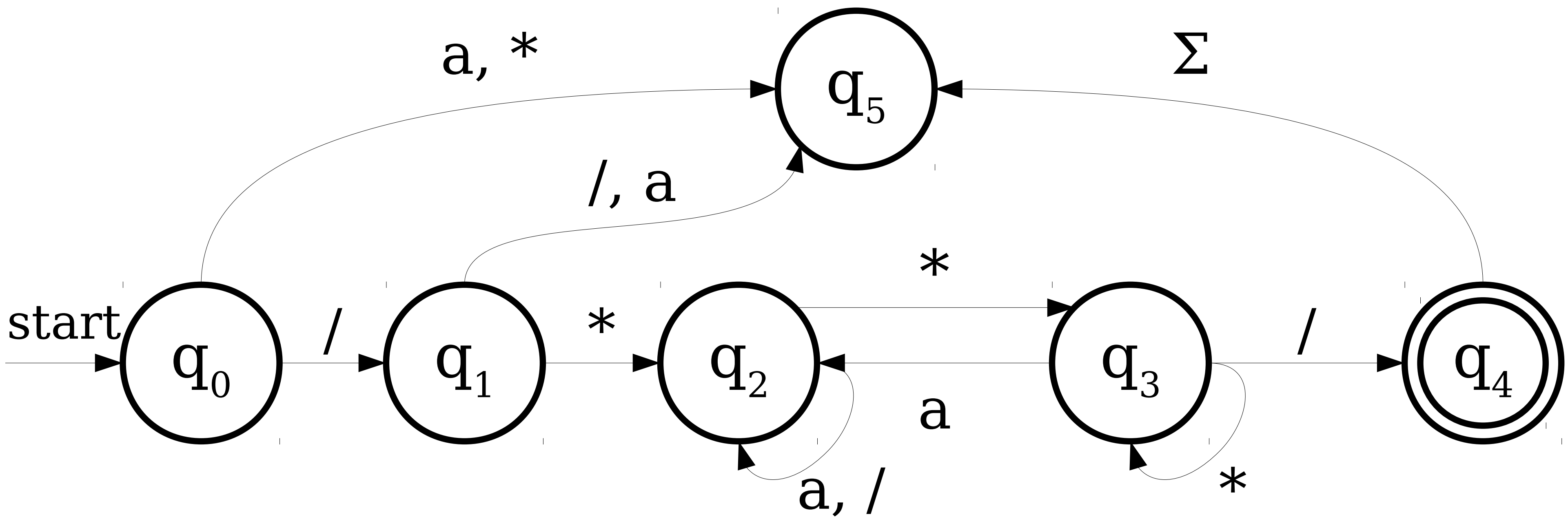
More Elaborate DFAs

$L = \{\ w \mid w \text{ is a C-style comment }\ \}$



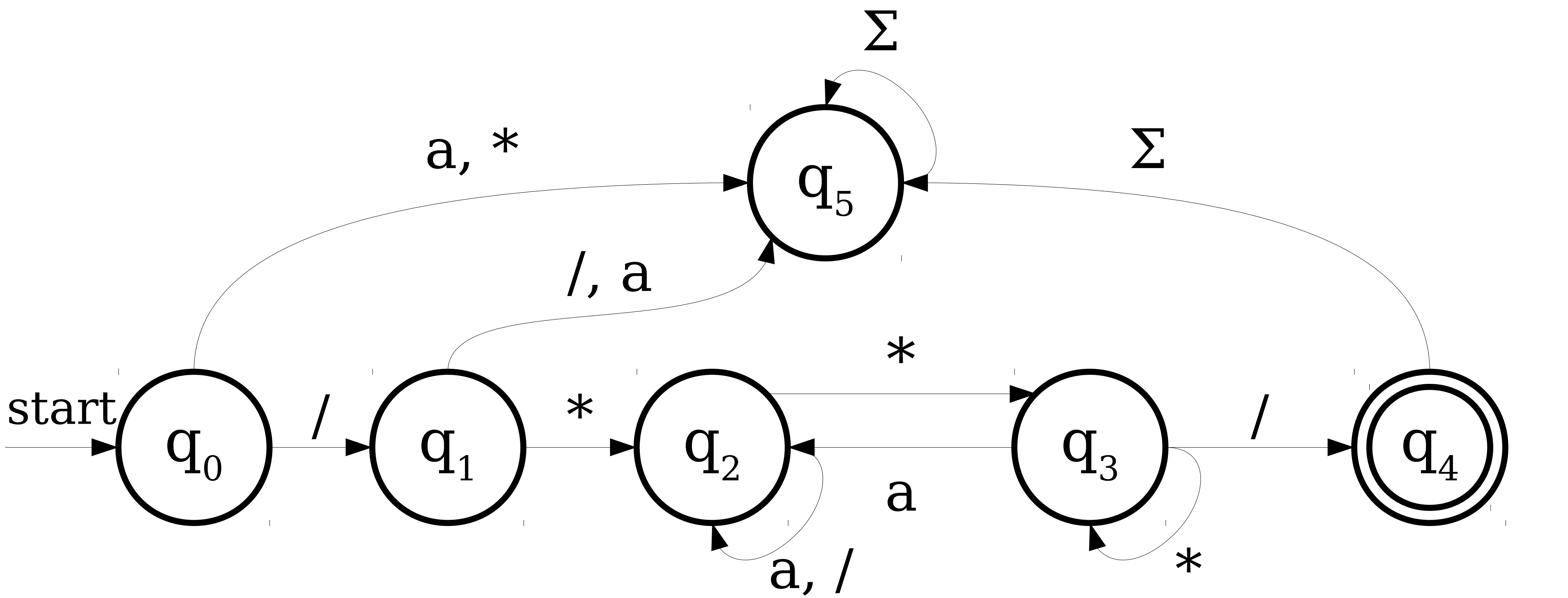
More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$

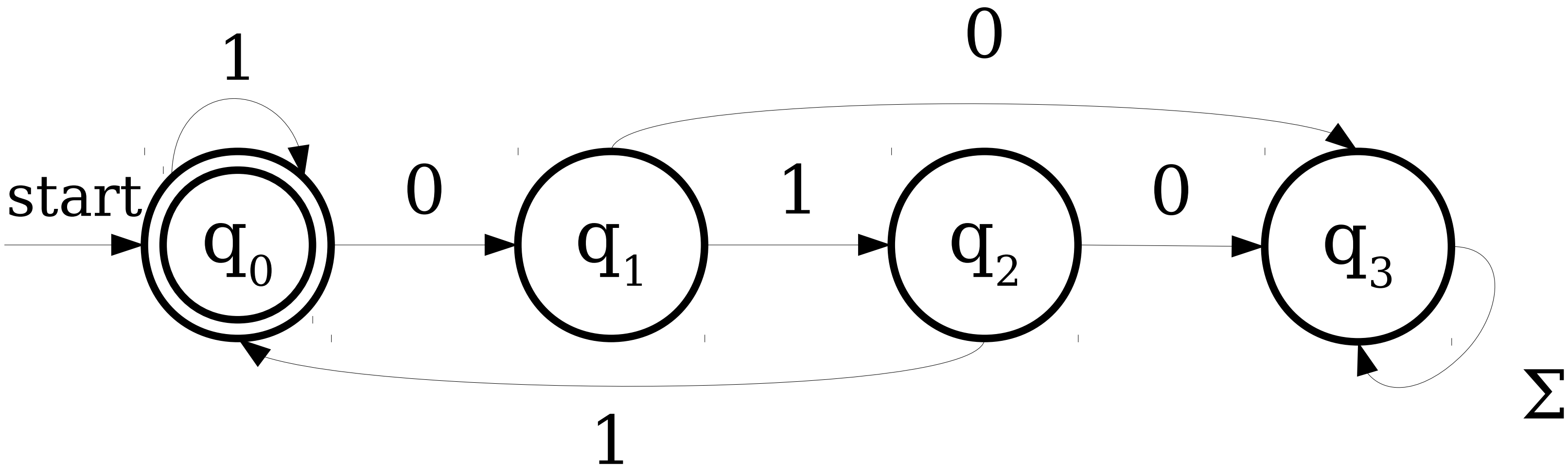


More Elaborate DFAs

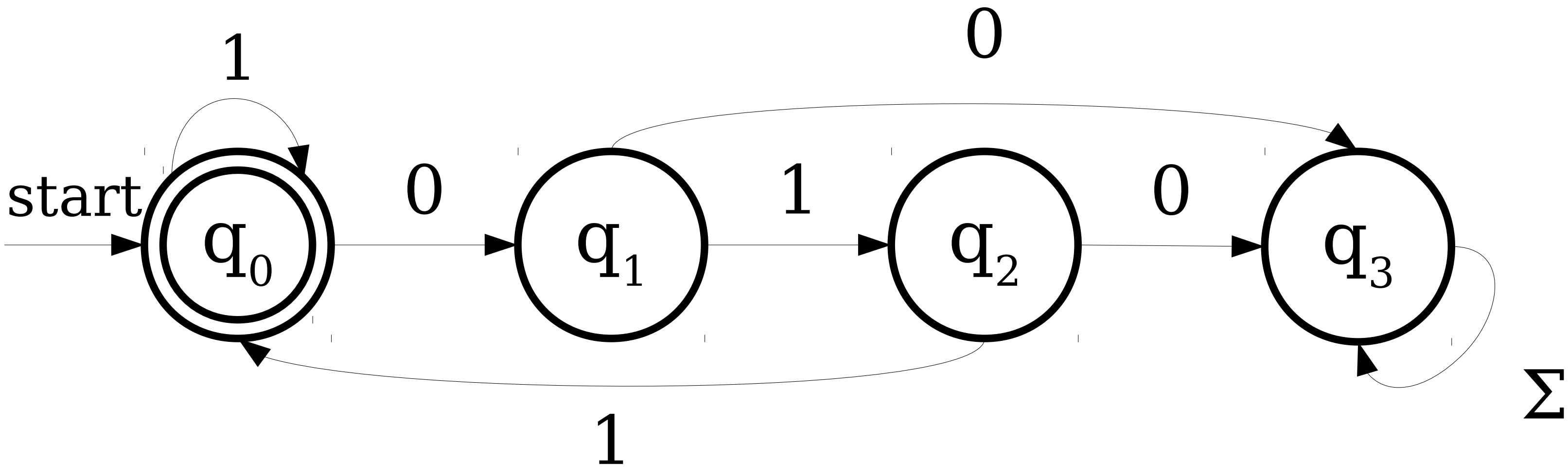
$L = \{ \ w \mid w \text{ is a C-style comment } \}$



Tabular DFAs

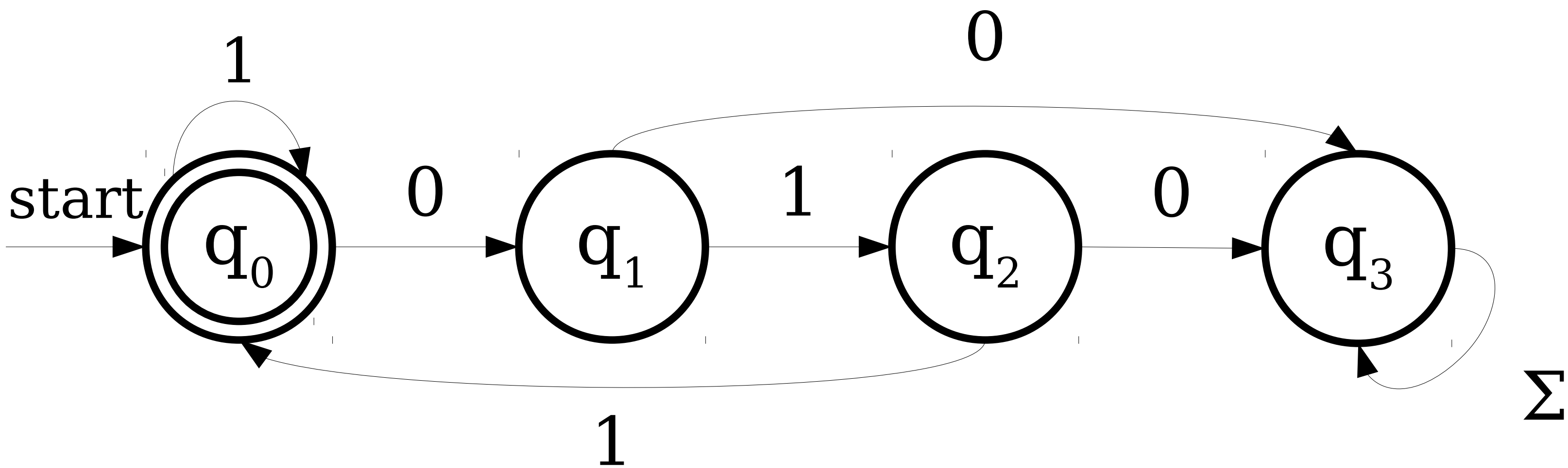


Tabular DFAs



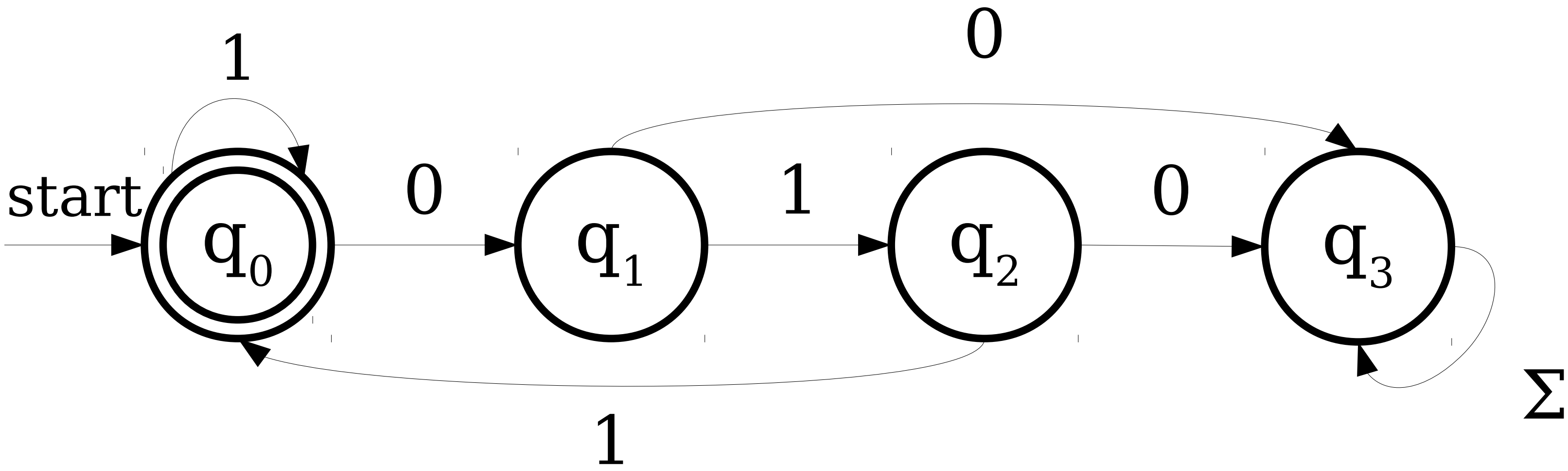
	0	1
q_0		
q_1		
q_2		
q_3		

Tabular DFAs



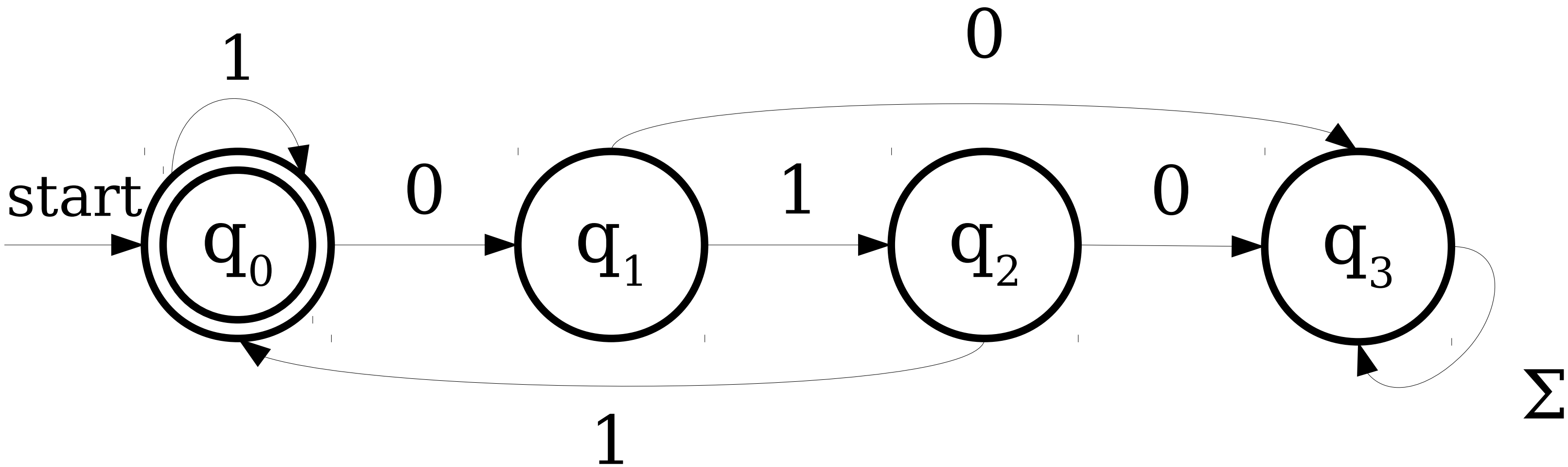
	0	1
q_0	q_1	
q_1		
q_2		
q_3		

Tabular DFAs



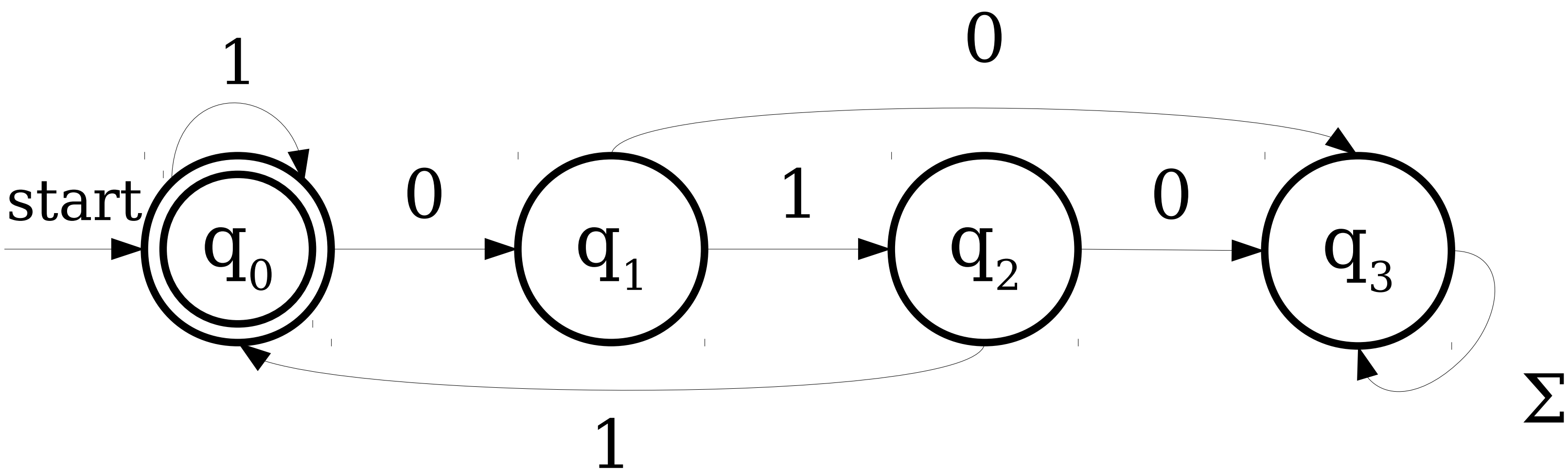
	0	1
q_0	q_1	q_0
q_1		
q_2		
q_3		

Tabular DFAs



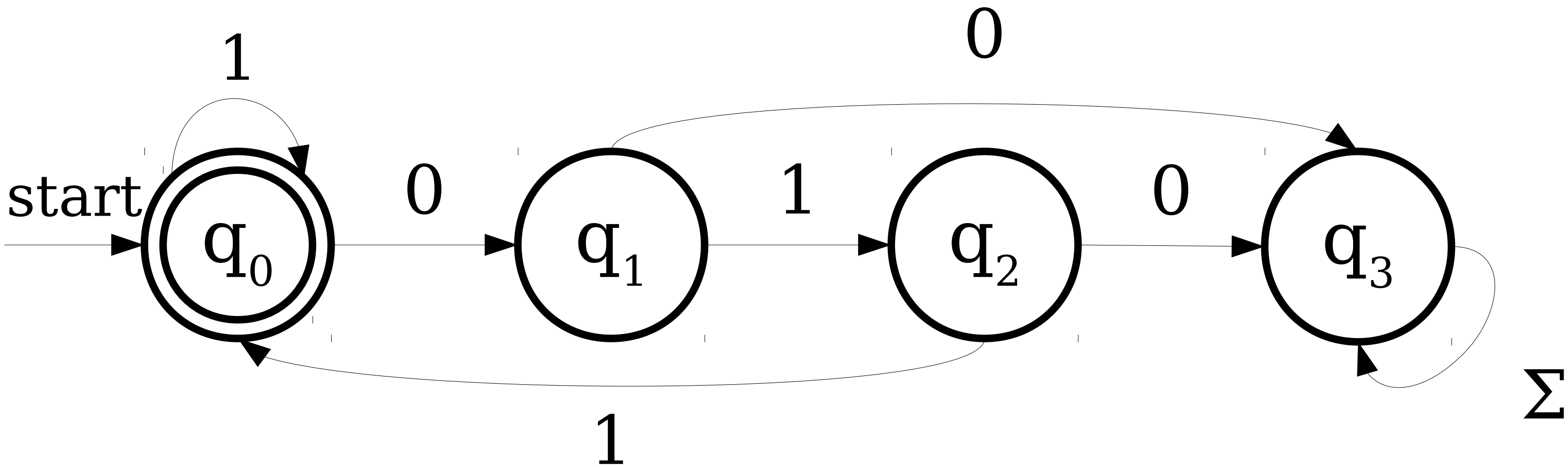
	0	1
q_0	q_1	q_0
q_1	q_3	
q_2		
q_3		

Tabular DFAs



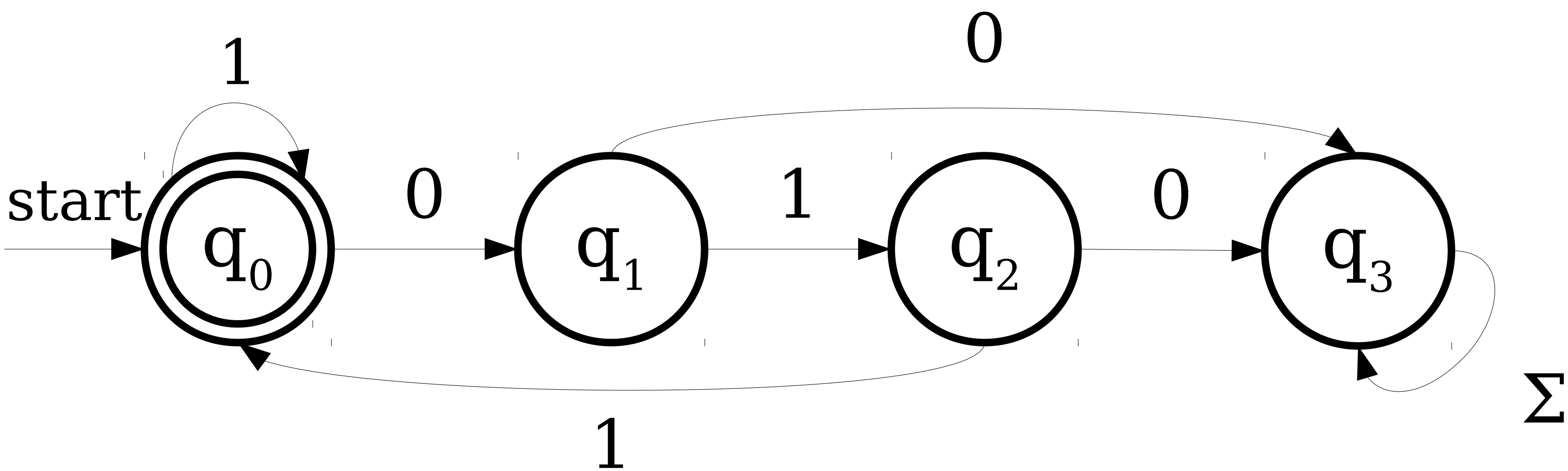
	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2		
q_3		

Tabular DFAs



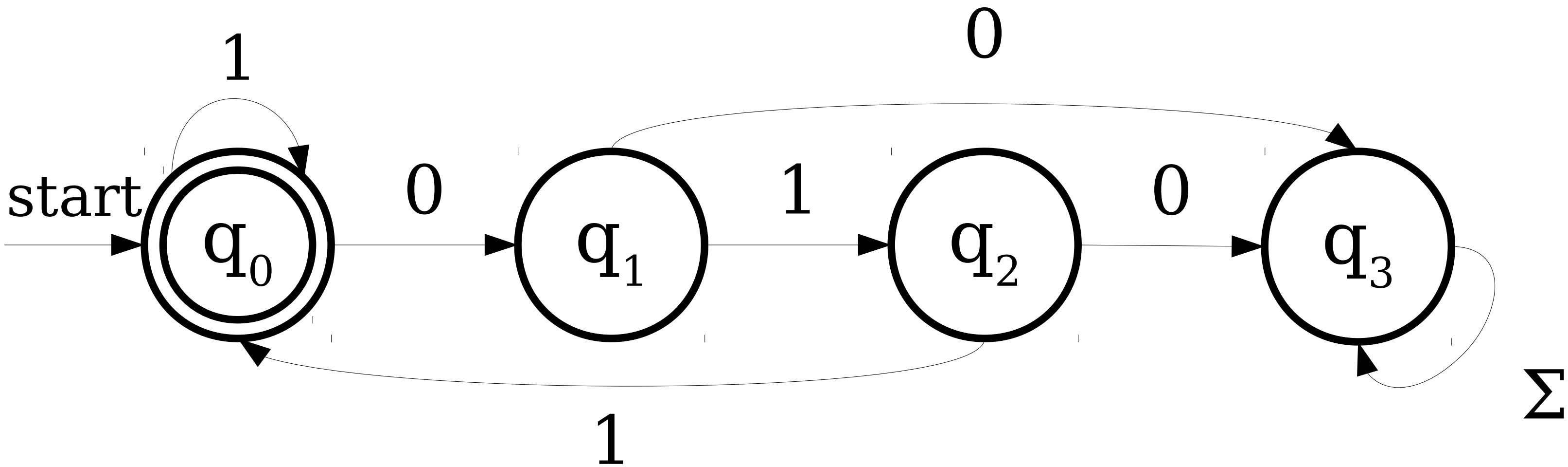
	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	
q_3		

Tabular DFAs



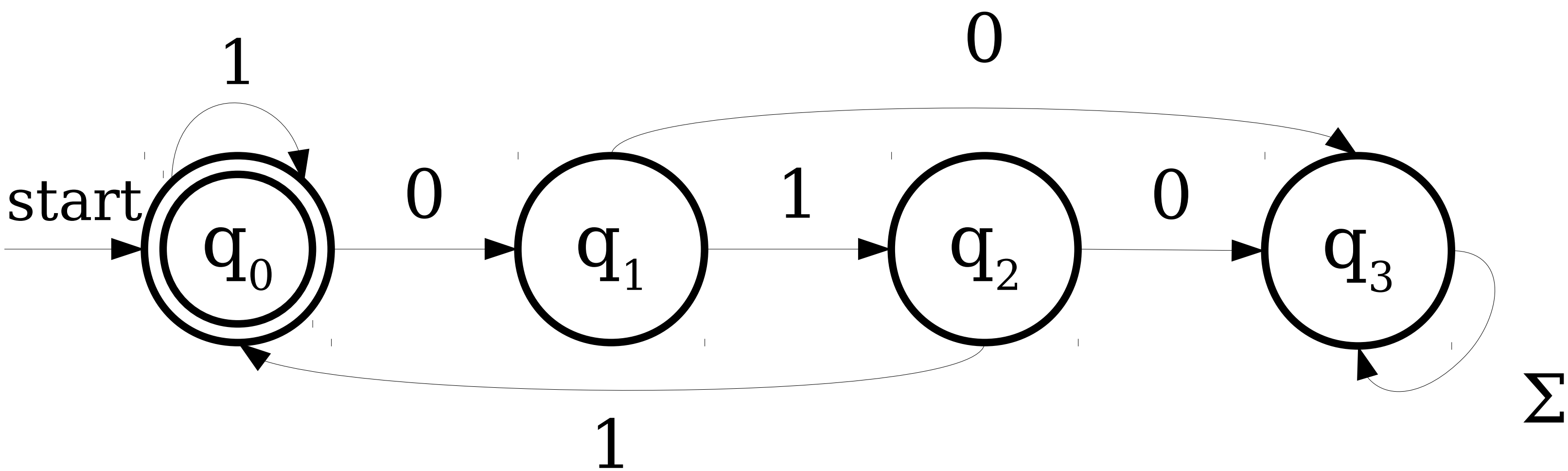
	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3		

Tabular DFAs



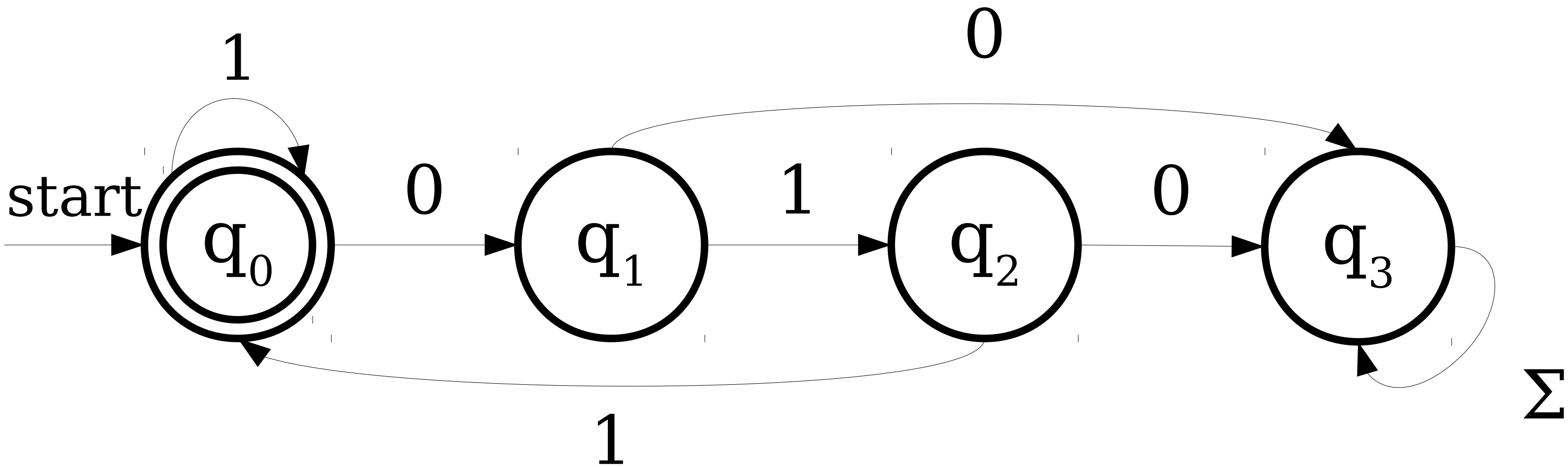
	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3	q_3	

Tabular DFAs



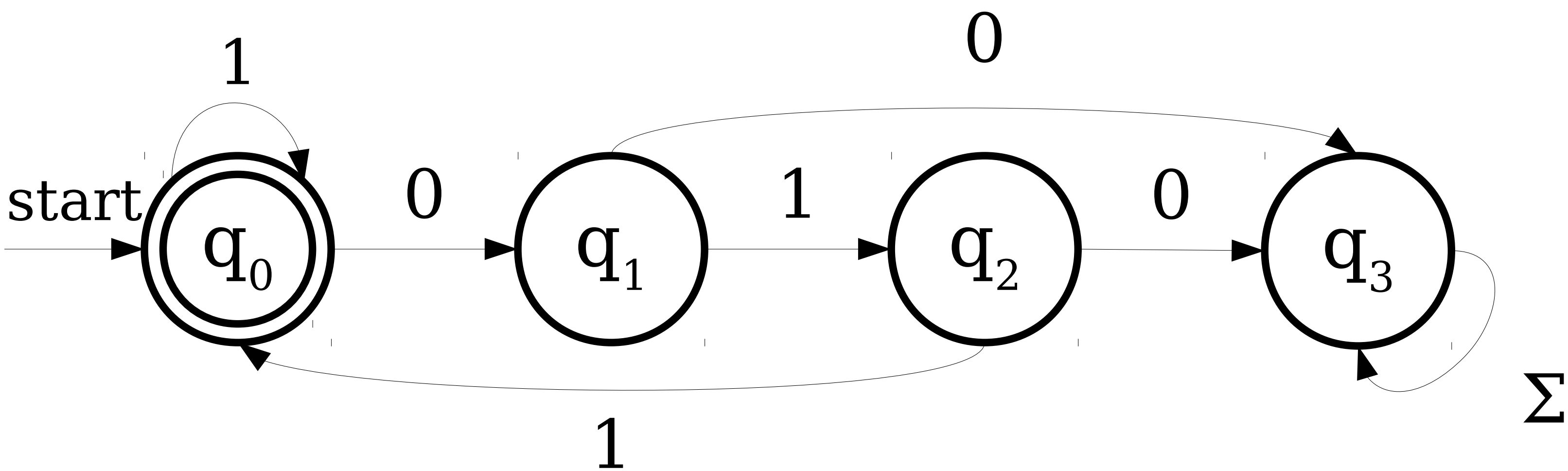
	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3	q_3	q_3

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

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

The star indicates that this is an accepting state.

Code? In a Theory Course?

```
int kTransitionTable[kNumStates][kNumSymbols] = {
     {0, 0, 1, 3, 7, 1, …},
      …
};
bool kAcceptTable[kNumStates] = {
    false,
    true,
    true,
    …
};
bool SimulateDFA(string input) {
    int state = 0;
    for (char ch: input)
        state = kTransitionTable[state][ch];
    return kAcceptTable[state];
}
```

The Regular Languages

A language L is called a ***regular language*** if there exists a DFA D such that $\mathscr{L}(D) = L$.

The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language (denoted $\overline{L}$) is the language of all strings in Σ^* not in L.
- Formally:

$$\overline{L} = \{\ w \mid w \in \Sigma^* \land w \notin L\ \}$$

The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language (denoted $\overline{L}$) is the language of all strings in Σ^* not in L.
- Formally:

$$\overline{L} = \boldsymbol{\{\ w \mid w \in \Sigma^* \land w \notin L\ \}}$$

The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language (denoted $\overline{L}$) is the language of all strings in Σ^* not in L.
- Formally:

$$\overline{L} = \boldsymbol{\Sigma^* - L}$$

The Complement of a Language

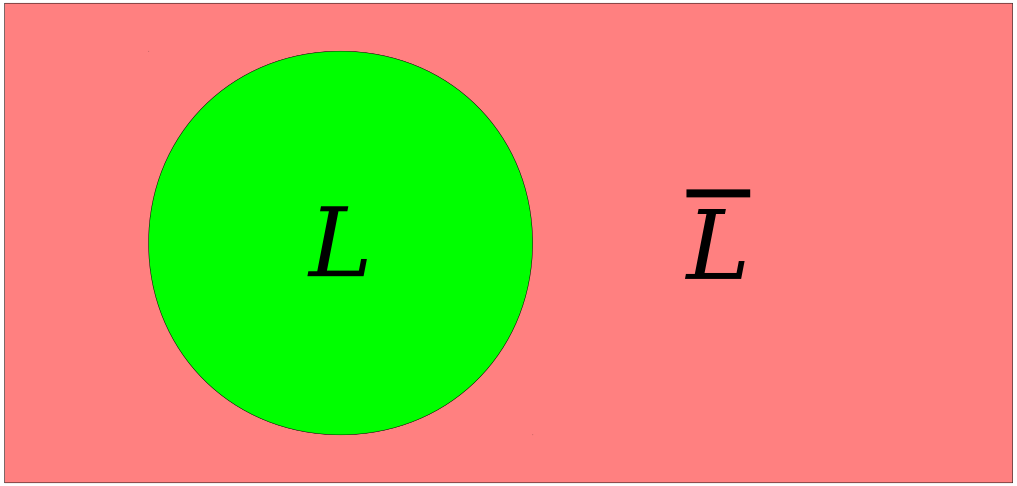
- Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language (denoted $\overline{L}$) is the language of all strings in Σ^* not in L.
- Formally:

$$\overline{L} = \Sigma^* - L$$

The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language (denoted $\overline{L}$) is the language of all strings in Σ^* not in L.
- Formally:

$$\overline{L} = \Sigma^* - L$$

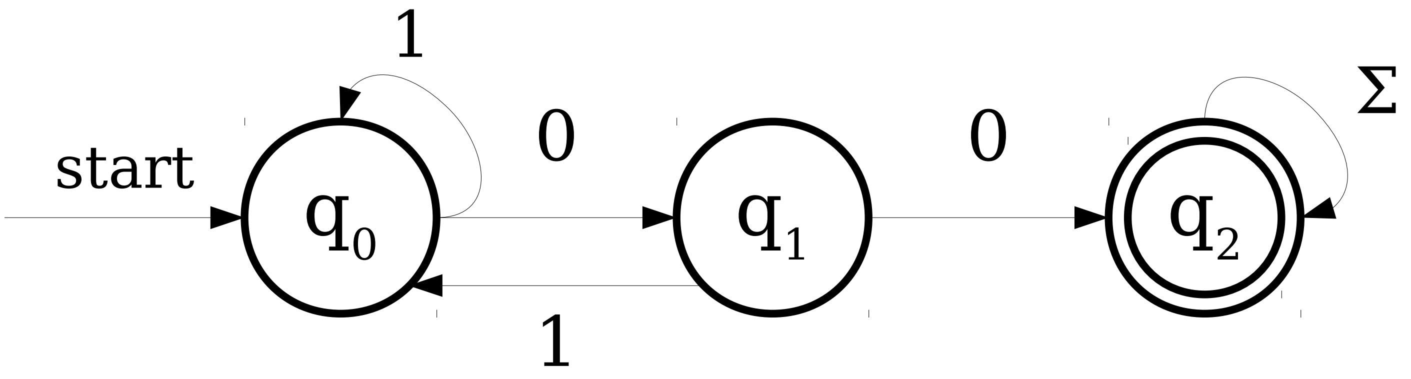


Complementing Regular Languages

- Recall: A ***regular language*** is a language accepted by some DFA.
- **Question:** If L is a regular language, is $\overline{L}$ a regular language?
- If the answer is "yes," then there must be some way to construct a DFA for $\overline{L}$.
- If the answer is "no," then some language L can be accepted by a DFA, but $\overline{L}$ cannot be accepted by any DFA.

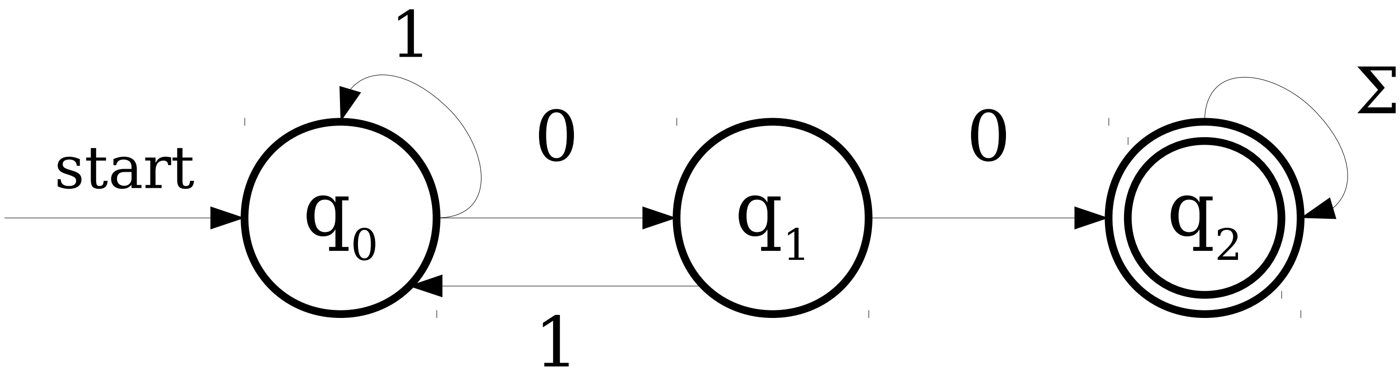
Complementing Regular Languages

$L = \{\ w \in \{\mathtt{0}, \mathtt{1}\}^* \mid w \text{ contains } \mathtt{00} \text{ as a substring } \}$



Complementing Regular Languages

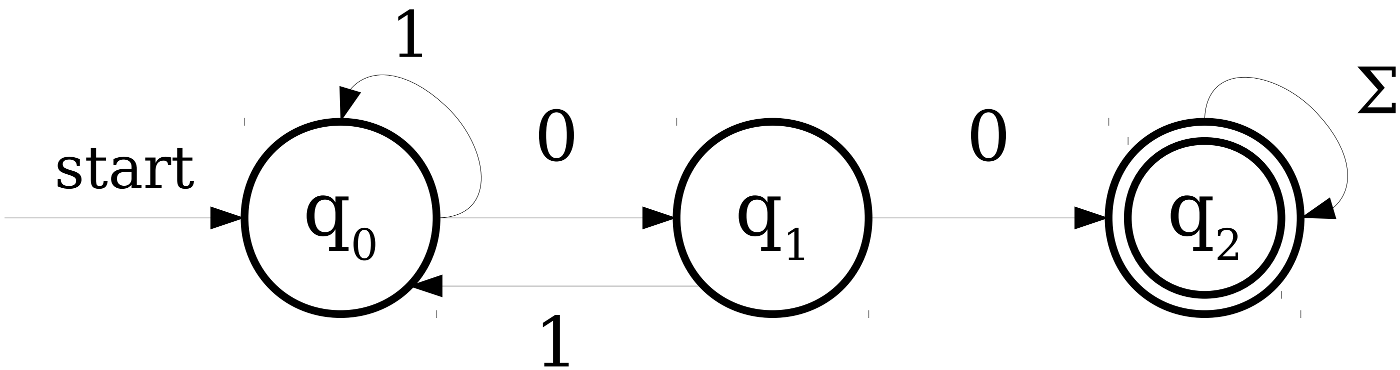
$L = \{ w \in \{0, 1\}^* \mid w \text{ contains } 00 \text{ as a substring } \}$



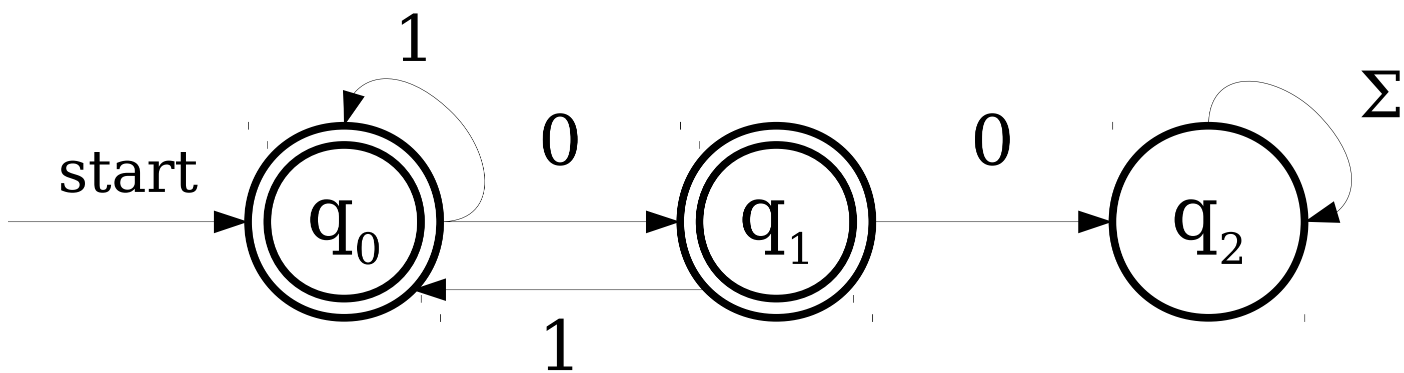
$\overline{L} = \{ w \in \{0, 1\}^* \mid w$ **does not** contain 00 as a substring $\}$

Complementing Regular Languages

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w$ contains $\texttt{00}$ as a substring $\}$

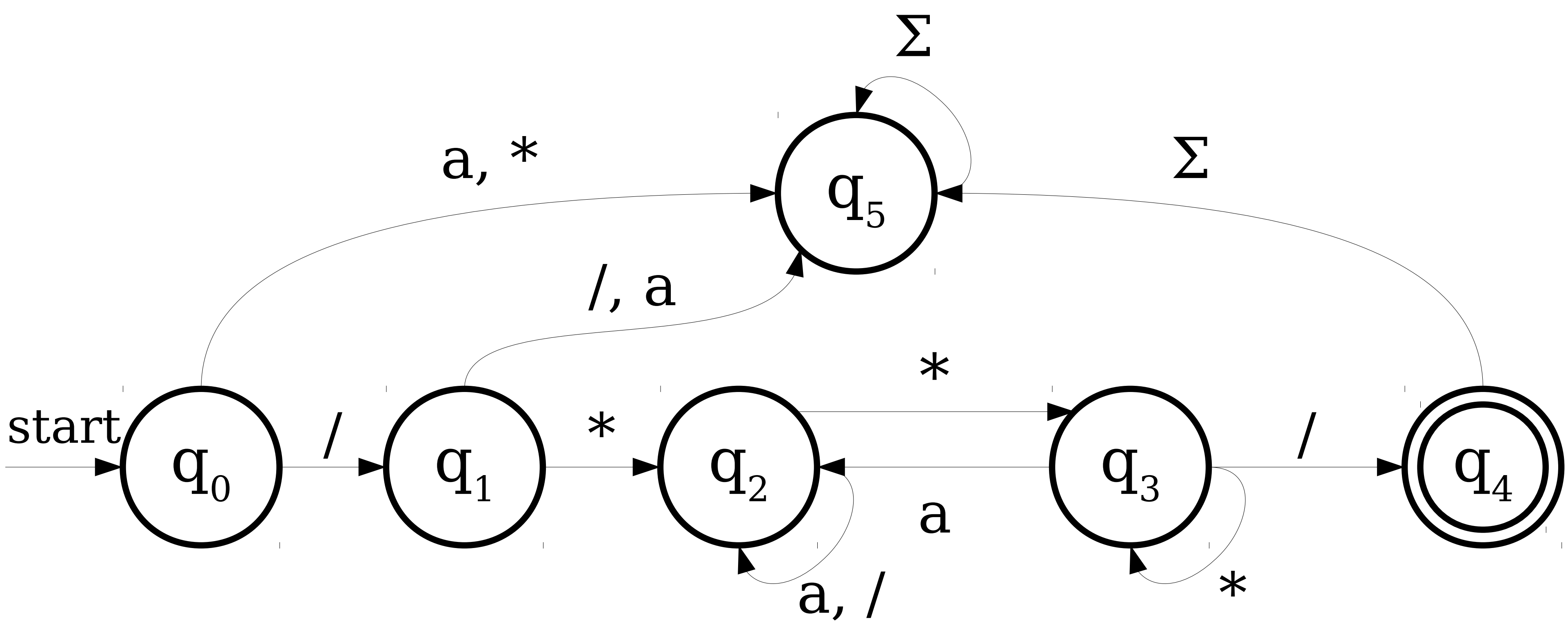


$\overline{L} = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w$ **does not** contain $\texttt{00}$ as a substring $\}$



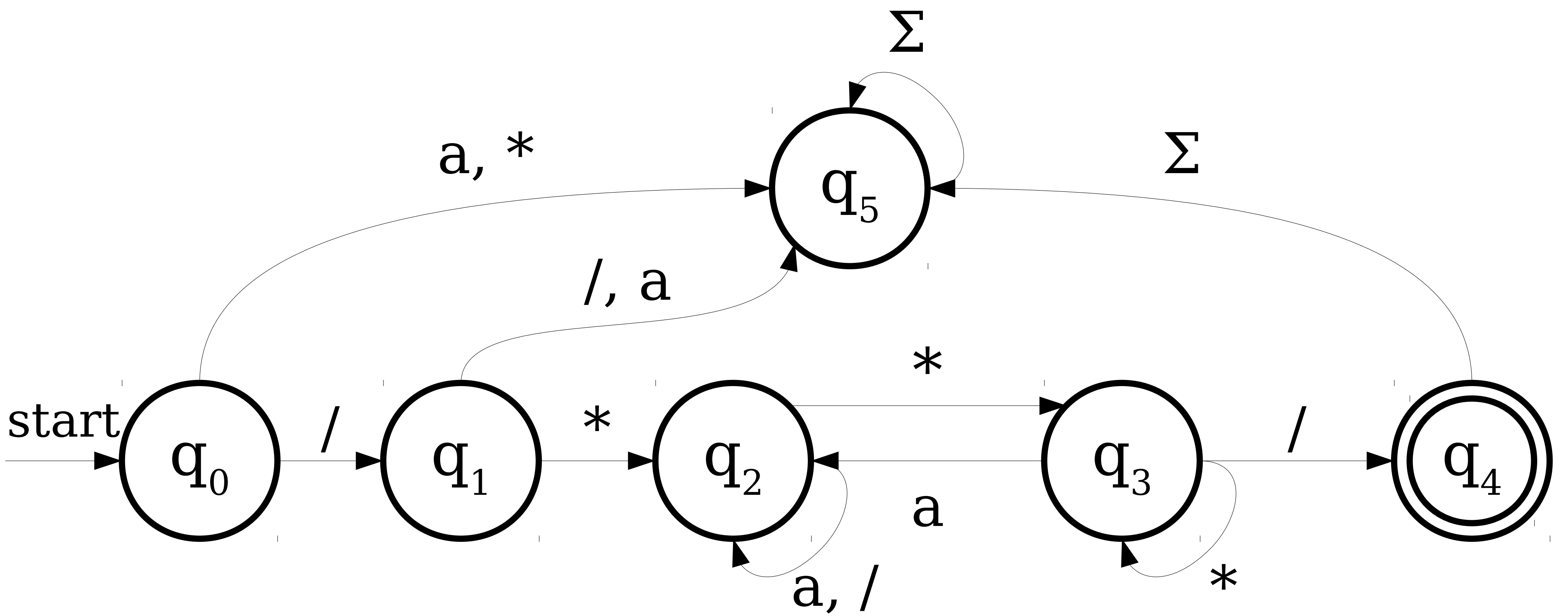
More Elaborate DFAs

$L = \{ \ w \mid w \text{ is a C-style comment } \}$



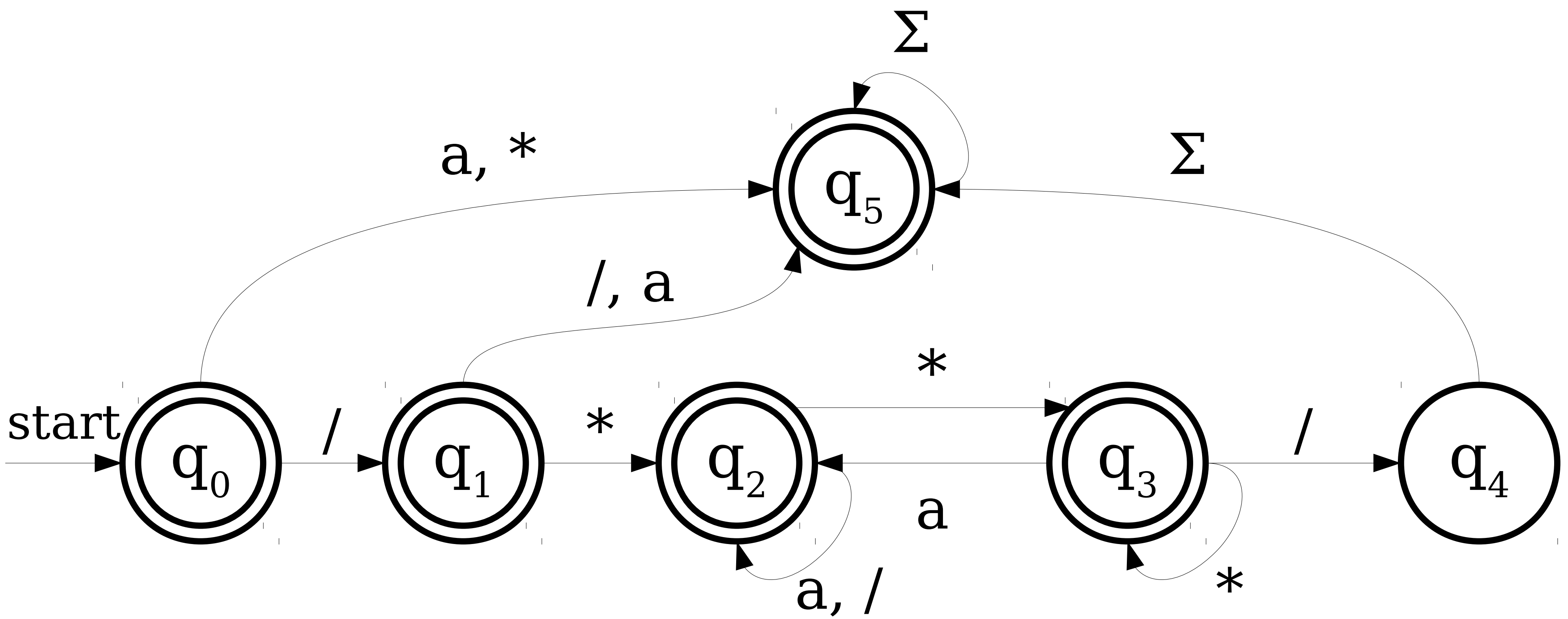
More Elaborate DFAs

$\overline{L}$ = { w | w is **not** a C-style comment }



More Elaborate DFAs

$\overline{L}$ = { w | w is **not** a C-style comment }



Closure Properties

- ***Theorem:*** If L is a regular language, then $\overline{L}$ is also a regular language.
- If we begin with a regular language and complement it, we end up with a regular language.
- This is an example of a **closure property of regular languages**.
  - The regular languages are **closed under complementation**.
  - We'll see more such properties later on.

