10. Finite Automata and Turing Machines

Frank Stephan

March 20, 2014

Alan Turing's 100th Birthday

"Alan Turing was a completely original thinker who shaped the modern world, but many people have never heard of him.
Before computers existed, he invented a type of theoretical machine now called a Turing Machine, which formalized what it means to compute a number."
Posted on Google by Jered Wierzbicki and Corrie Scalisi, Software Engineers, and Sophia Foster-Dimino, Doodler

23 June 1912 – 7 June 1954

Google Doodle and Alan Turing's 100th Birthday

"Our doodle for his 100th birthday shows a live action Turing Machine with twelve interactive programming puzzles."

Alan Turing's 100th Birthday

The Quest for the Universal Machine

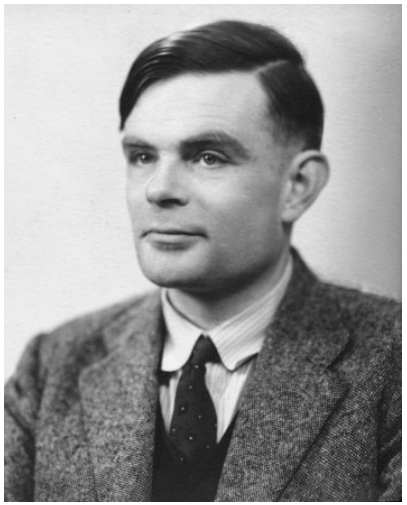
In order to study and understand what computer can and cannot do we need a mathematical model of computers, algorithms and computation.

The Quest only Begins

The λ-calculus was introduced by Alonzo Church in the 1932. The Turing machine was invented in 1936 by Alan Turing. Emil Post proposed Post canonical systems. Many other systems have been proposed since. More and more refined ones are being studied.

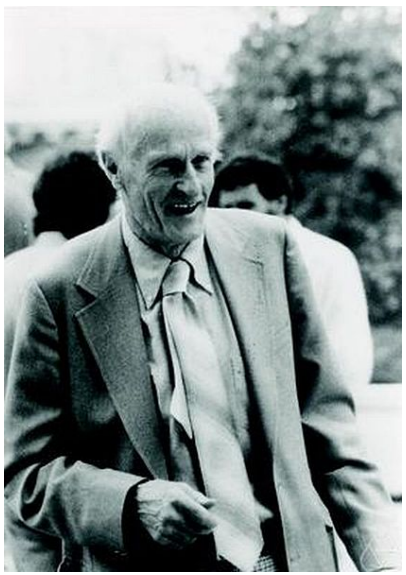
Church-Turing Thesis

The Church-Turing Thesis is a conjecture that the functions that can be computed (mechanically) by an algorithm are exactly those functions that can be computed by a Turing Machine (or by λ-calculus or by Post systems, or as counter programs, or as recursive functions as proved by Rosser in 1939, or $\cdots$).



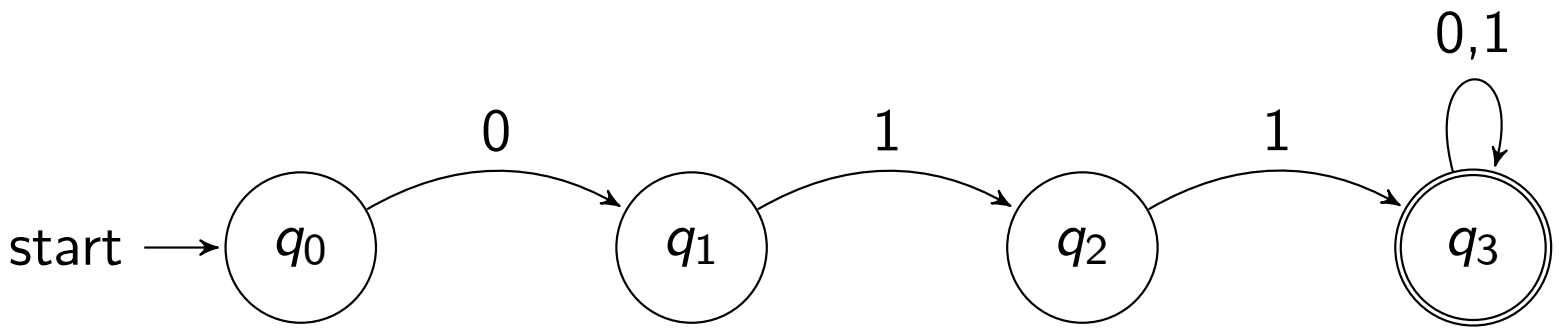
Recursive Functions

Stephen Kleene and logician John Rosser (students of Church) defined, following up some ideas by Gödel, a class of mathematical functions on natural numbers whose values can be calculated by recursion.