Time-Out For Announcements!

Midterm Denouement

- We'll be grading the midterm exam over the weekend; we're aiming to get it graded and returned by Monday.
- Solutions will be released along with statistics when the exam is returned.
- Have any questions in the meantime? Feel free to email us!

Old Solution Sets

- All solution sets released before the midterm will be recycled next week to make more space.
- Please pick them up by then if you're interested!

Problem Set Four

- PS4 is due on Monday at 2:15PM; due Wednesday at 2:15PM with a late period.
- We've slightly shifted our OH schedule; there's now two sets of office hours after today's lecture.
- Check the website for details.

A Point of Clarification

Theorem: If $\mathscr{U}$ is the universal set, then $|\wp(\mathscr{U})| \leq |\mathscr{U}|$

Proof: The universal set $\mathscr{U}$ contains all objects. Therefore, $\wp(\mathscr{U}) \in \mathscr{U}$. Consequently, $|\wp(\mathscr{U})| \leq |\mathscr{U}|$. ■

Theorem: If $\mathscr{U}$ is the universal set, then $|\wp(\mathscr{U})| \leq |\mathscr{U}|$

Proof: The universal set $\mathscr{U}$ contains all objects.
Therefore, $\wp(\mathscr{U}) \in \mathscr{U}$. Consequently, $|\wp(\mathscr{U})| \leq |\mathscr{U}|$. ■

Theorem: If $\mathscr{U}$ is the universal set, then $|\wp(\mathscr{U})| \leq |\mathscr{U}|$

Proof: The universal set $\mathscr{U}$ contains all objects.
Therefore, $\wp(\mathscr{U}) \in \mathscr{U}$. Consequently, $|\wp(\mathscr{U})| \leq |\mathscr{U}|$. ■

Does this reasoning work?

Theorem: If $\mathscr{U}$ is the universal set, then $|\wp(\mathscr{U})| \le |\mathscr{U}|$

Proof: The universal set $\mathscr{U}$ contains all objects. Therefore, every element of $\wp(\mathscr{U})$ is an element of $\mathscr{U}$. Accordingly, $\wp(\mathscr{U}) \subseteq \mathscr{U}$, so $|\wp(\mathscr{U})| \le |\mathscr{U}|$. ■

Remember that $\in$ and $\subseteq$
are different concepts!

Your Questions

“What are some classes you wish you took as a student but never did?”

“Are you teaching any classes next quarter? If so, what are you teaching?”

Back to CS103!

NFAs

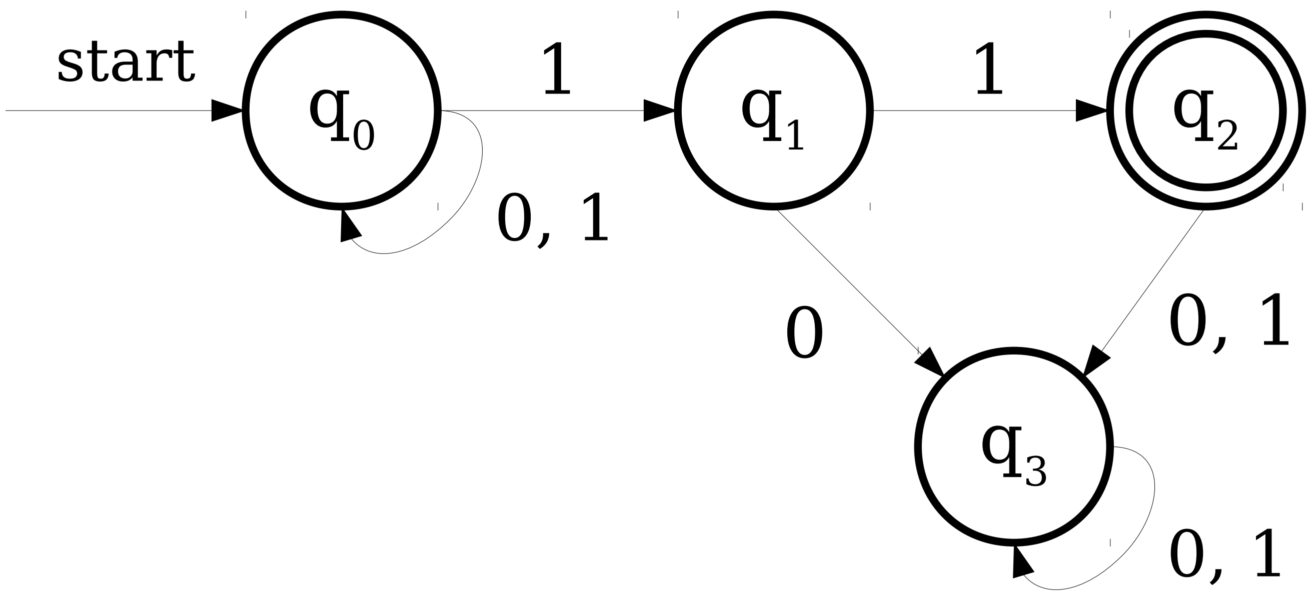
NFAs

- An **NFA** is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- Conceptually similar to a DFA, but equipped with the vast power of *nondeterminism*.

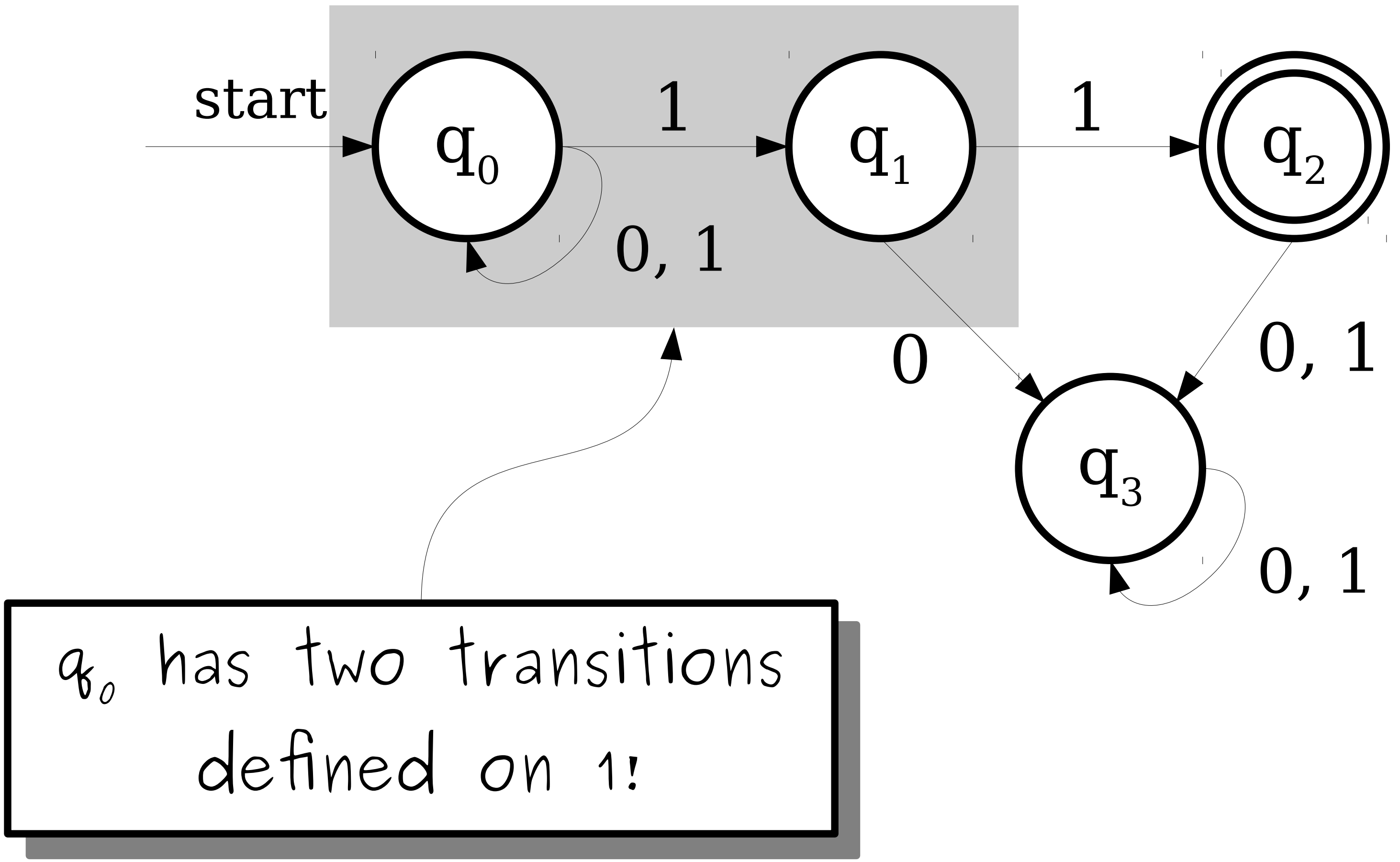
(Non)determinism

- A model of computation is ***deterministic*** if at every point in the computation, there is exactly one choice that can make.
- The machine accepts if that series of choices leads to an accepting state.
- A model of computation is ***nondeterministic*** if the computing machine may have multiple decisions that it can make at one point.
- The machine accepts if ***any*** series of choices leads to an accepting state.

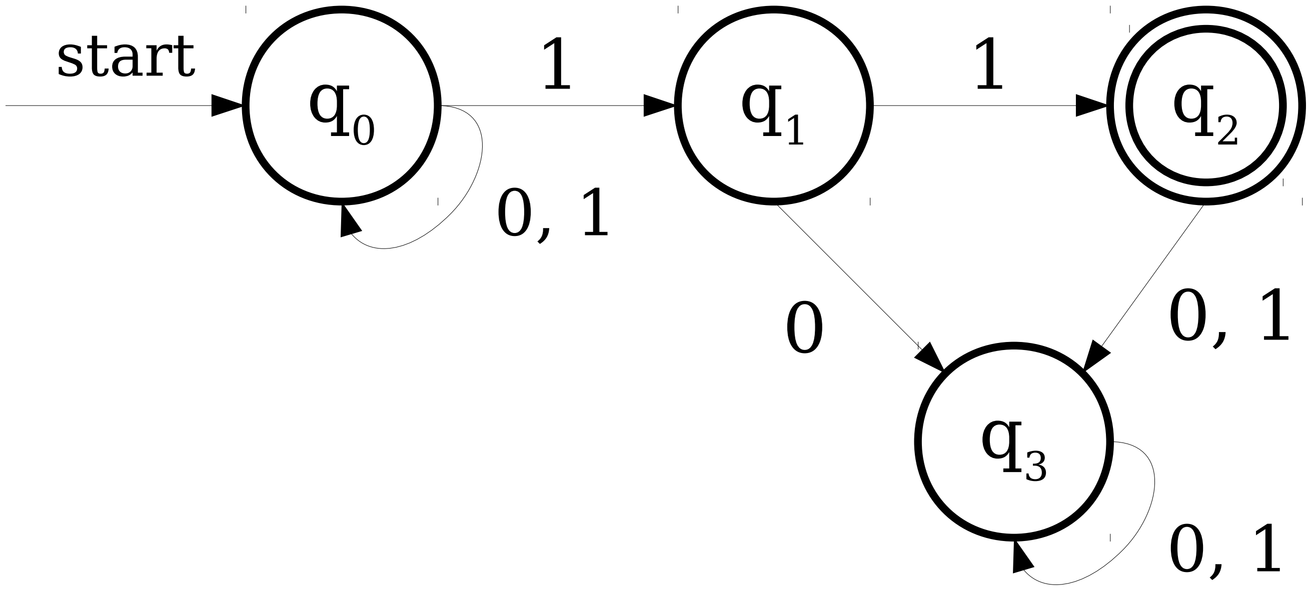
A Simple NFA



A Simple NFA

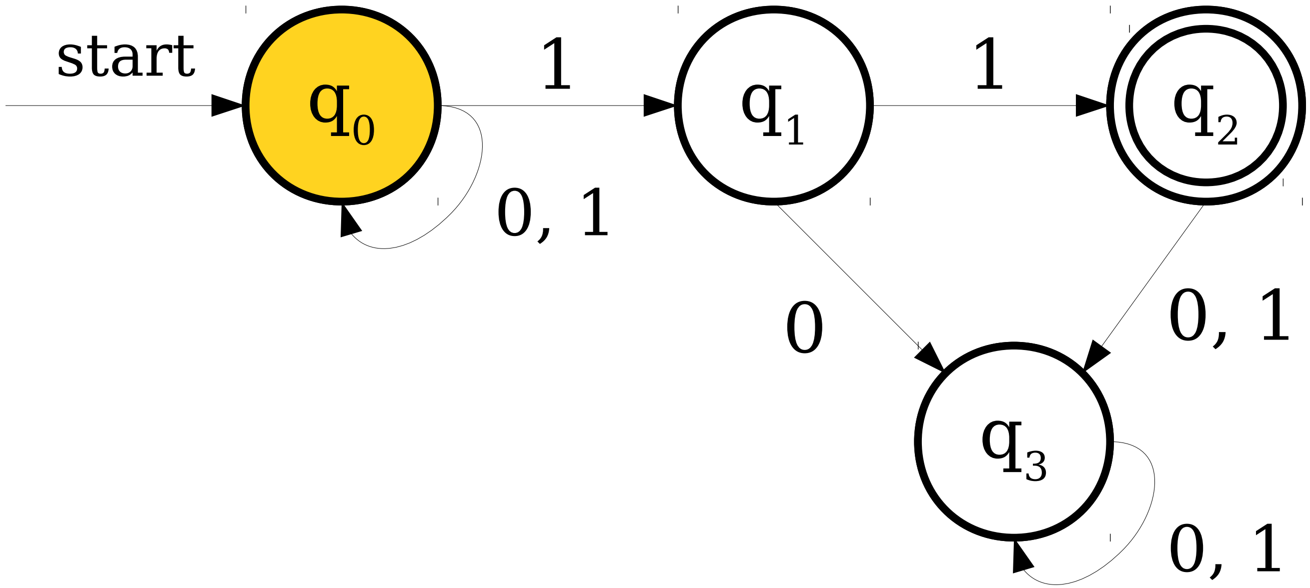

start
q0
q1
q2
q3
1
1
0, 1
0
0, 1
0, 1
q0 has two transitions defined on 1!

A Simple NFA



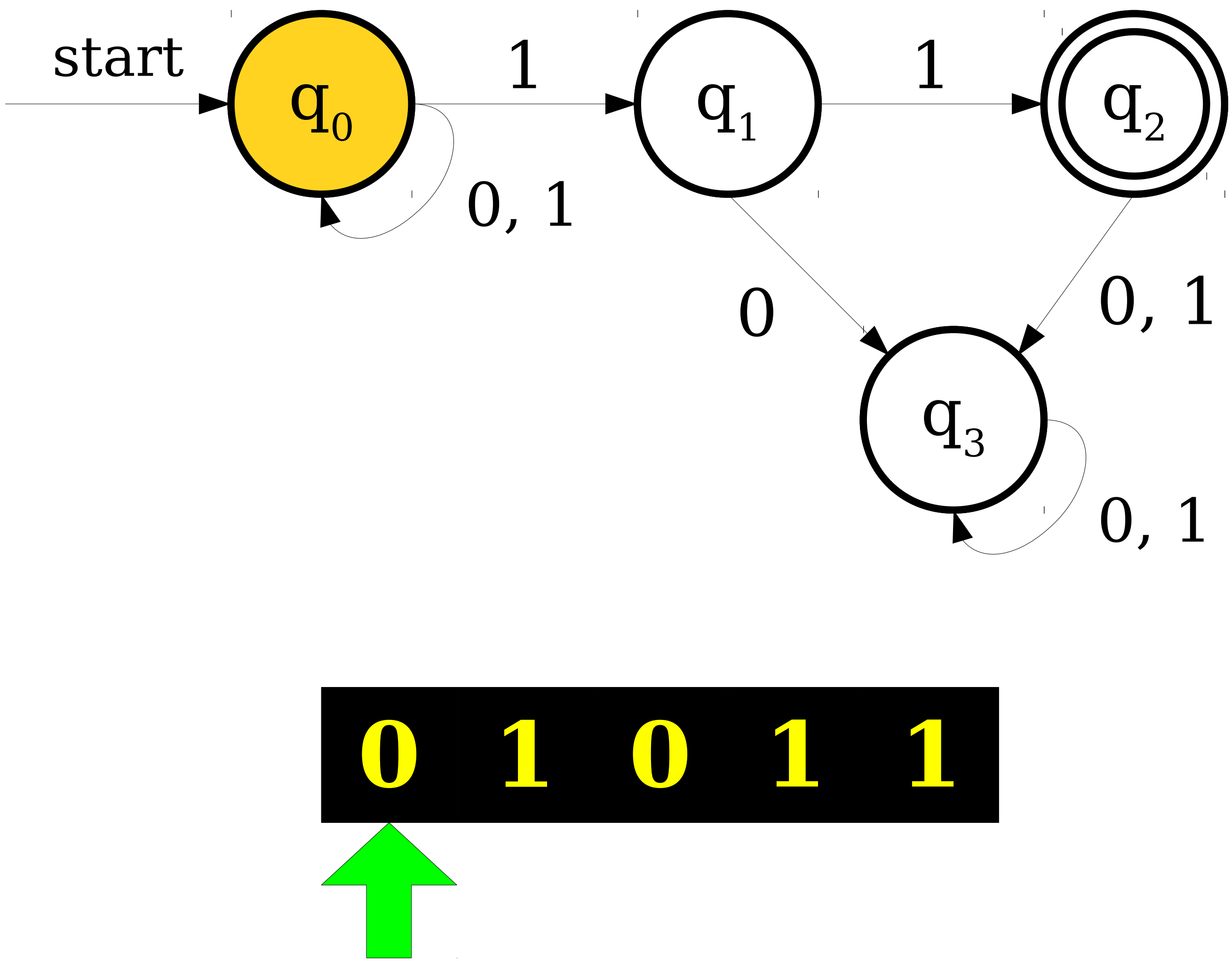
0 1 0 1 1

A Simple NFA

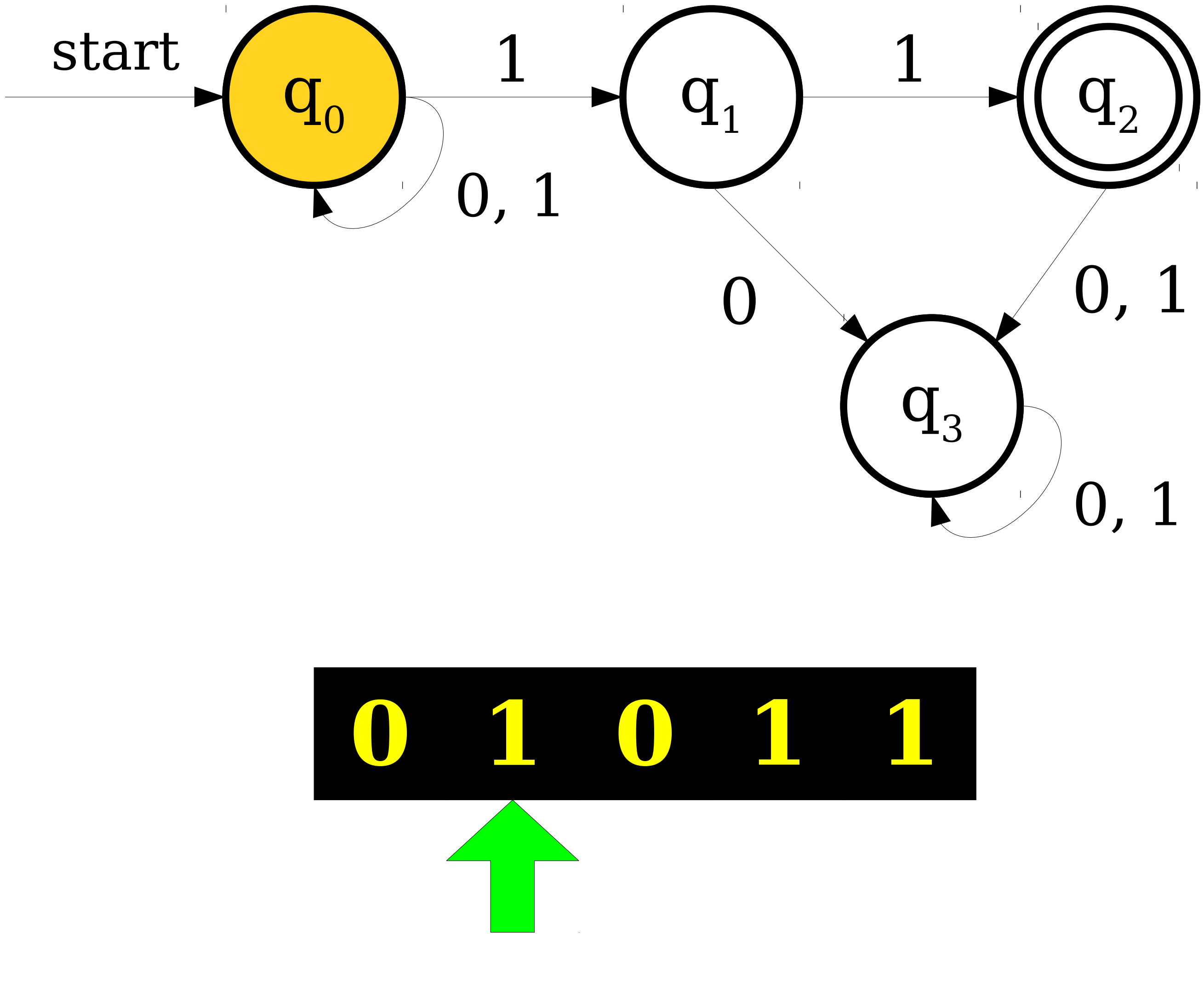


0 1 0 1 1

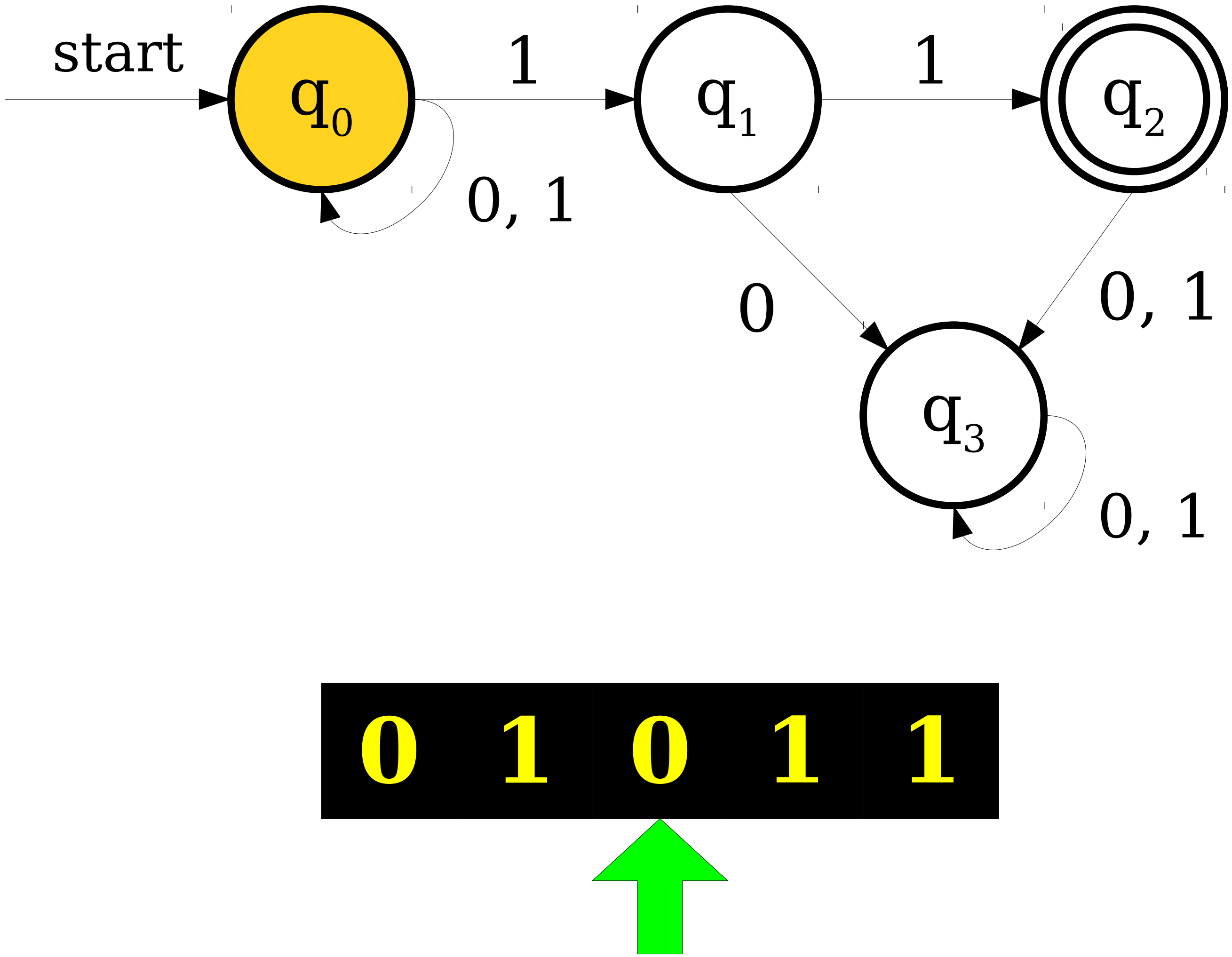
A Simple NFA



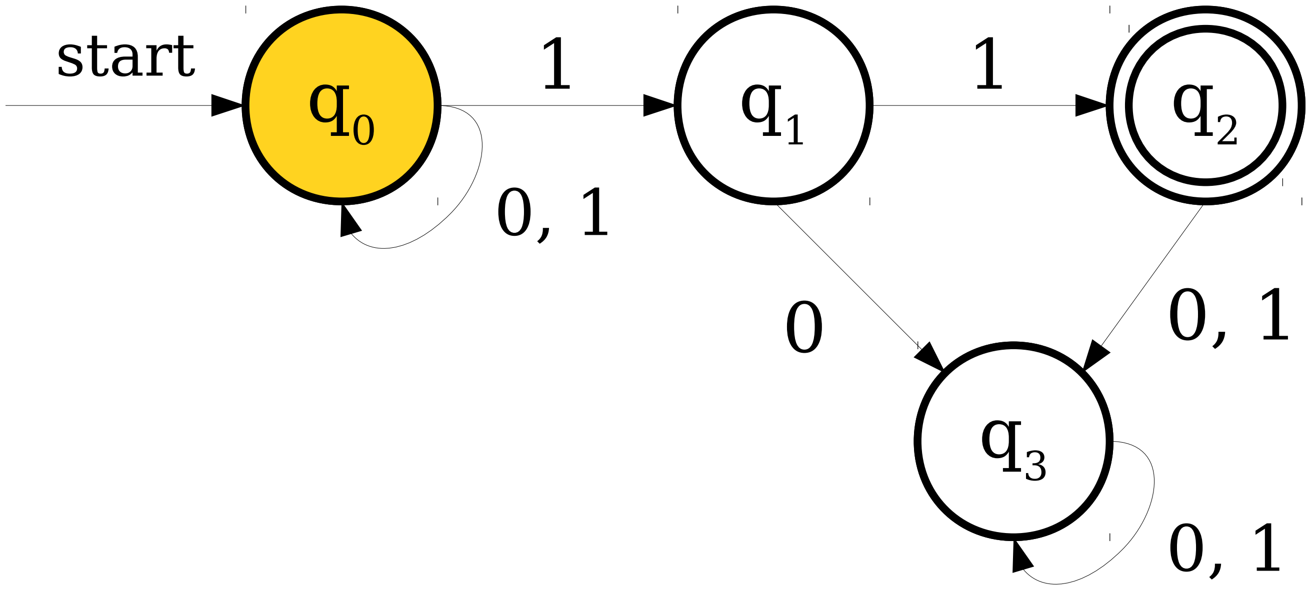
A Simple NFA

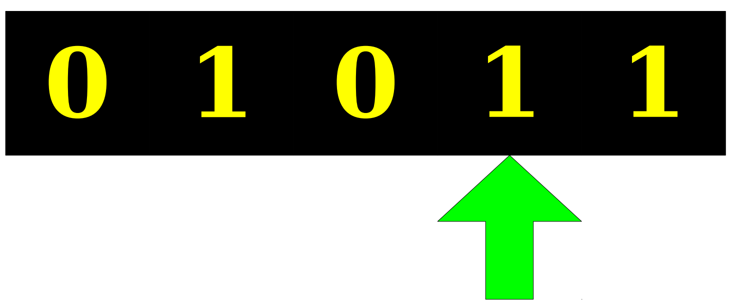


A Simple NFA

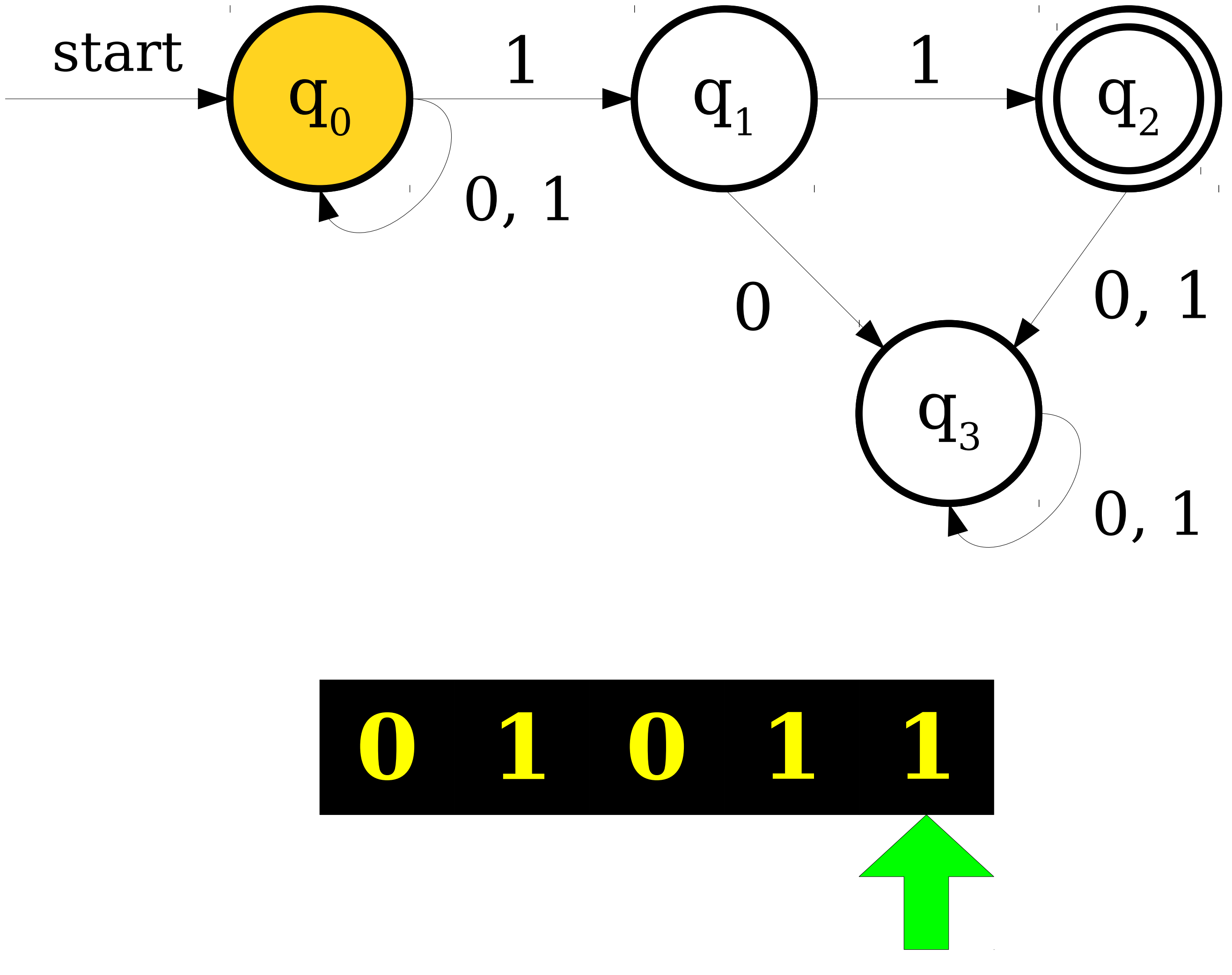


A Simple NFA

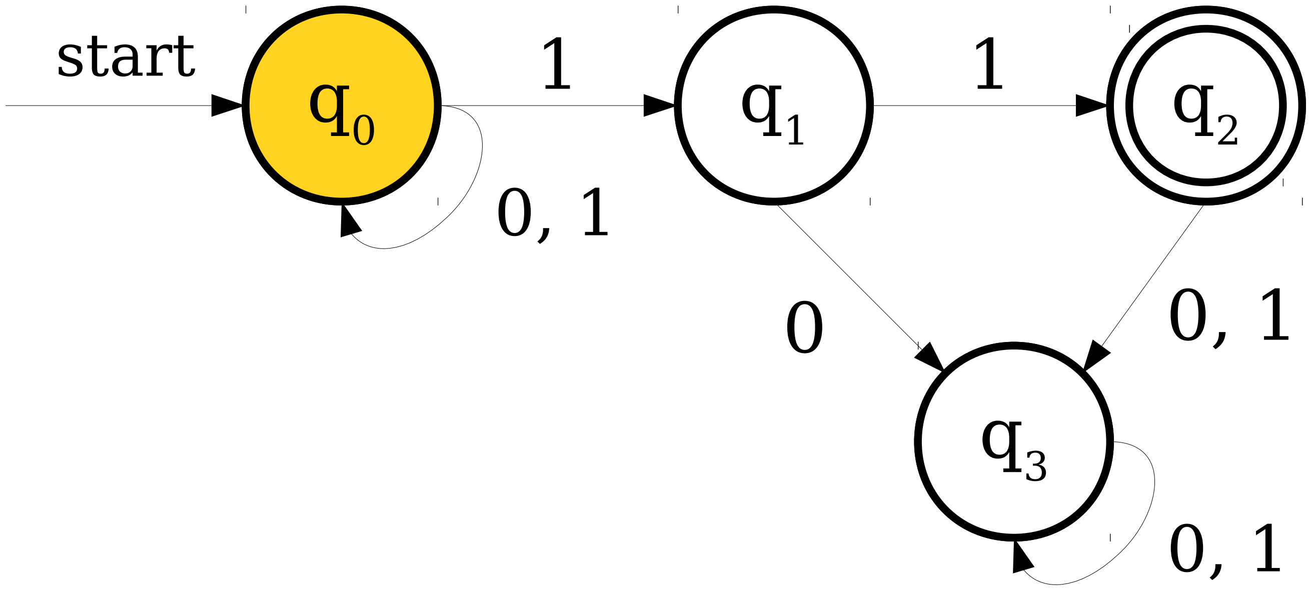




A Simple NFA

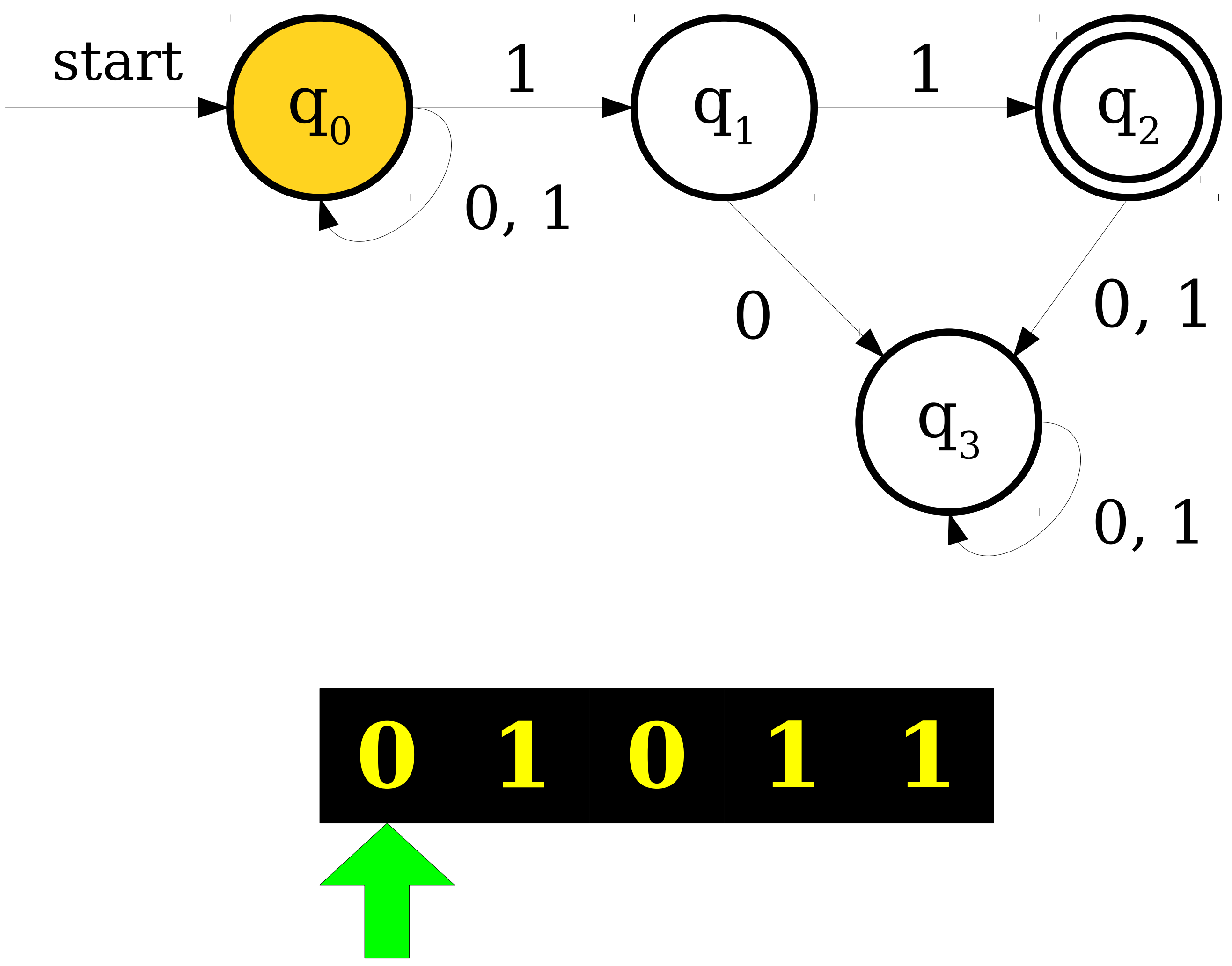


A Simple NFA

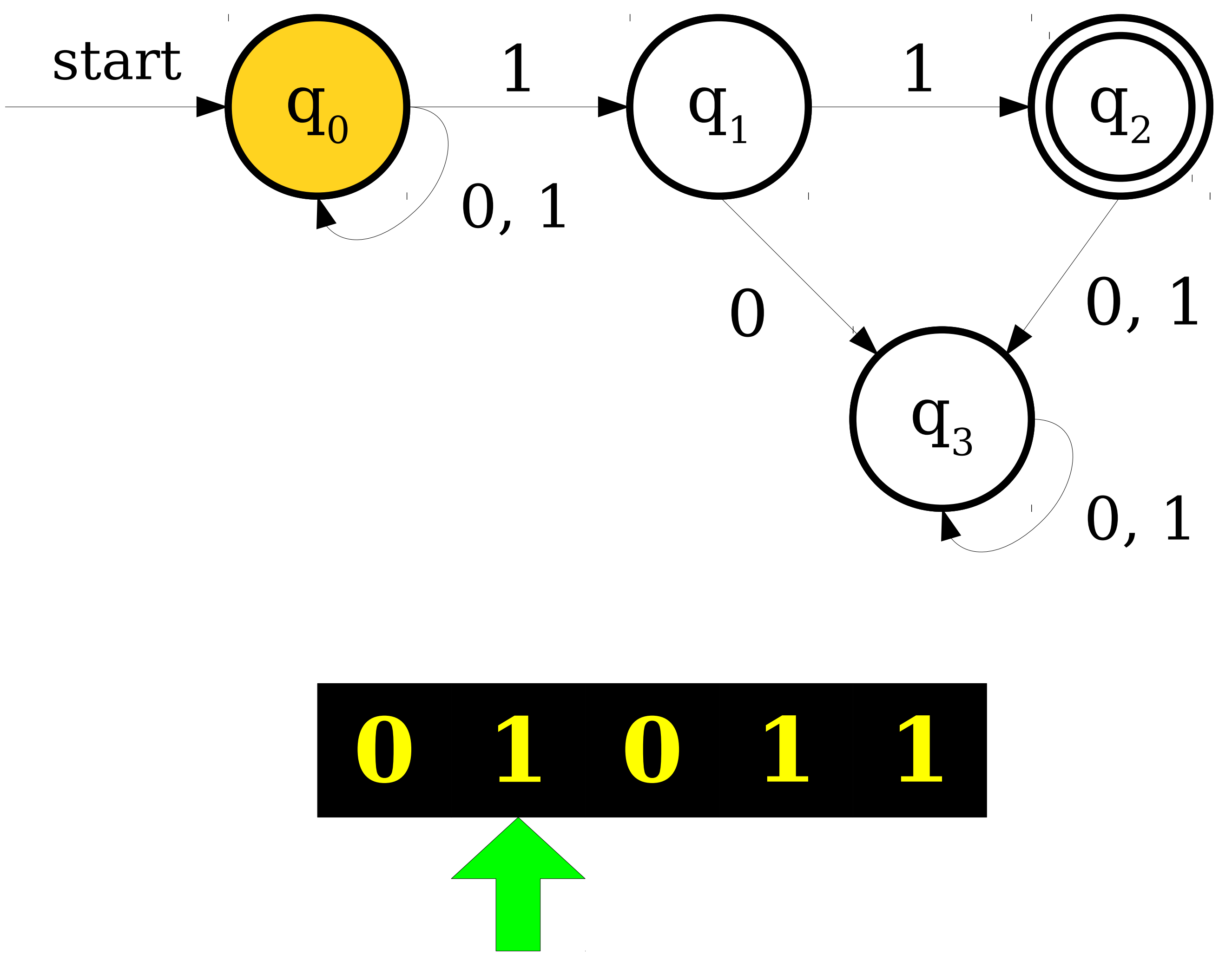


0 1 0 1 1

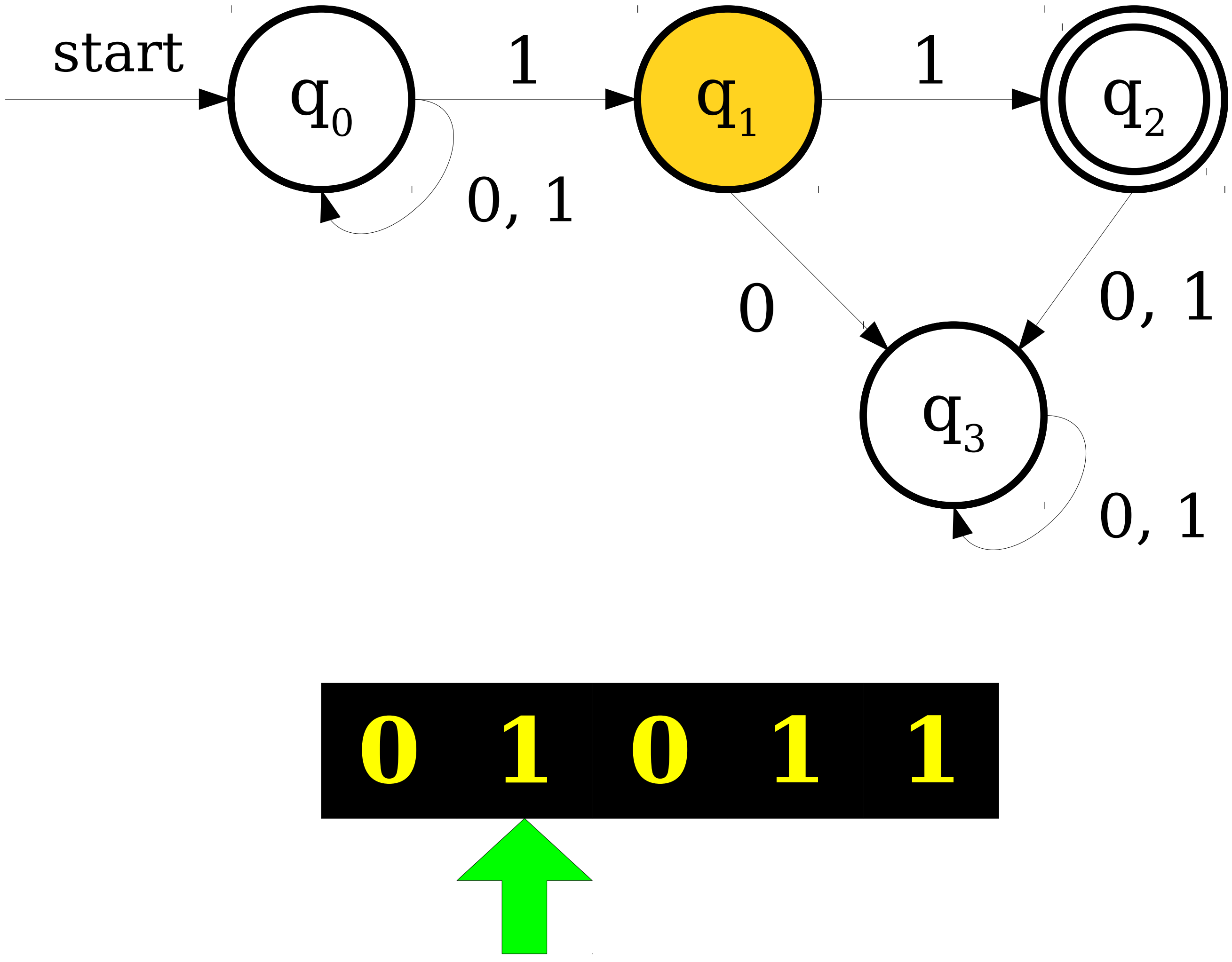
A Simple NFA



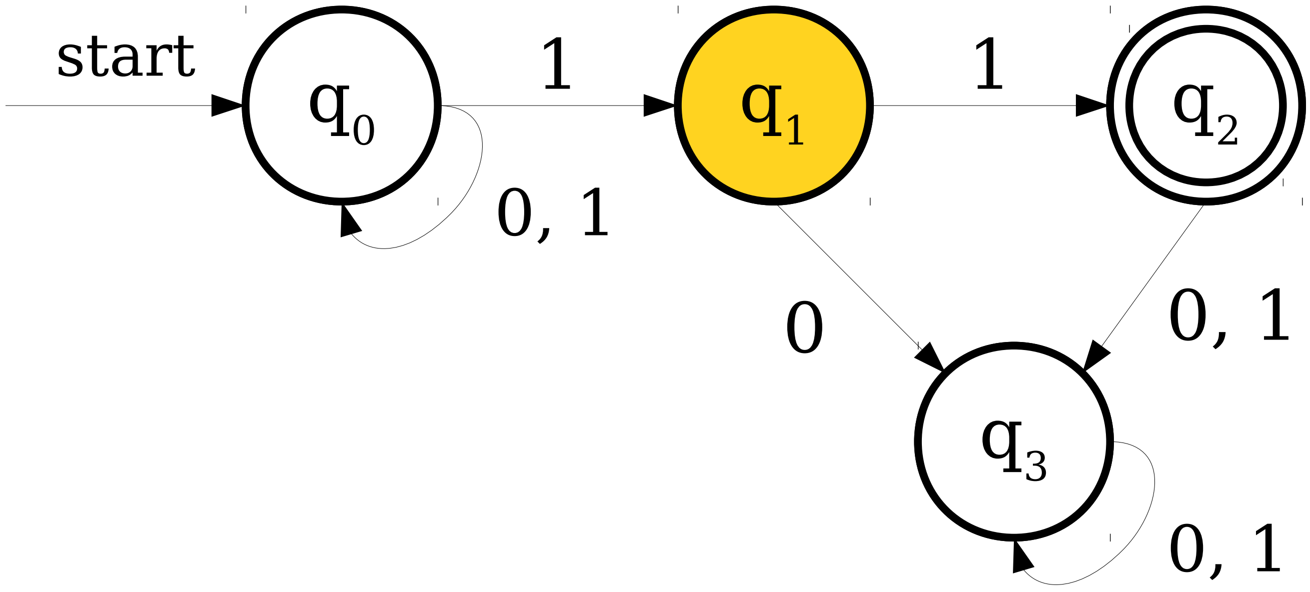
A Simple NFA

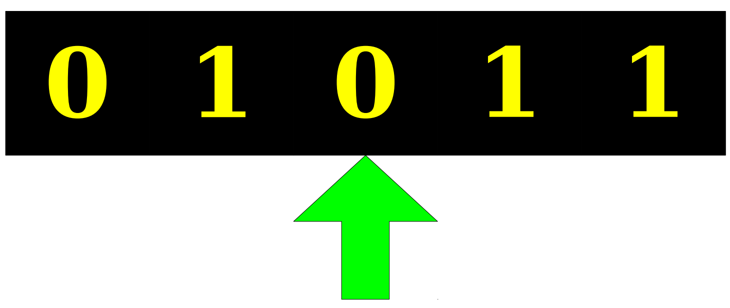


A Simple NFA

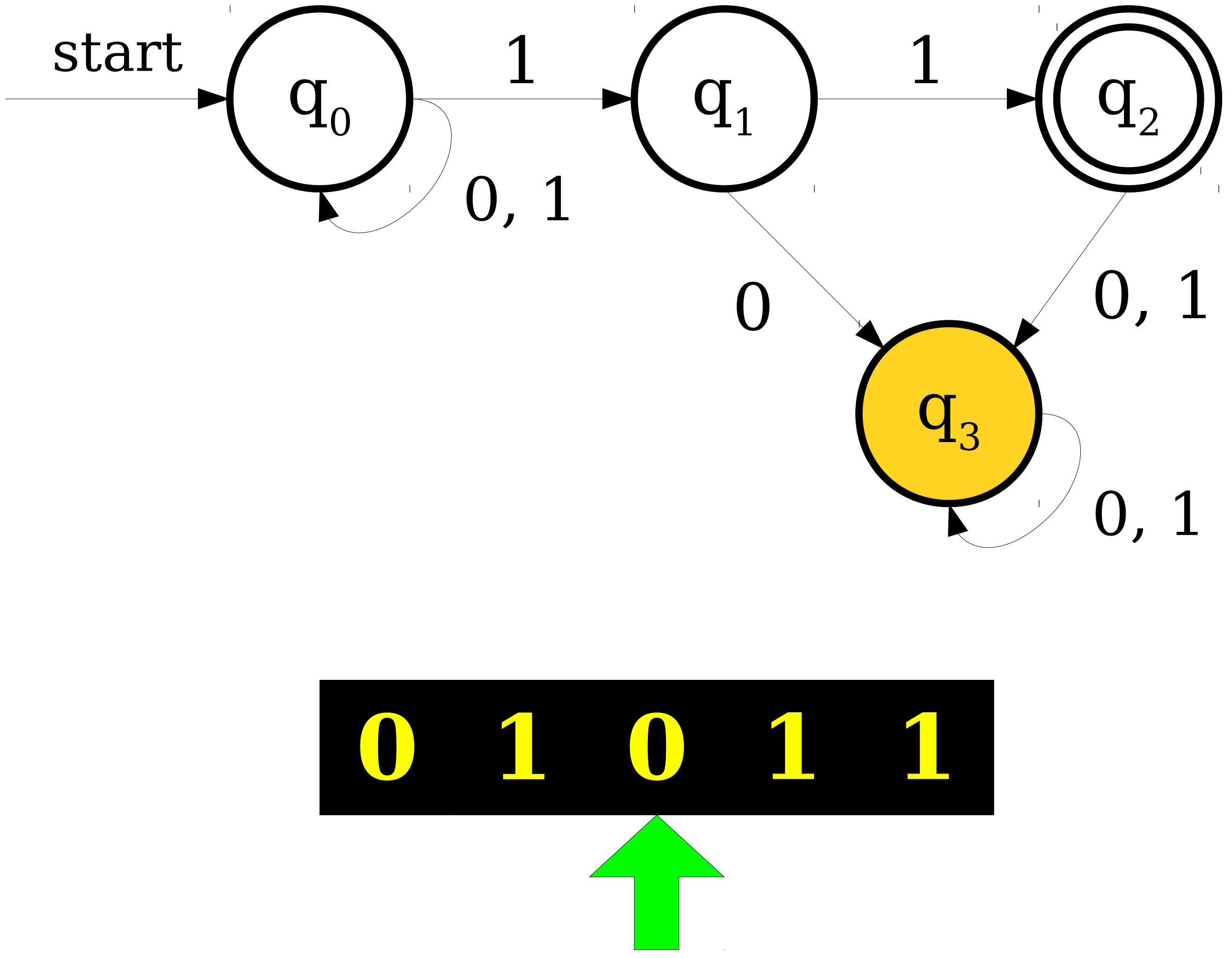


A Simple NFA

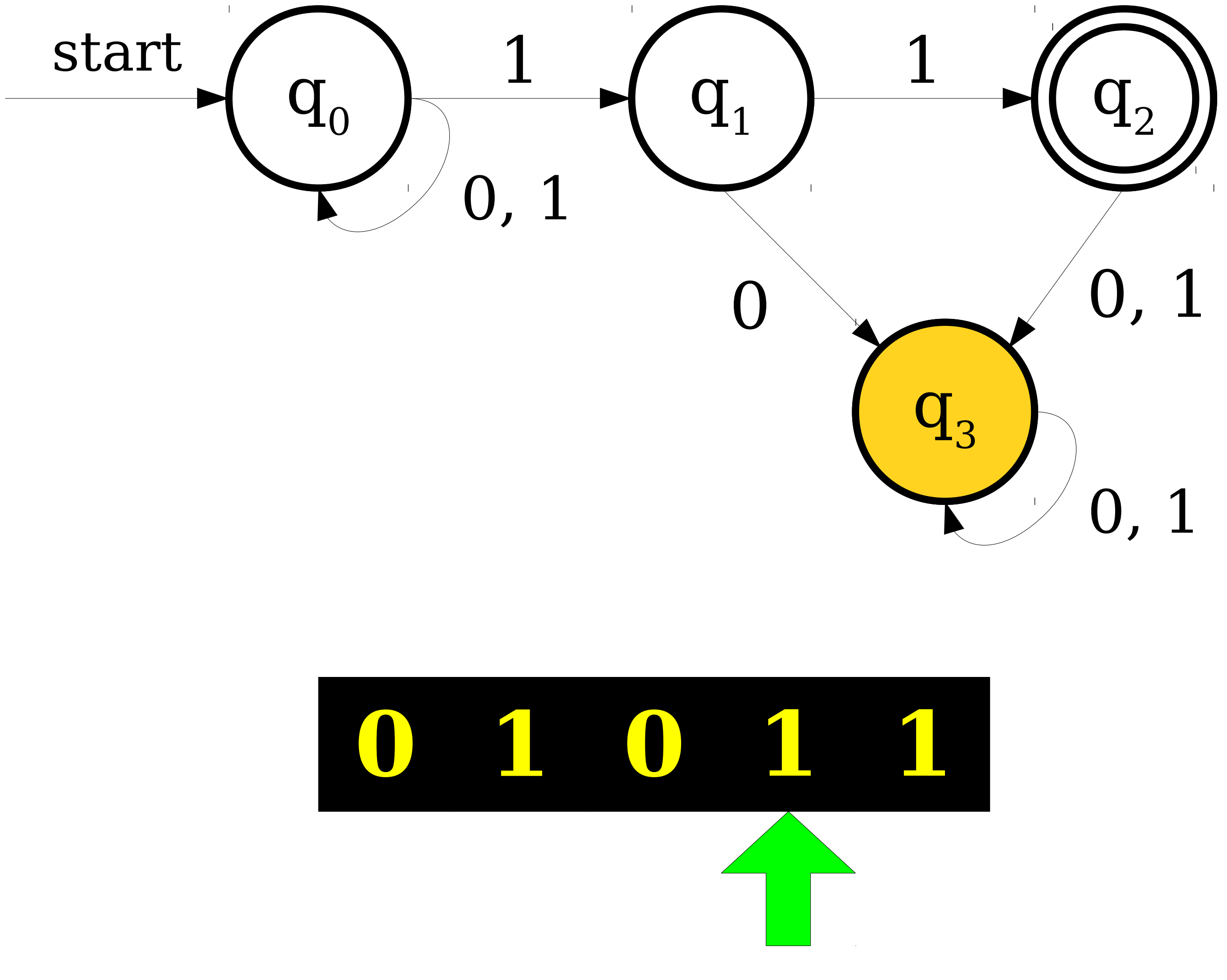




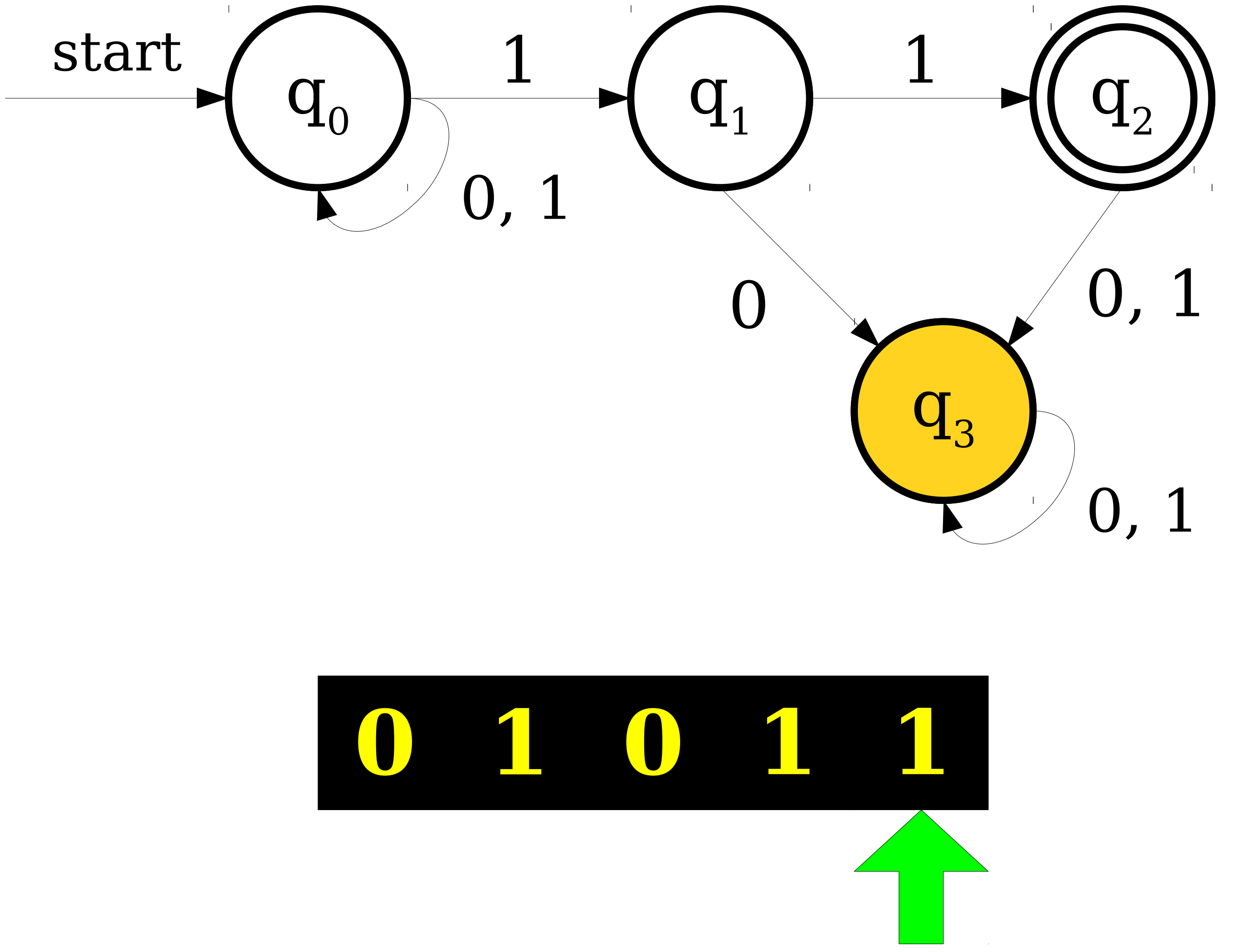
A Simple NFA



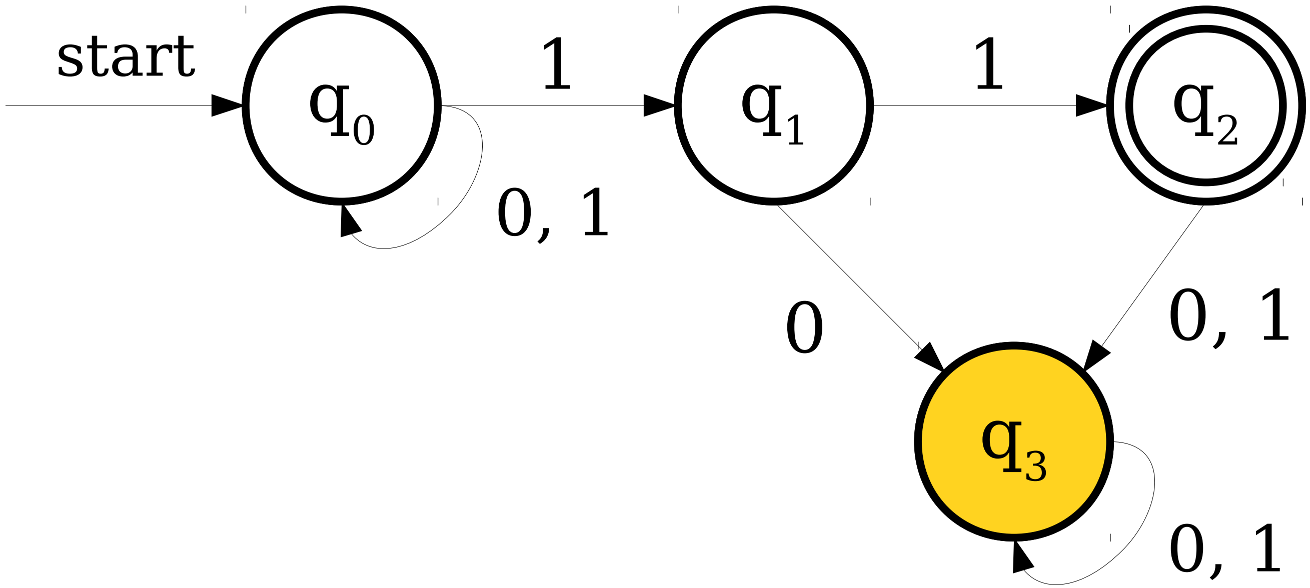
A Simple NFA



A Simple NFA

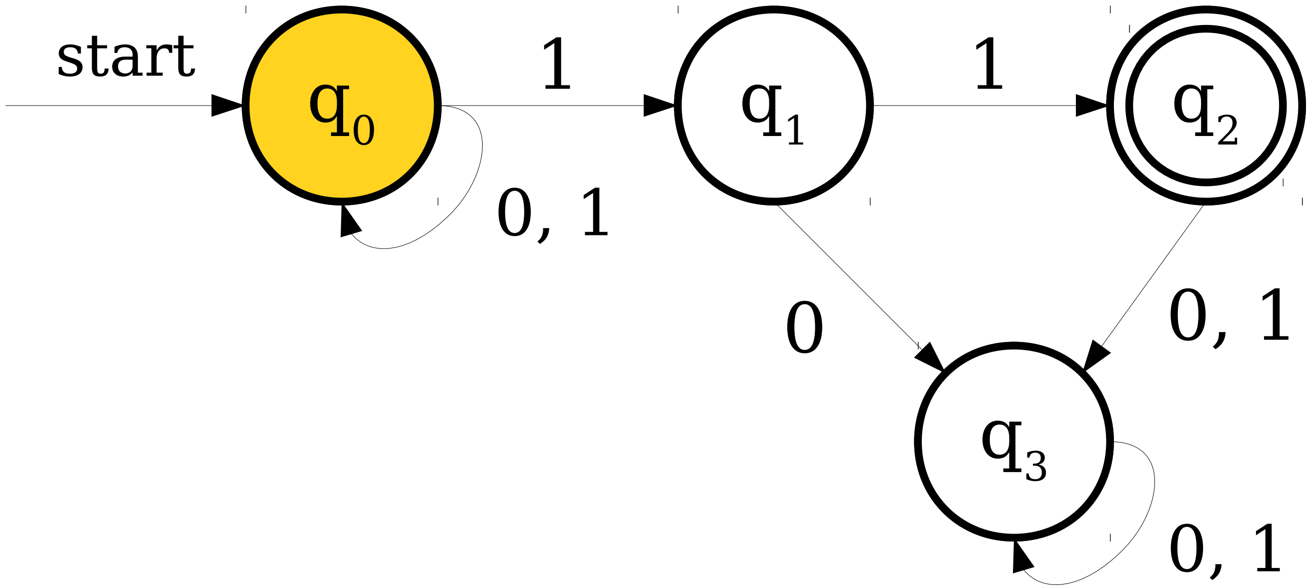


A Simple NFA

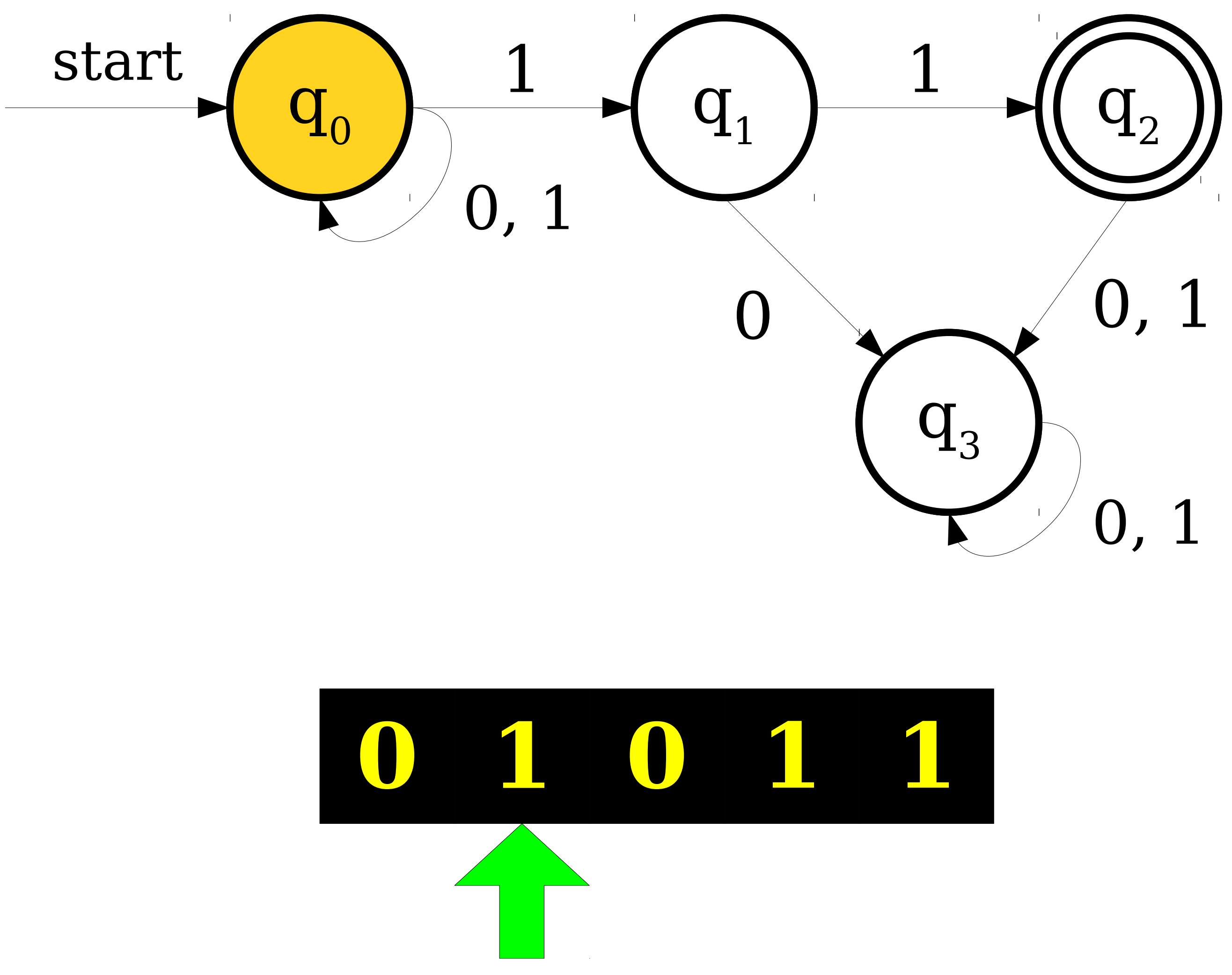


0 1 0 1 1

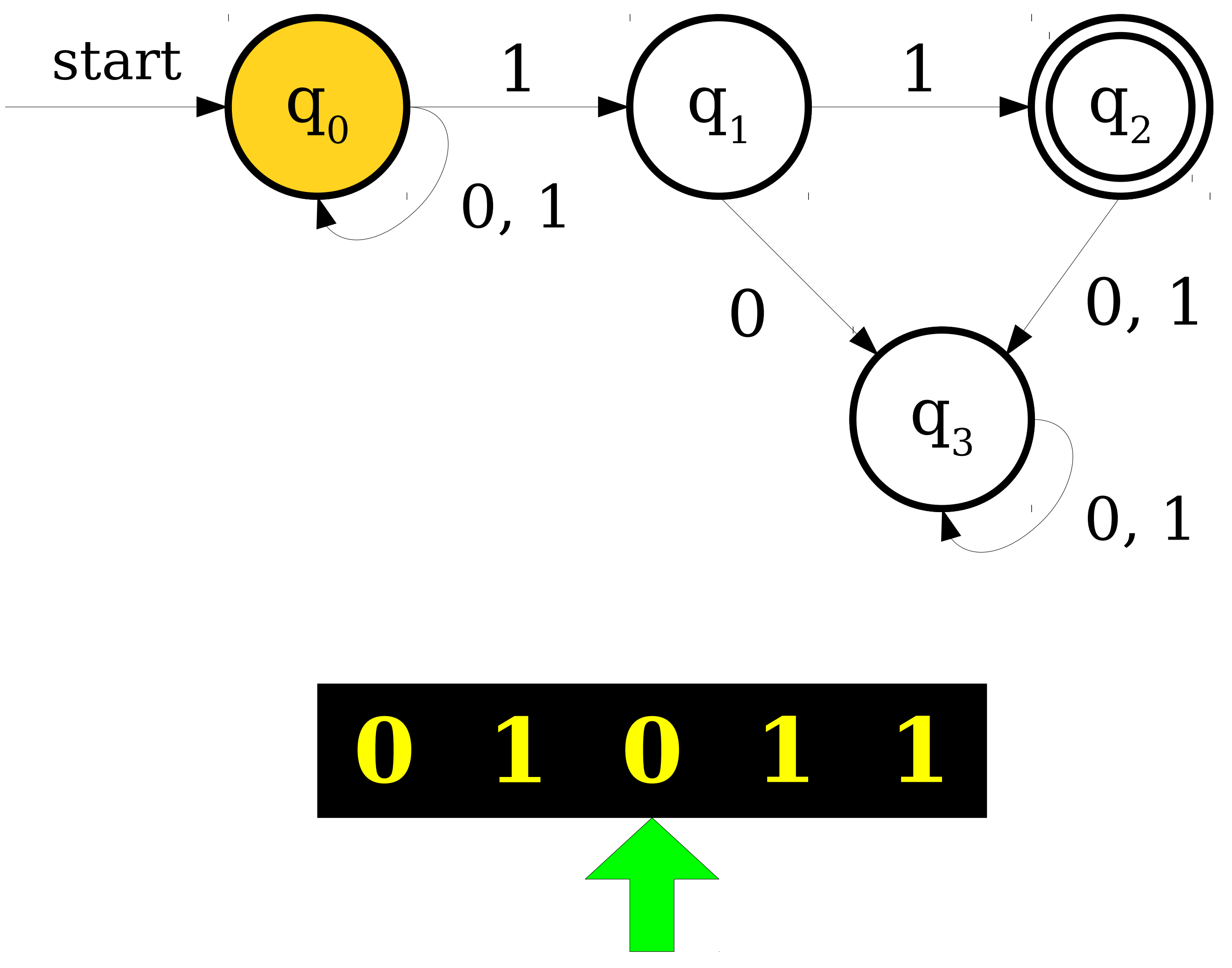
A Simple NFA



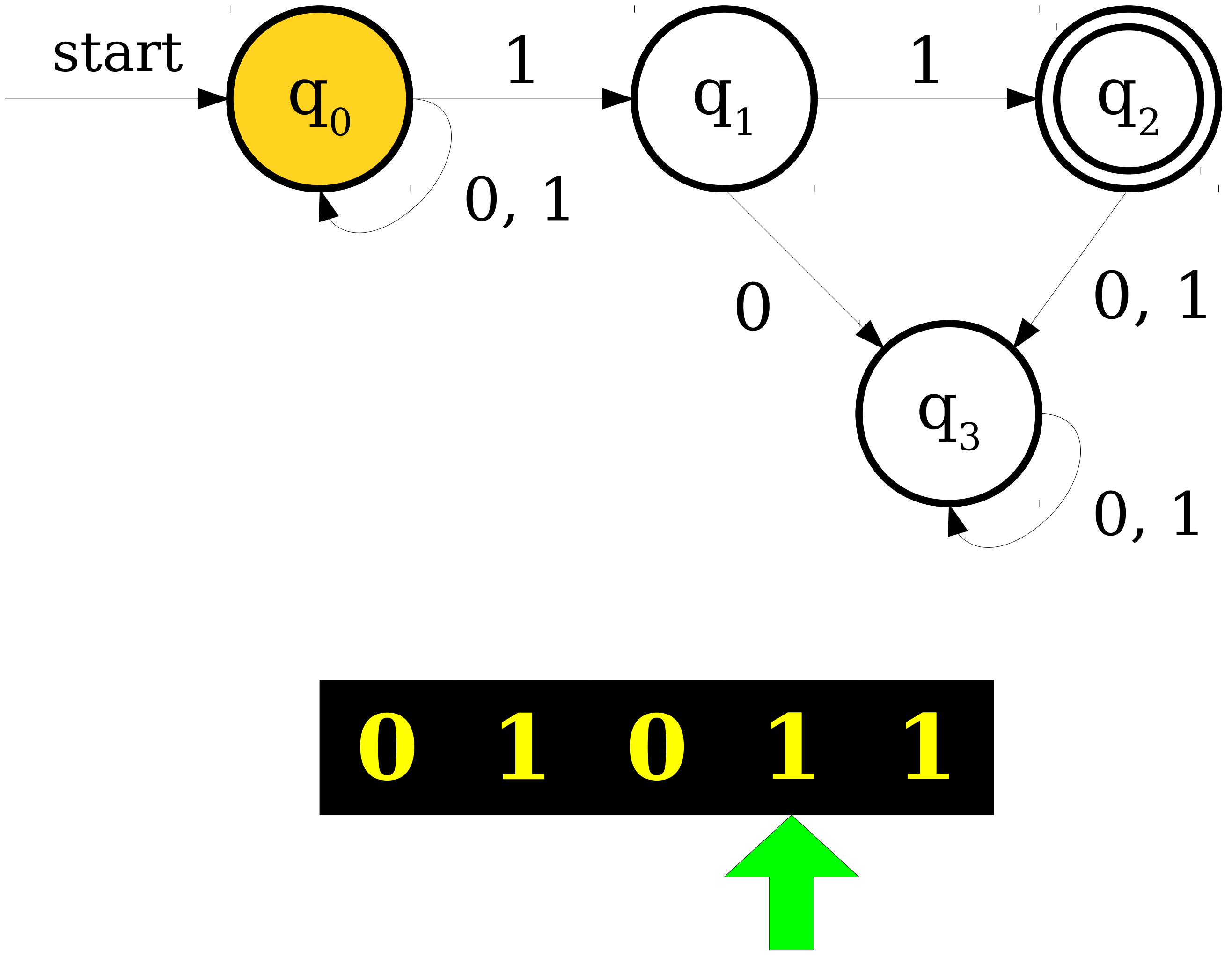
A Simple NFA



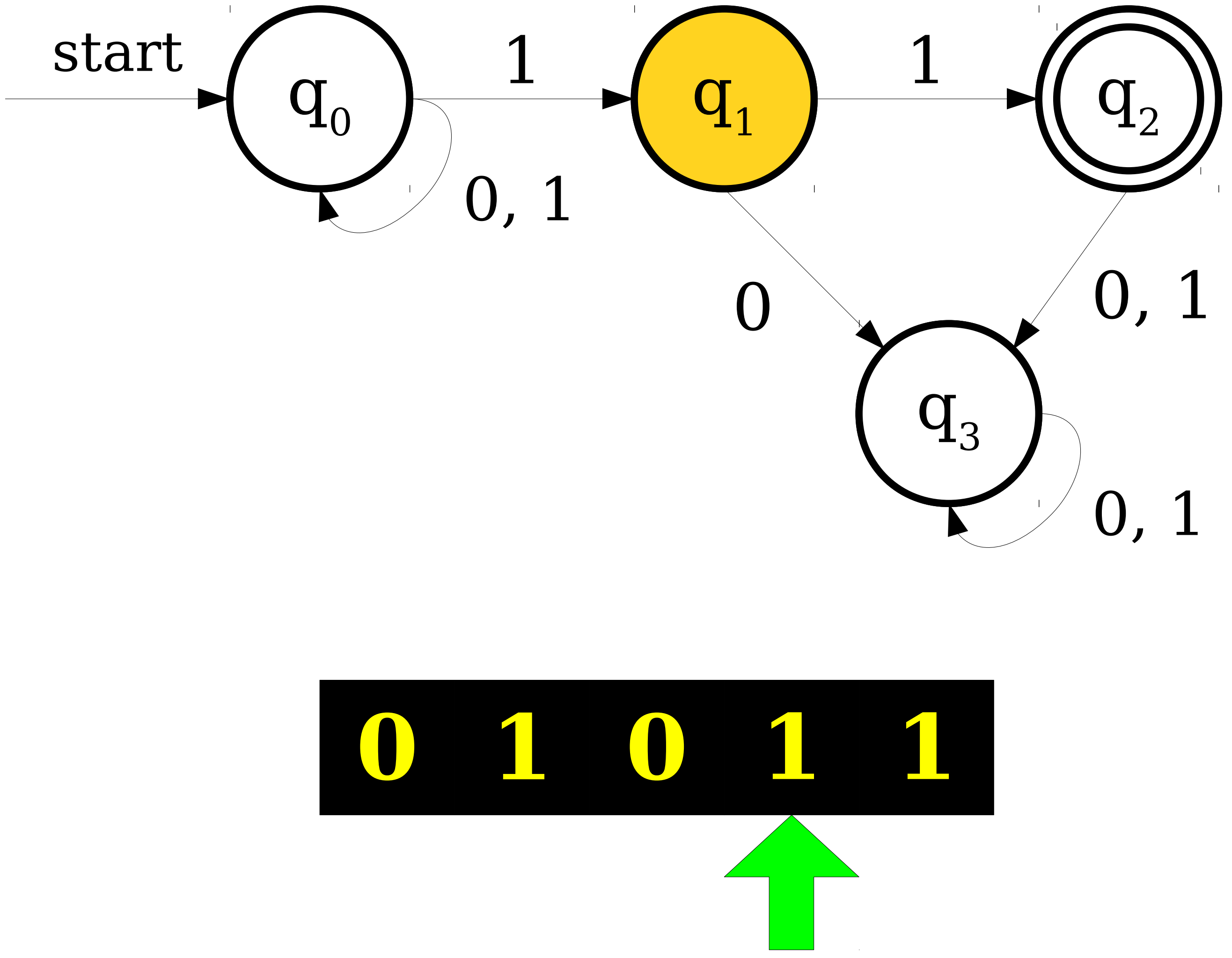
A Simple NFA



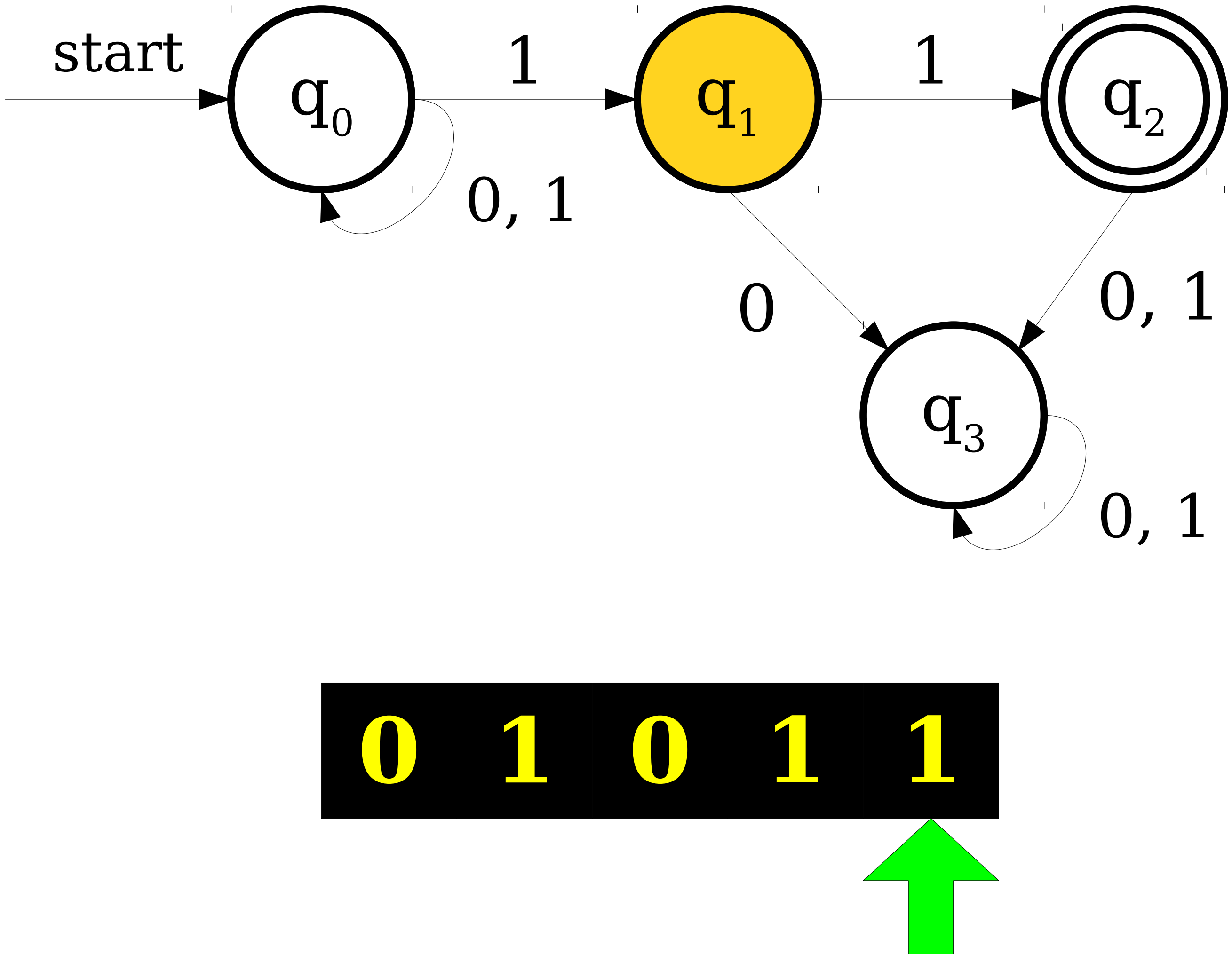
A Simple NFA



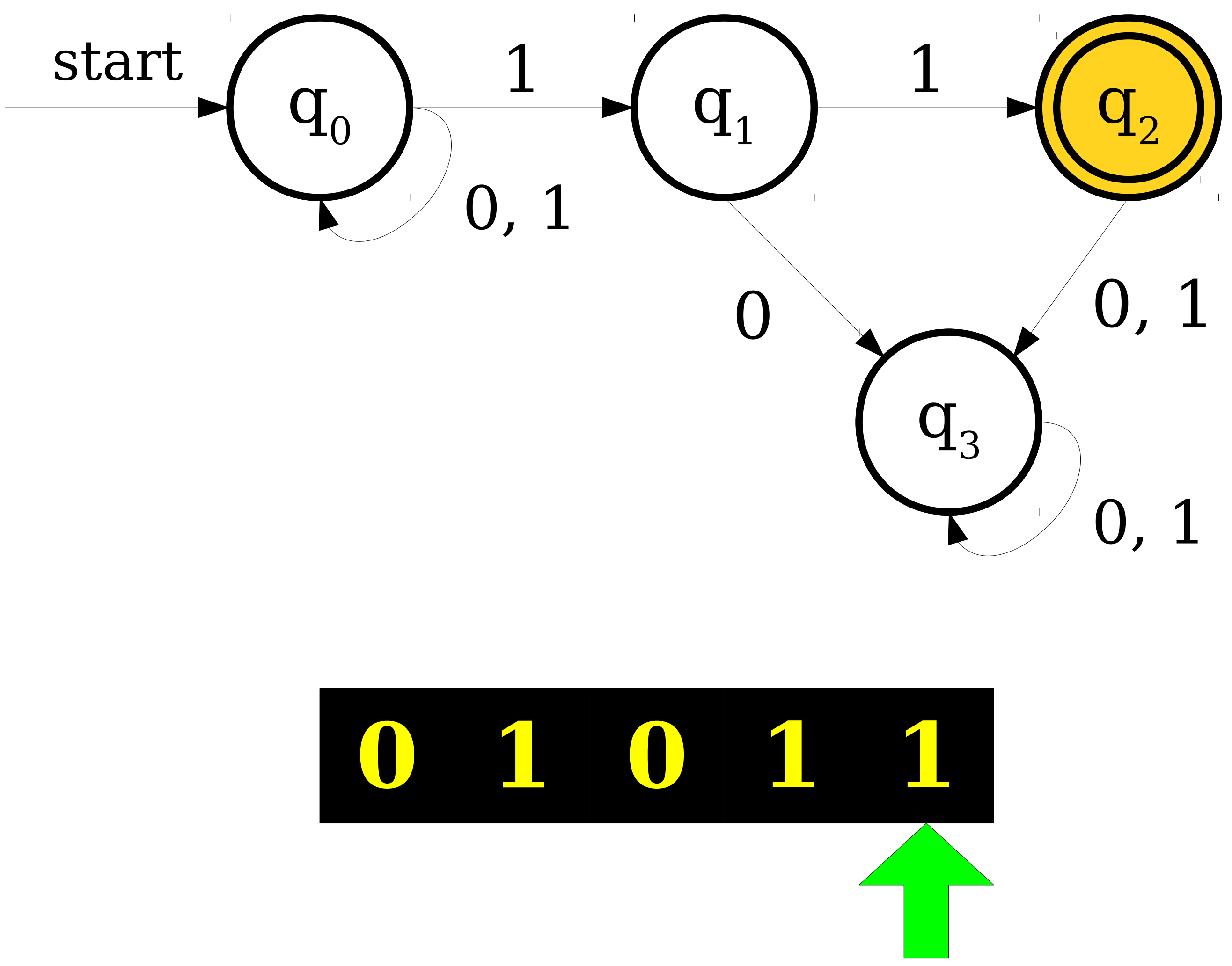
A Simple NFA



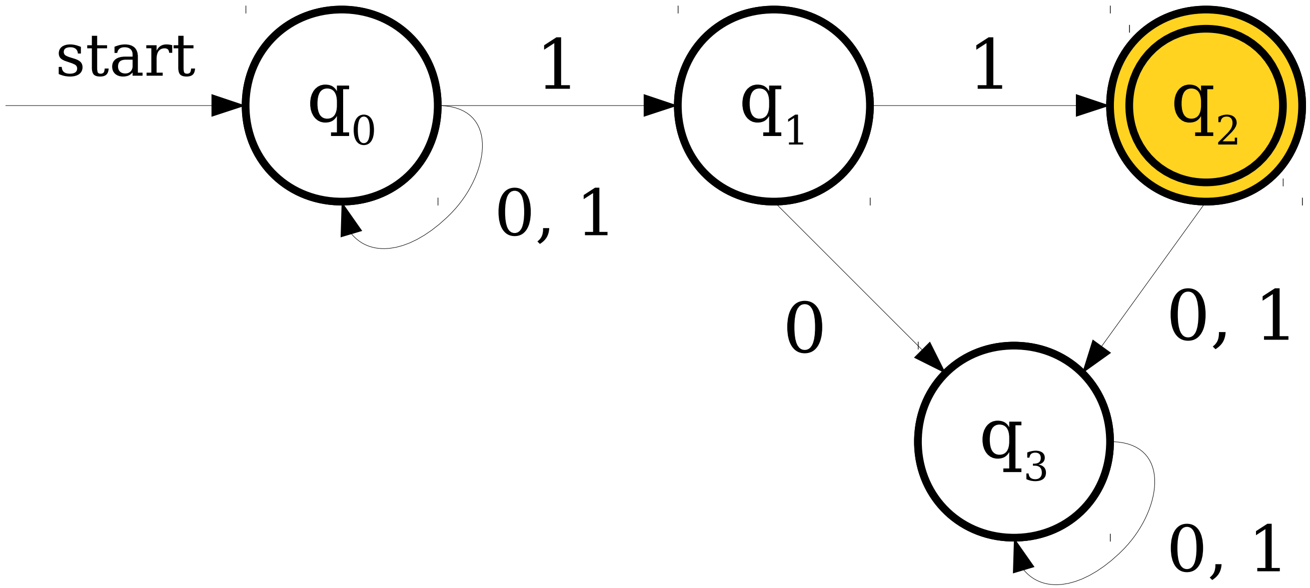
A Simple NFA



A Simple NFA

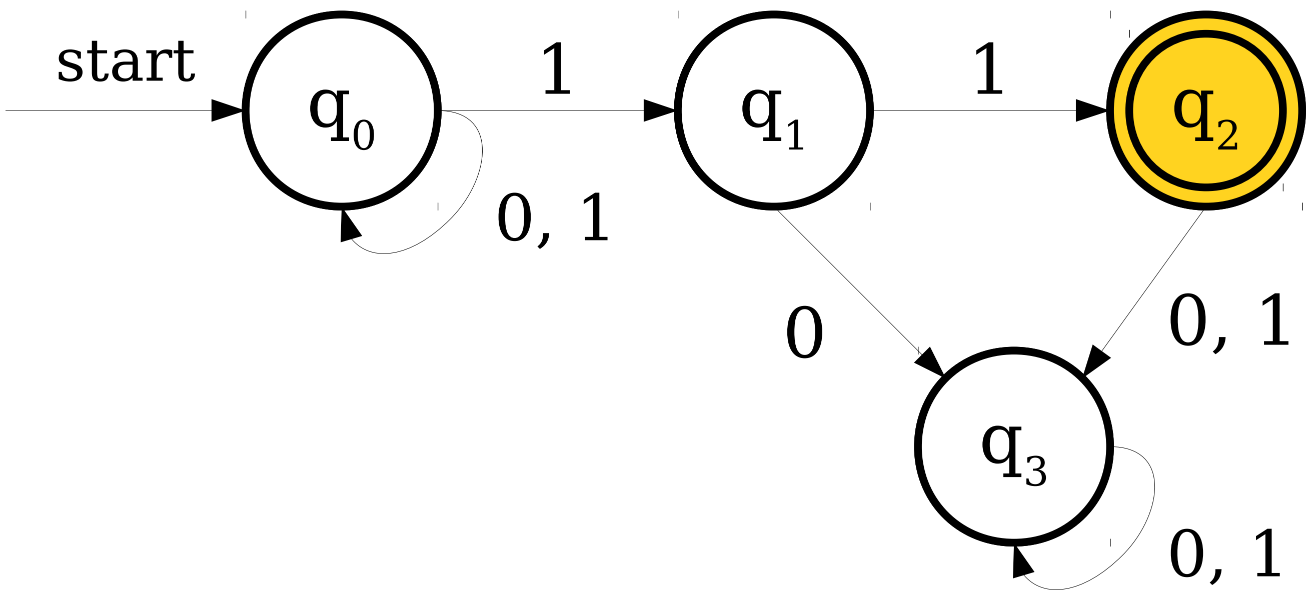


A Simple NFA



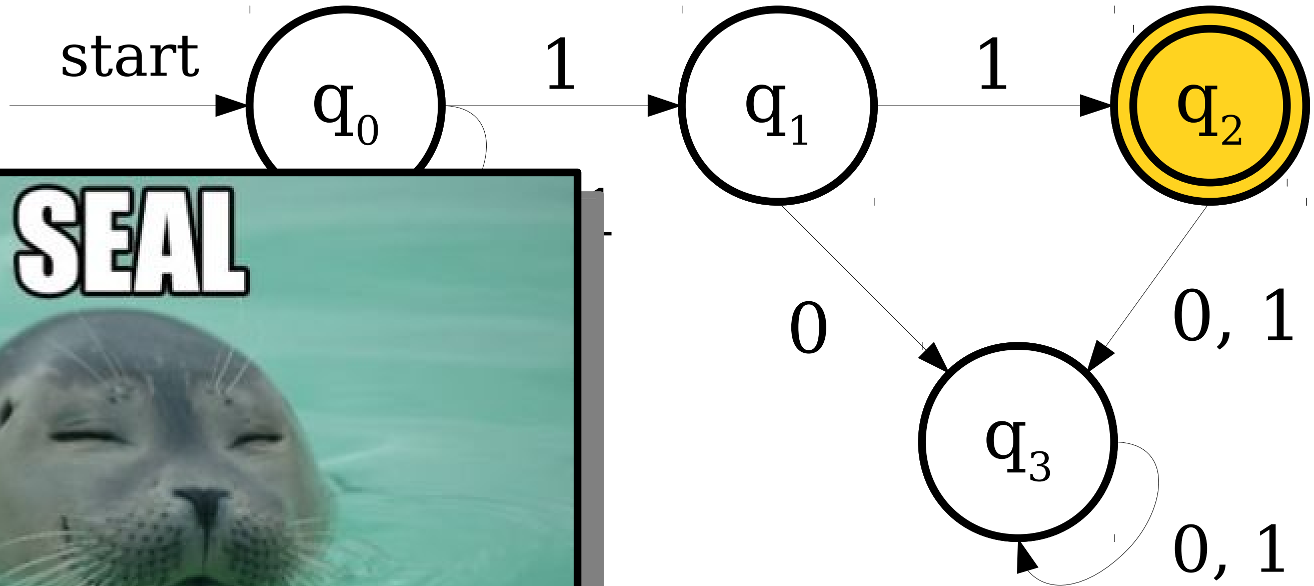
0 1 0 1 1

A Simple NFA



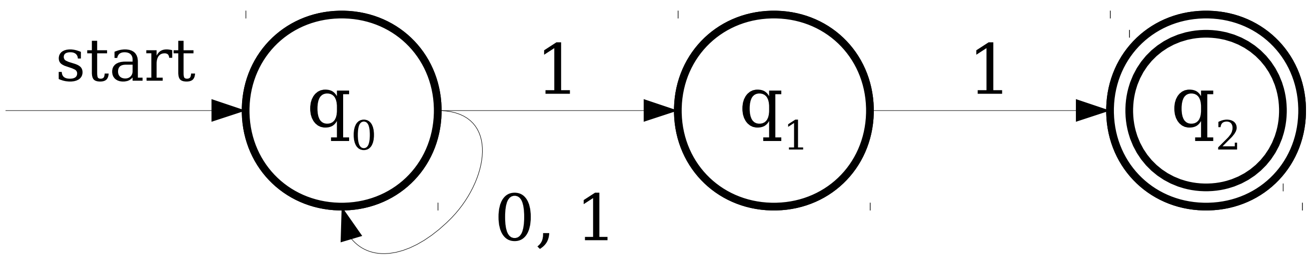
0 1 0 1 1

A Simple NFA

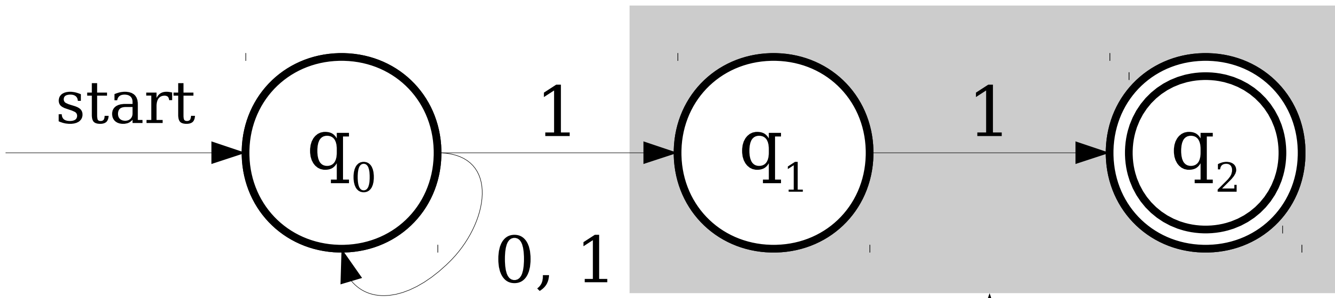


0 1 0 1 1

A More Complex NFA

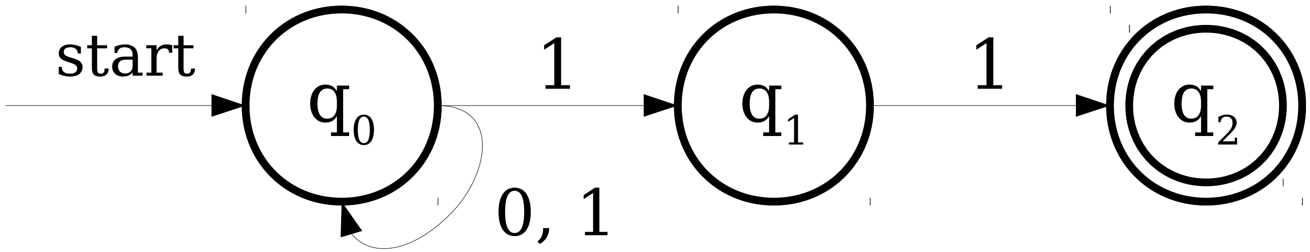


A More Complex NFA



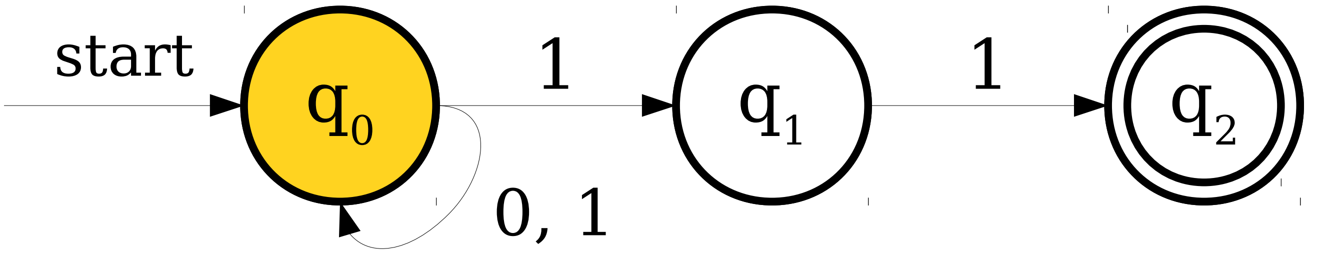
If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path rejects.