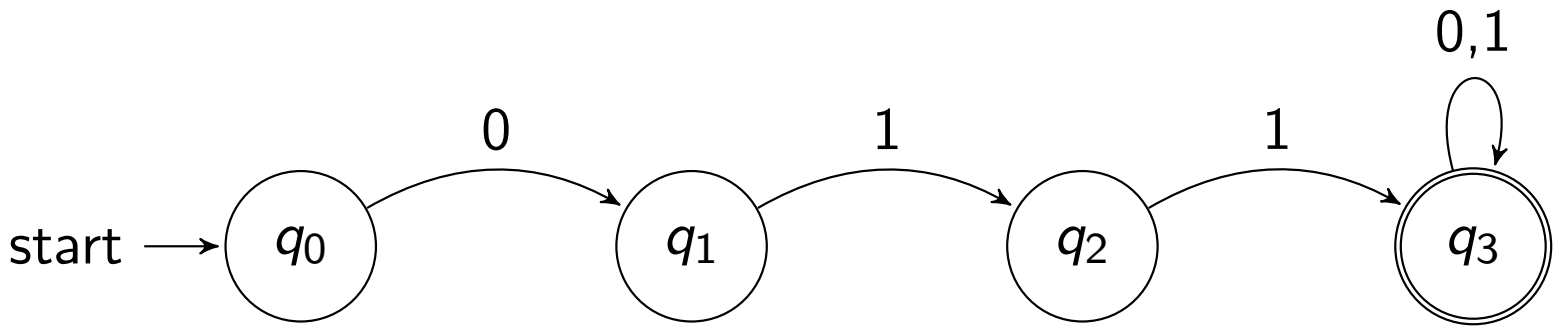
Non Deterministic Finite State Automaton (NDFA)

A finite state automaton is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where Q is set of states, Σ is a finite alphabet or set of symbols, δ is a transition relation $(Q \times \Sigma \times Q)$, q_0 is the initial state and F is a set of accepting states.

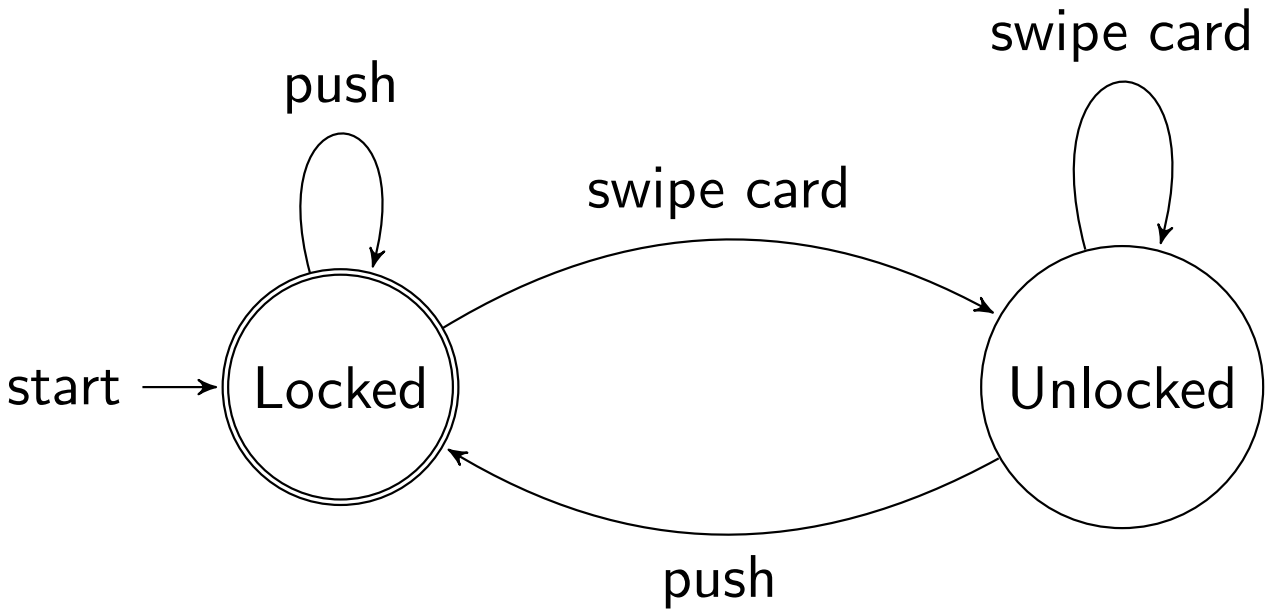


Language

A language on the alphabet Σ is a set of strings of symbols of Σ.