A More Complex NFA



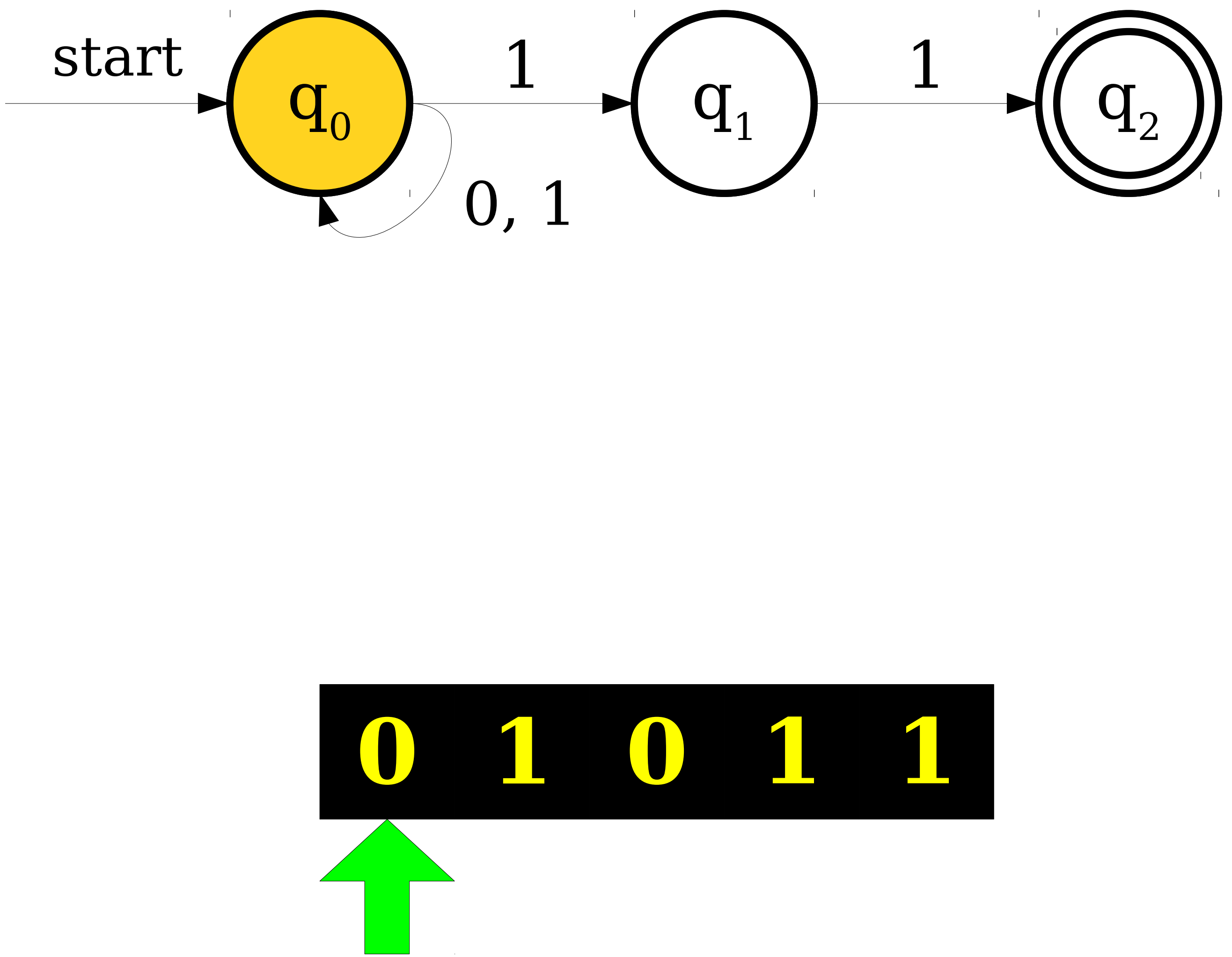
0 1 0 1 1

A More Complex NFA

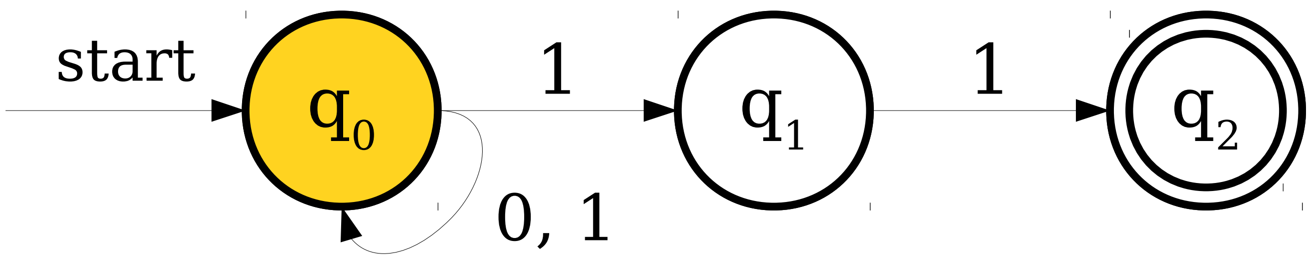


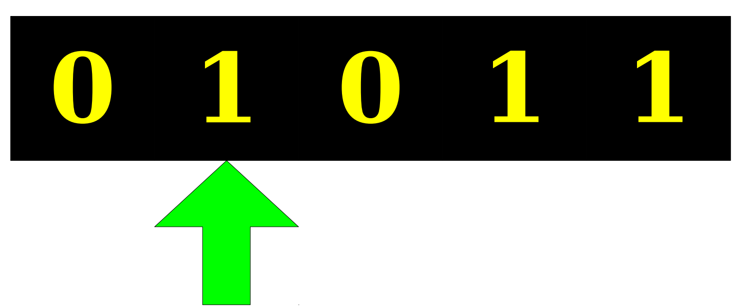
0 1 0 1 1

A More Complex NFA

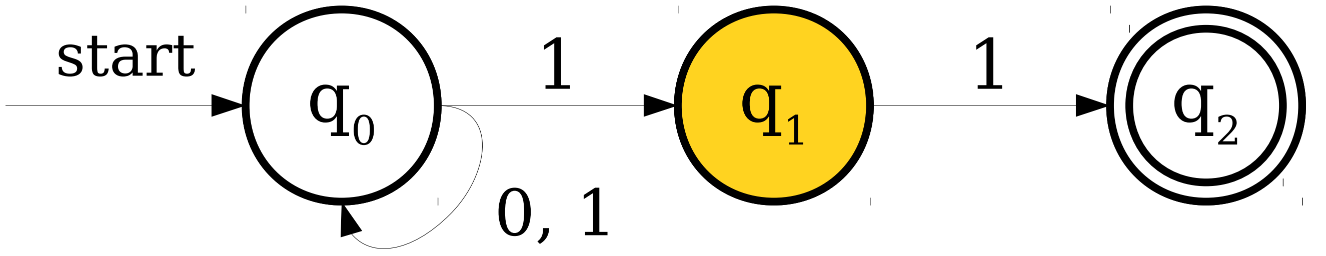


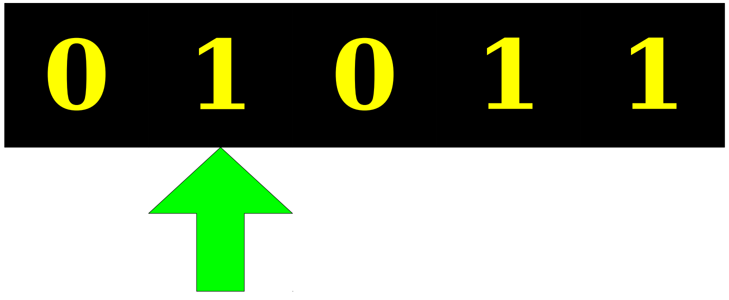
A More Complex NFA



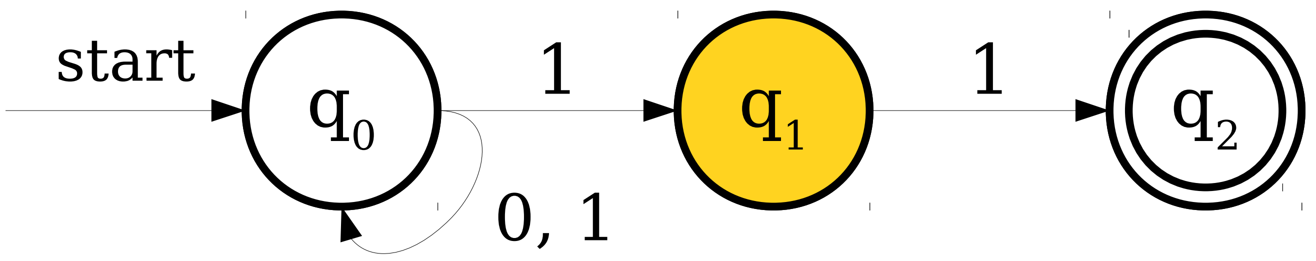


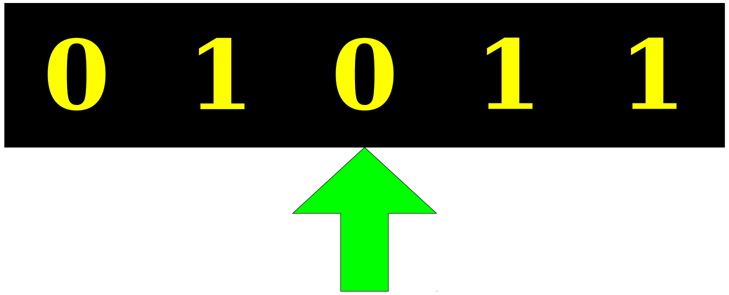
A More Complex NFA



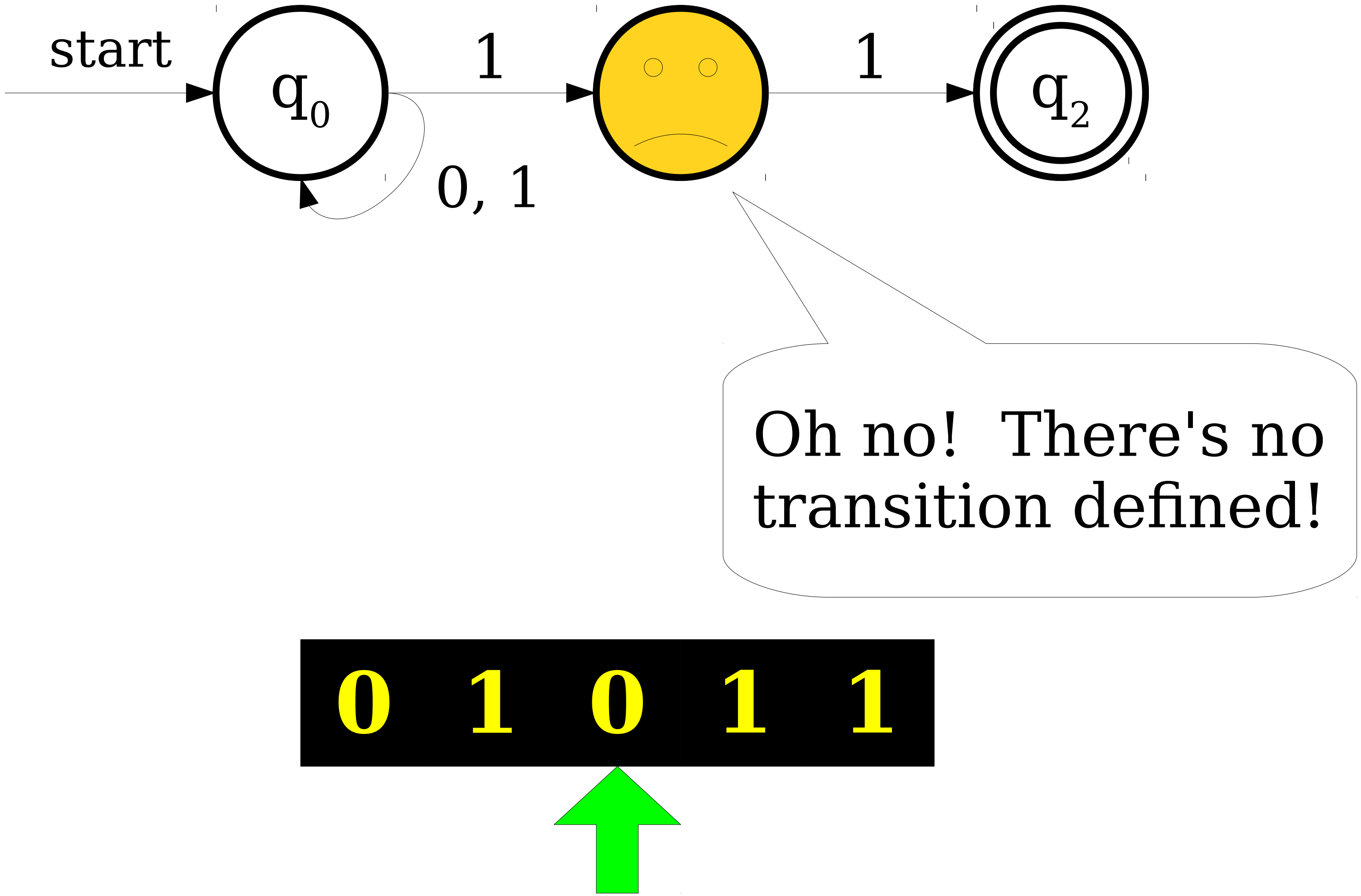


A More Complex NFA

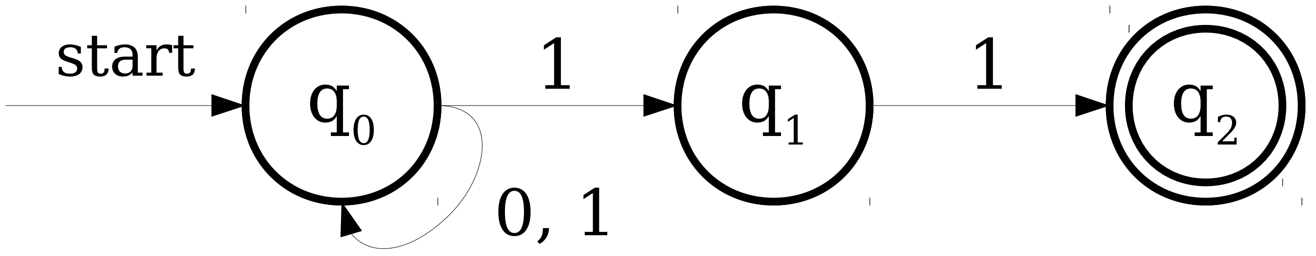


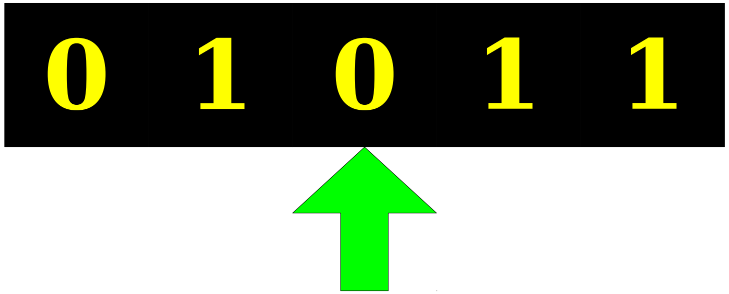


A More Complex NFA

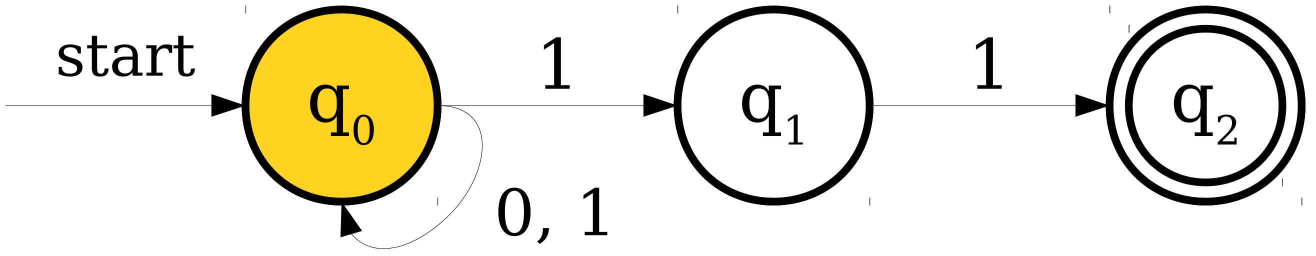


A More Complex NFA

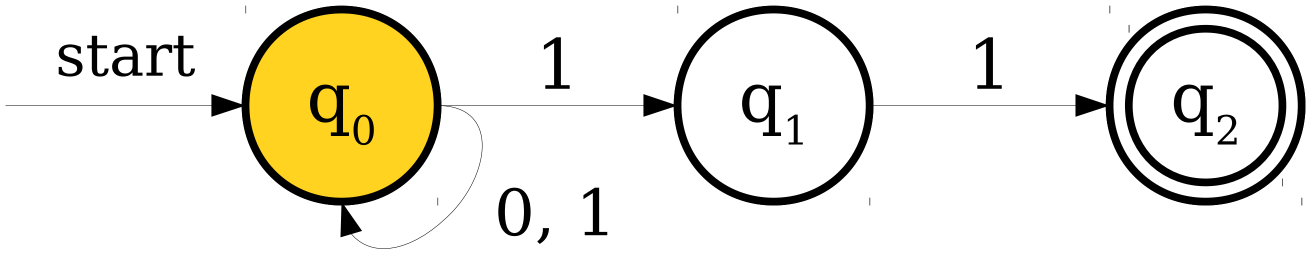


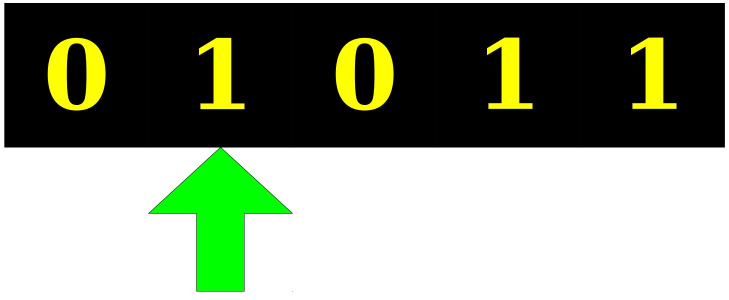


A More Complex NFA

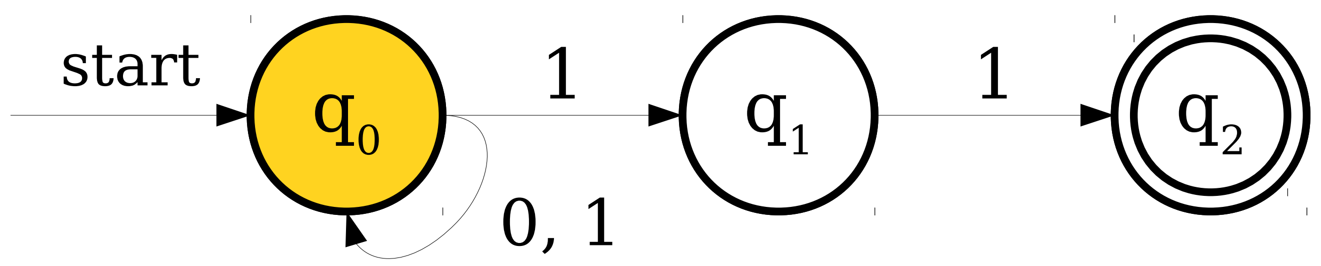


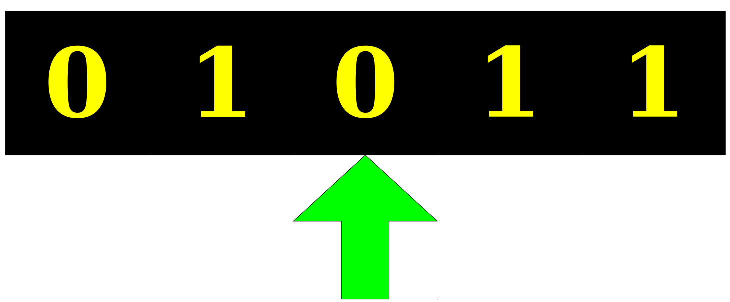
A More Complex NFA



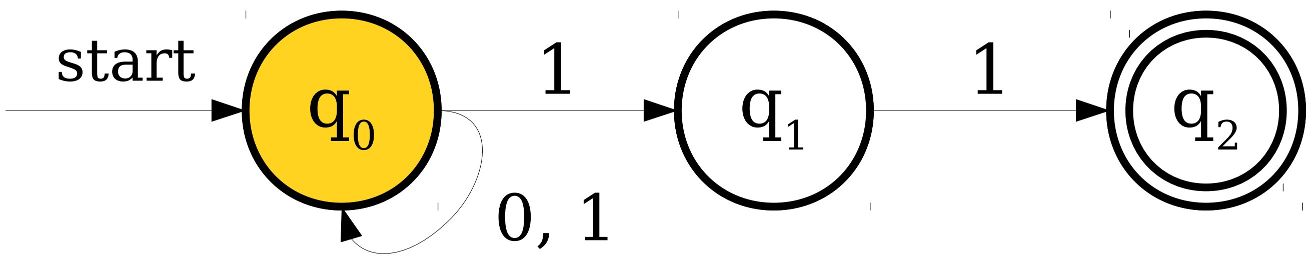


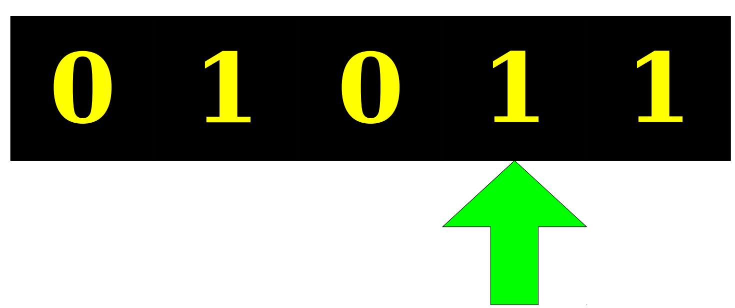
A More Complex NFA



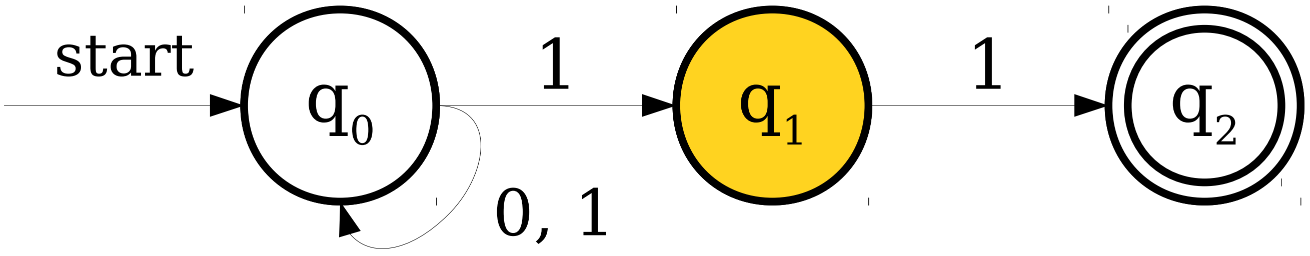


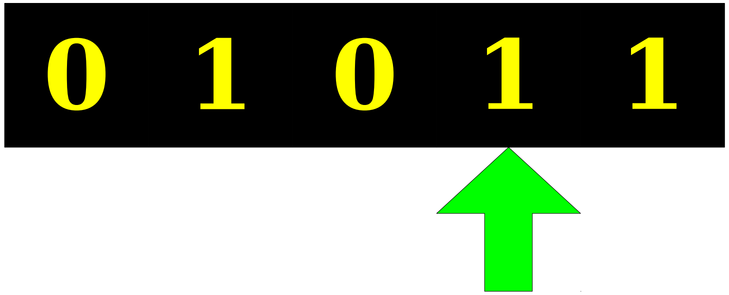
A More Complex NFA



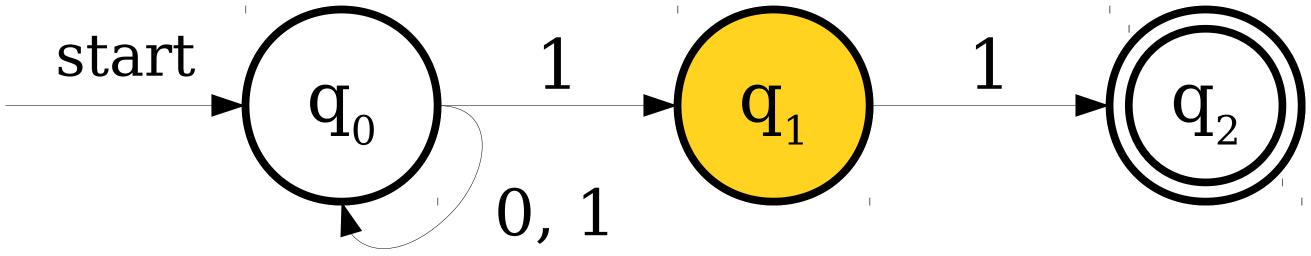


A More Complex NFA



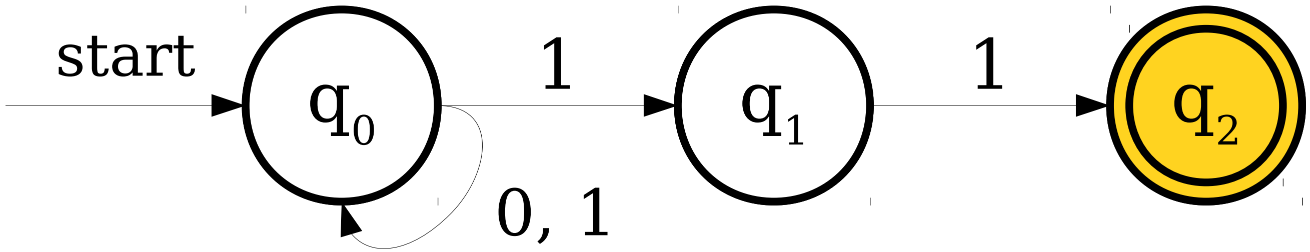


A More Complex NFA

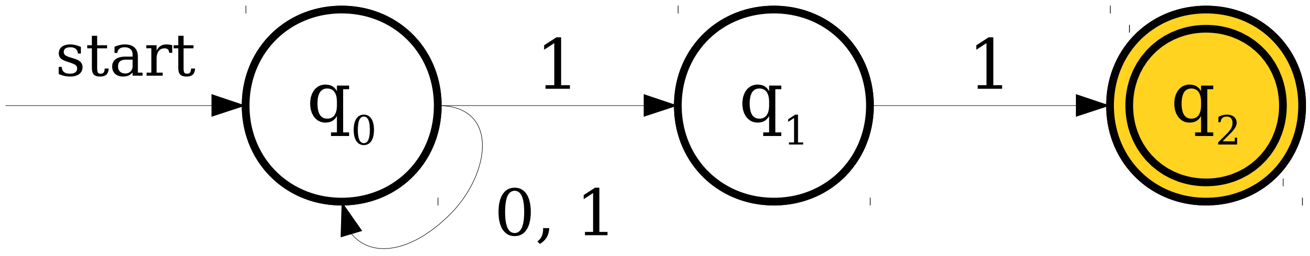


0 1 0 1 1

A More Complex NFA

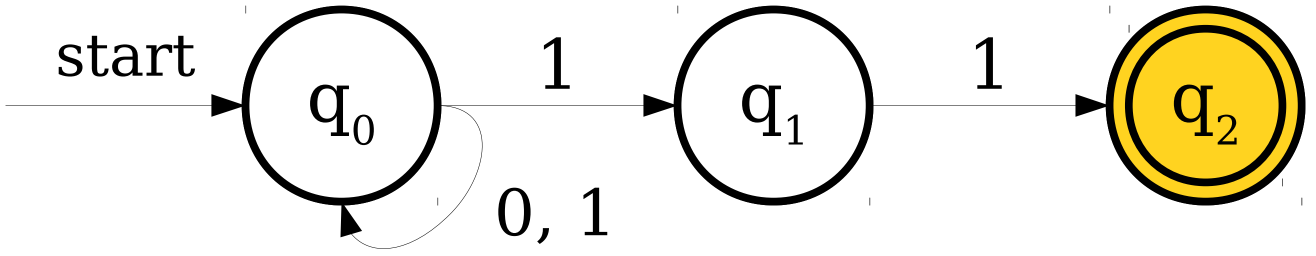


A More Complex NFA



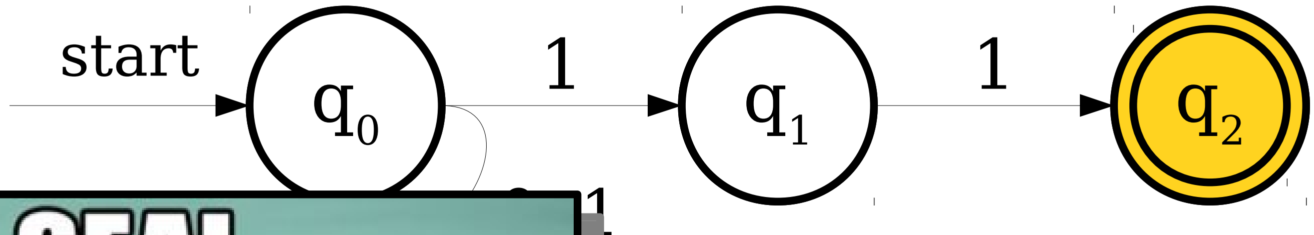
0 1 0 1 1

A More Complex NFA



0 1 0 1 1

A More Complex NFA



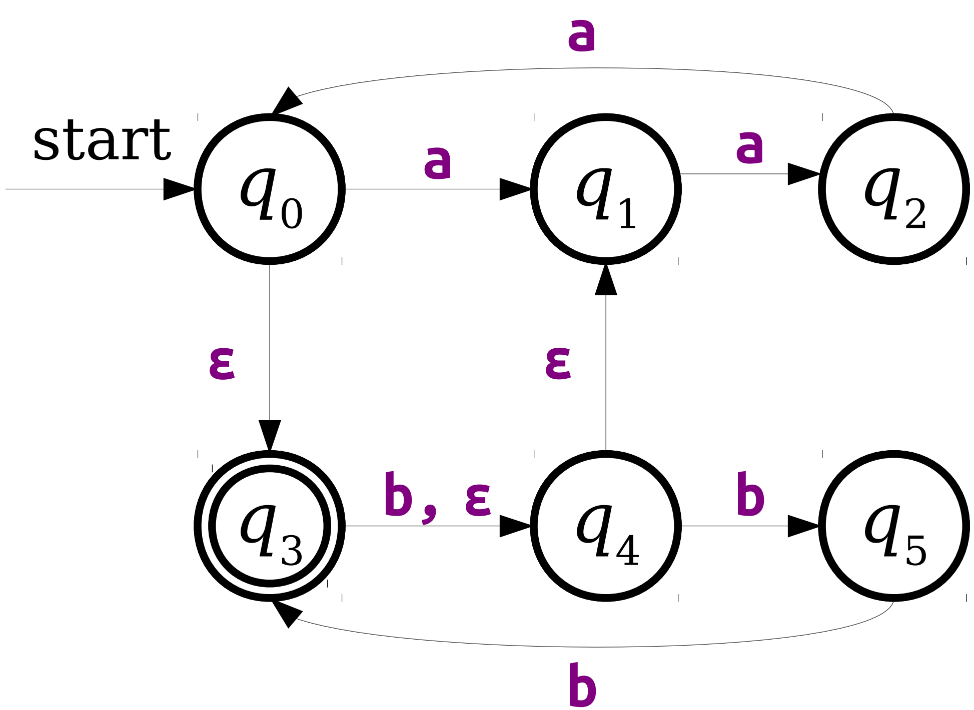
0 1 1

ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

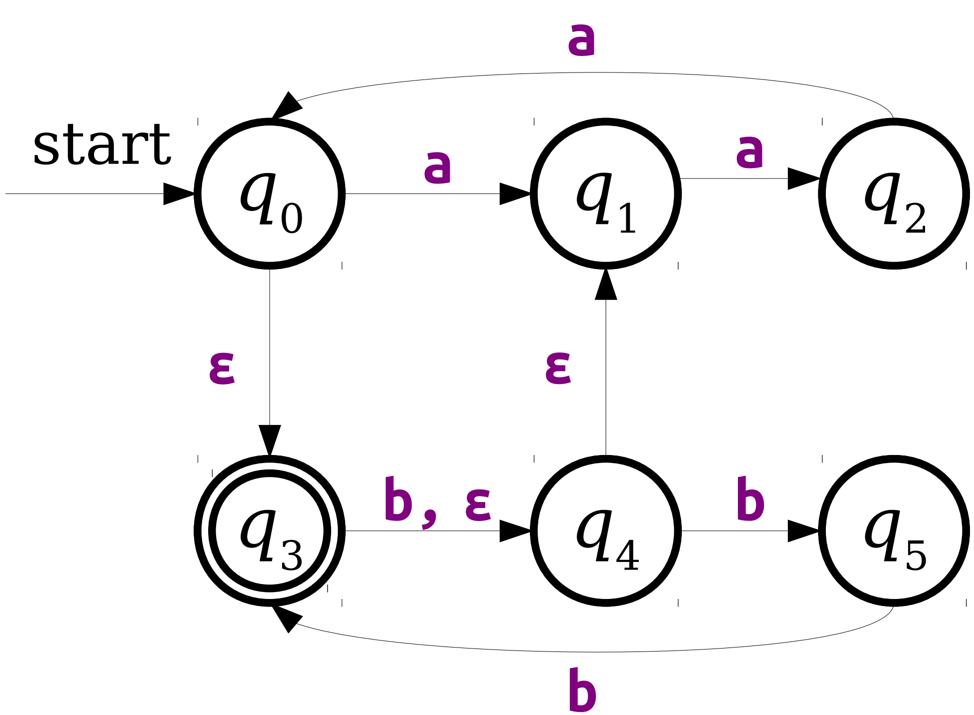
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

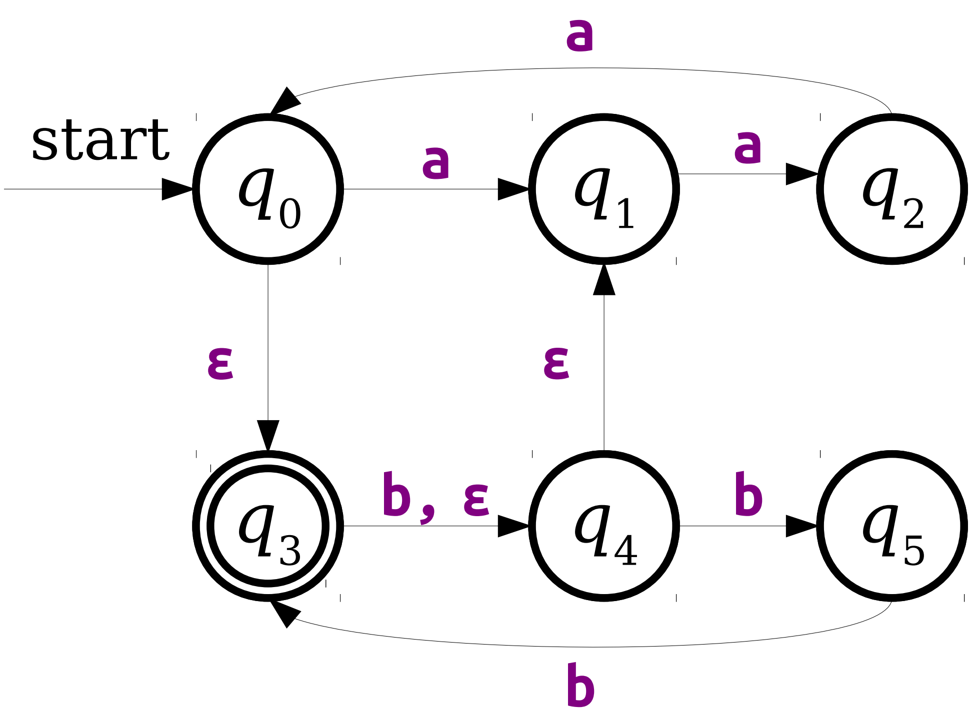
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



b a a b b

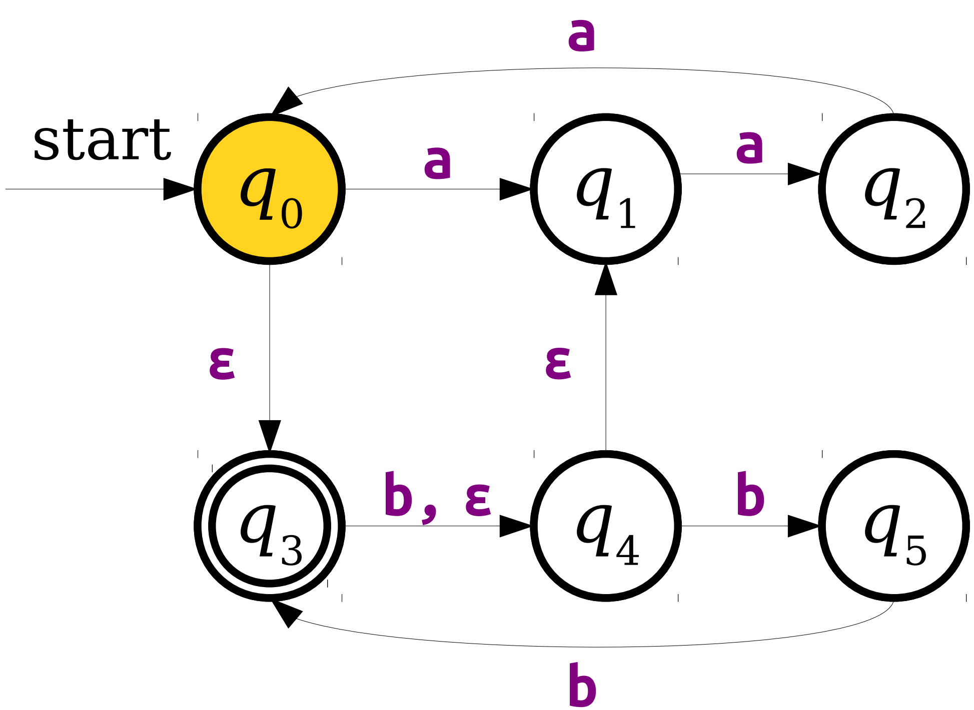
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



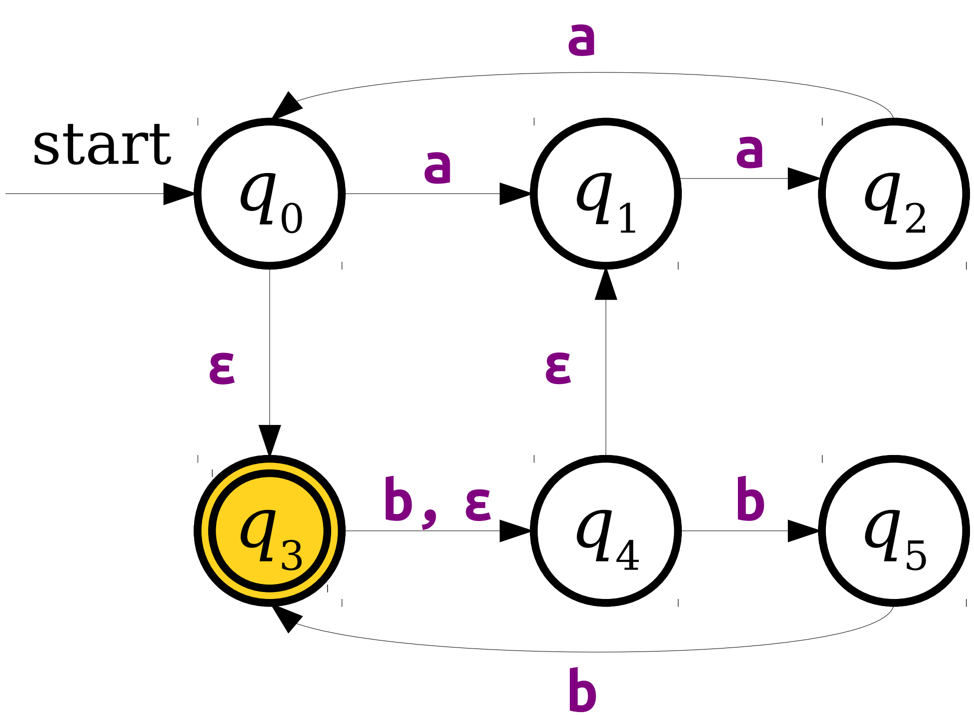
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



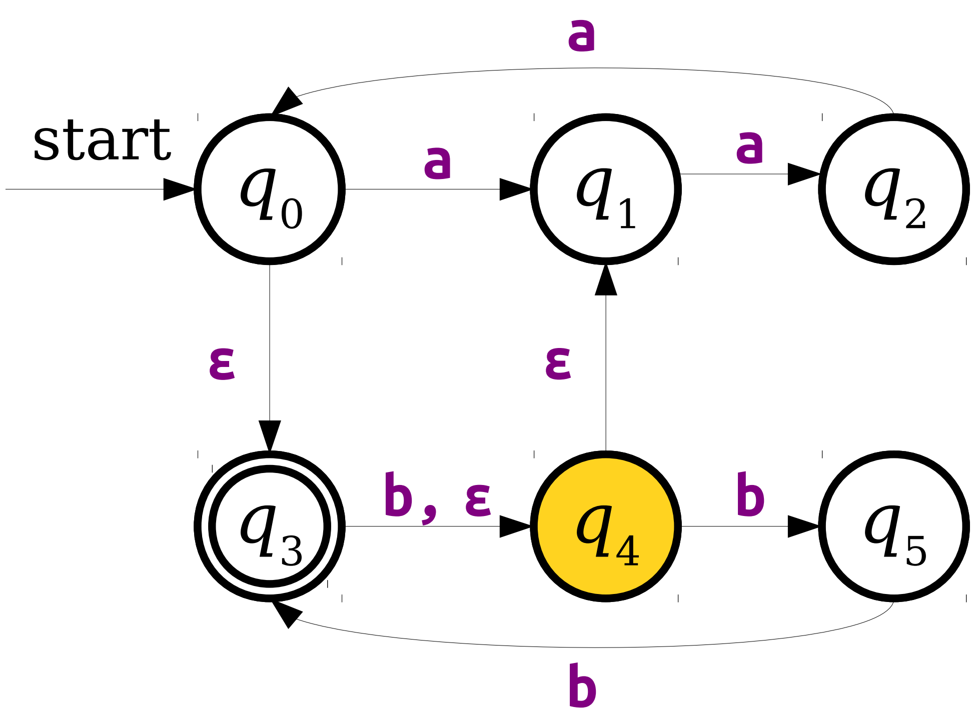
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



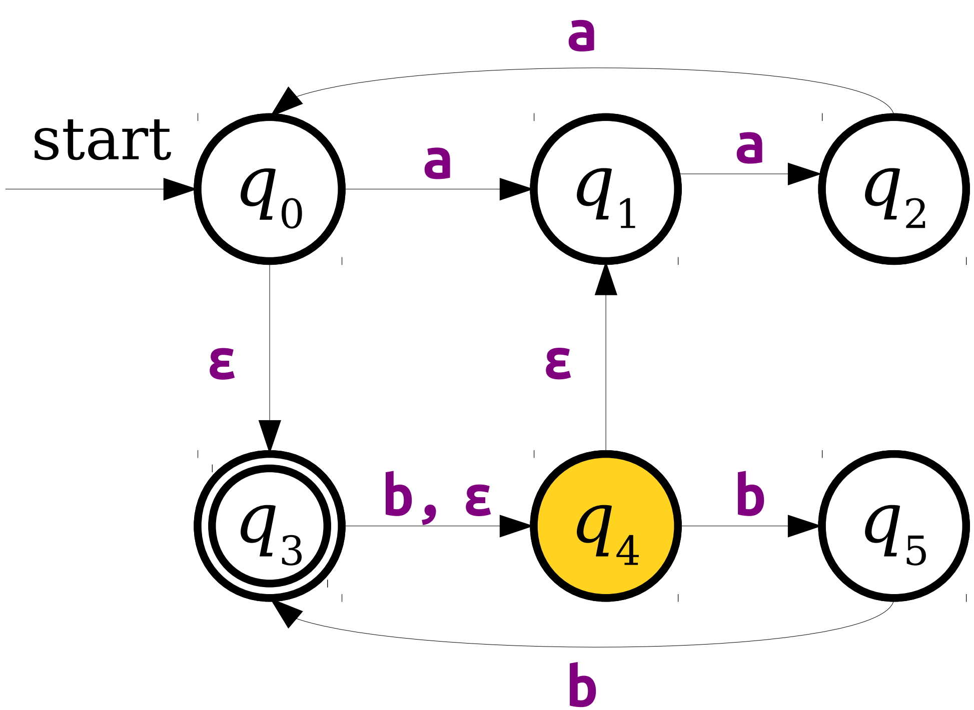
ε-Transitions

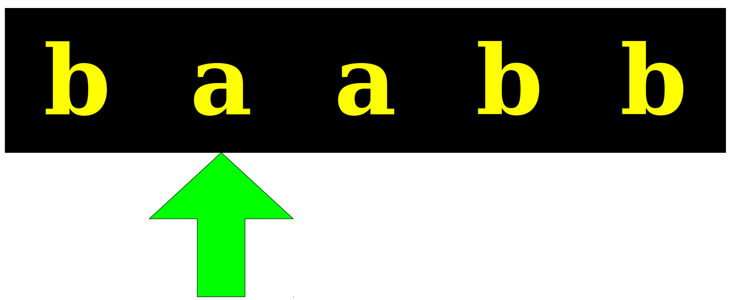
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

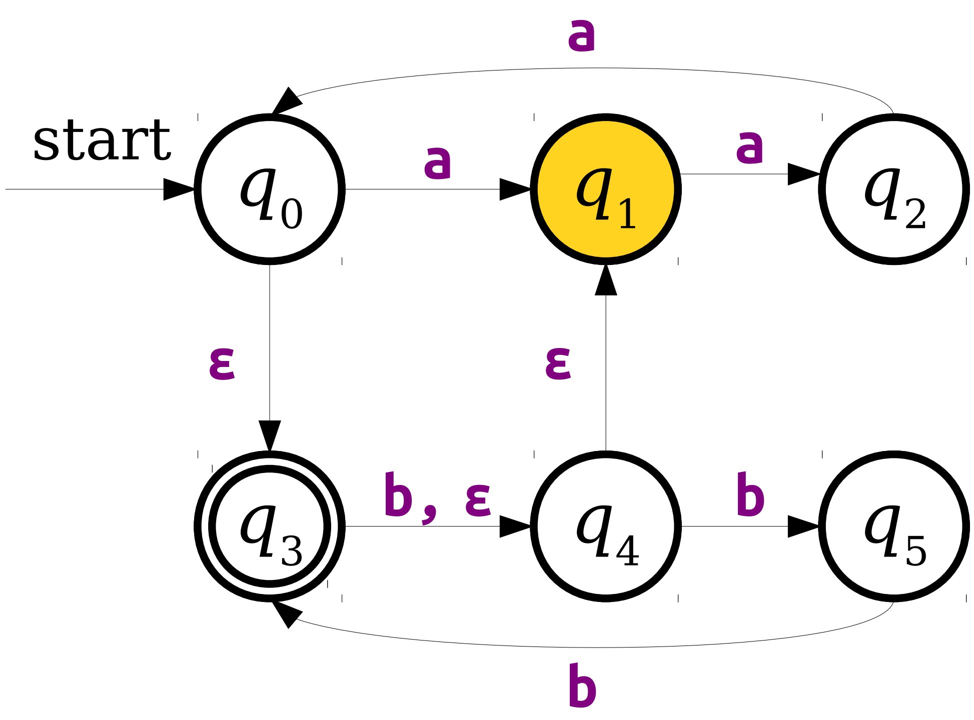
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

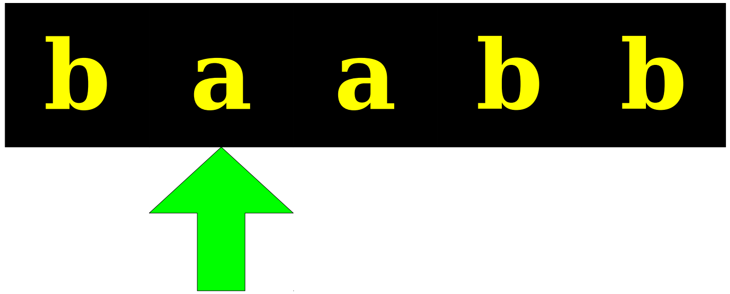




ε-Transitions

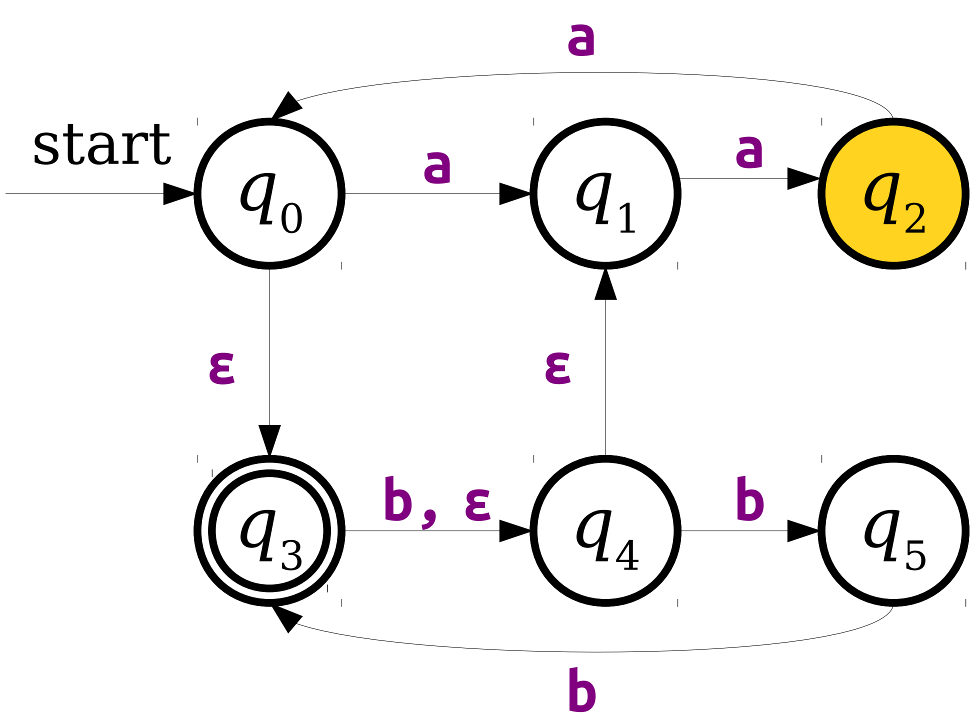
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

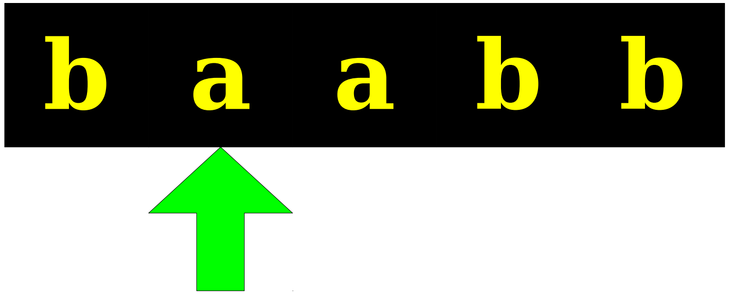




ε-Transitions

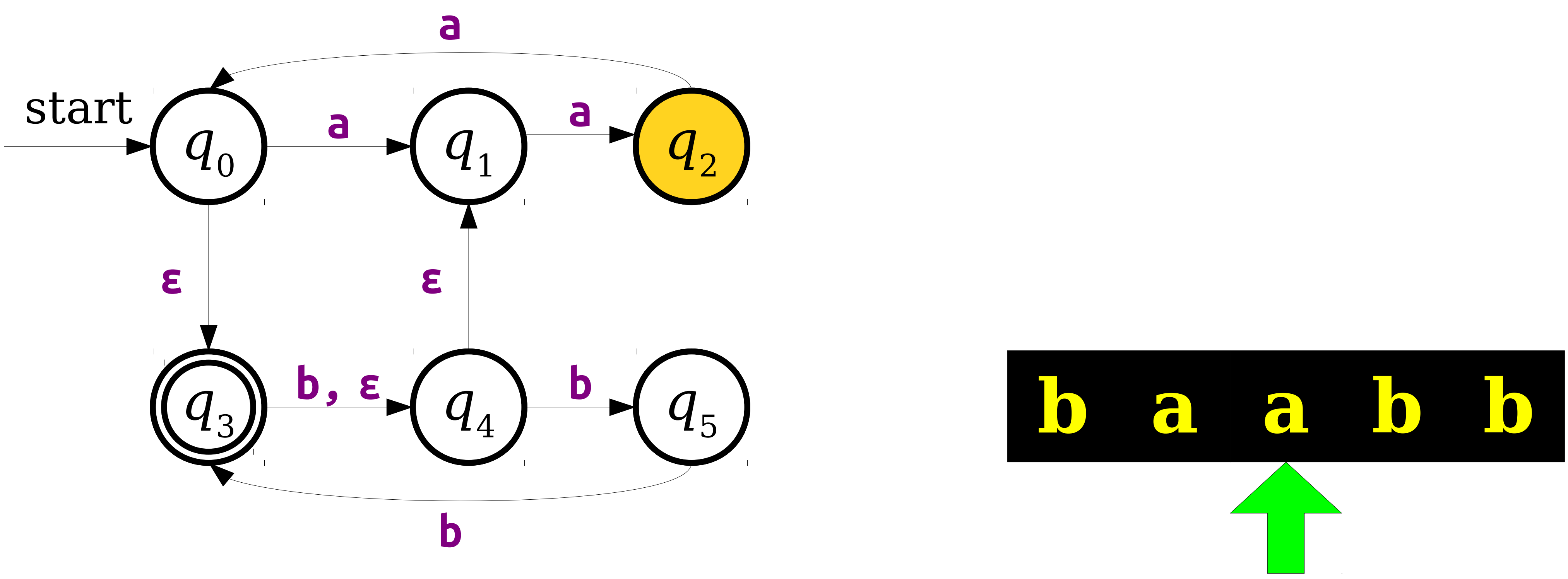
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.





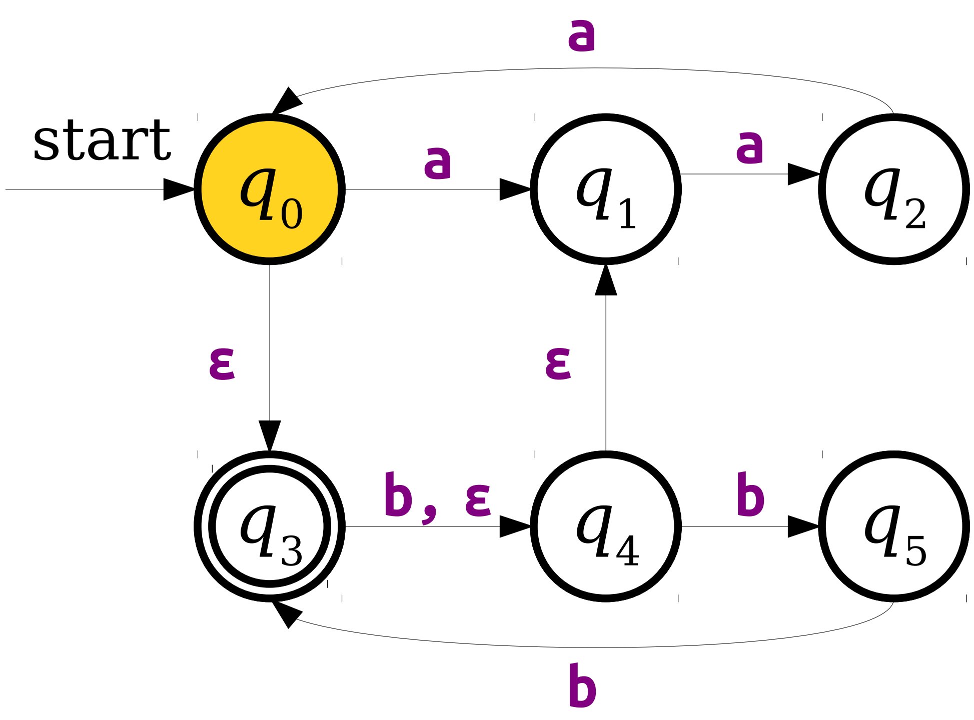
ε-Transitions

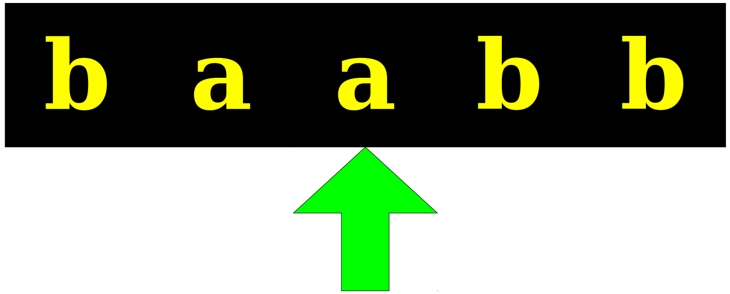
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

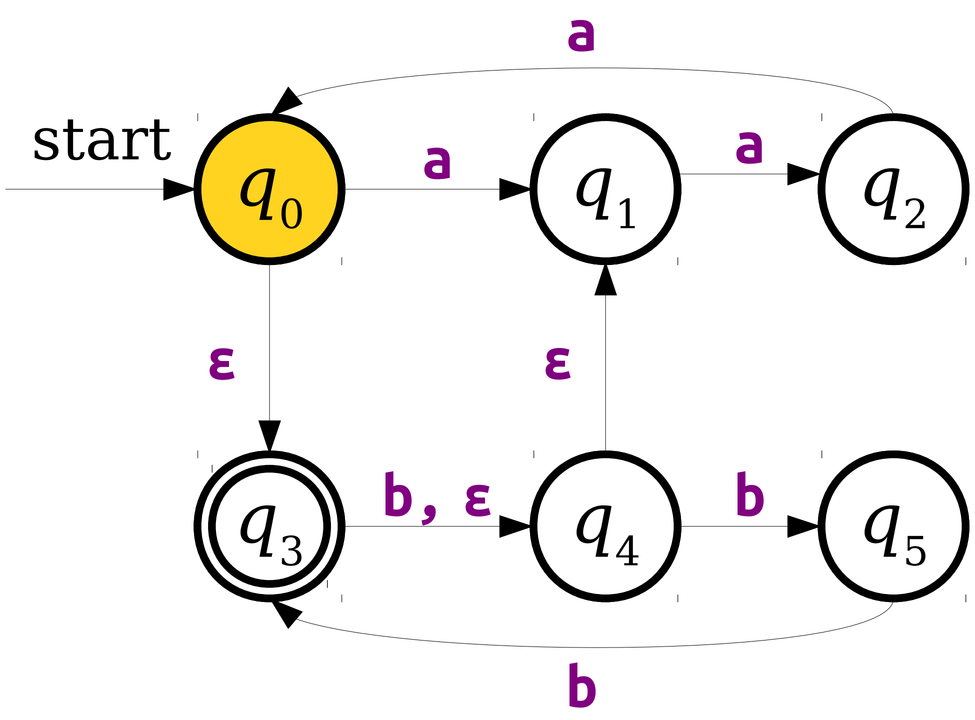
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

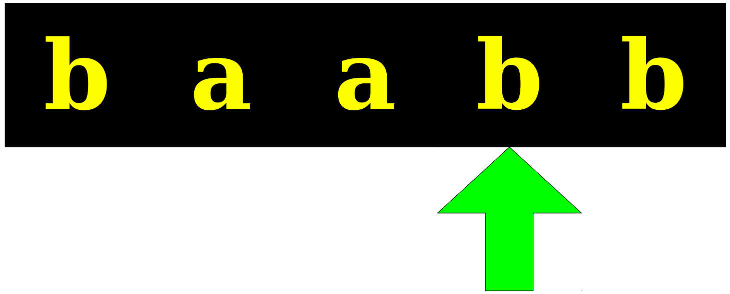




ε-Transitions

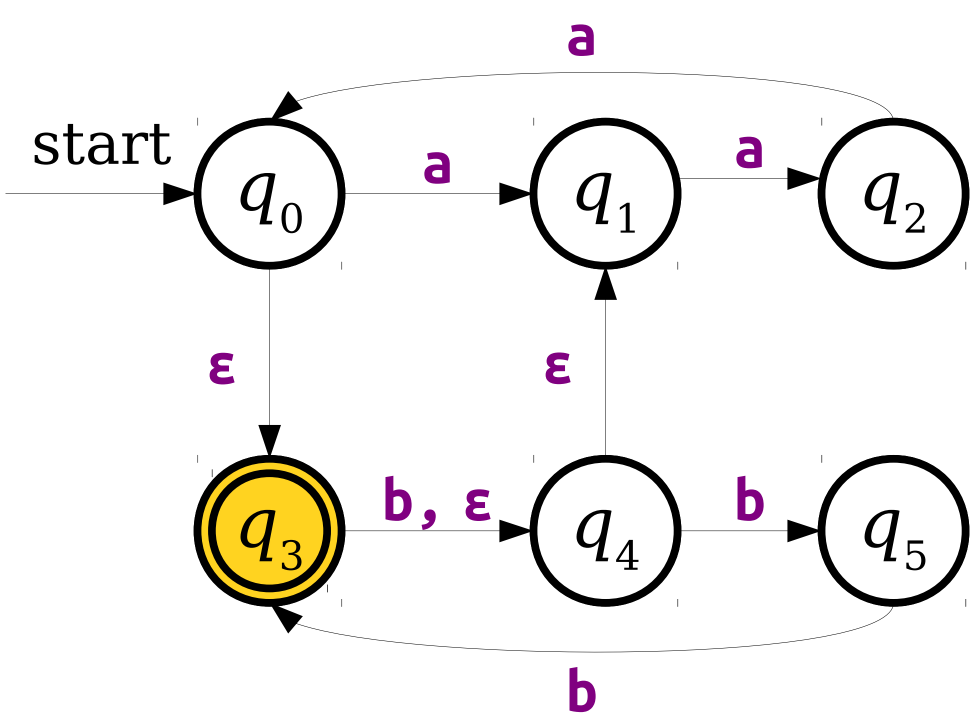
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

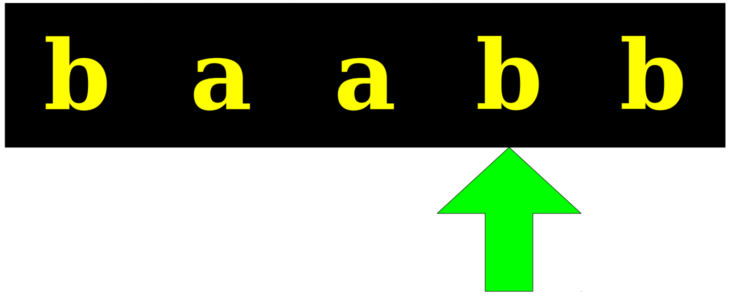




ε-Transitions

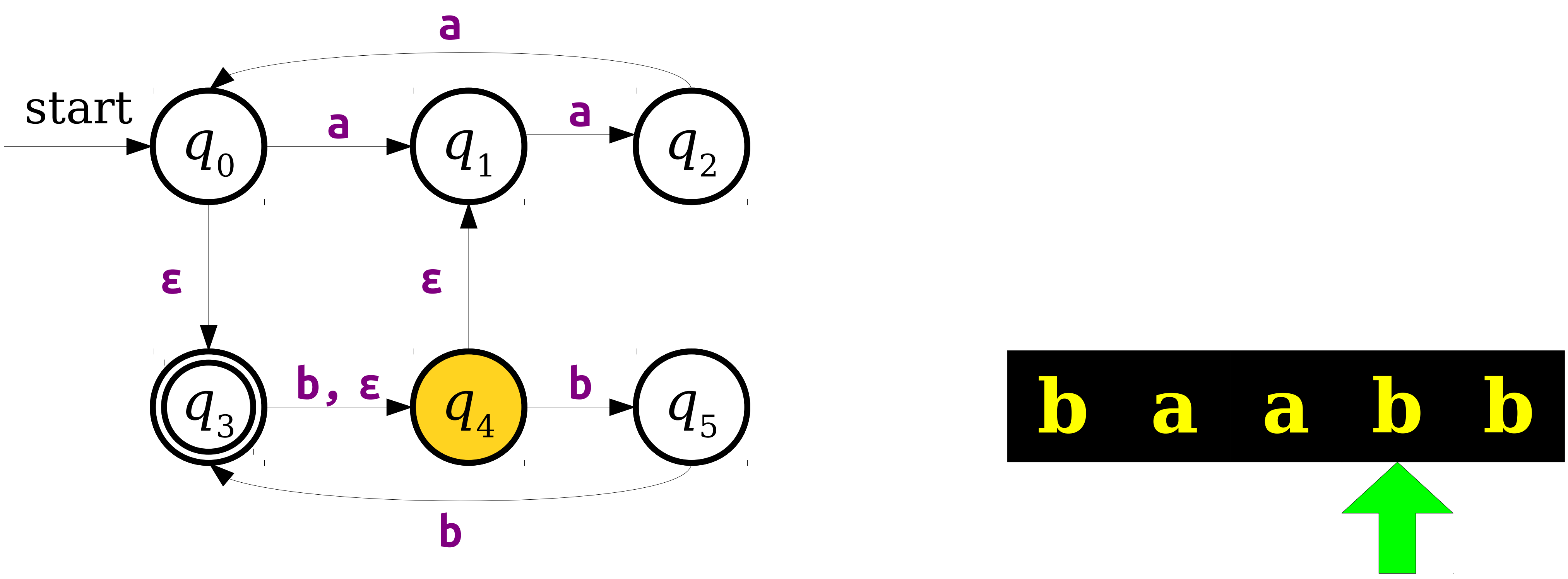
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.





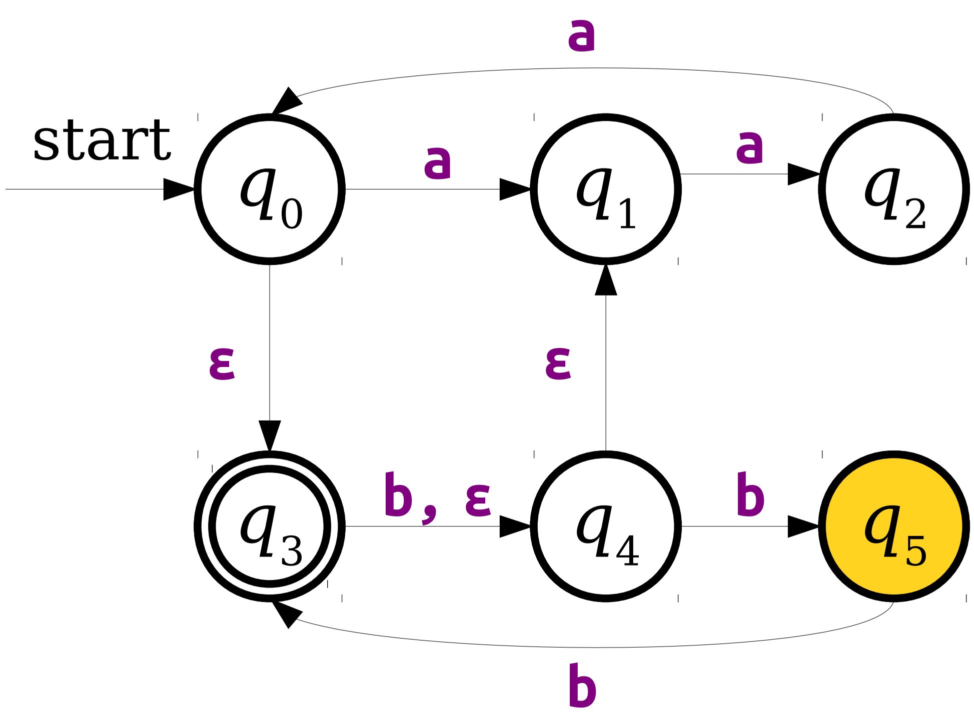
ε-Transitions

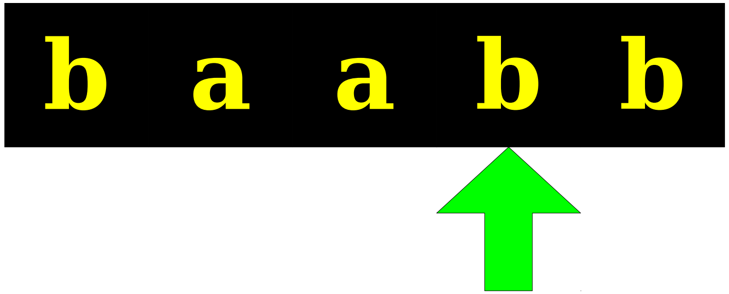
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

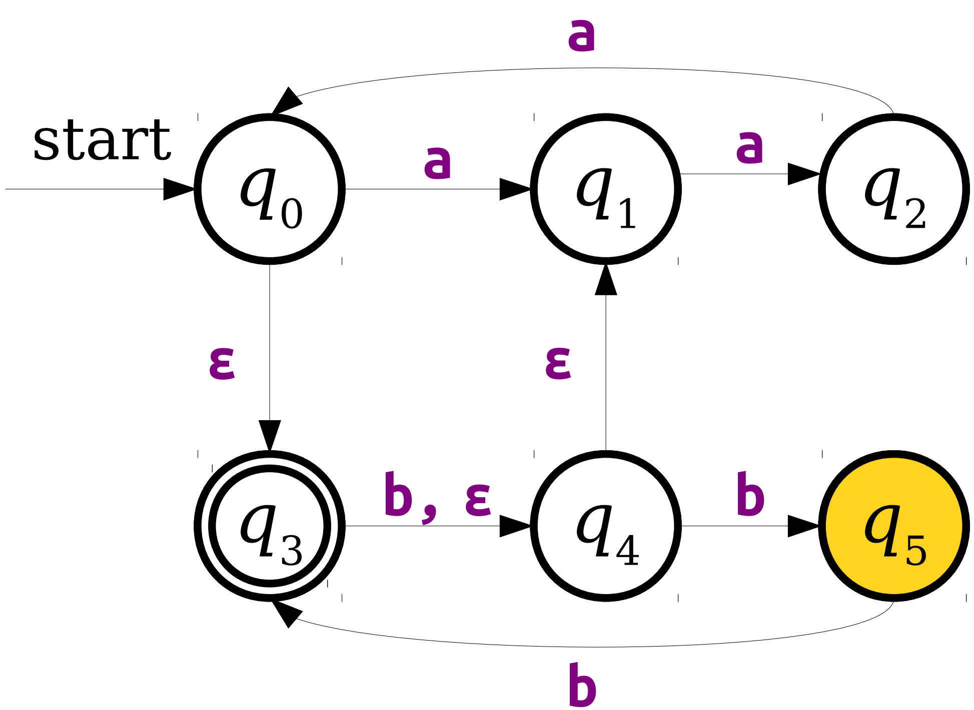
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.





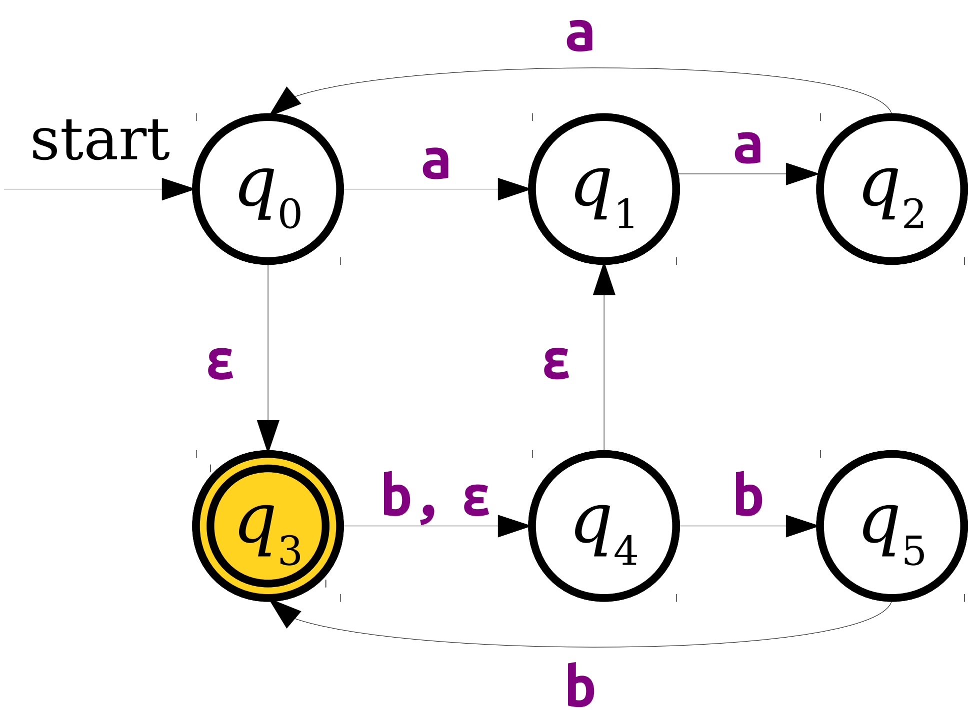
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



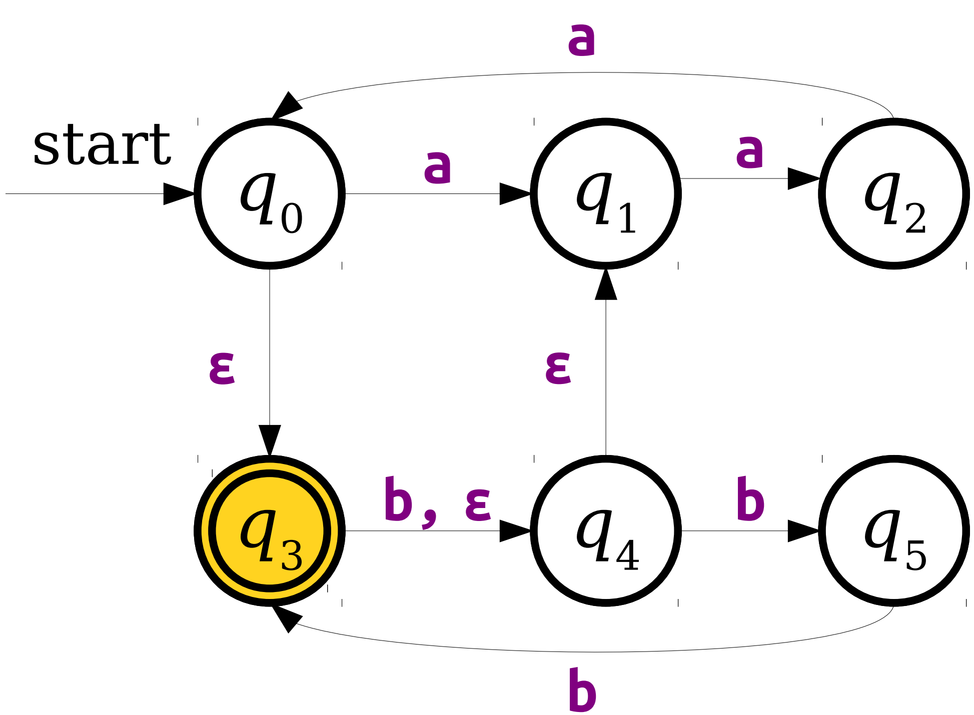
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



b a a b b

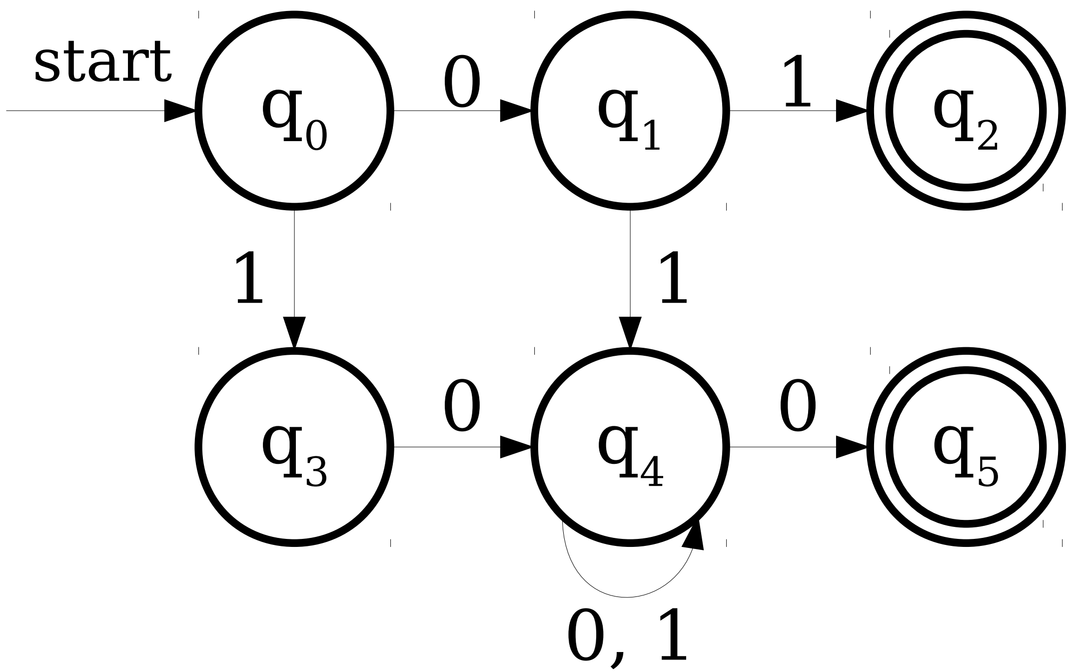
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
- NFAs are not *required* to follow ε-transitions. It's simply another option at the machine's disposal.

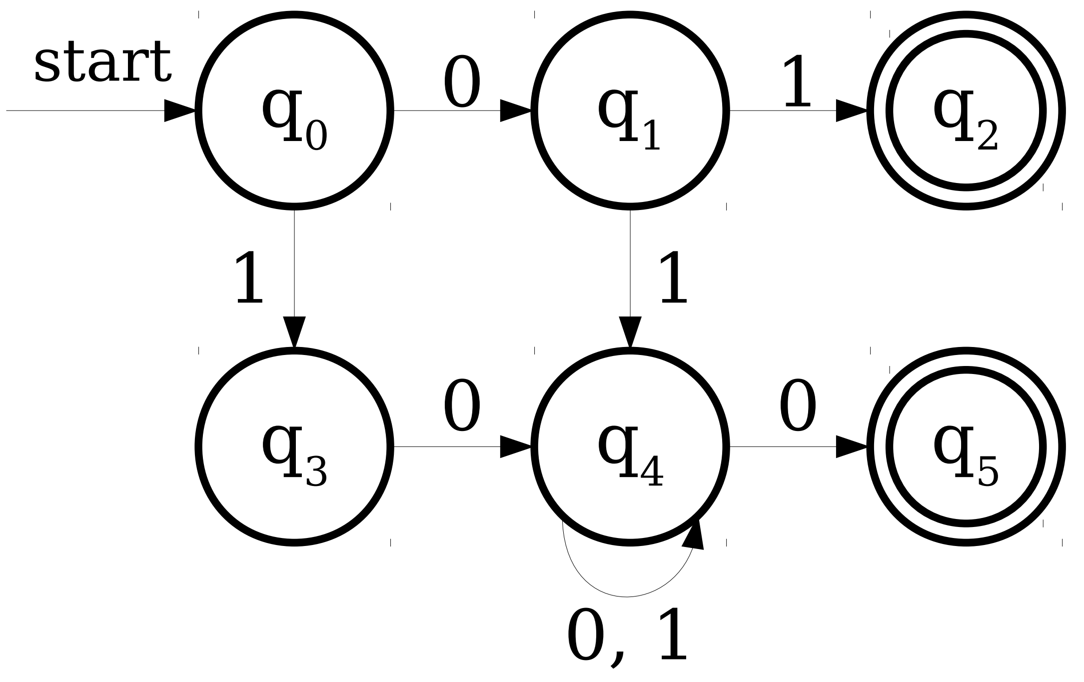
Intuiting Nondeterminism

- Nondeterministic machines are a serious departure from physical computers.
- How can we build up an intuition for them?
- Three approaches:
    - **Tree computation**
    - **Perfect guessing**
    - **Massive parallelism**