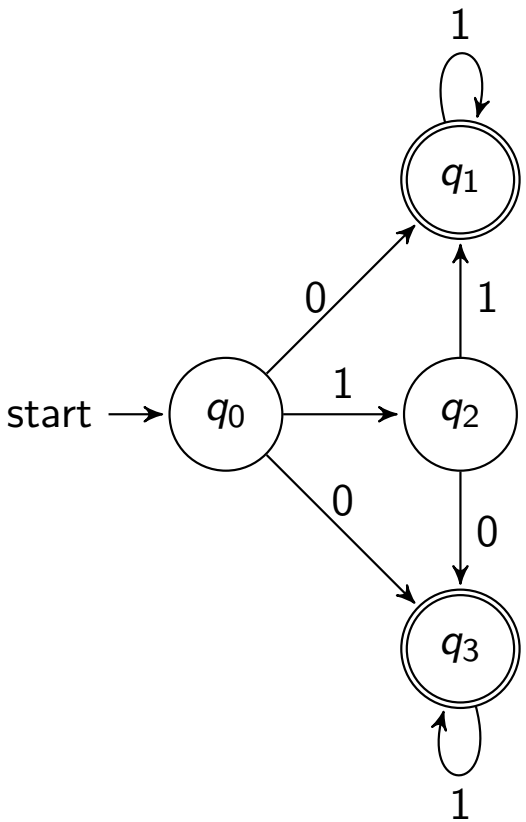
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$\delta = \{(q_0, 0, q_1), (q_1, 1, q_2), (q_2, 1, q_3), (q_3, 0, q_3), (q_3, 1, q_3)\}$
q_0
$F = \{q_3\}$
Load the dfa into a browser (additional input slide07)



Turnstile

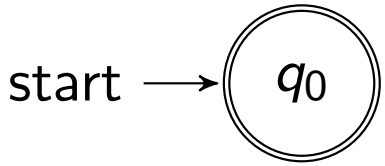
Recognizers

An automaton recognizes a language. It accepts (or generates) strings of symbols of that language.



This non-deterministic automaton accepts strings such as 0, 01, 011, 01111, 11, 111111, 10 and 101111.

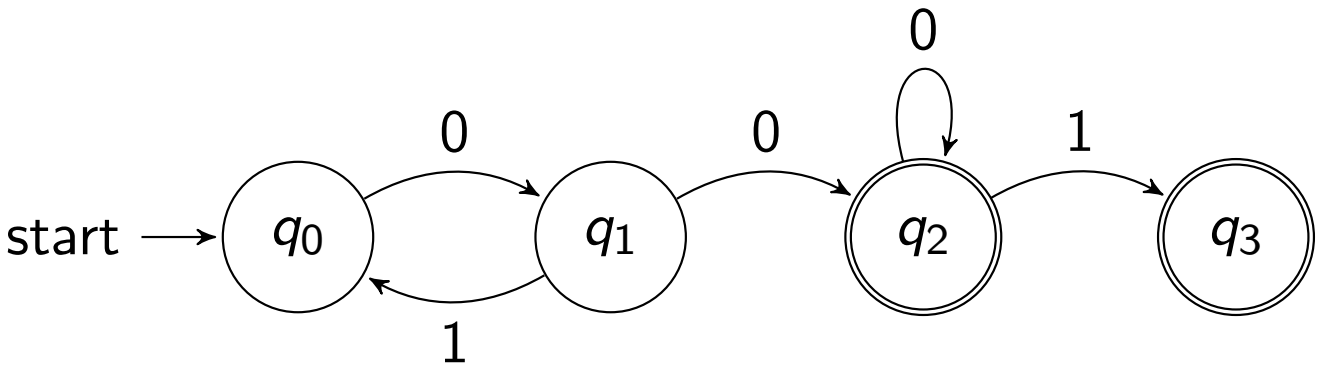
It recognizes the language $L = (01^*) \mid (111^*) \mid (01^*) \mid (101^*)$