Tree Computation

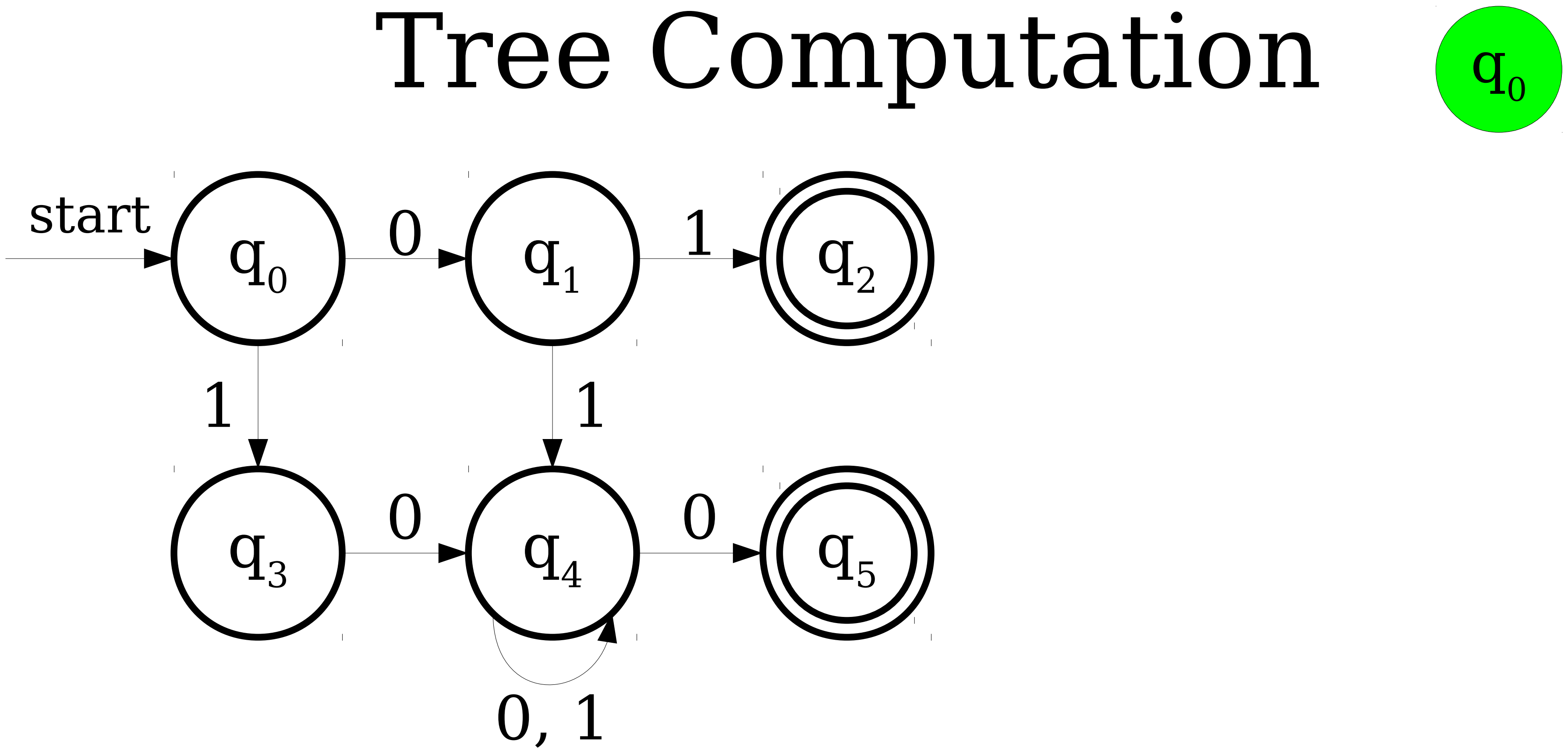


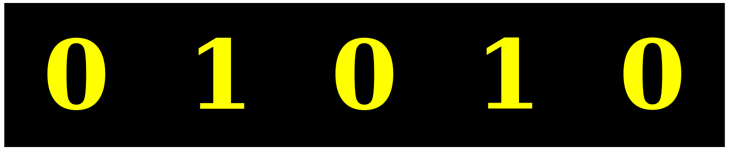
Tree Computation



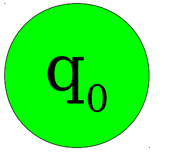
0 1 0 1 0

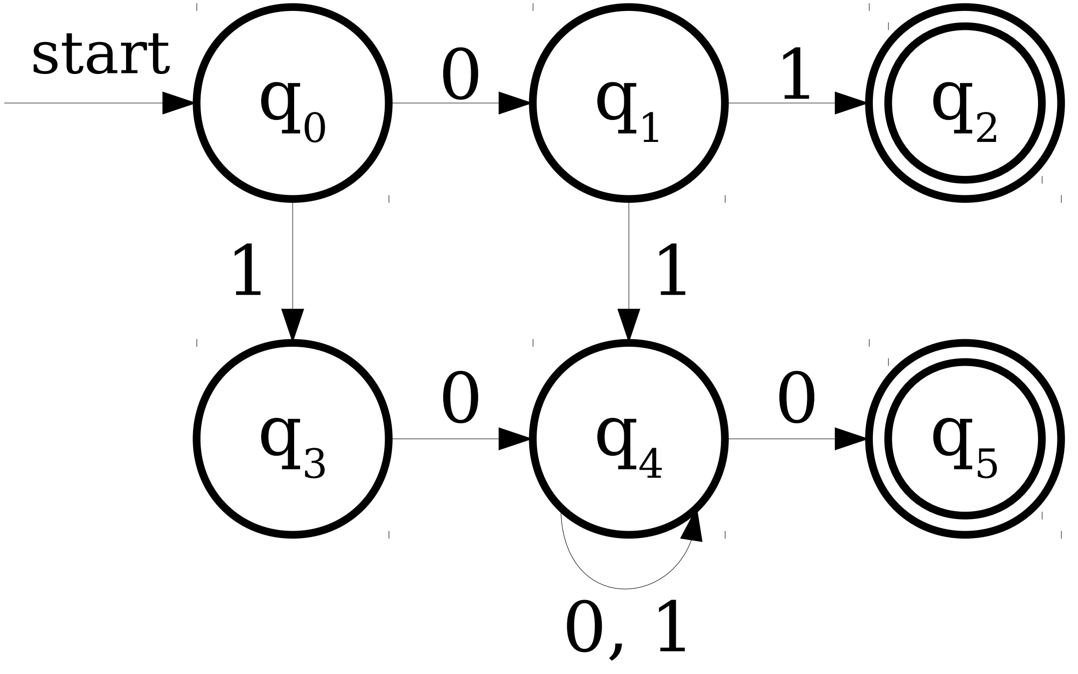
Tree Computation



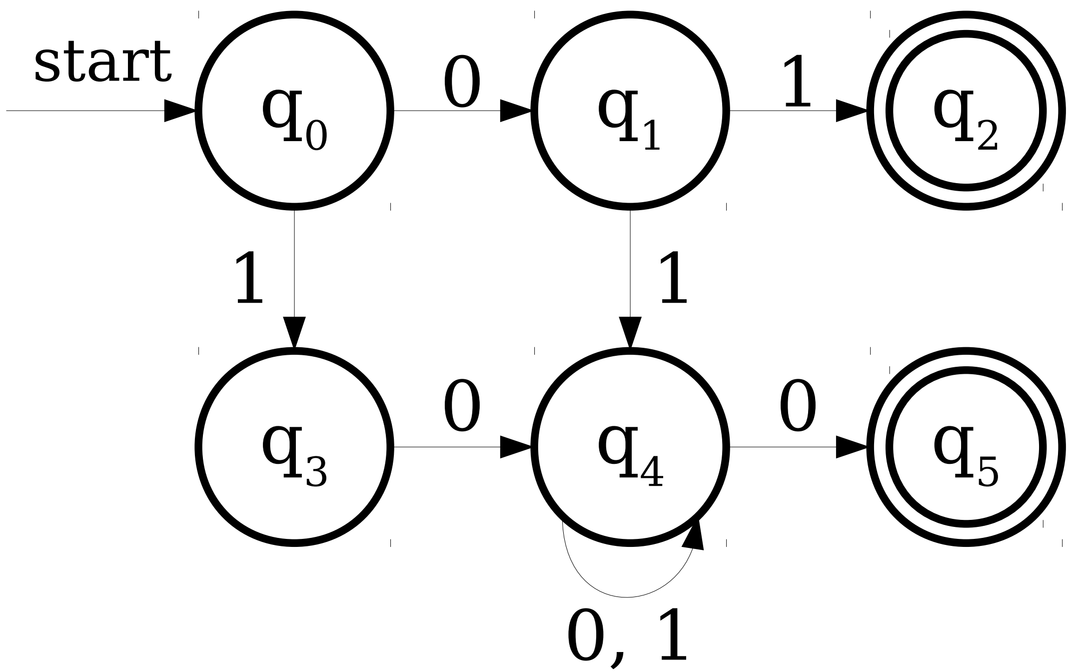


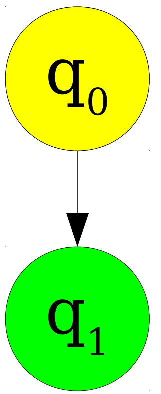
Tree Computation



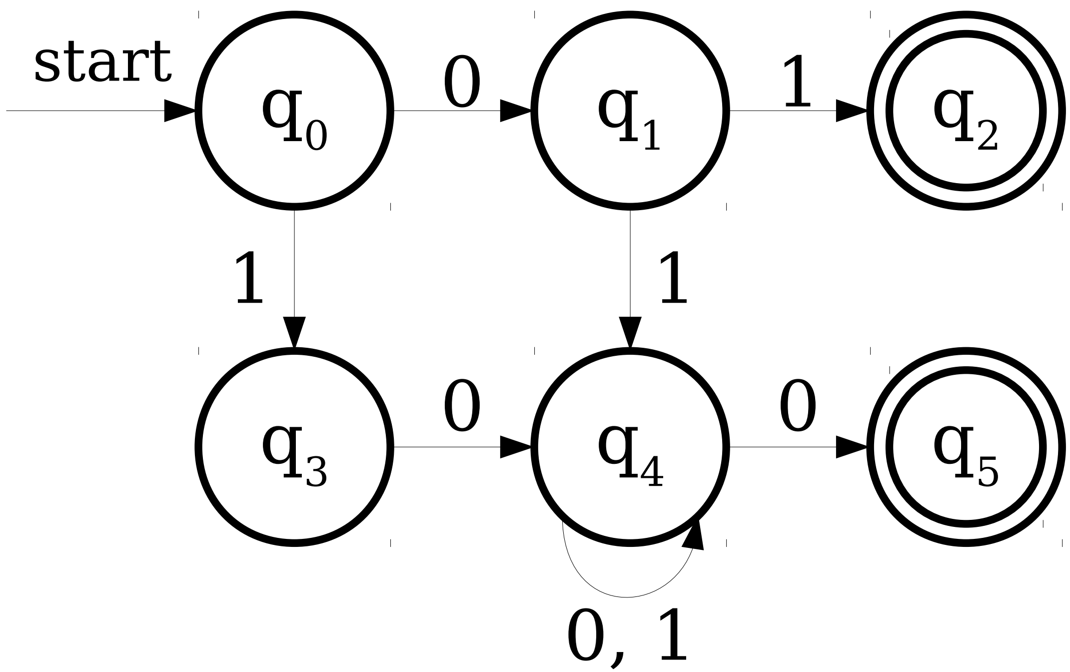


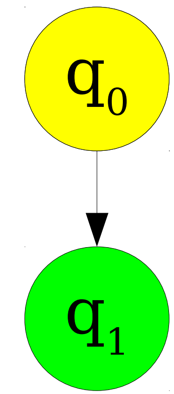
Tree Computation

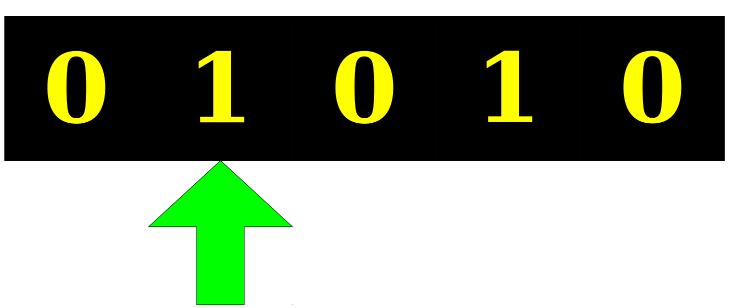




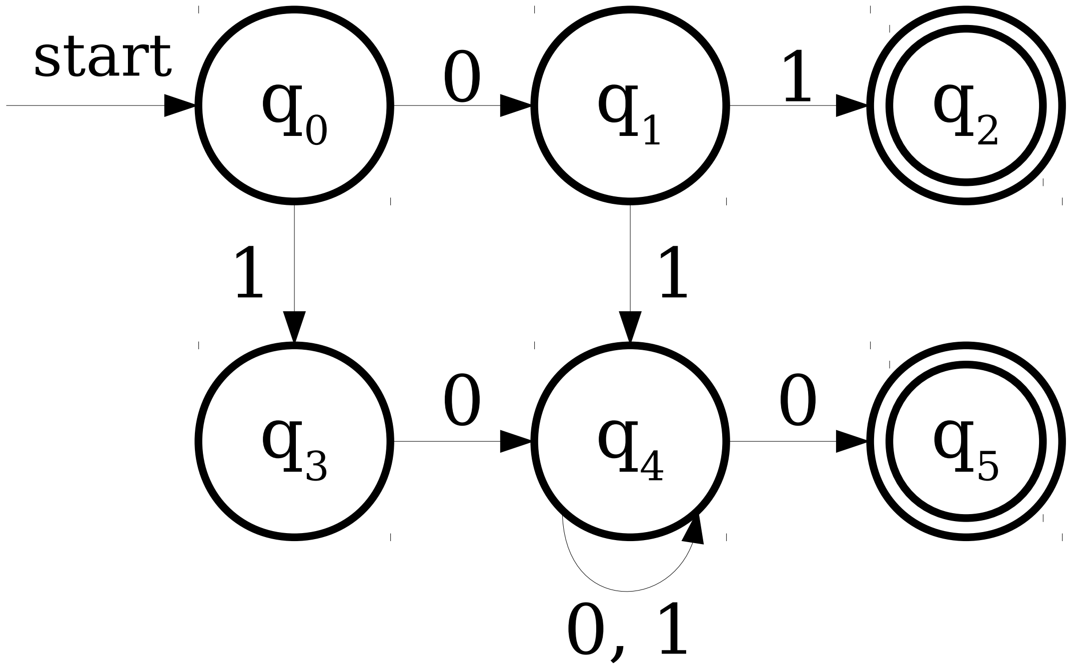
Tree Computation

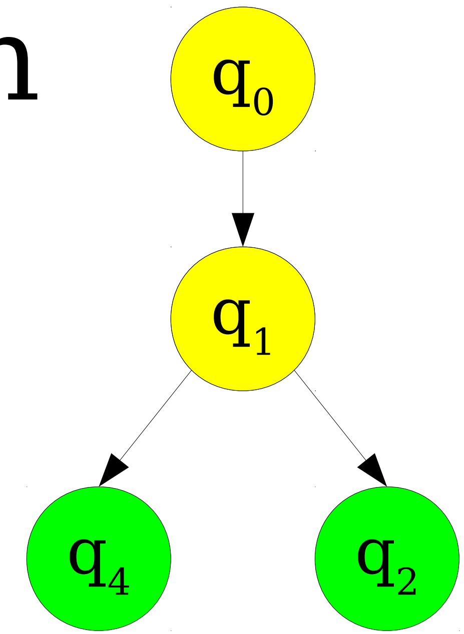


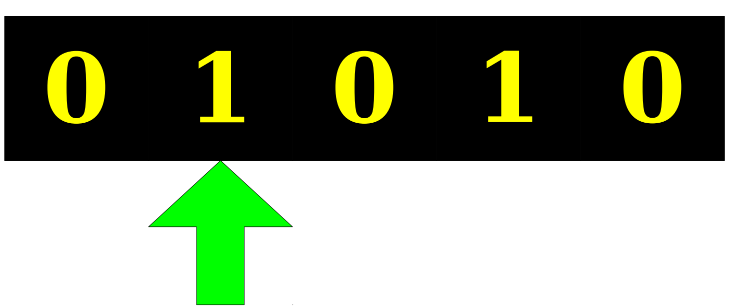




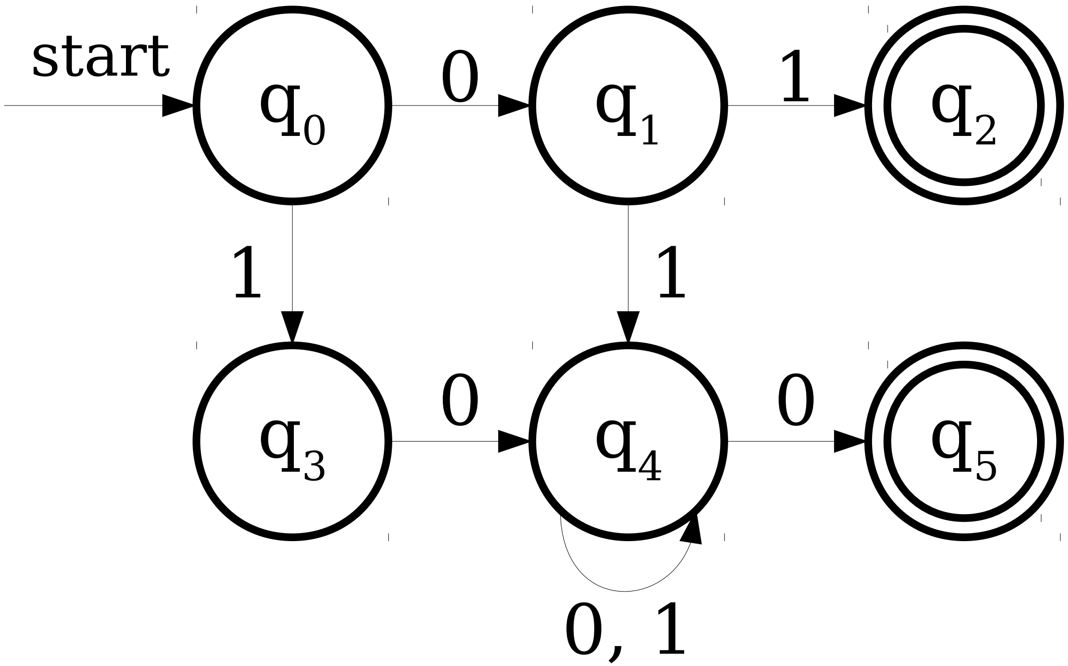
Tree Computation

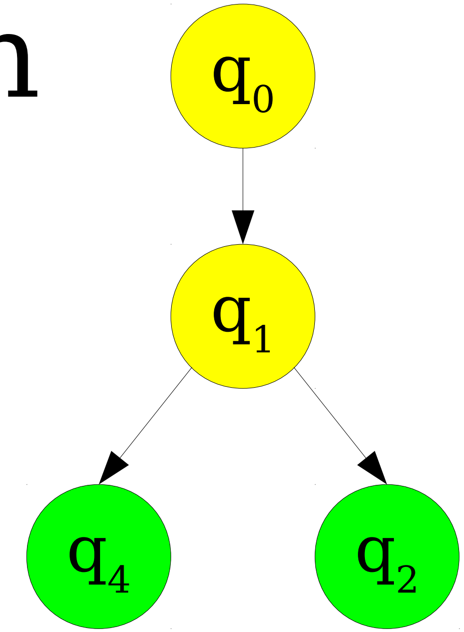


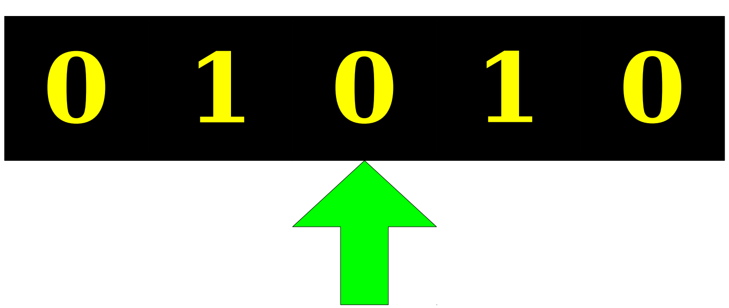




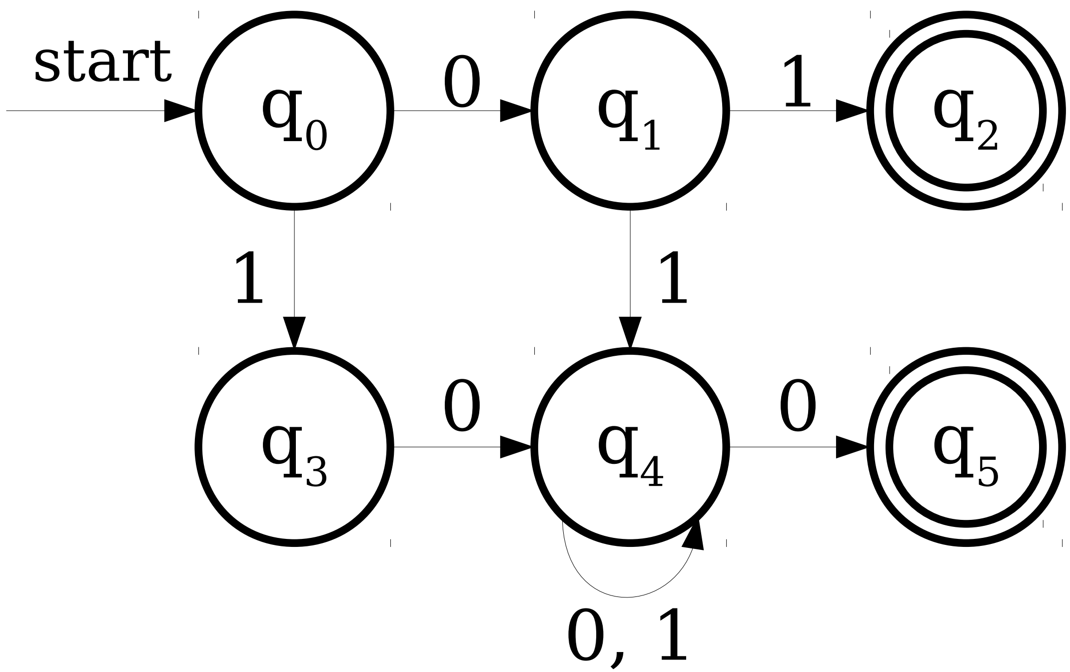
Tree Computation

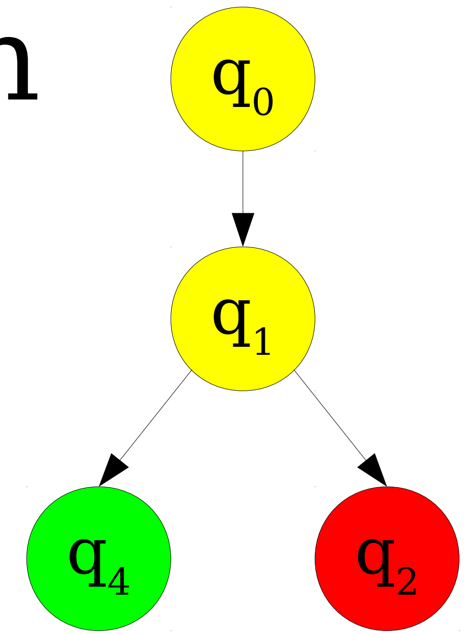


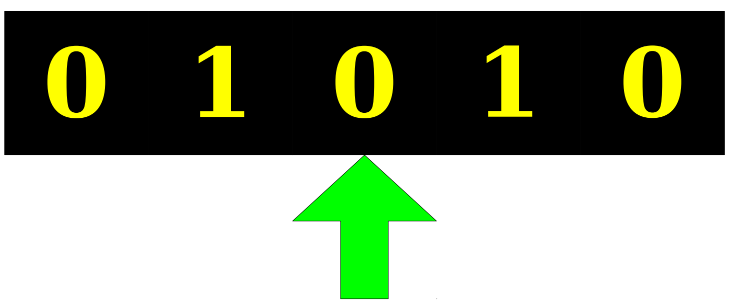




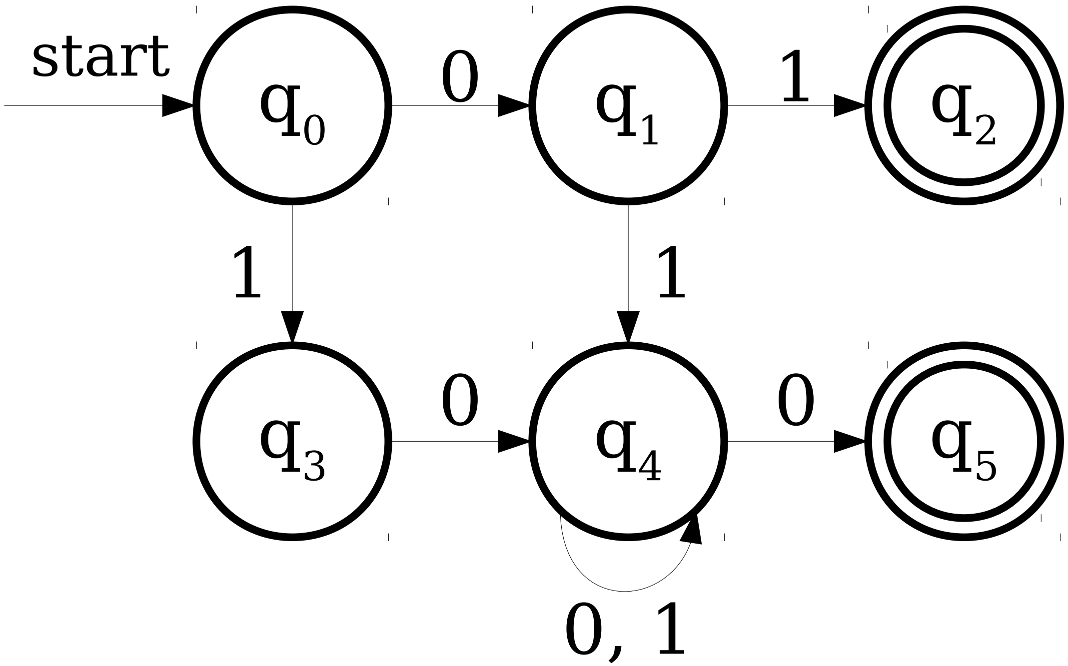
Tree Computation

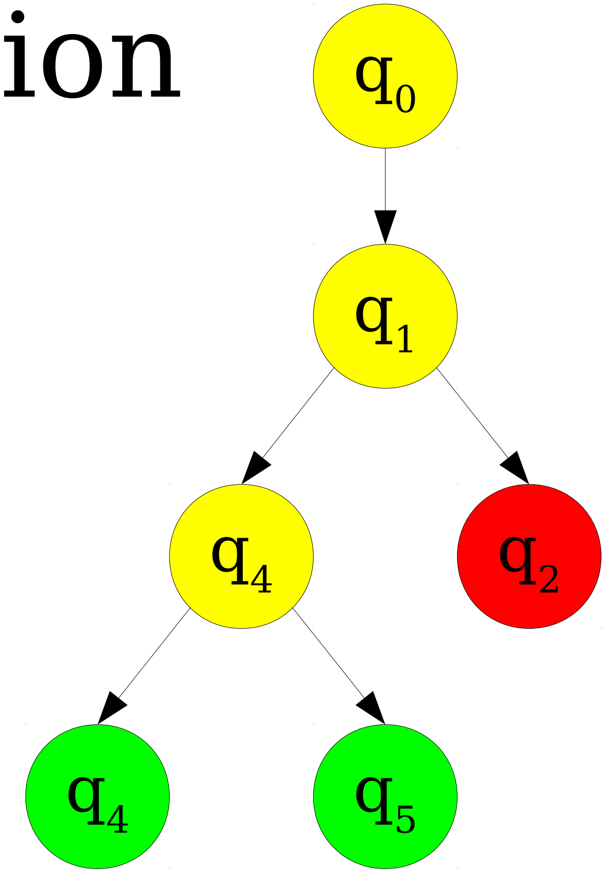


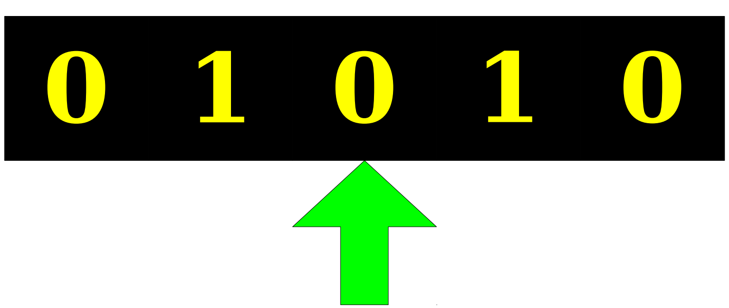




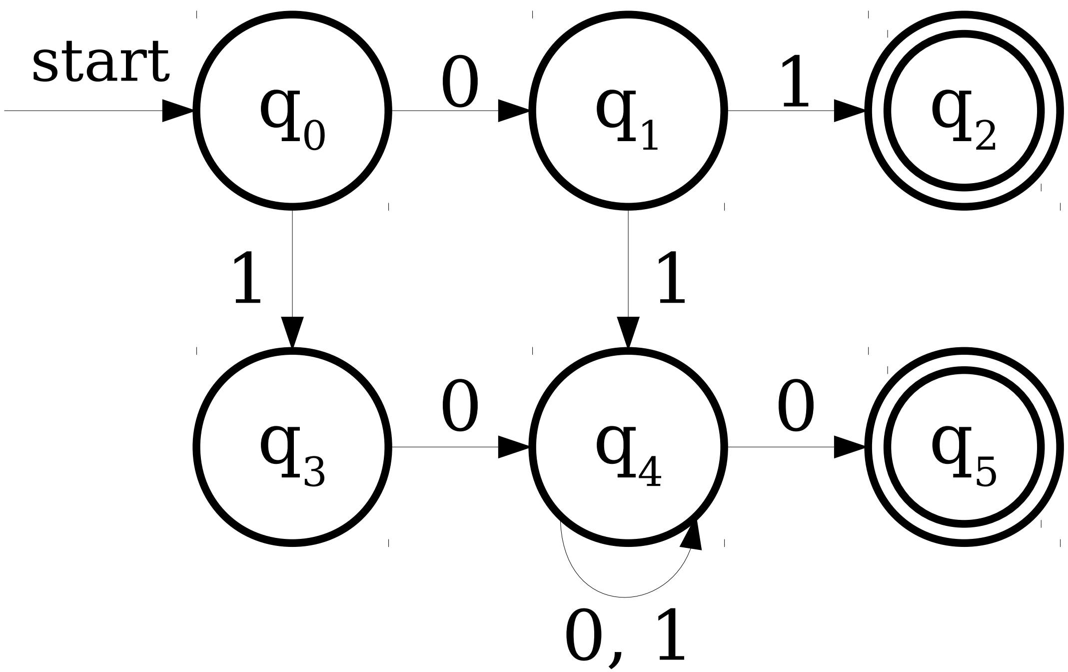
Tree Computation

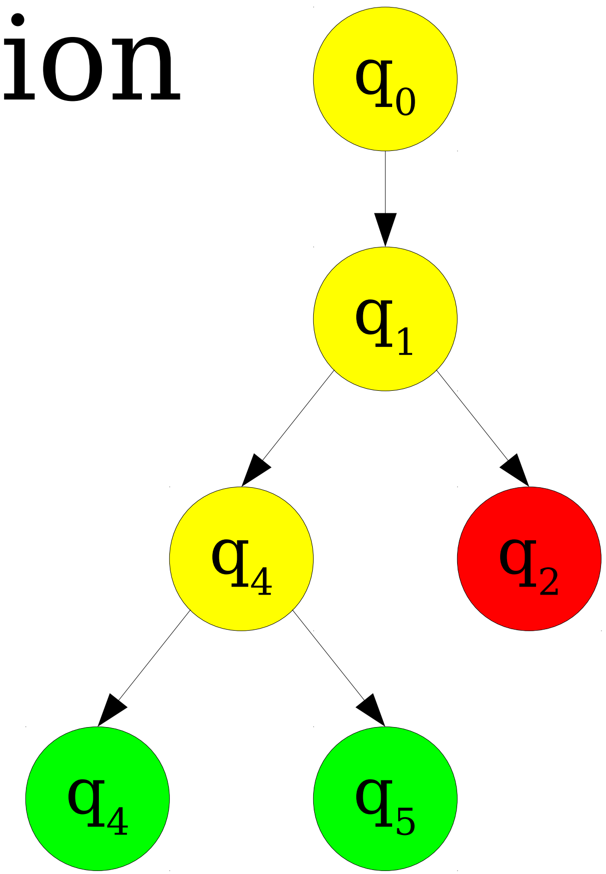




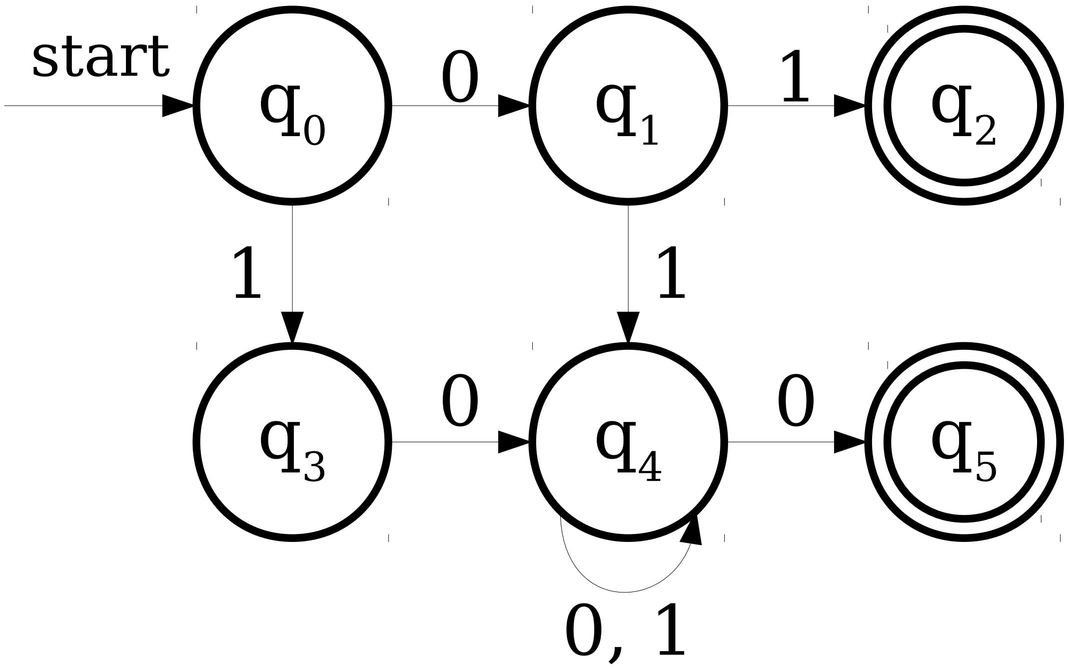


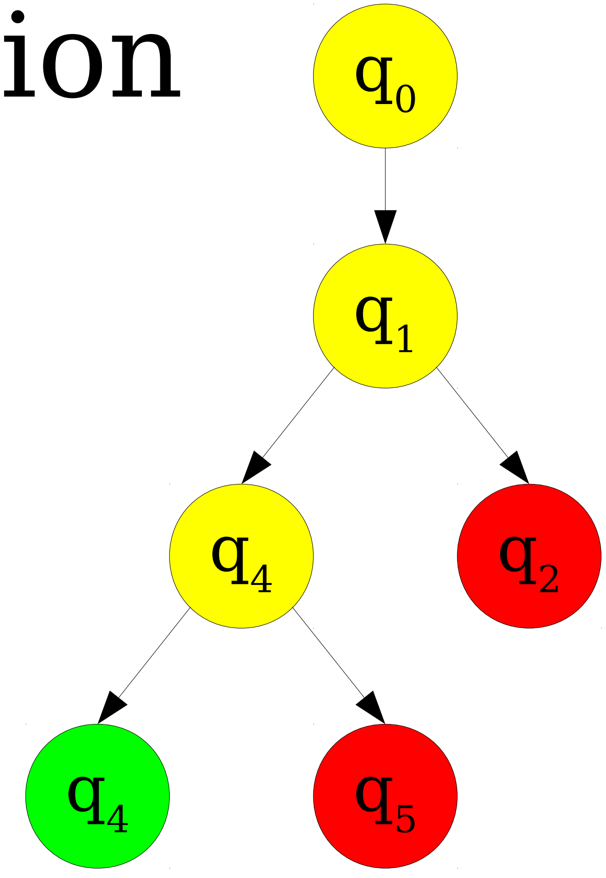
Tree Computation

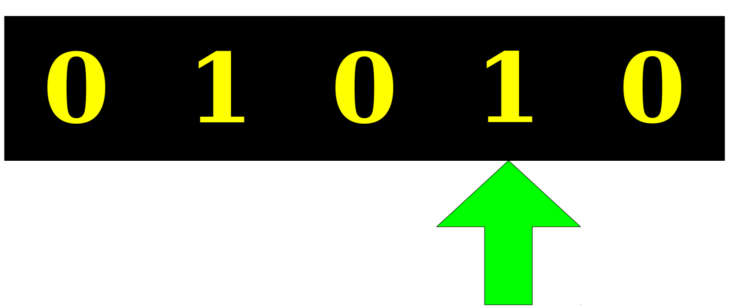




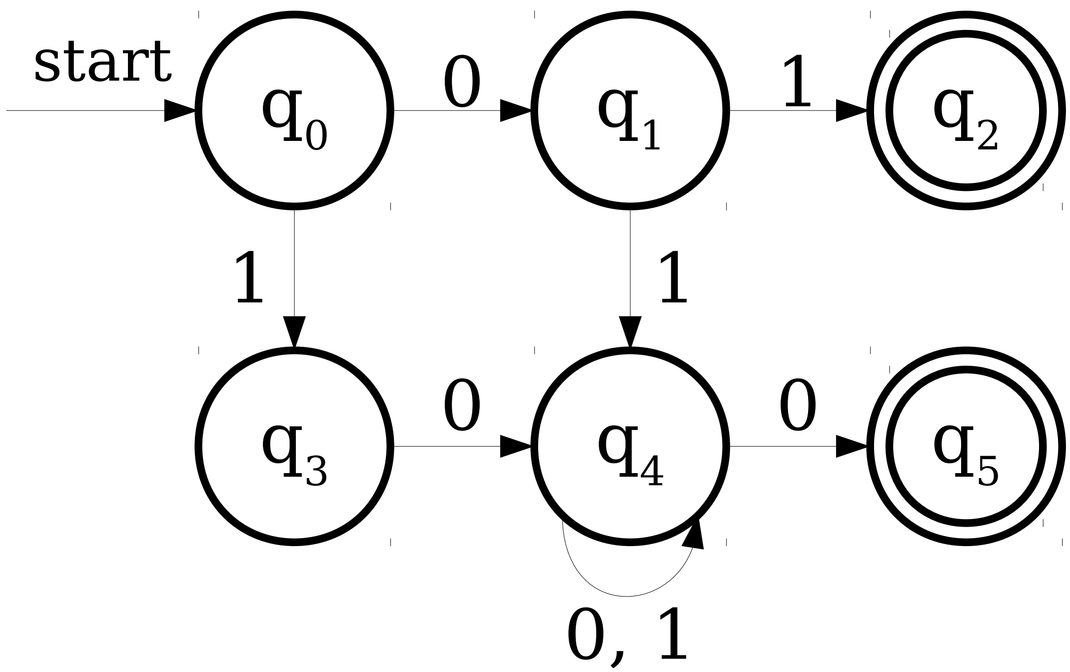
Tree Computation

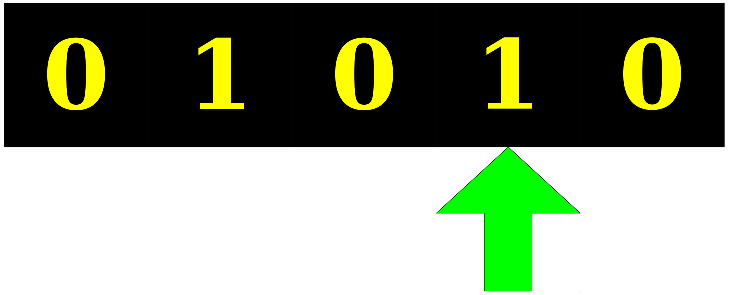


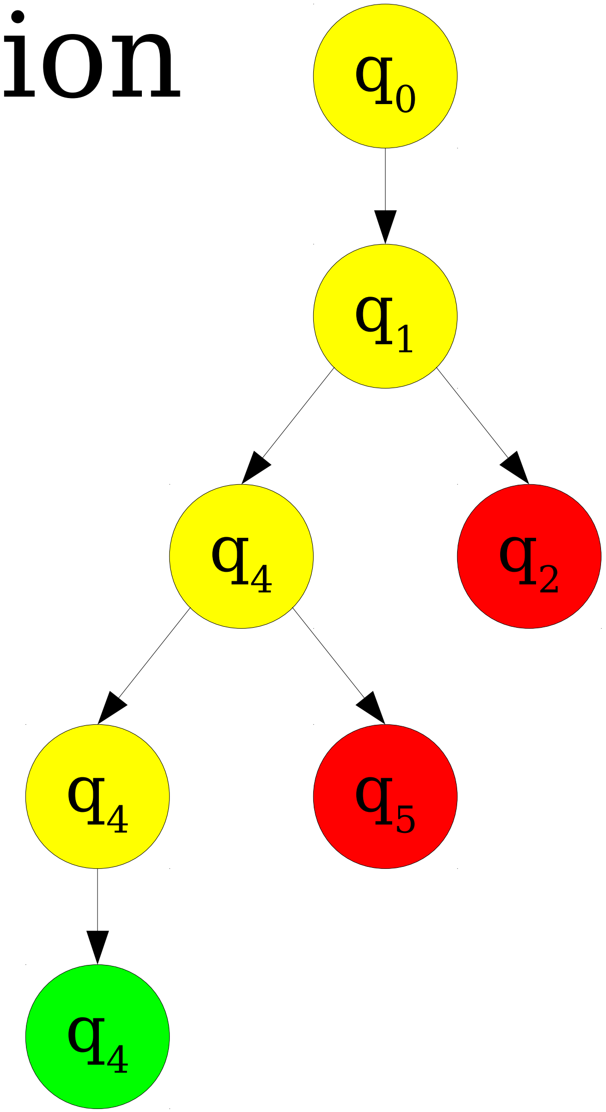




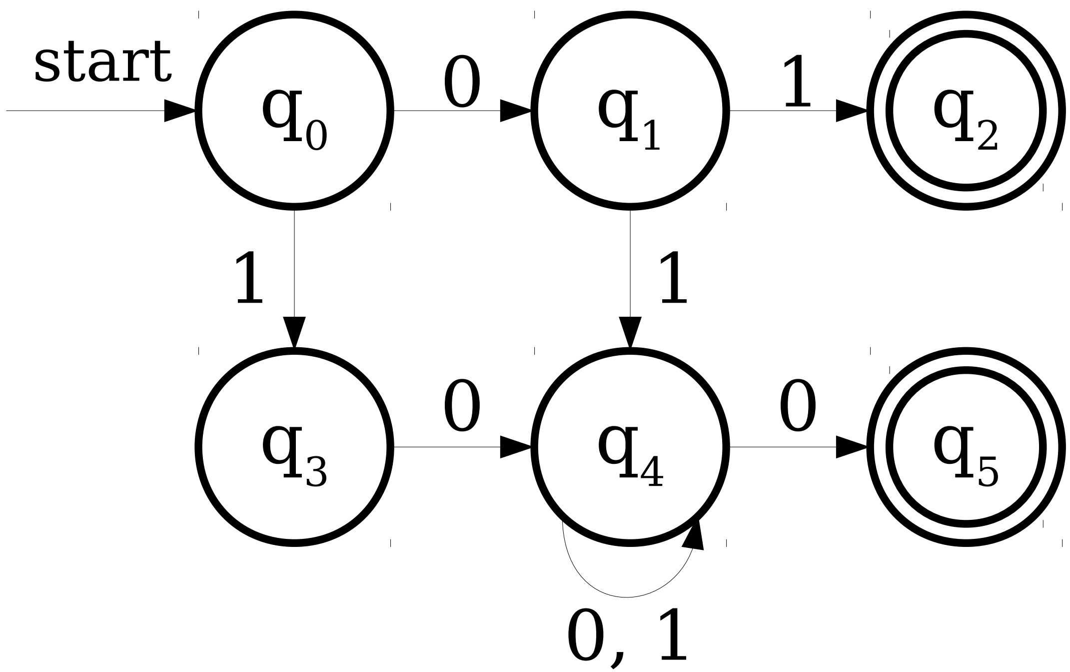
Tree Computation

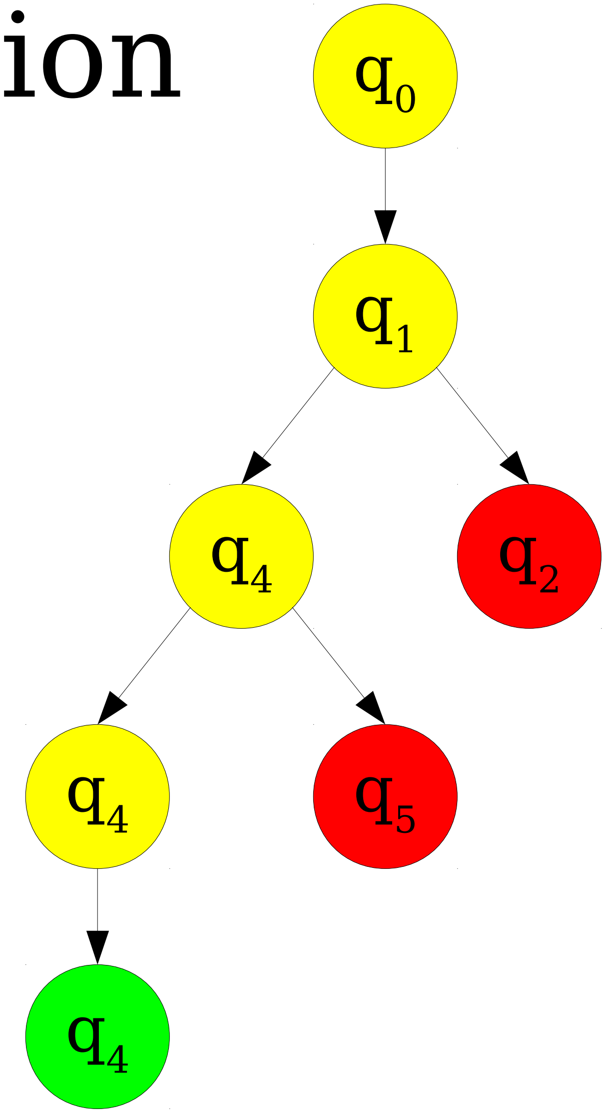




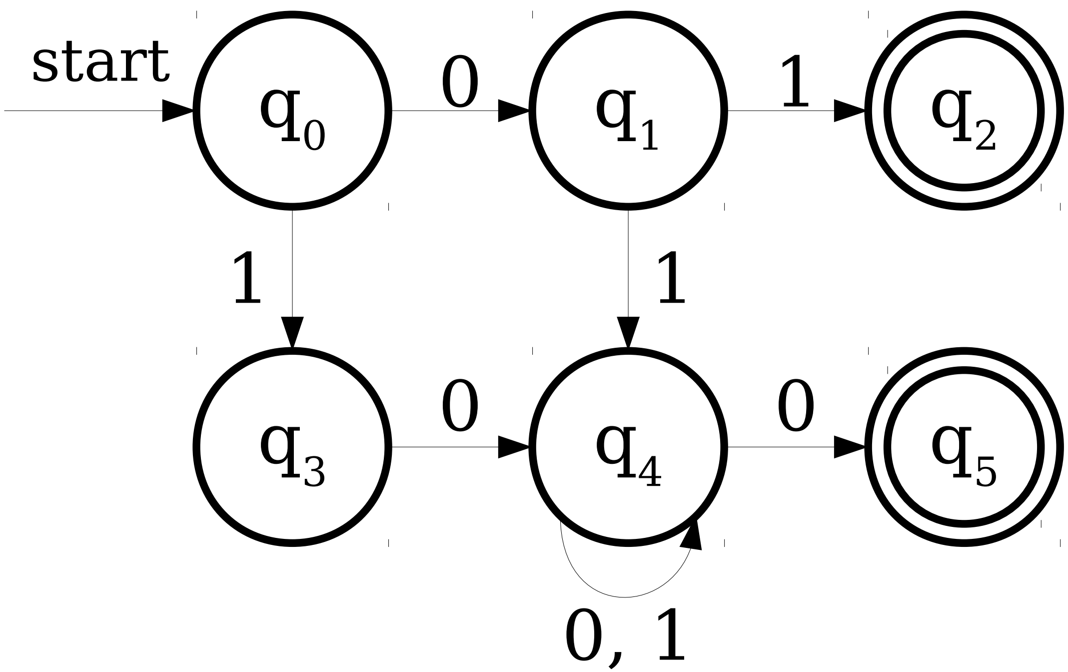


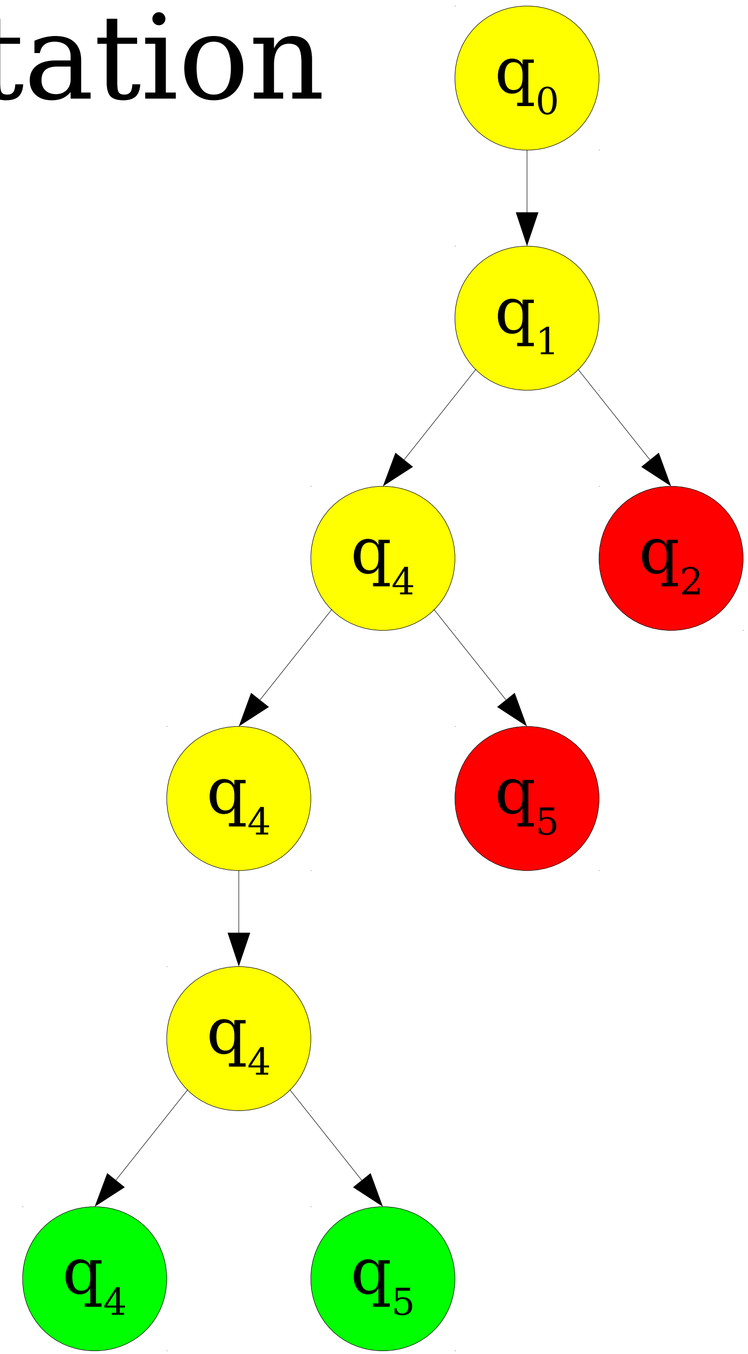
Tree Computation



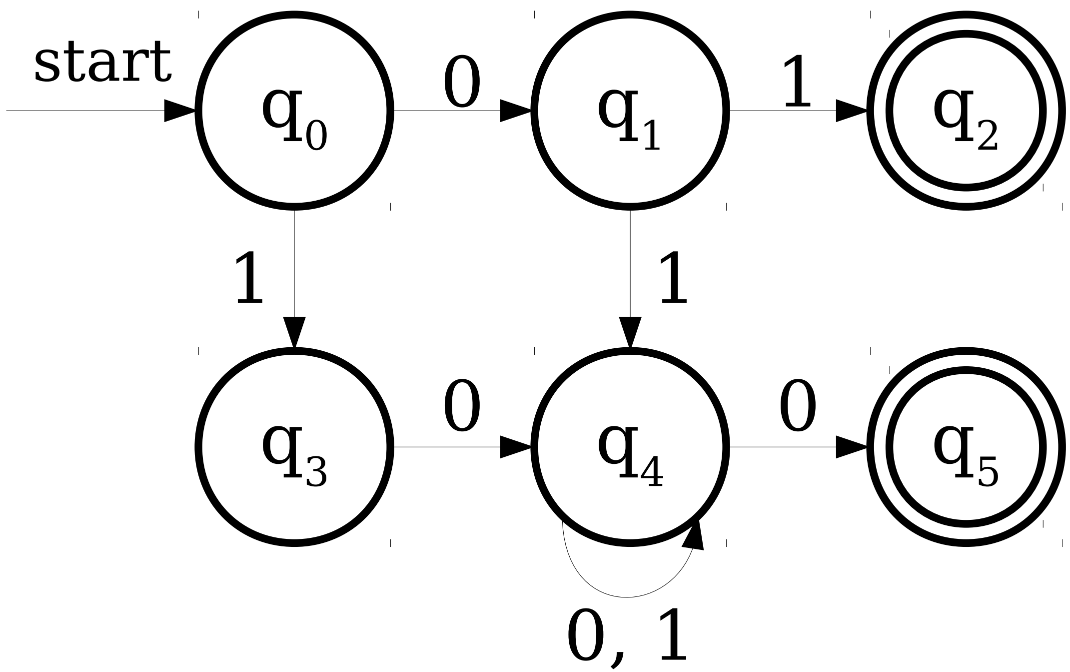


Tree Computation

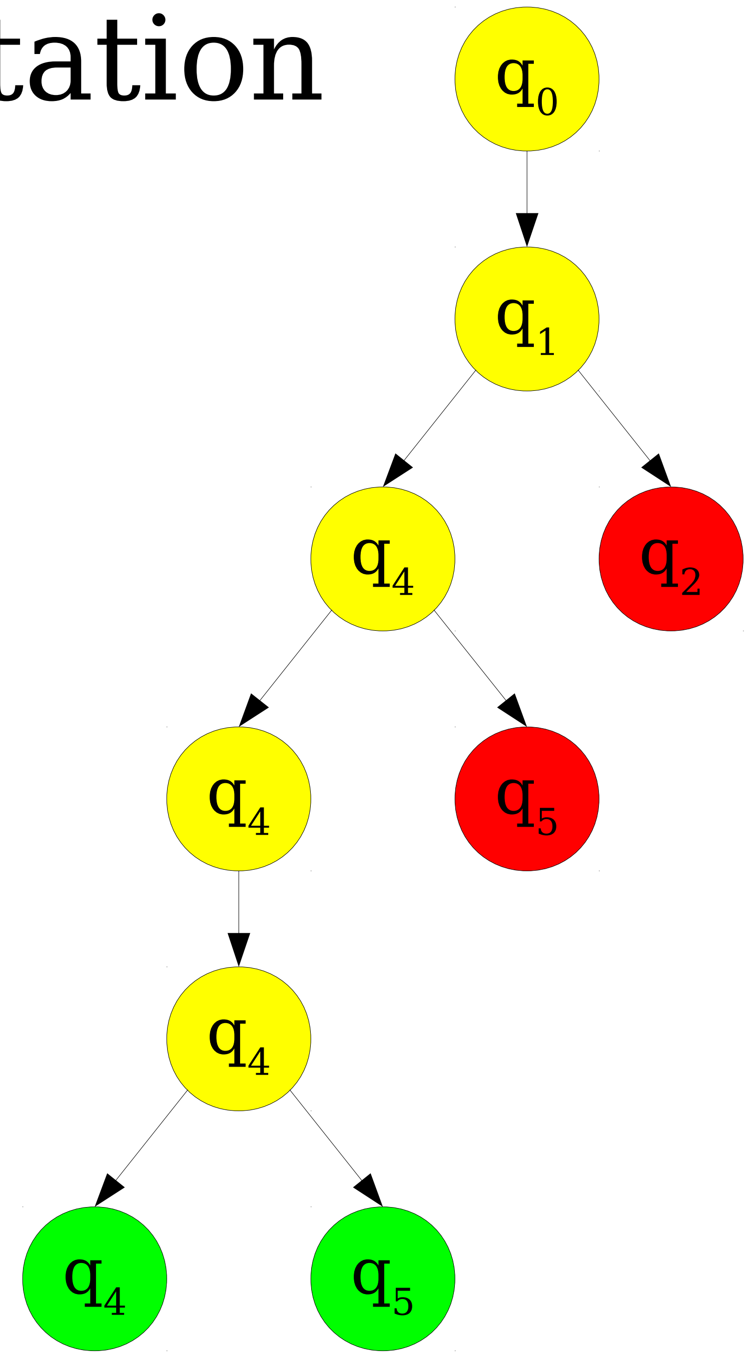




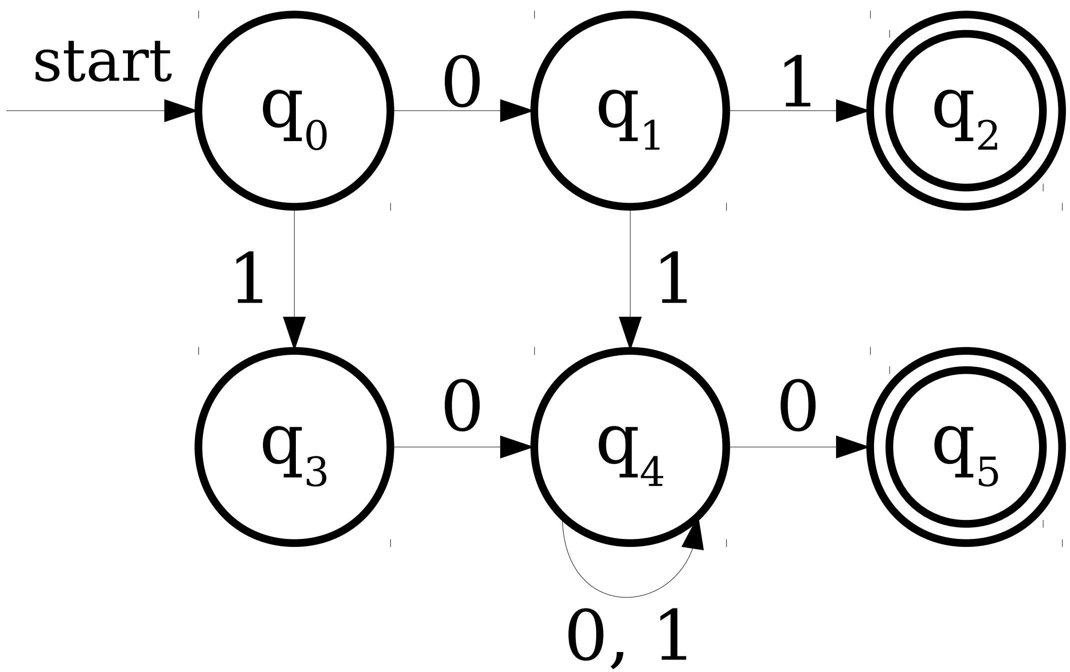
Tree Computation



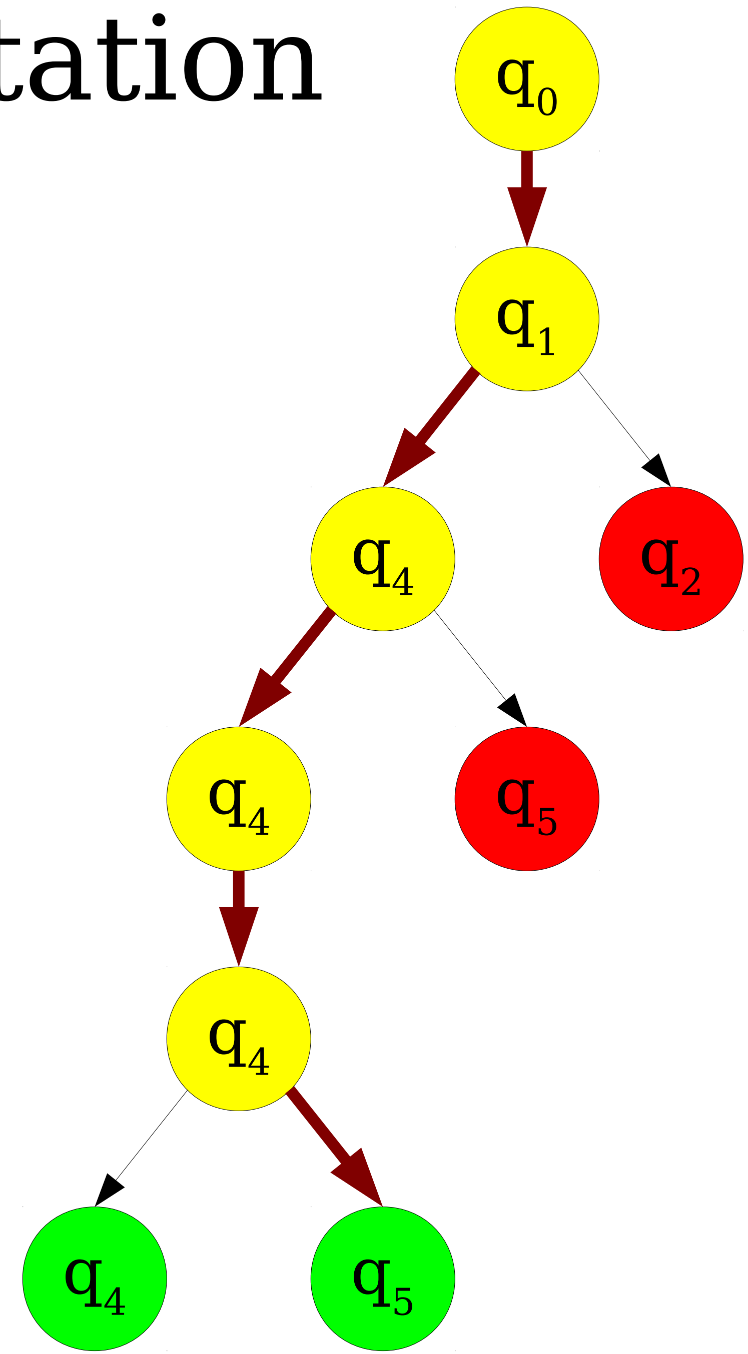
0 1 0 1 0



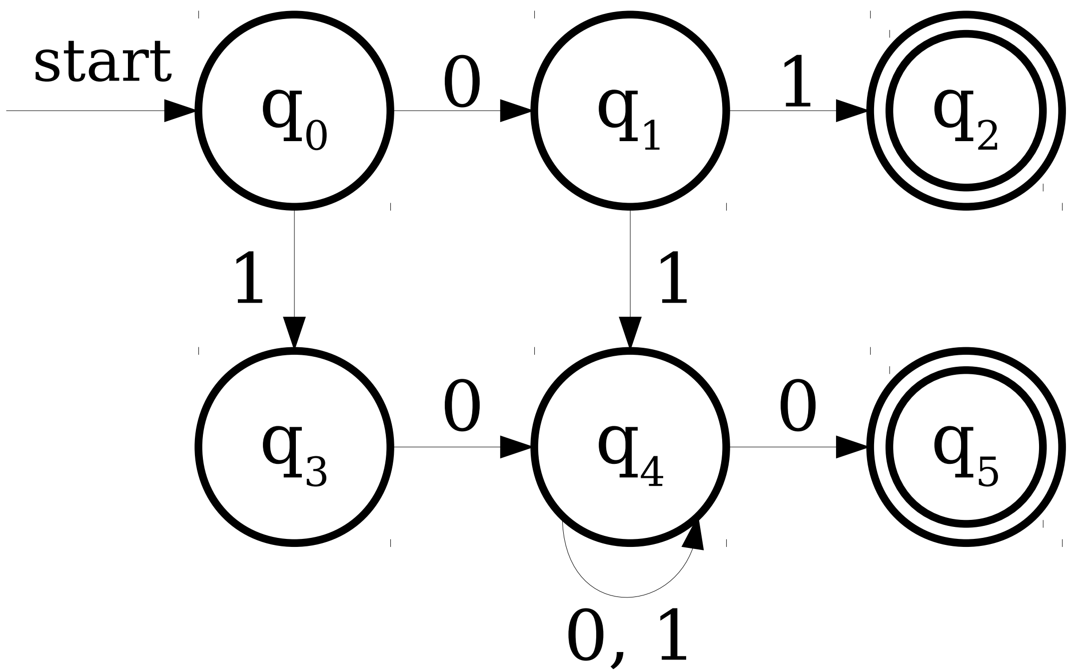
Tree Computation



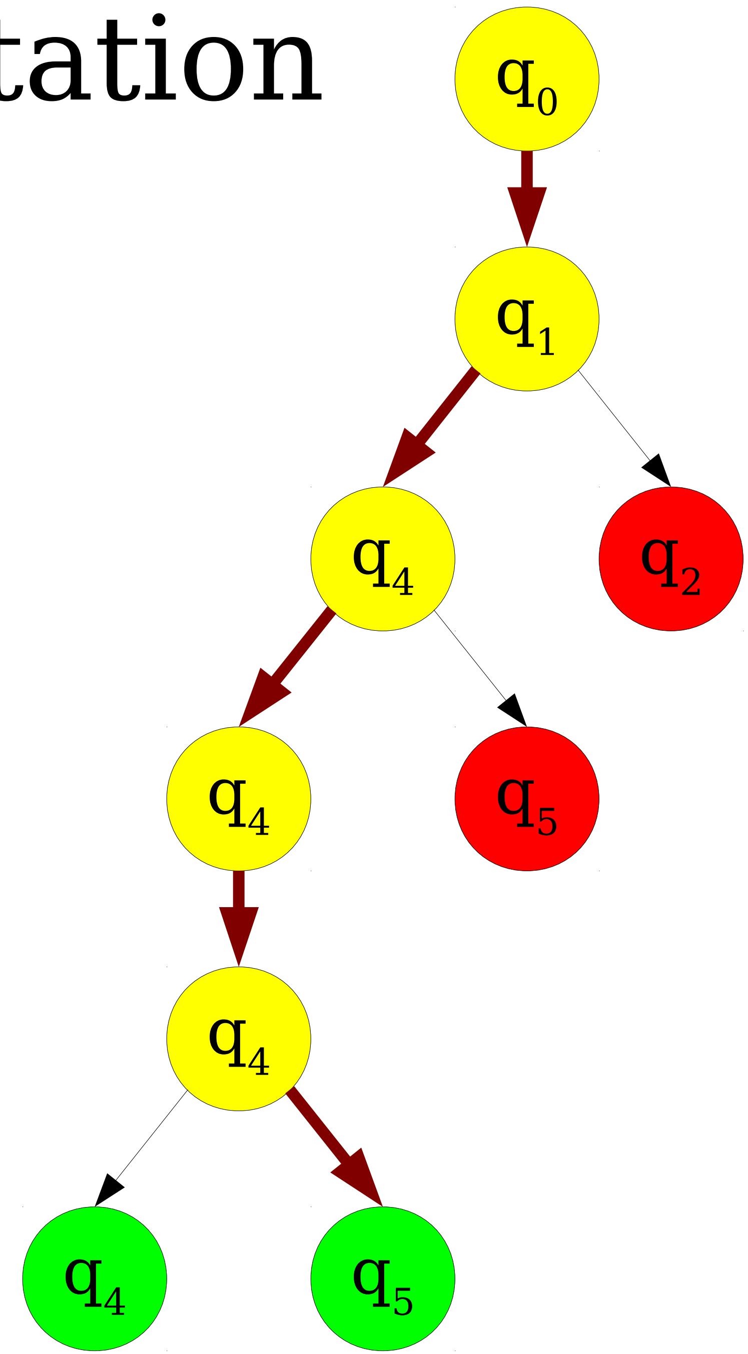
0 1 0 1 0



Tree Computation



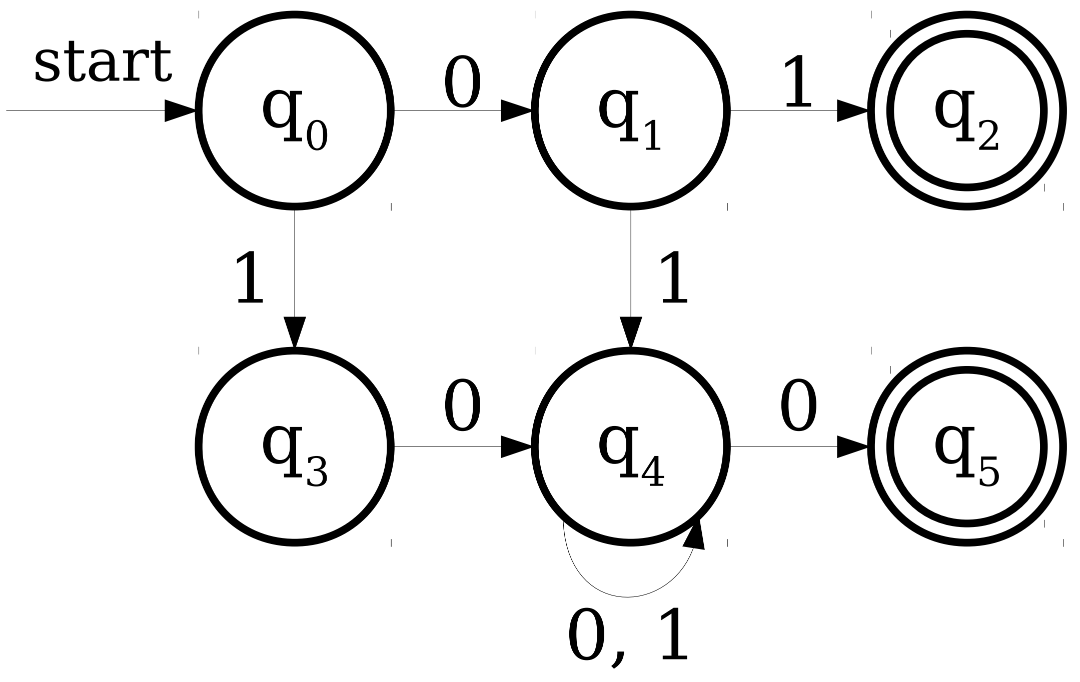
0 1 0 1 0



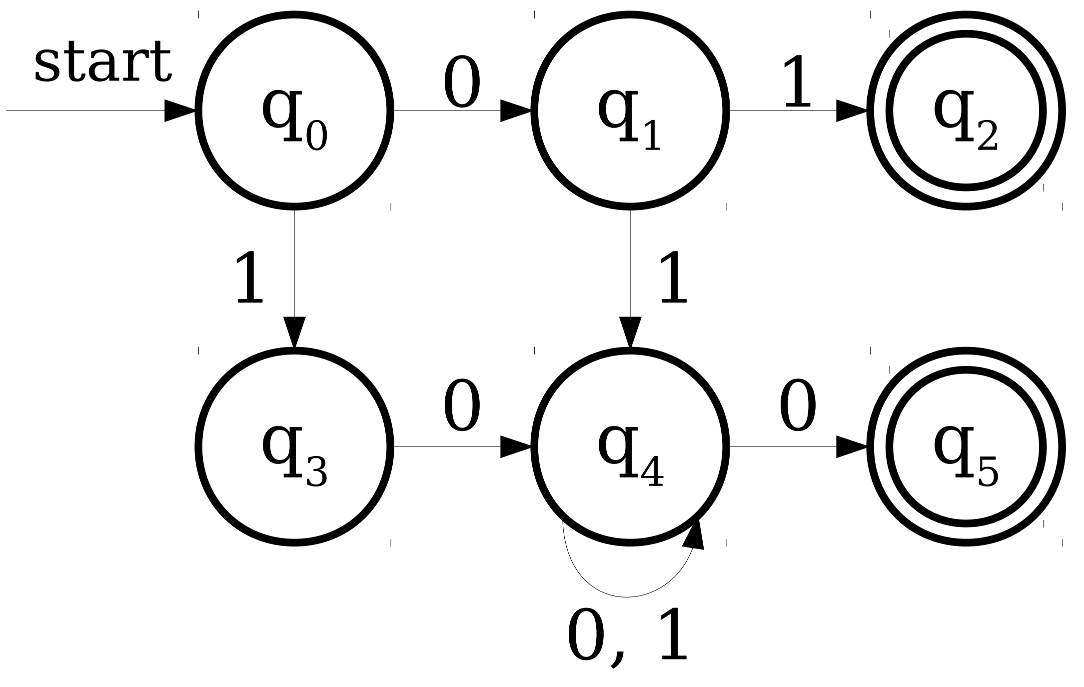
Nondeterminism as a Tree

- At each decision point, the automaton clones itself for each possible decision.
- The series of choices forms a directed, rooted tree.
- At the end, if any active accepting states remain, we accept.

Perfect Guessing

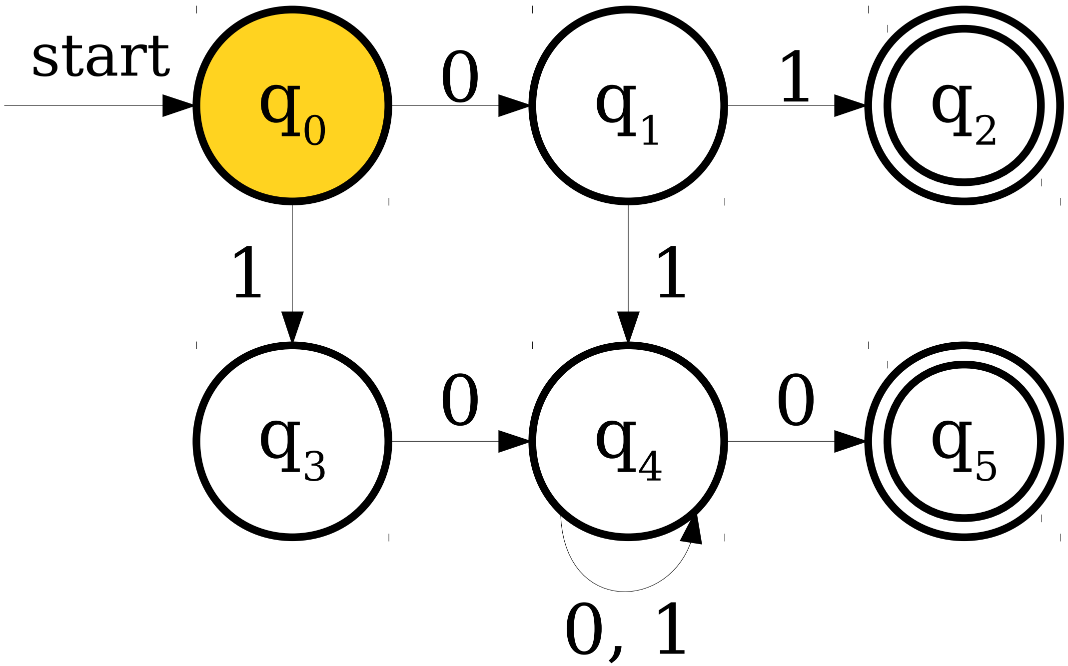


Perfect Guessing



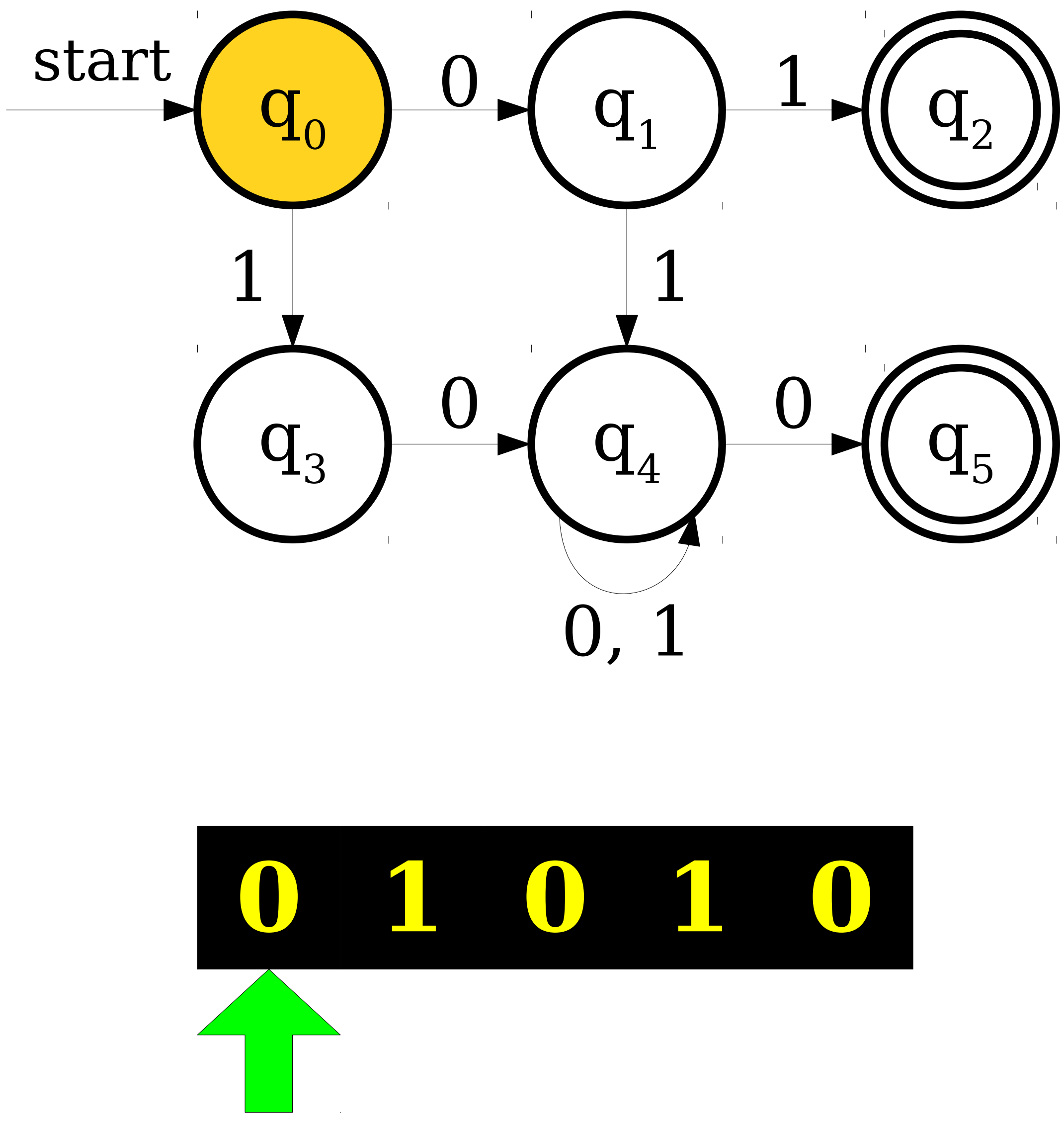
0 1 0 1 0

Perfect Guessing

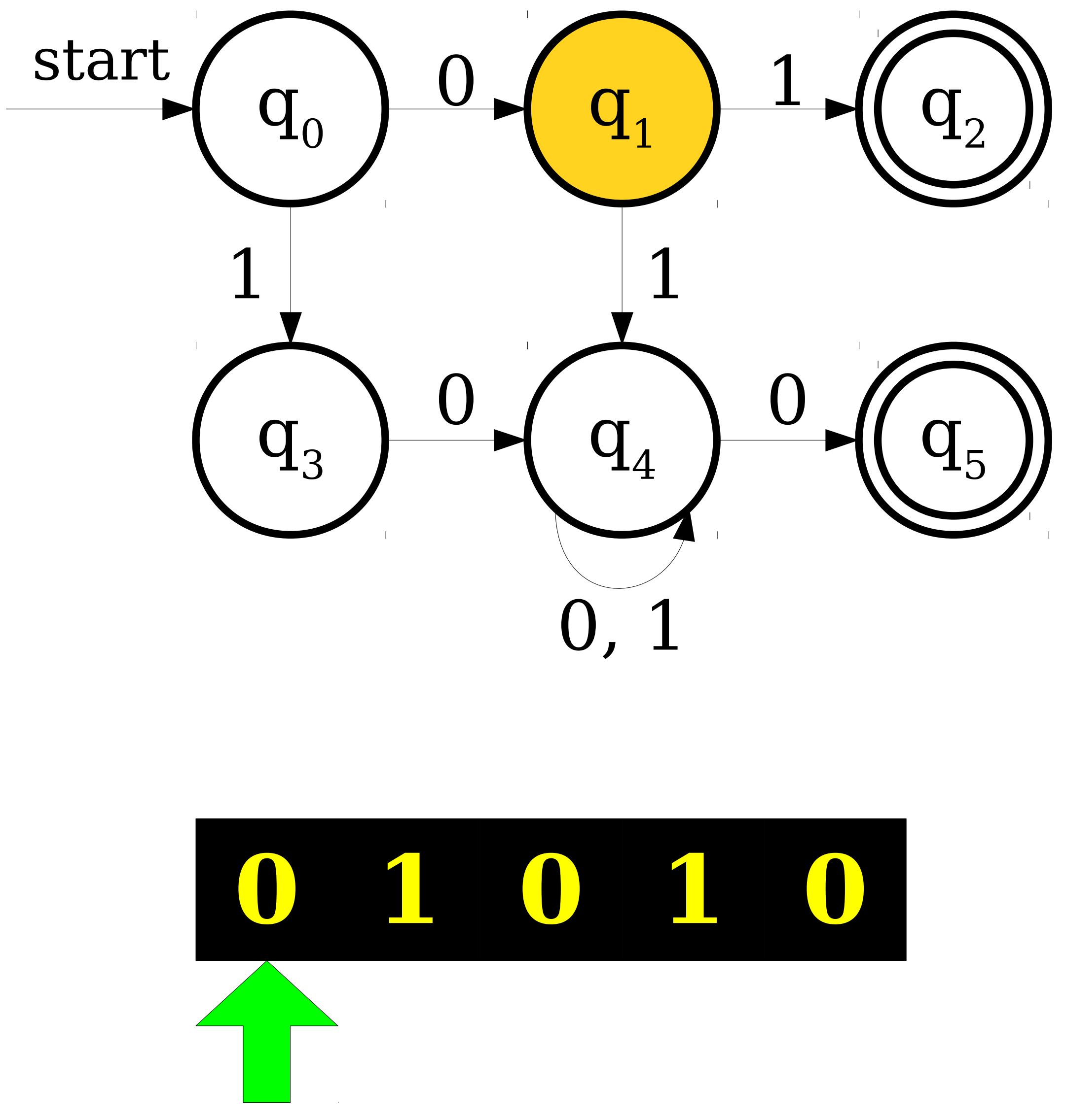


0 1 0 1 0

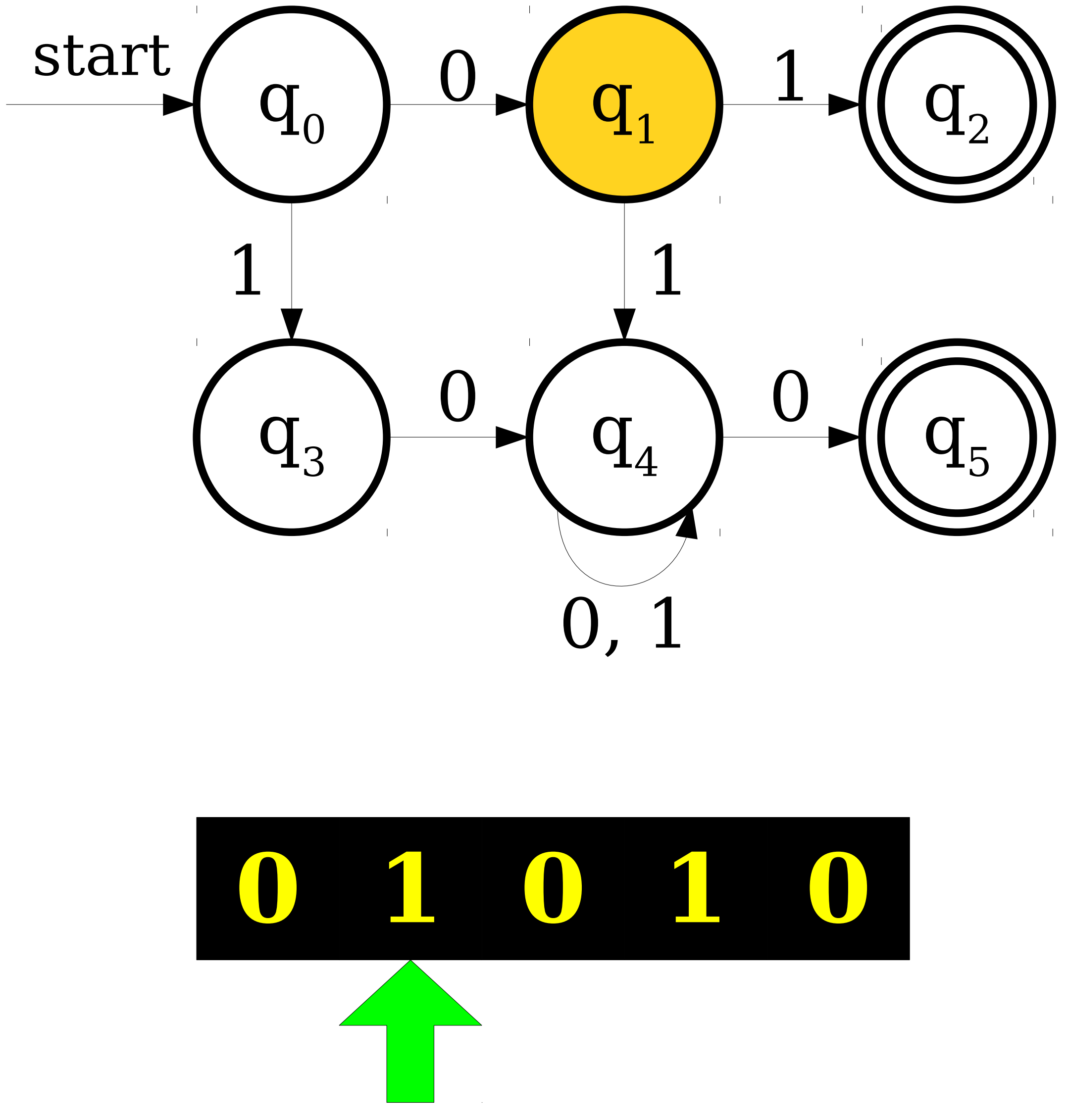
Perfect Guessing



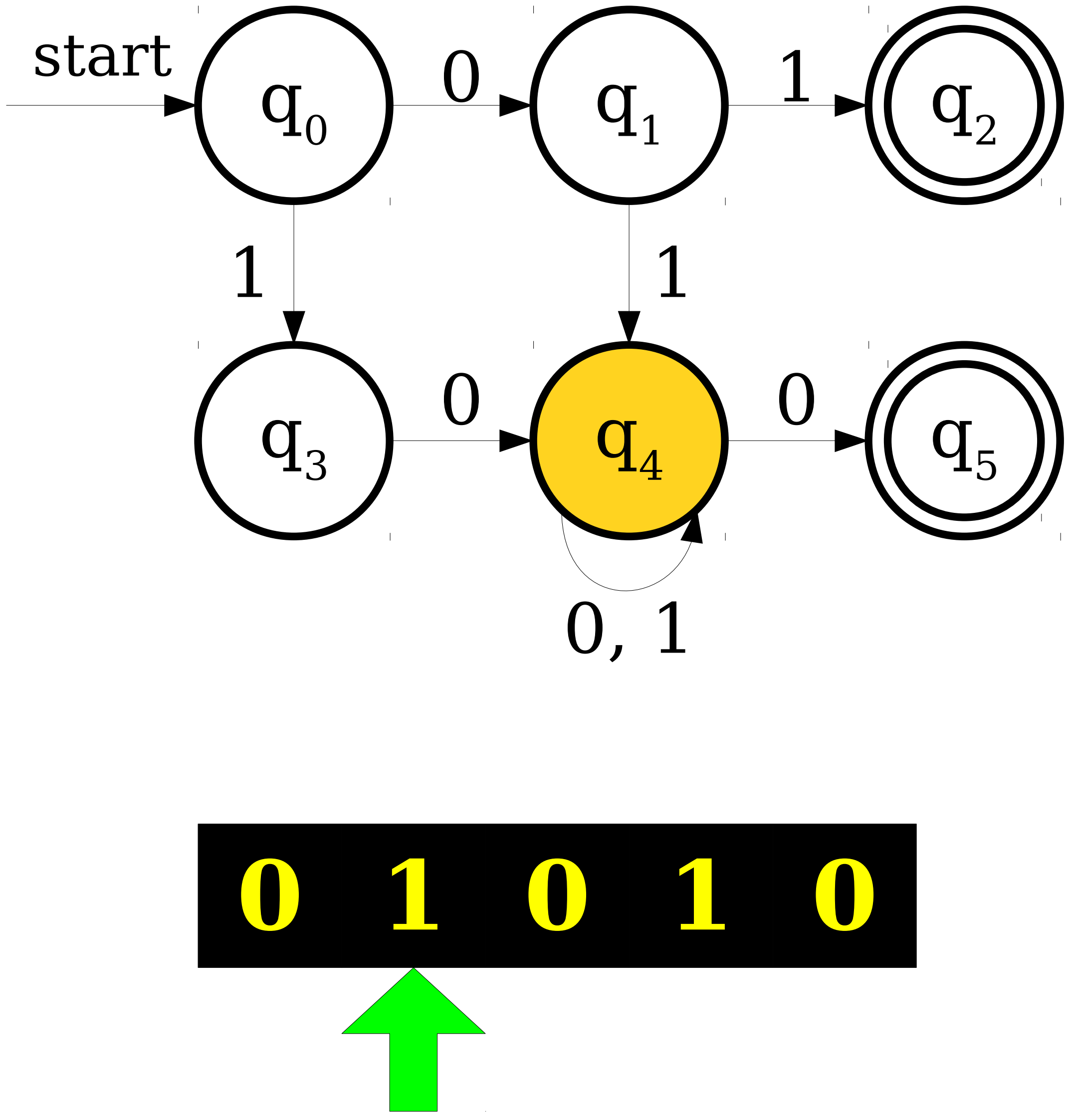
Perfect Guessing



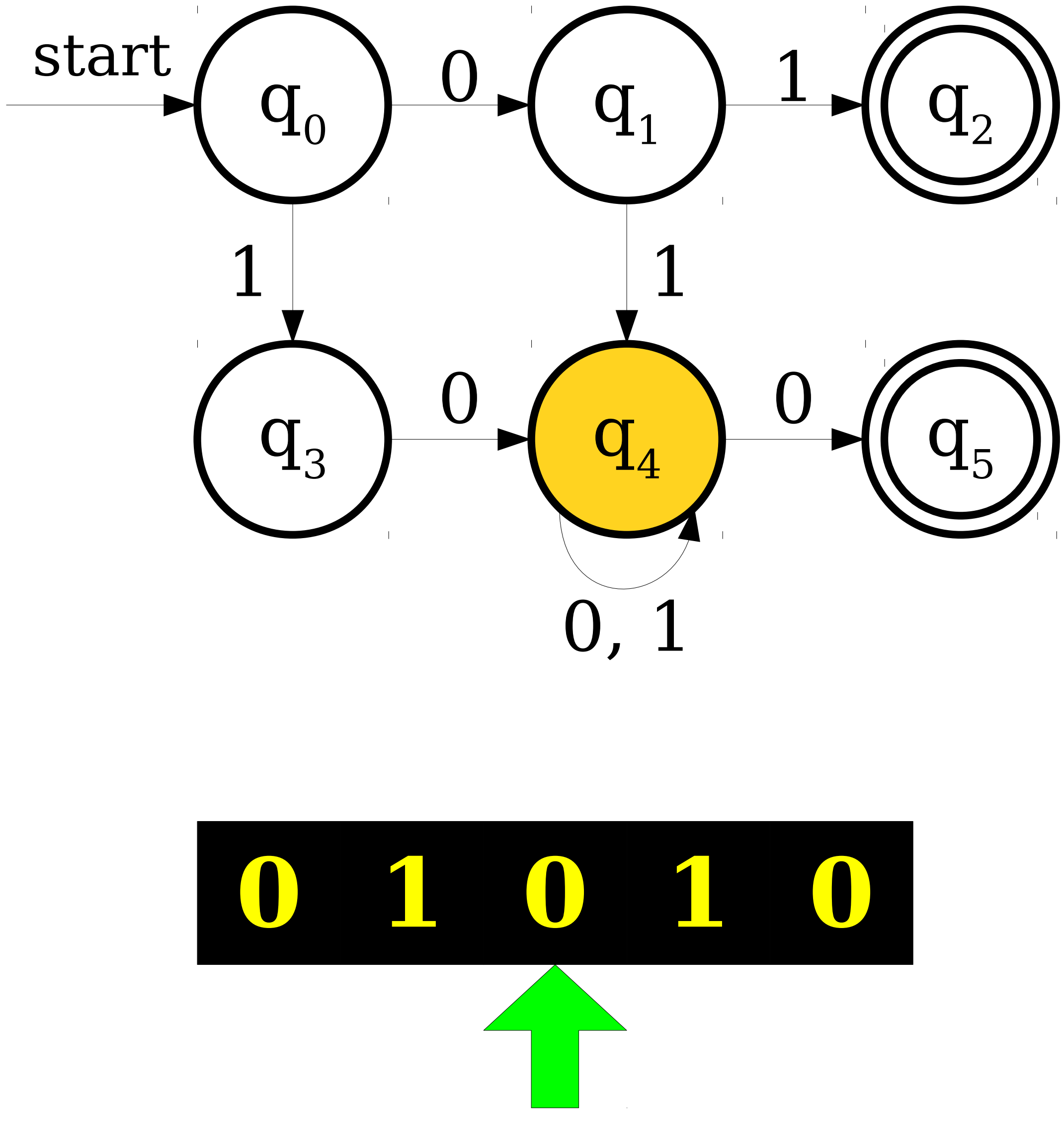
Perfect Guessing



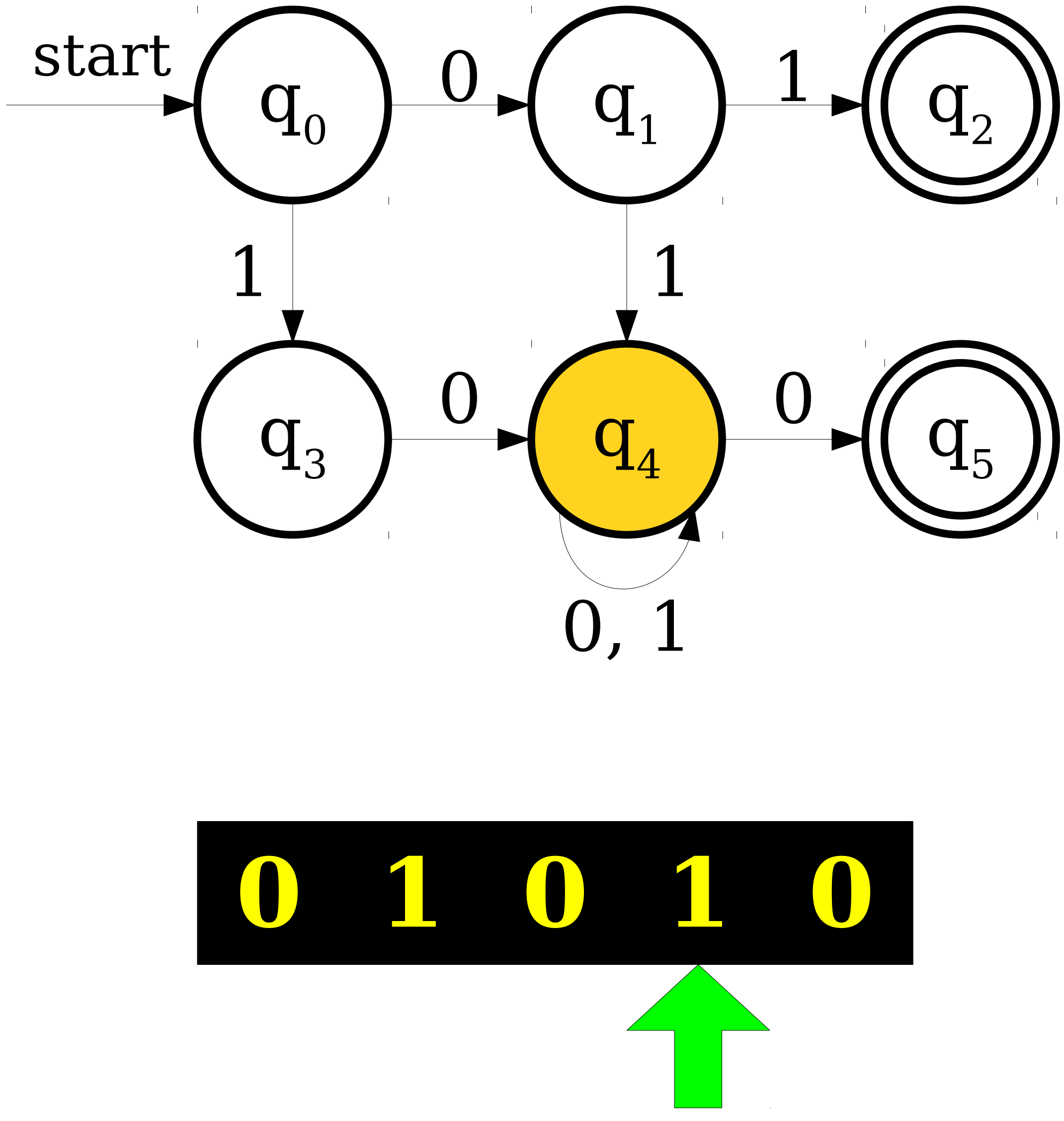
Perfect Guessing



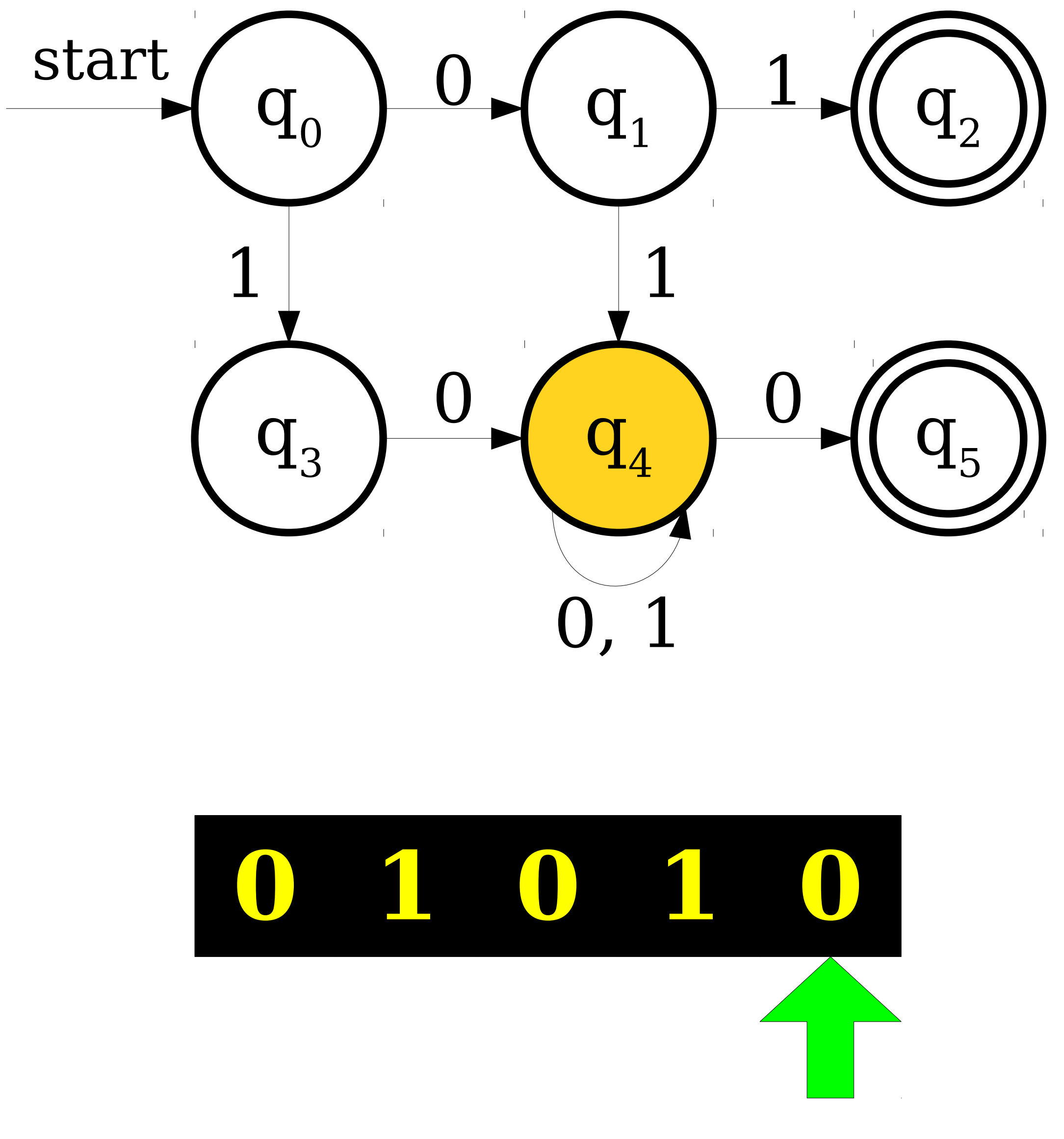
Perfect Guessing



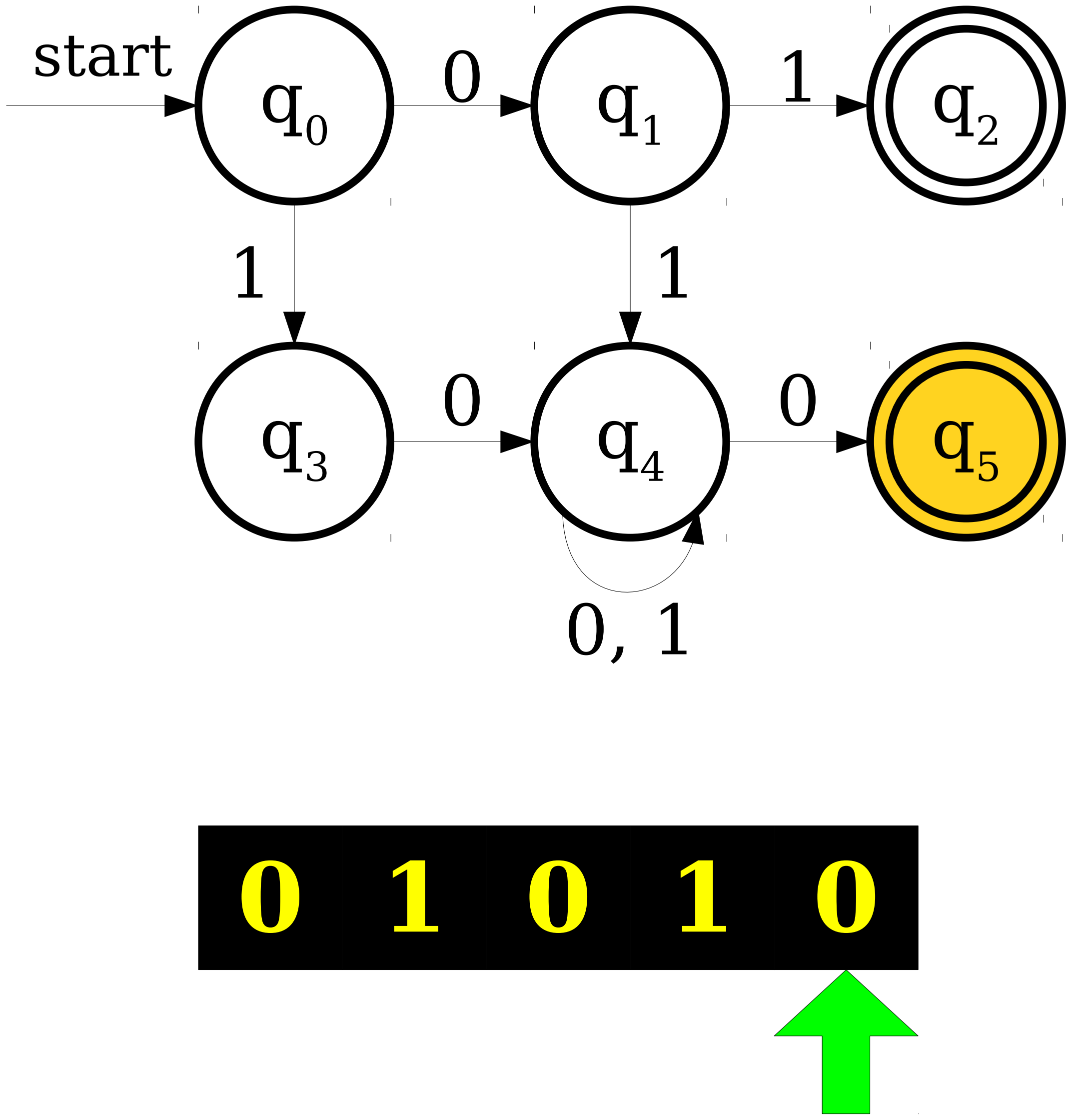
Perfect Guessing



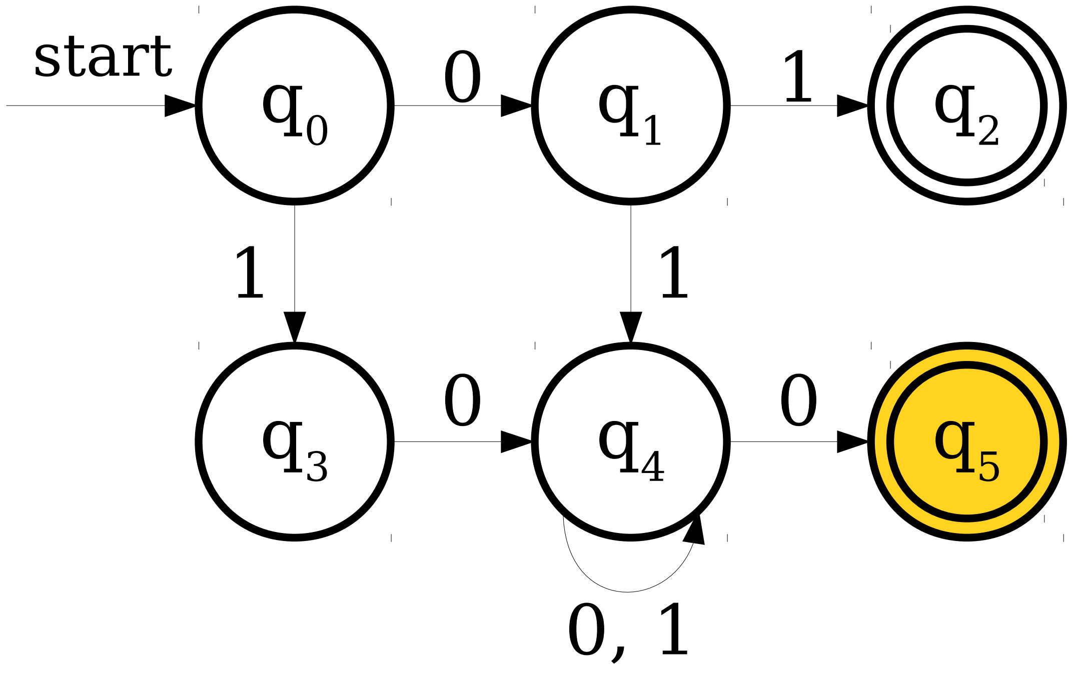
Perfect Guessing



Perfect Guessing

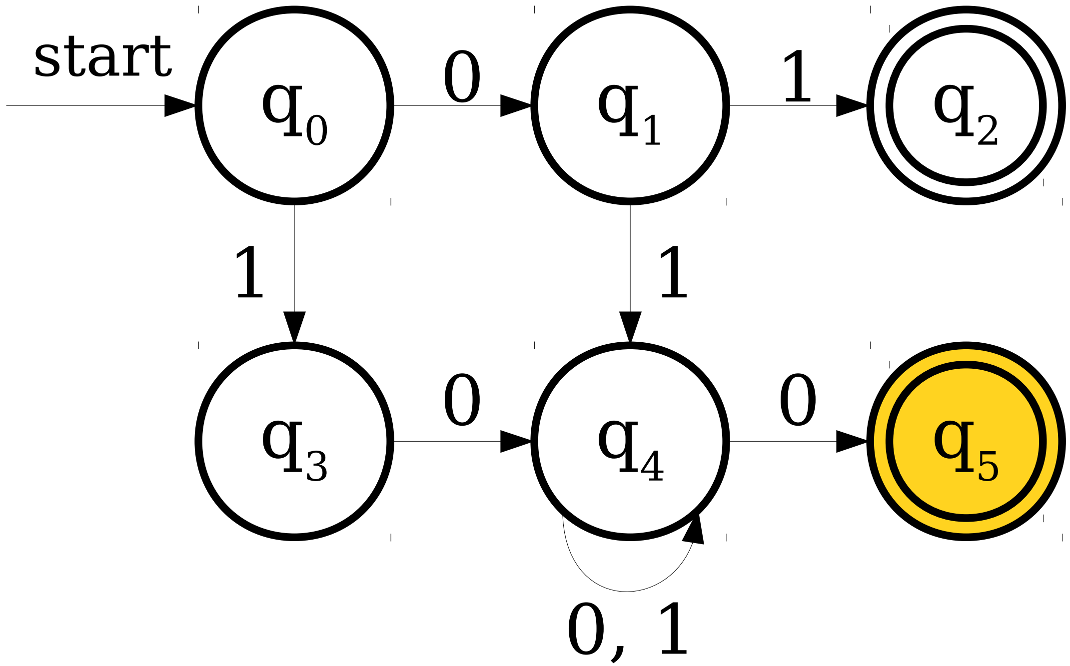


Perfect Guessing



0 1 0 1 0

Perfect Guessing

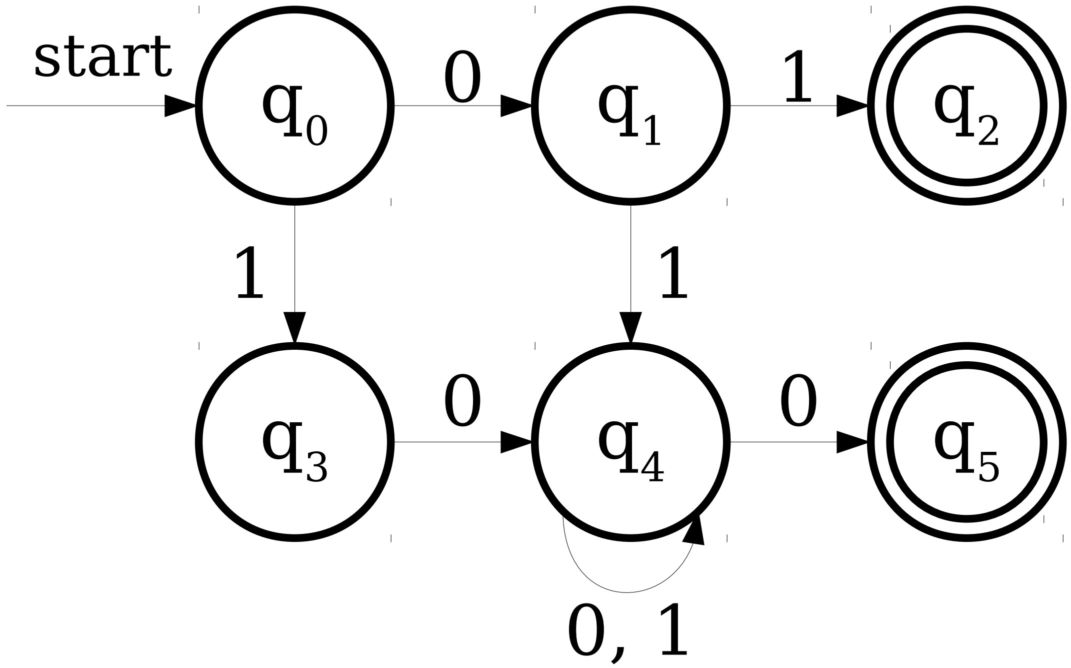


0 1 0 1 0

Perfect Guessing

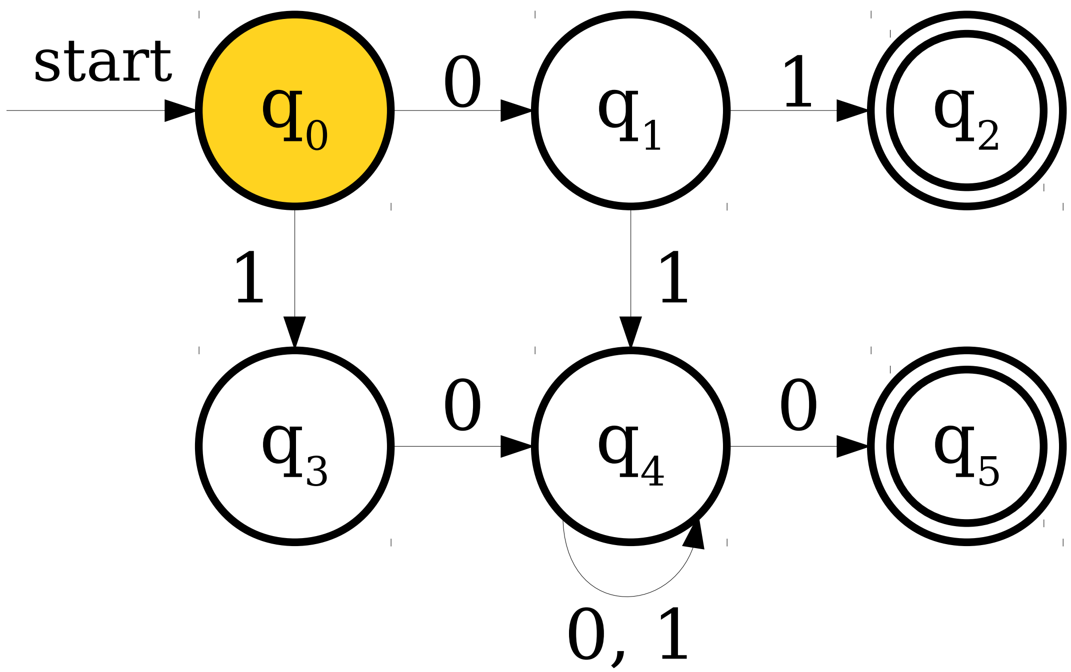
- We can view nondeterministic machines as having ***Magic Superpowers*** that enable them to guess the correct choice of moves to make.
- Idea: Machine can always guess a path that leads to an accepting state if one exists.
- No known physical analog for this style of computation.

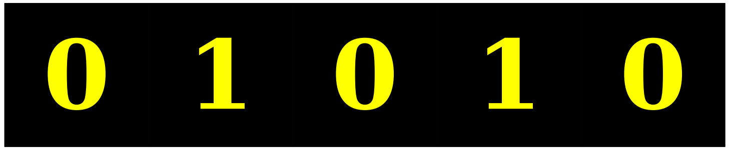
Massive Parallelism



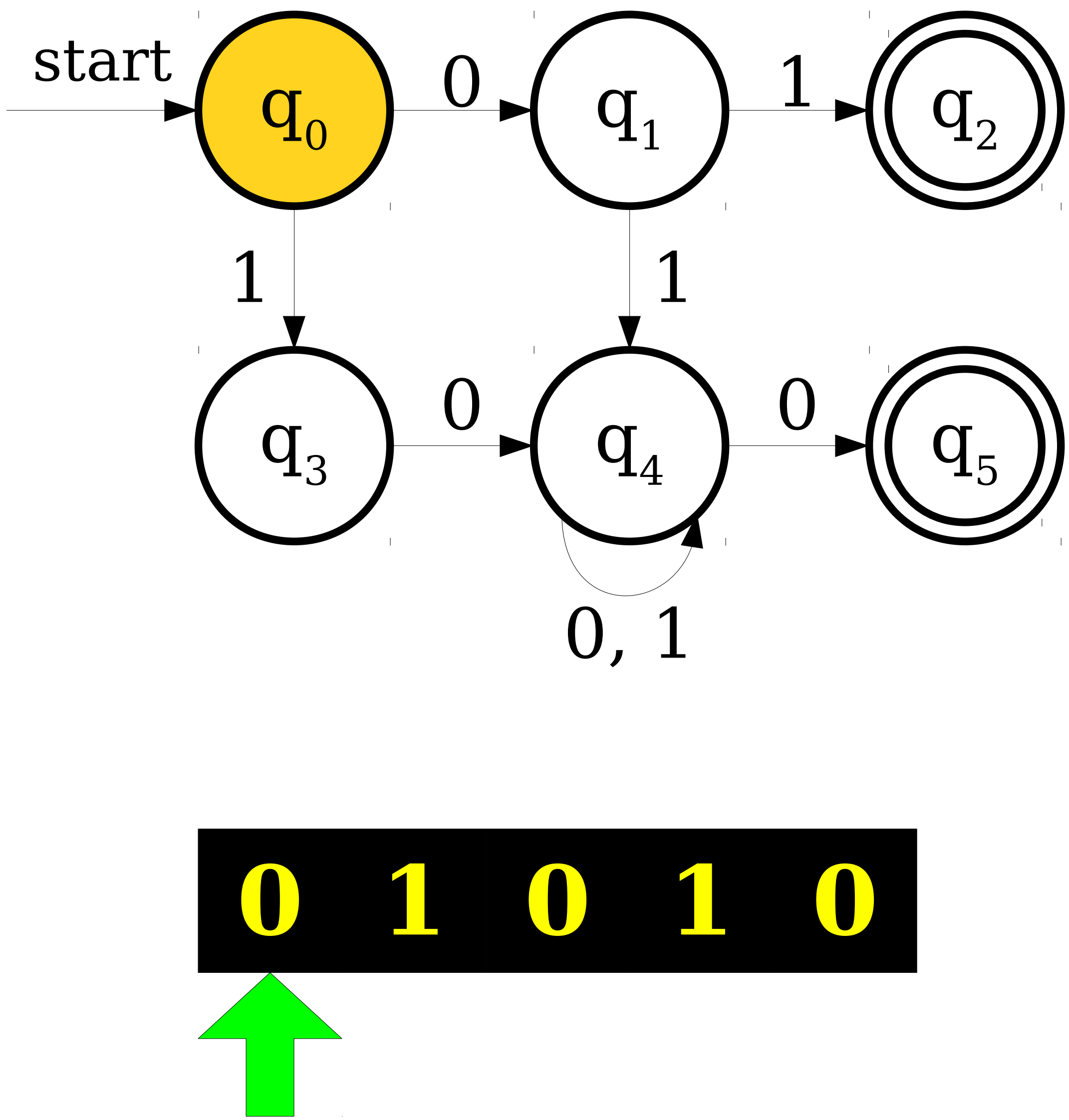
0 1 0 1 0

Massive Parallelism

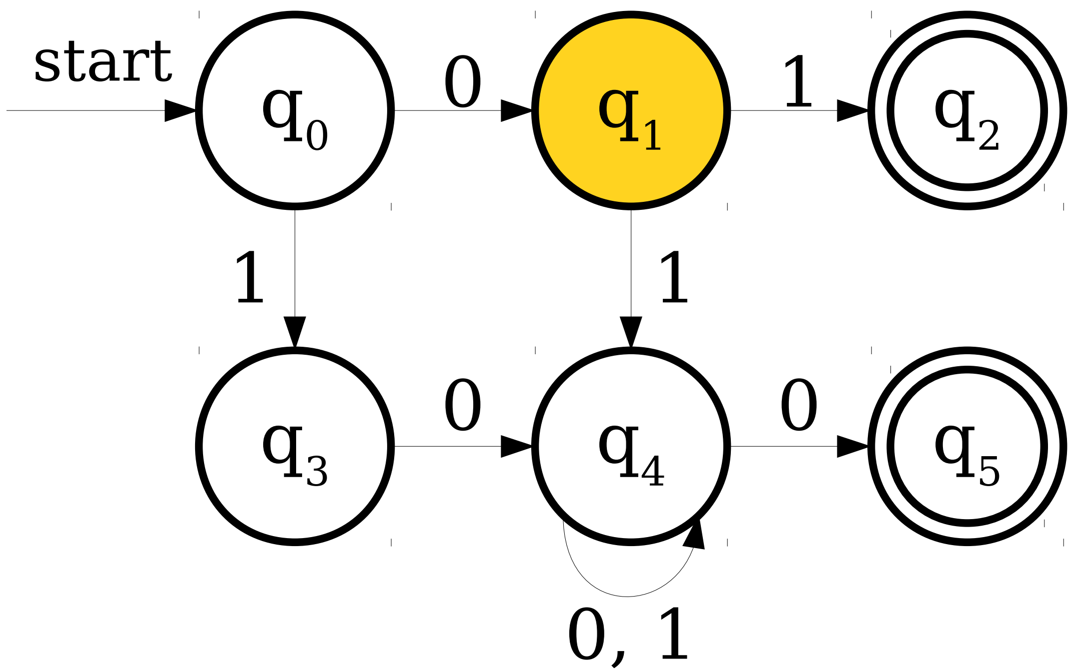




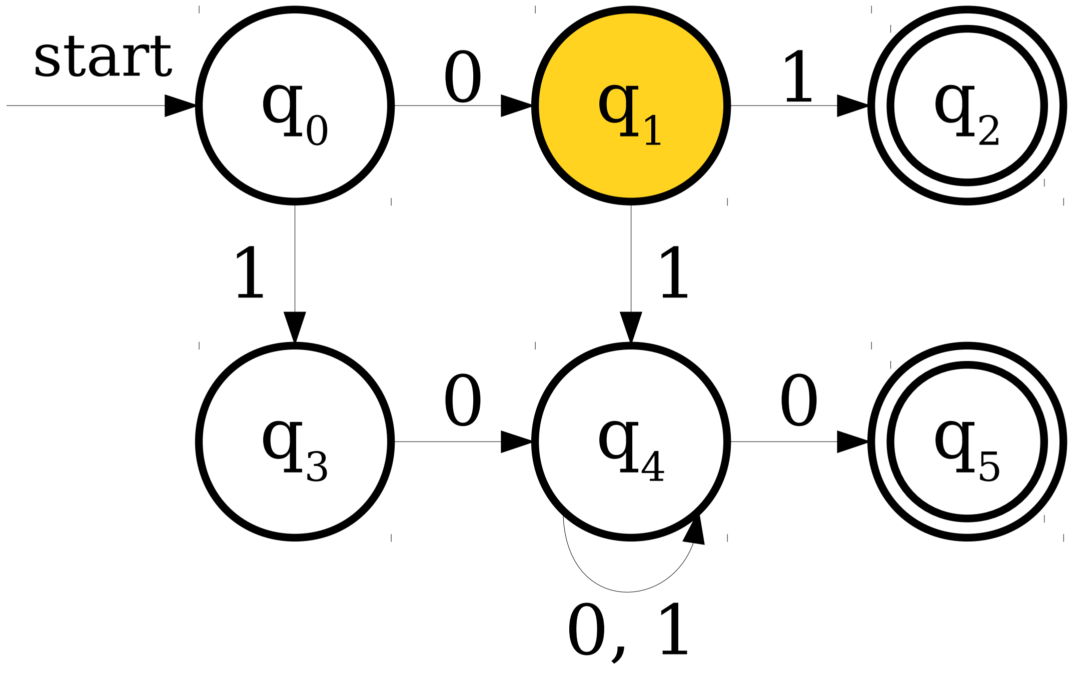
Massive Parallelism

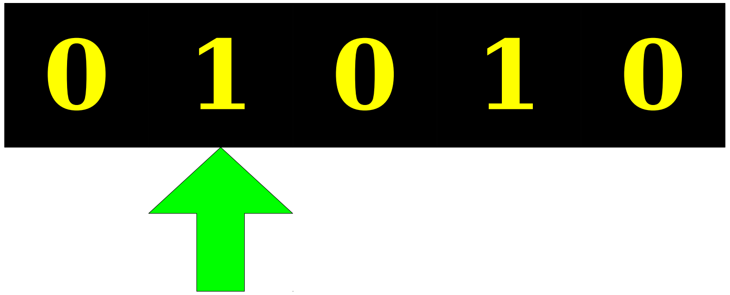


Massive Parallelism

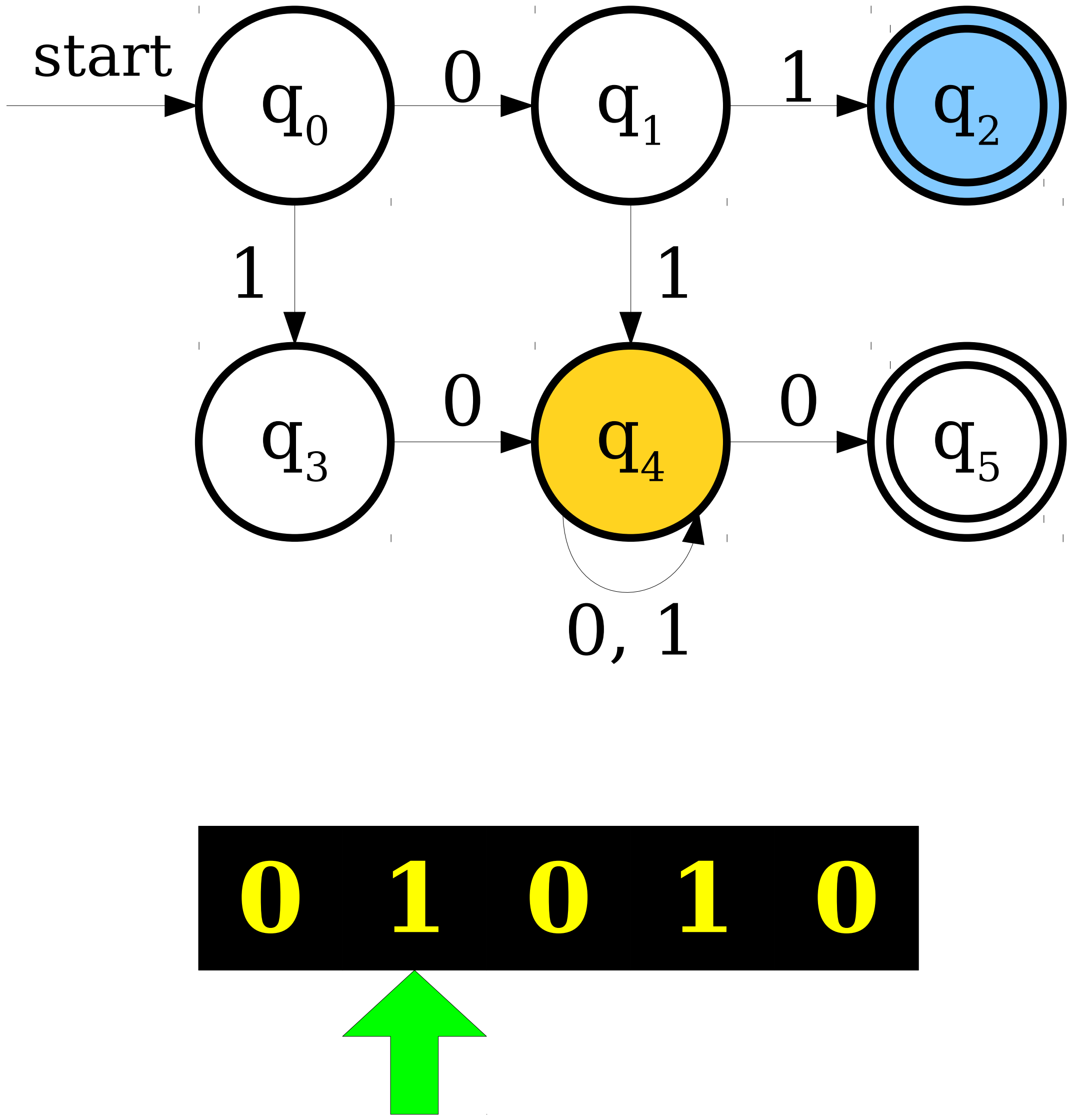


Massive Parallelism

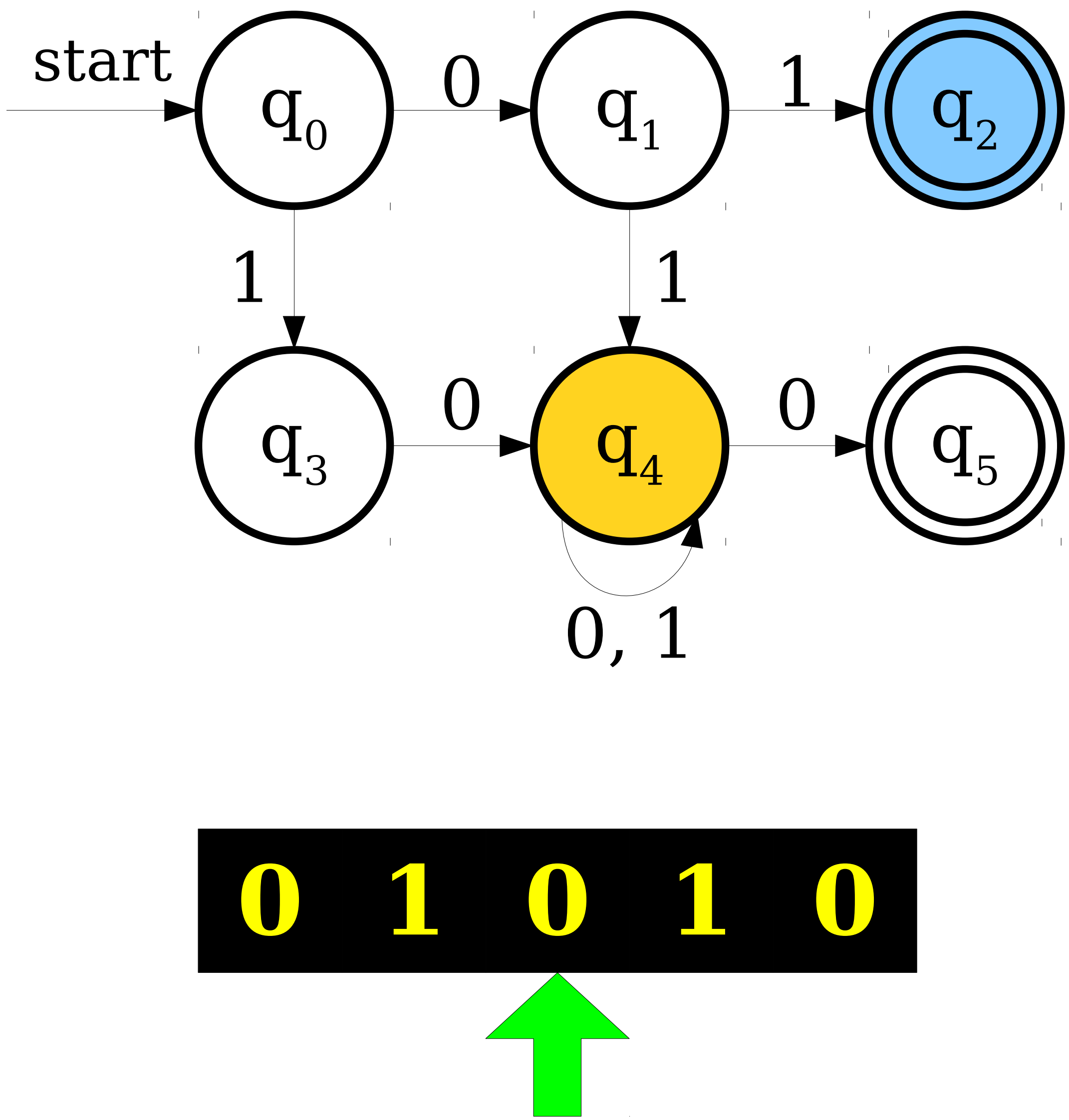




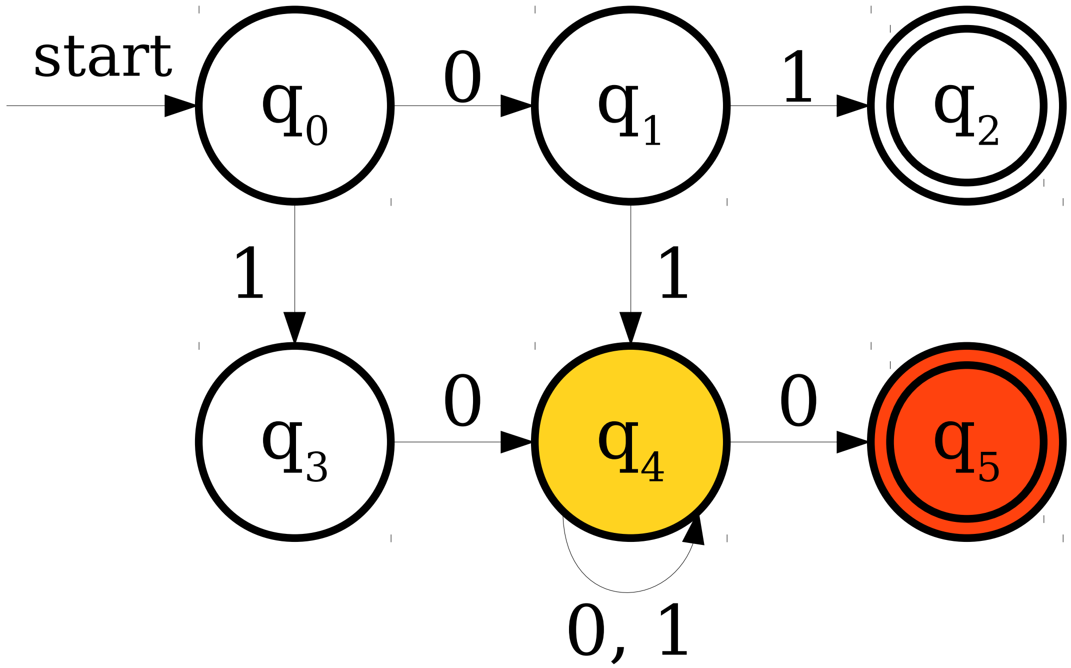
Massive Parallelism

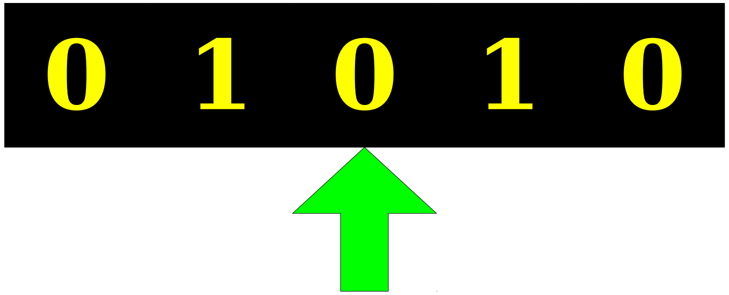


Massive Parallelism

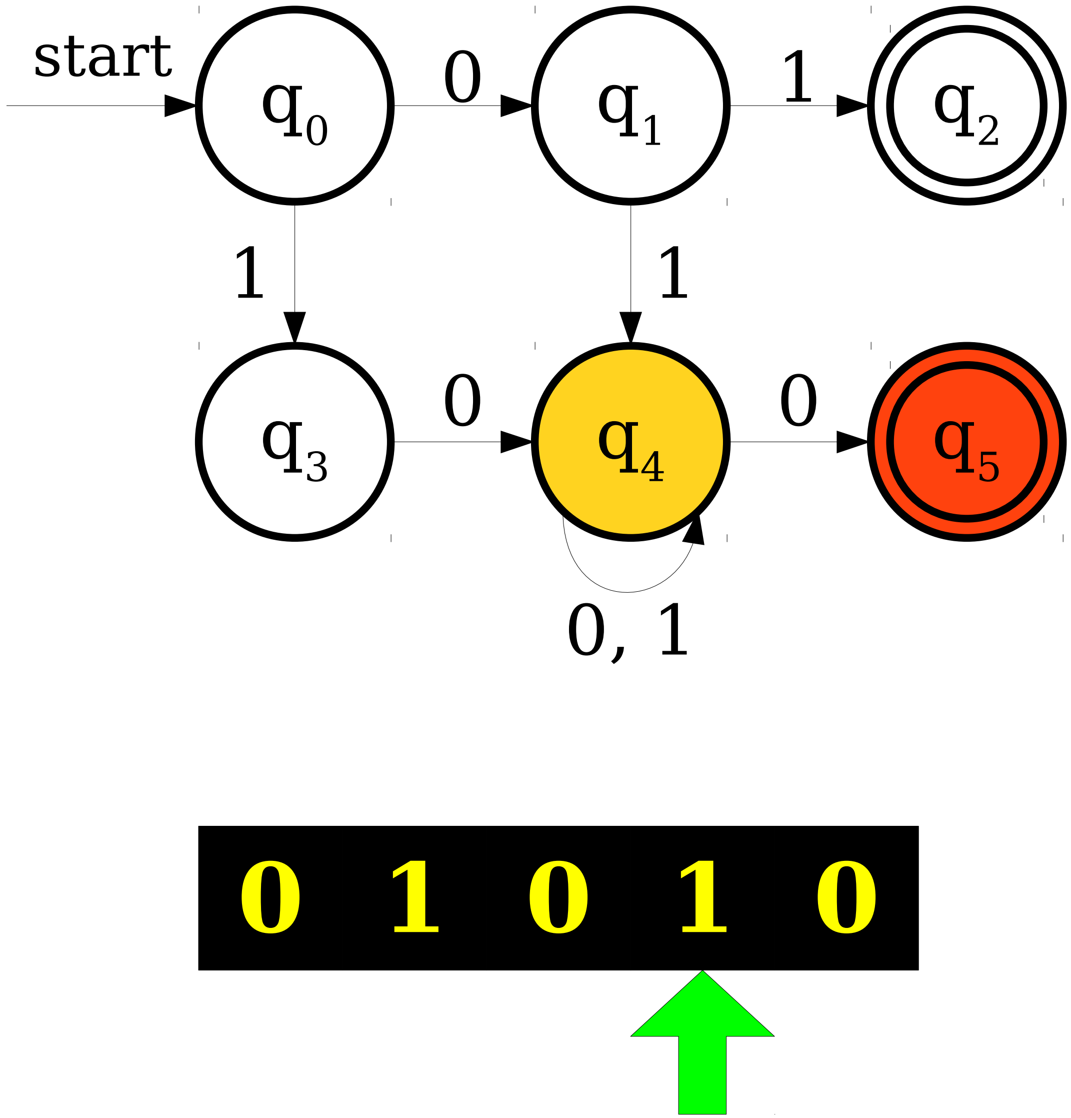


Massive Parallelism

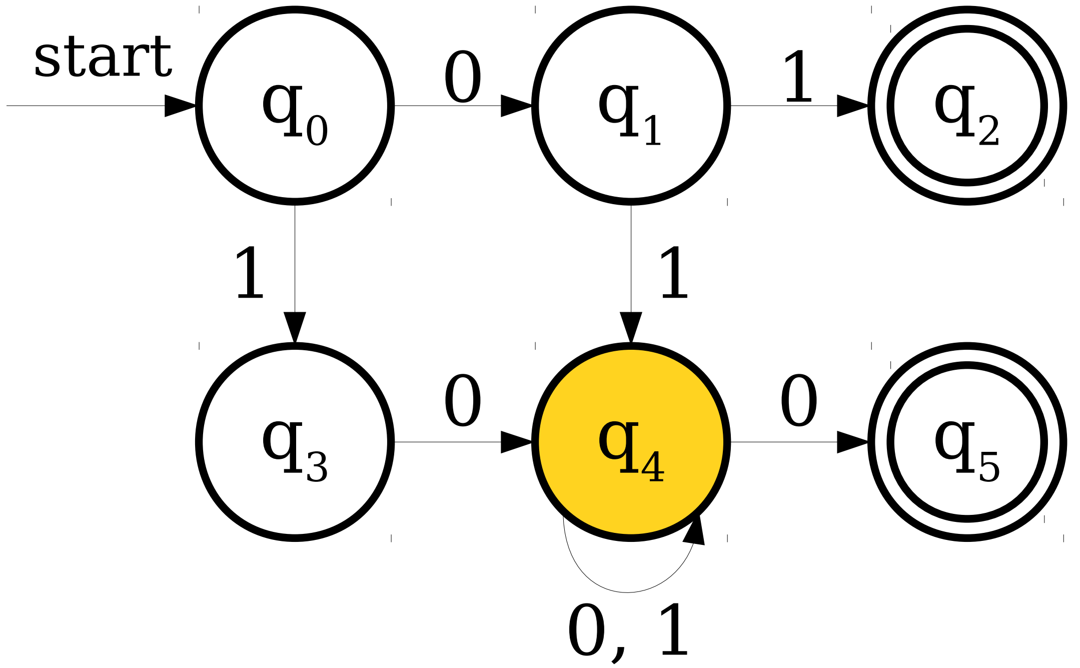


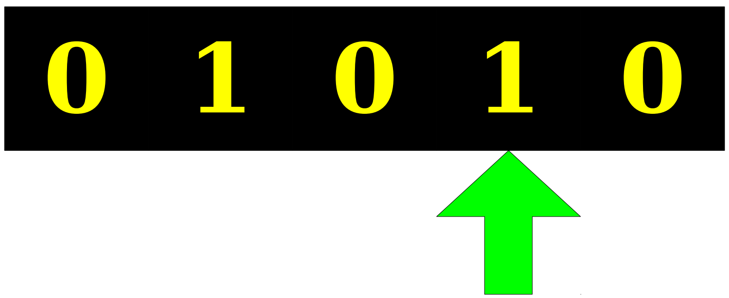


Massive Parallelism

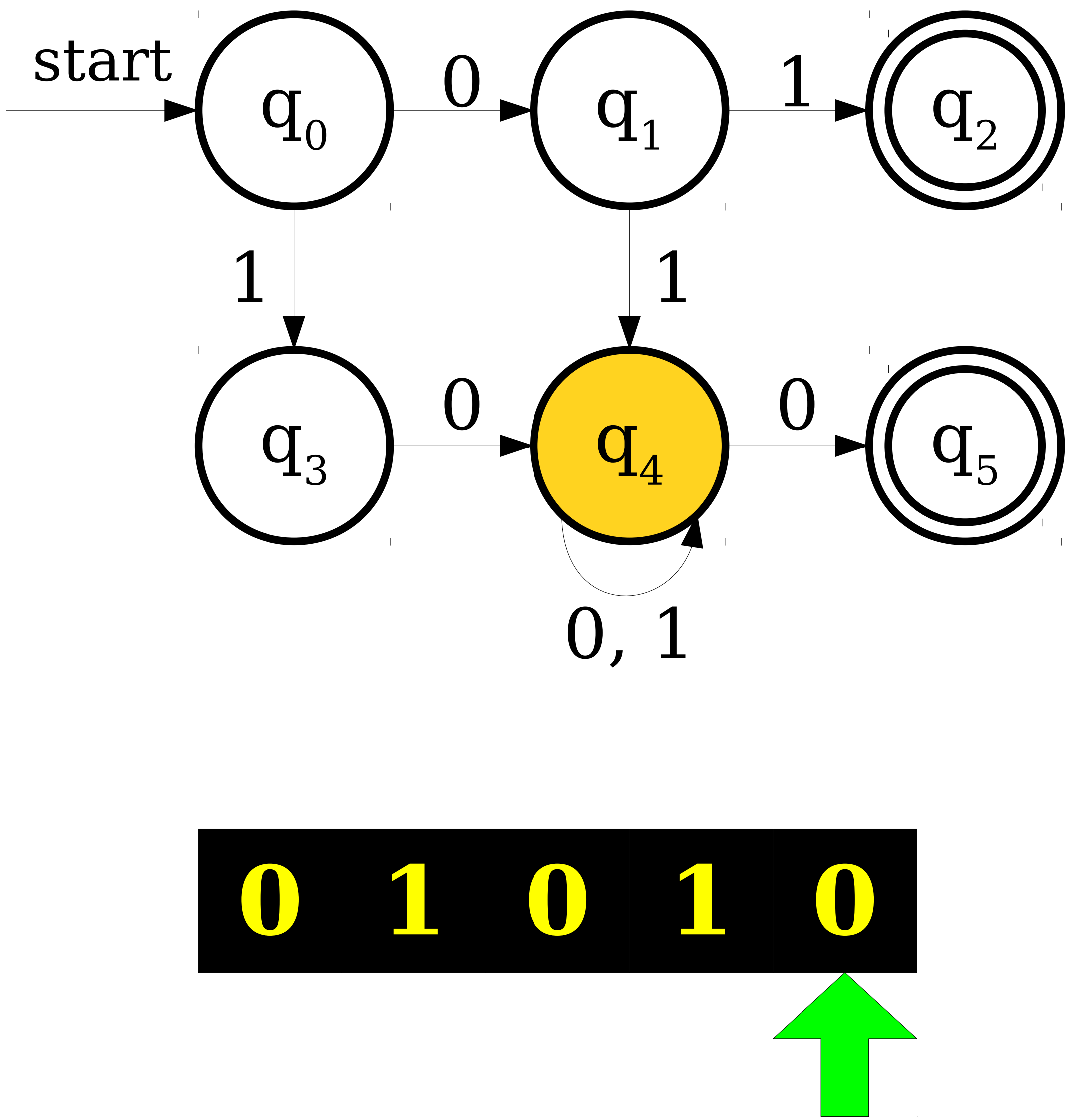


Massive Parallelism

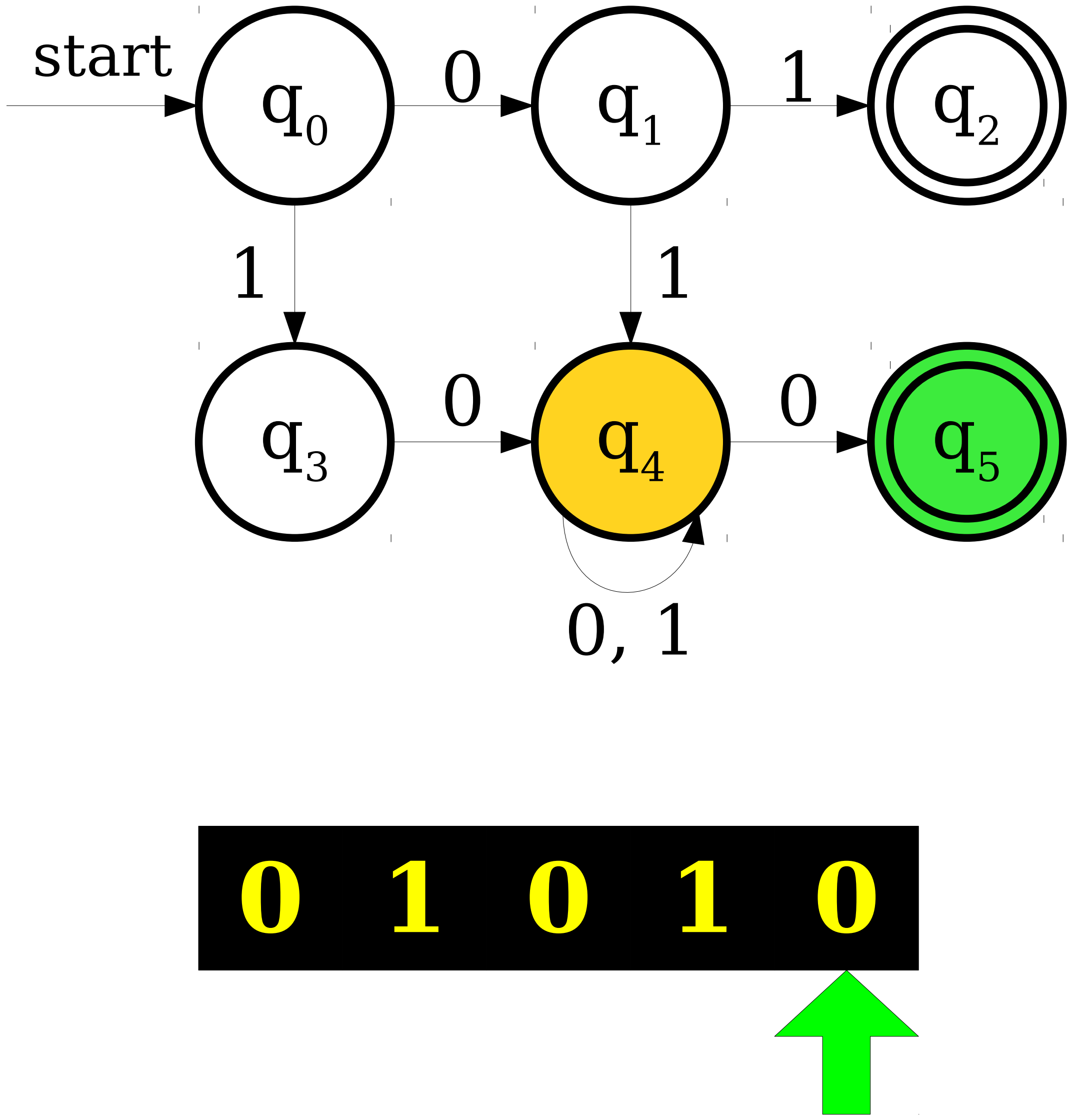




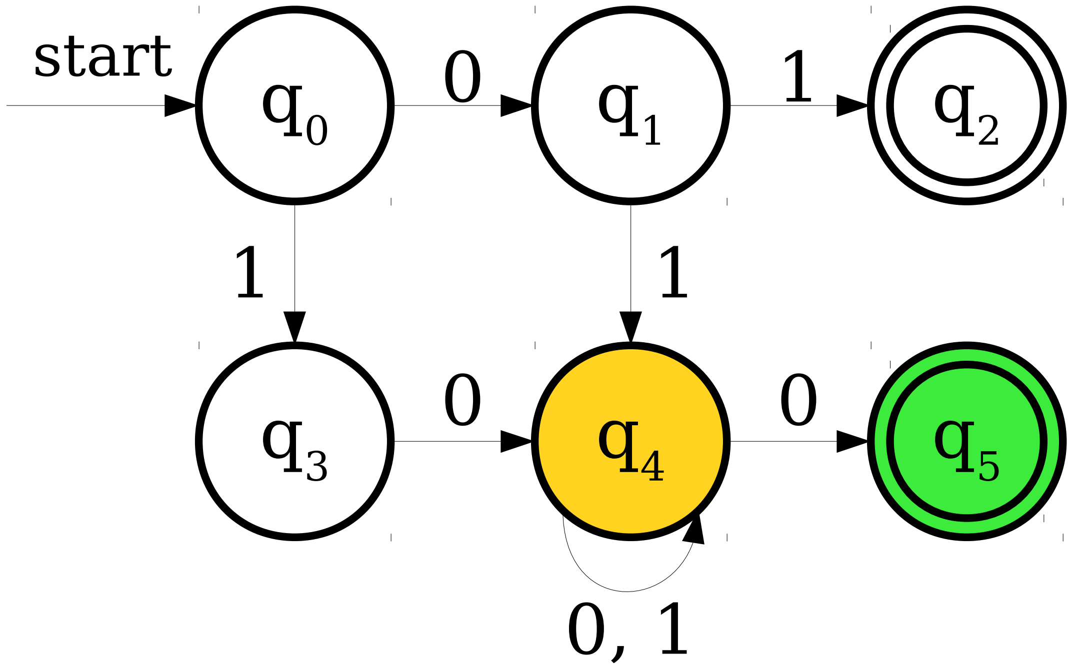
Massive Parallelism



Massive Parallelism

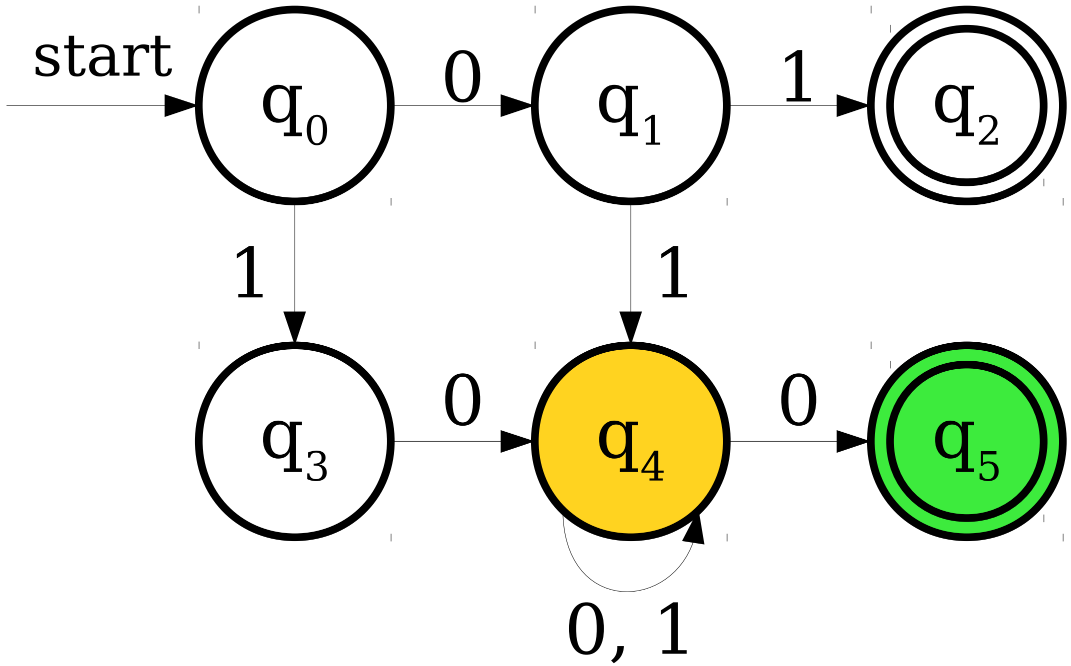


Massive Parallelism



0 1 0 1 0

Massive Parallelism



0 1 0 1 0

Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states at once.
- Each symbol read causes a transition on every active state into each potential state that could be visited.
- Nondeterministic machines can be thought of as machines that can try any number of options in parallel.
  - No fixed limit on processors; makes multicore machines look downright wimpy!

So What?

- We will turn to these three intuitions for nondeterminism more later in the quarter.
- Nondeterministic machines may not be feasible, but they give a great basis for interesting questions:
  - Can any problem that can be solved by a nondeterministic machine be solved by a deterministic machine?
  - Can any problem that can be solved by a nondeterministic machine be solved ***efficiently*** by a deterministic machine?
- The answers vary from automaton to automaton.

Designing NFAs

Designing NFAs

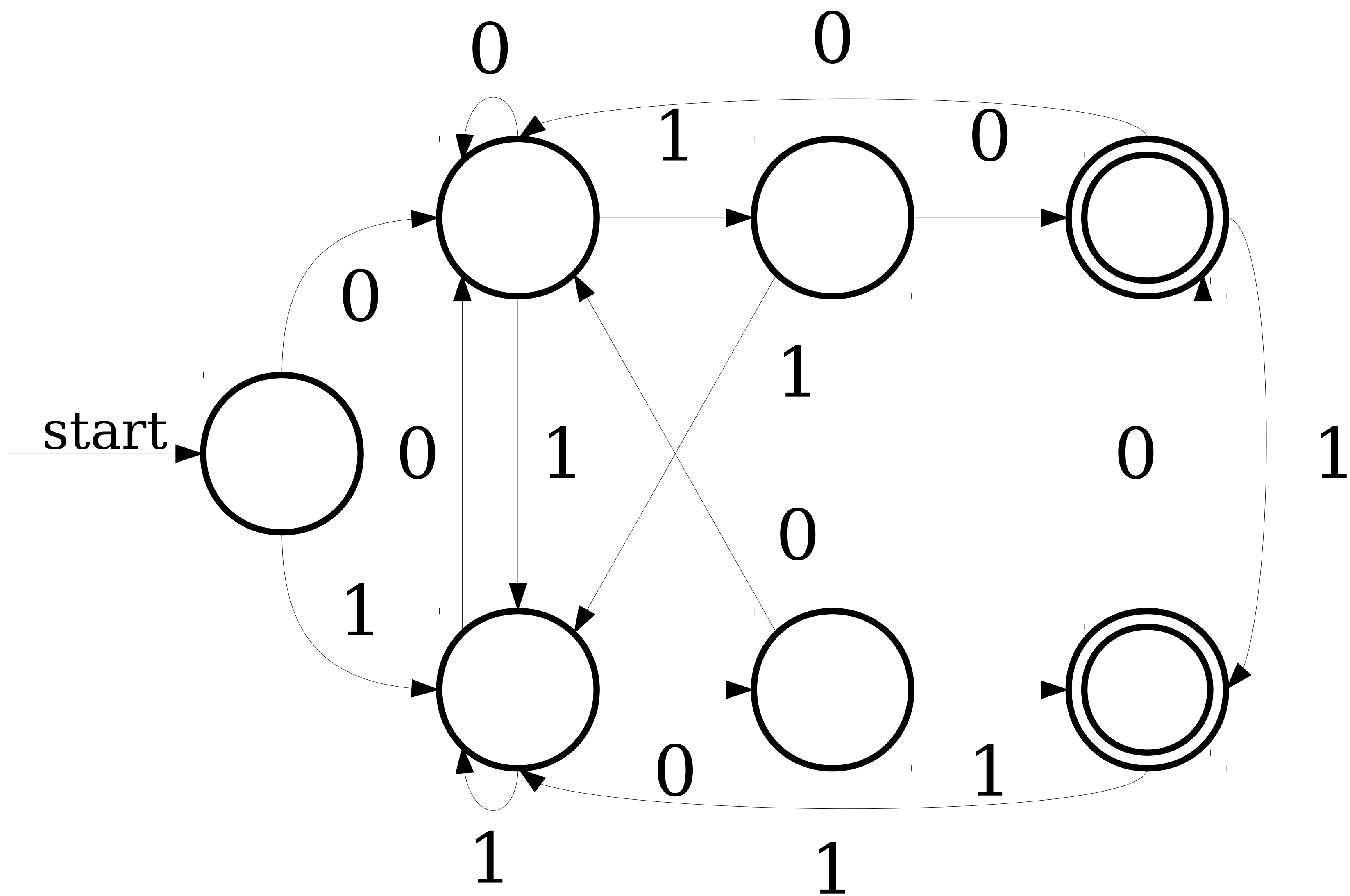
- When designing NFAs, *embrace the nondeterminism!*
- Good model: **Guess-and-check**:
  - Have the machine *nondeterministically guess* what the right choice is.
  - Have the machine *deterministically check* that the choice was correct.
- The *guess* phase corresponds to trying lots of different options.
- The *check* phase corresponds to filtering out bad guesses or wrong options.

Guess-and-Check

$L = \{\ w \in \{\mathtt{0}, \mathtt{1}\}^* \mid w \text{ ends in } \mathtt{010} \text{ or } \mathtt{101}\ \}$

Guess-and-Check

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101} \}$

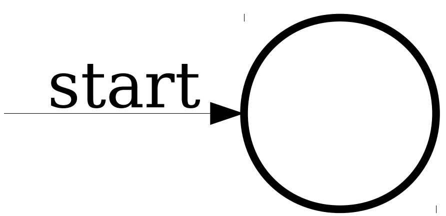


Guess-and-Check

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101} \}$

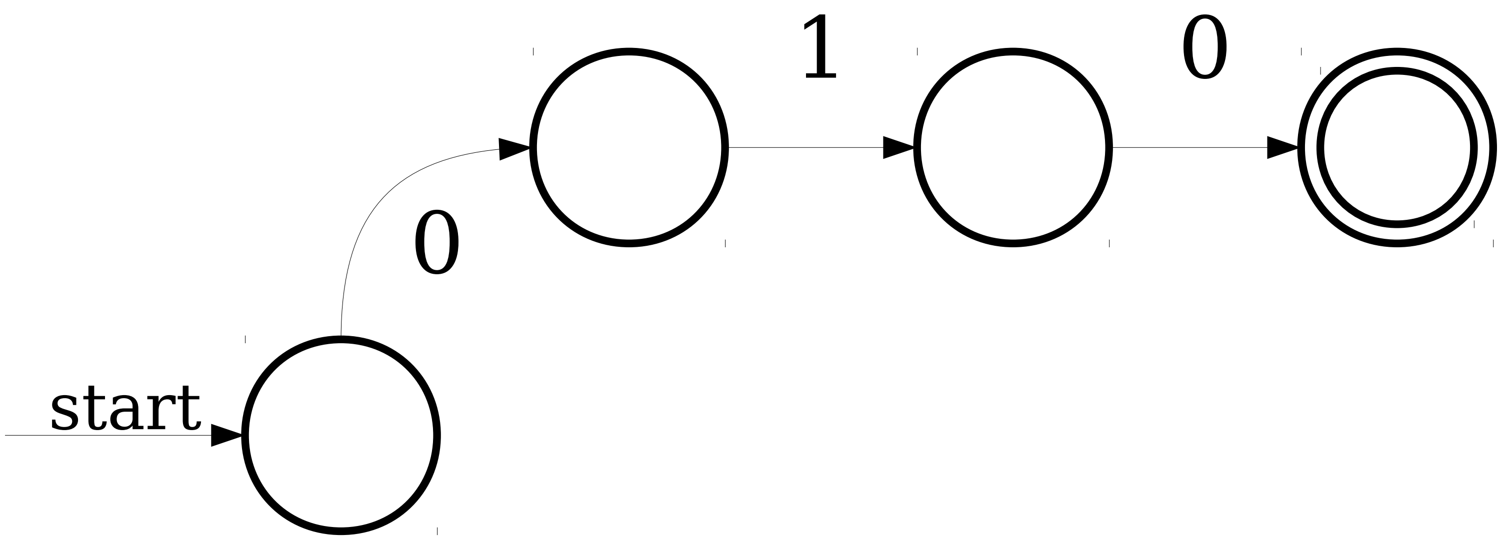
Guess-and-Check

$$L = \{ \ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101} \ \}$$



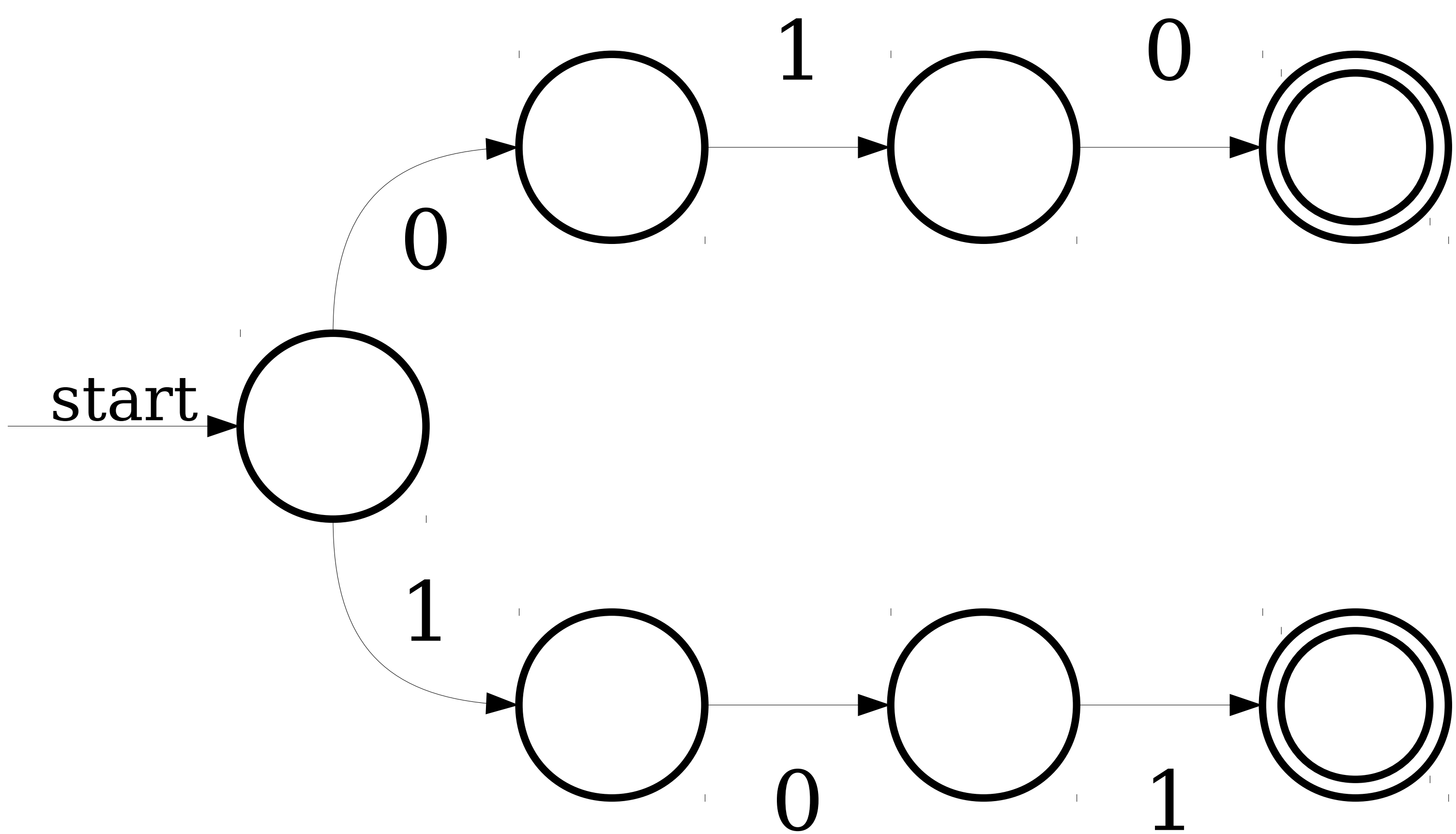
Guess-and-Check

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101} \}$



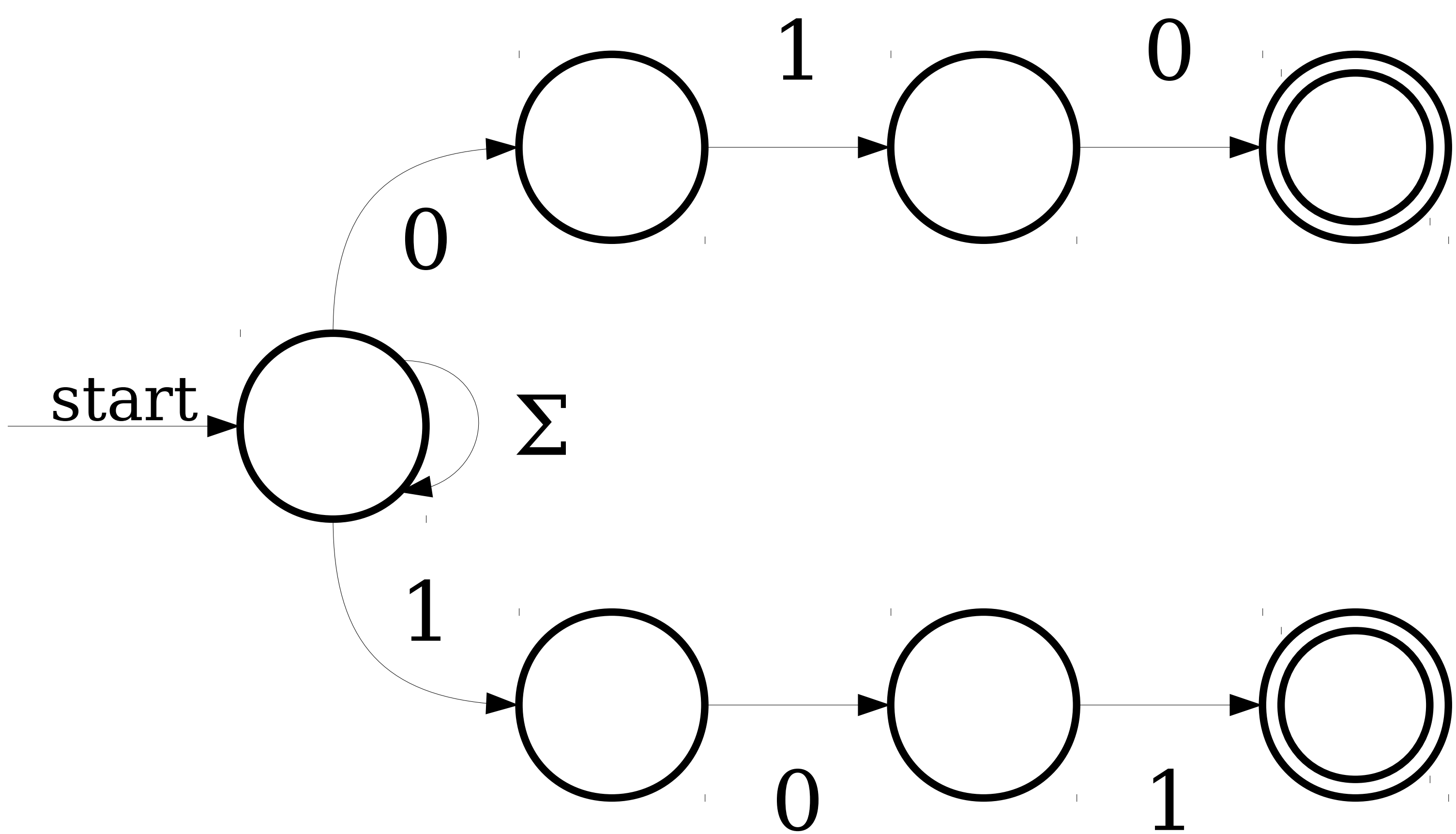
Guess-and-Check

$$L = \{\ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101}\ \}$$



Guess-and-Check

$L = \{\ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101}\ \}$

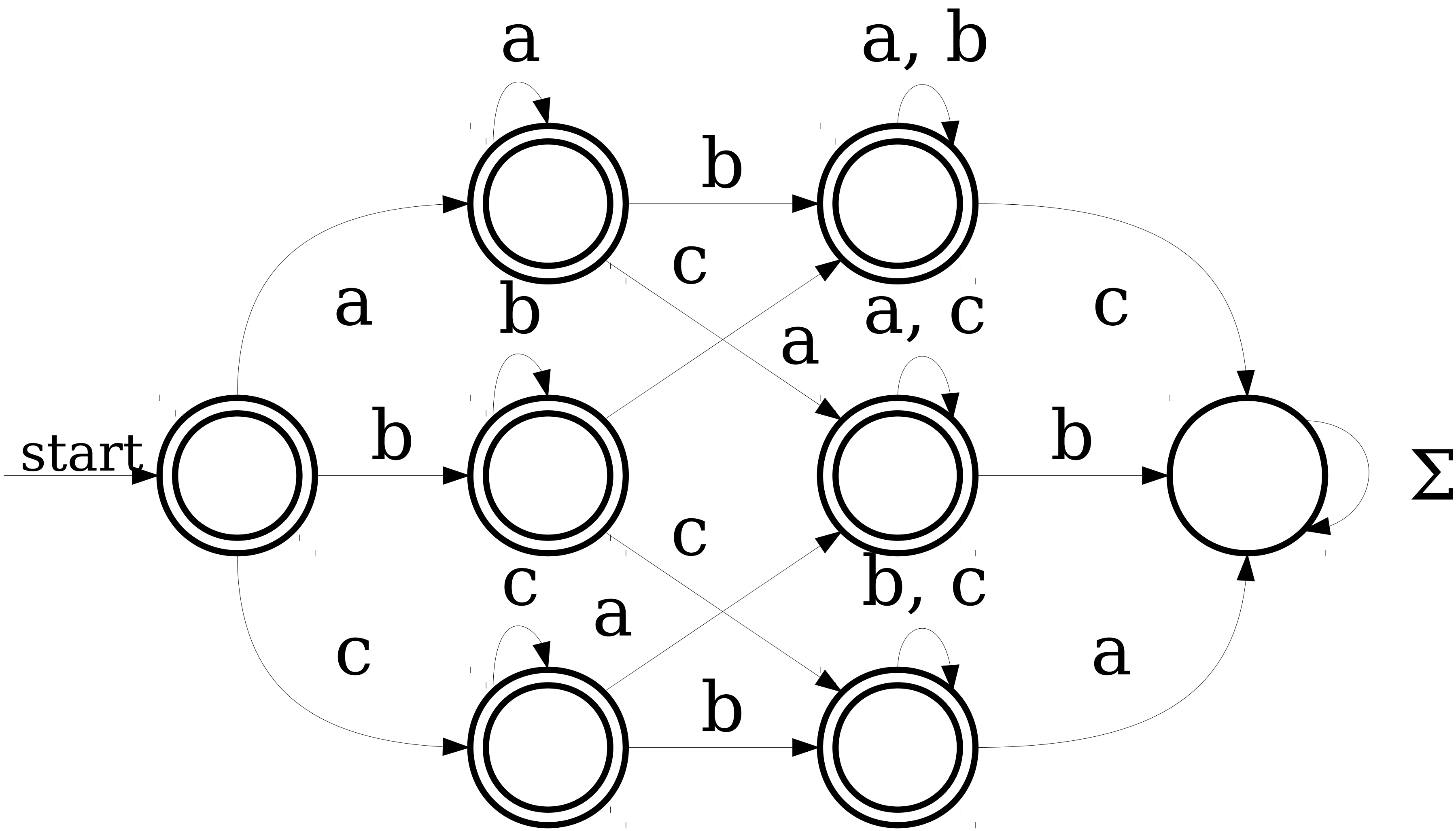


Guess-and-Check

$L = \{ w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* \mid \text{at least one of } \texttt{a}, \texttt{b}, \text{ or } \texttt{c} \text{ is not in } w \}$

Guess-and-Check

$L = \{\ w \in \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}^* \mid \text{at least one of } \mathbf{a}, \mathbf{b}, \text{ or } \mathbf{c} \text{ is not in } w\ \}$

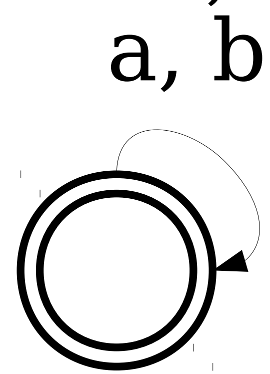


Guess-and-Check

$L = \{ w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \text{at least one of } \mathtt{a}, \mathtt{b}, \text{ or } \mathtt{c} \text{ is not in } w \}$

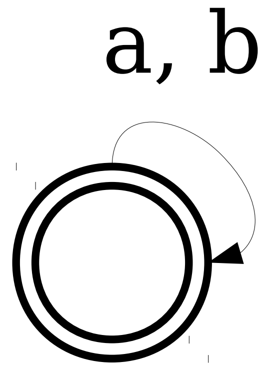
Guess-and-Check

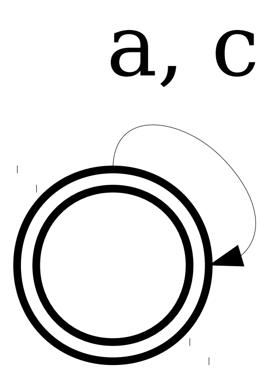
$L = \{\ w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid$ at least one of $\mathtt{a}$, $\mathtt{b}$, or $\mathtt{c}$ is not in w $\}$



Guess-and-Check

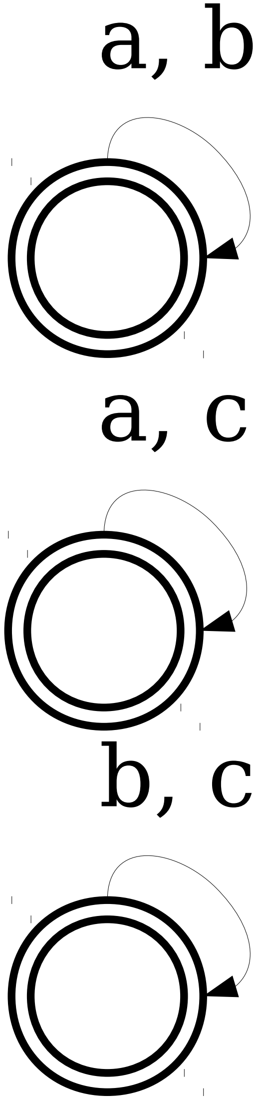
$L = \{ w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \text{at least one of } \mathtt{a}, \mathtt{b}, \text{ or } \mathtt{c} \text{ is not in } w \}$





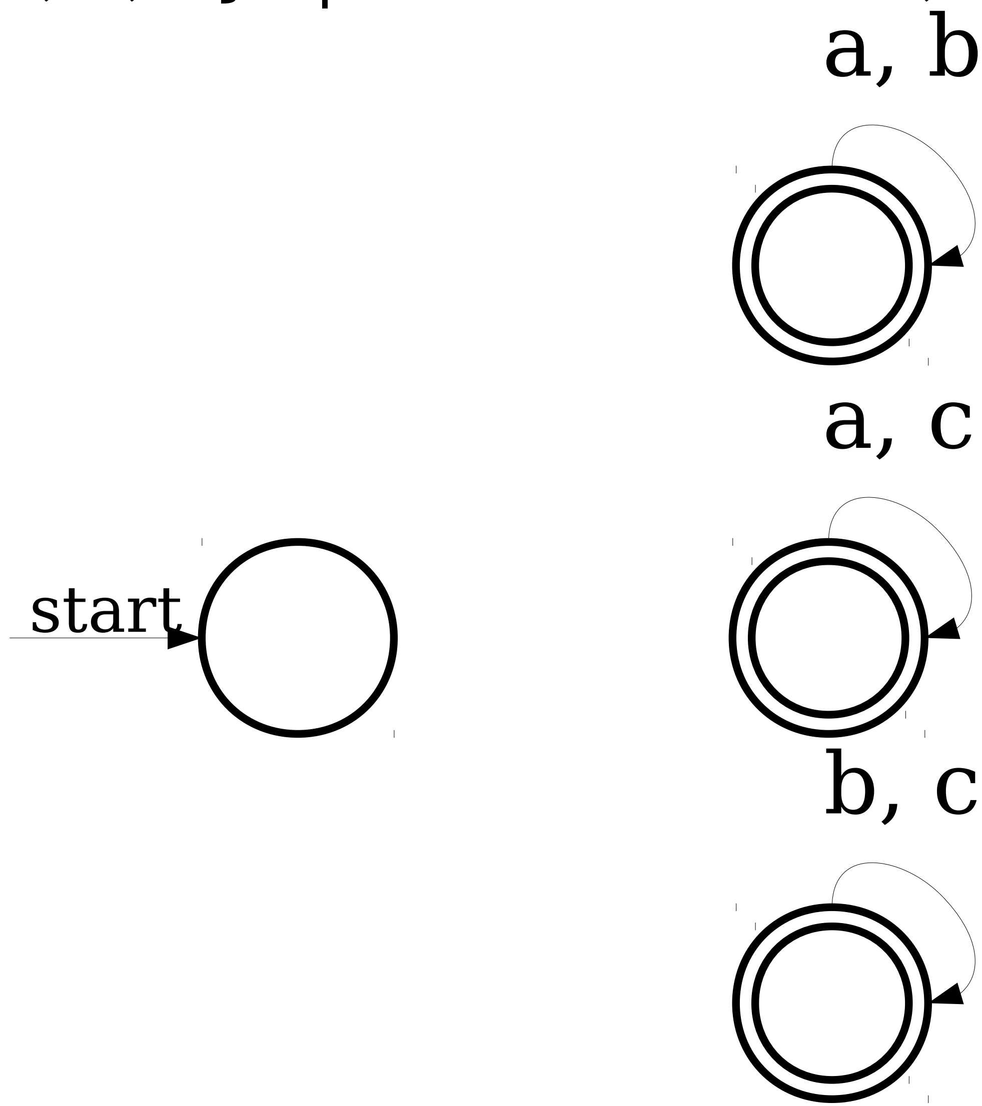
Guess-and-Check

$L = \{ \ w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* \mid \text{at least one of } \texttt{a}, \texttt{b}, \text{ or } \texttt{c} \text{ is not in } w \ \}$



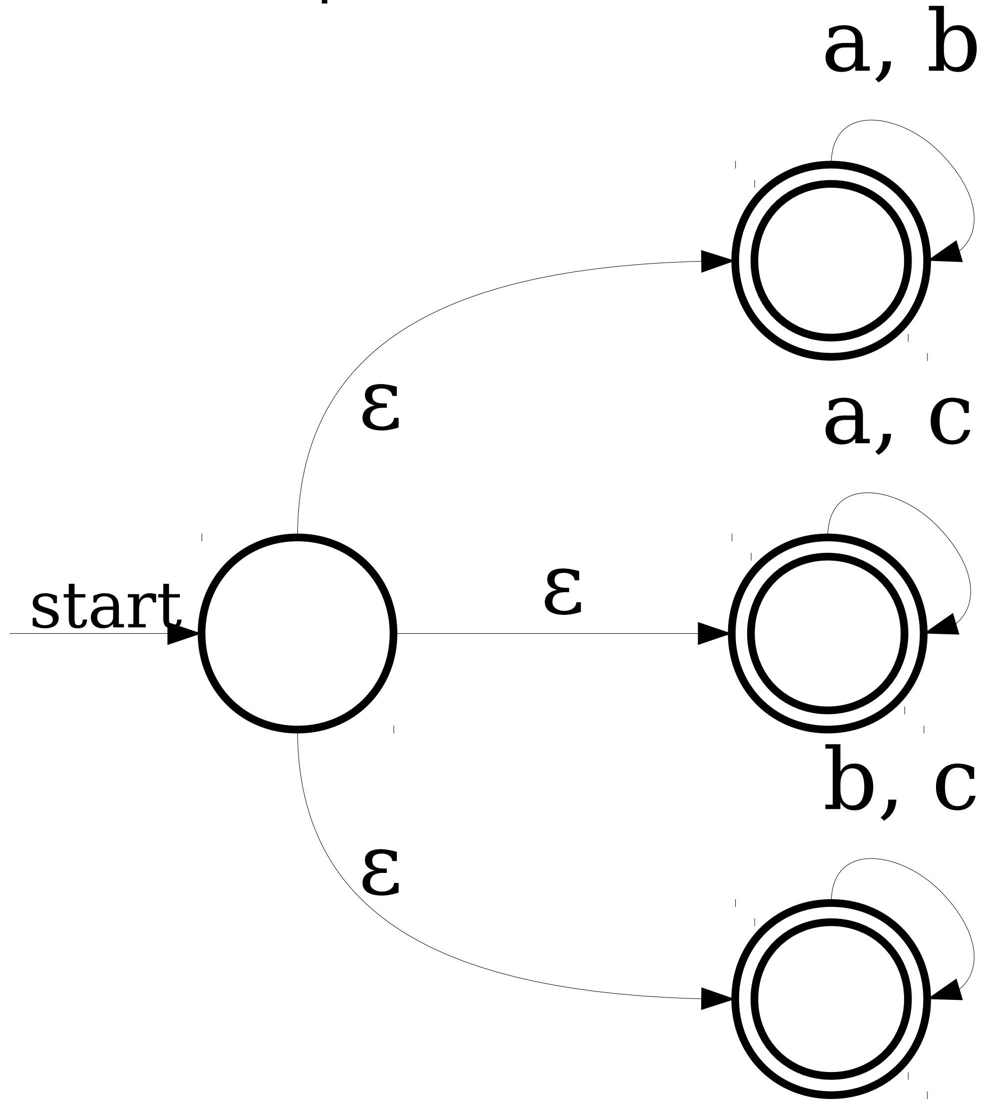
Guess-and-Check

$L = \{ w \in \{$**a**, **b**, **c**$\}^* \mid$ at least one of **a**, **b**, or **c** is not in w $\}$



Guess-and-Check

$L = \{ w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \text{at least one of } \mathtt{a}, \mathtt{b}, \text{or } \mathtt{c} \text{ is not in } w \}$



Next Time

- **NFAs and DFAs**
  - Are NFAs more powerful than DFAs?
- **Closure Properties**
  - More ways of transforming regular languages.
- **Regular Expressions**
  - A different perspective on regular languages.