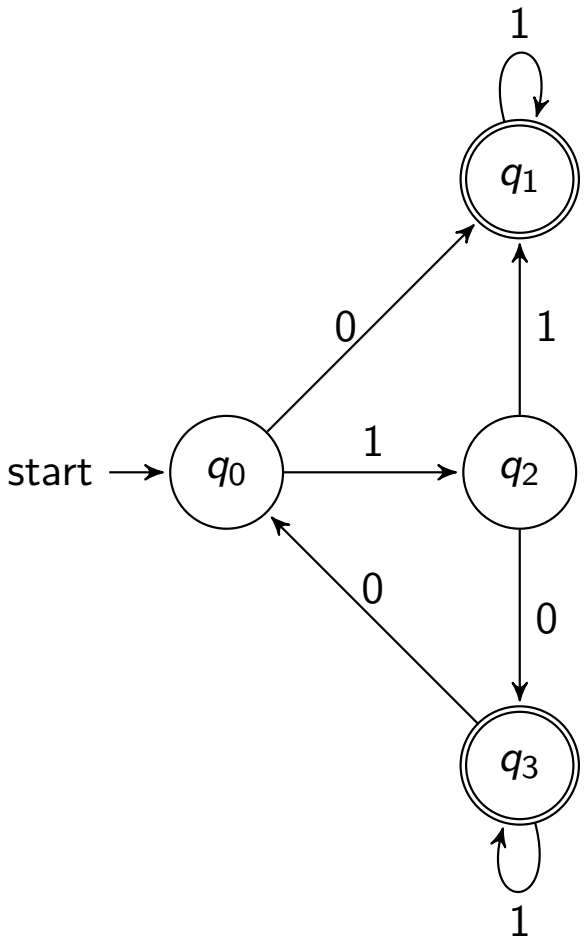
This automaton accepts the empty string ϵ. It recognizes the language $L = \{\epsilon\}$

Deterministic Finite State Automaton (DFA)

A finite state automaton is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where Q is set of states, Σ is a finite alphabet or set of symbols, δ is a transition function $(Q \times \Sigma \to Q)$, q_0 is the initial state and F is a set of accepting states.



This DFA accepts all words consiting of arbitrary many 01 followed by 00 followed by arbitrary many 0 followed by up to one 1.



This automaton accepts strings such as 0, 11, 10, 1000, 1011111100 and 101111111.

It recognizes the language $L = (101^*0)^*((01^*) \mid (111^*) \mid (101^*))$

Load the dfa into a browser (additional input slide13)

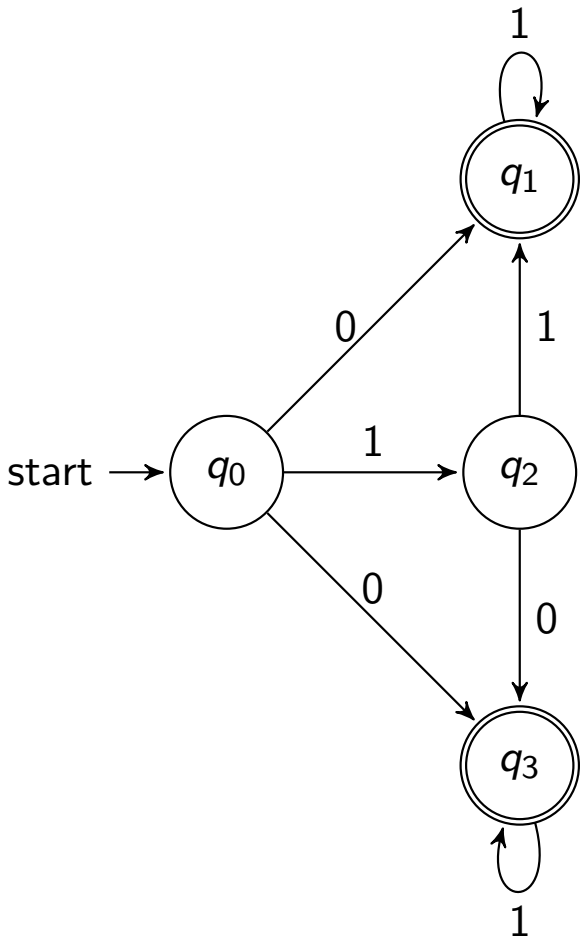
Equivalence (Büchi's Theorem)

For every non deterministic finite state automaton there exists a deterministic finite state automaton that recognizes the same language.

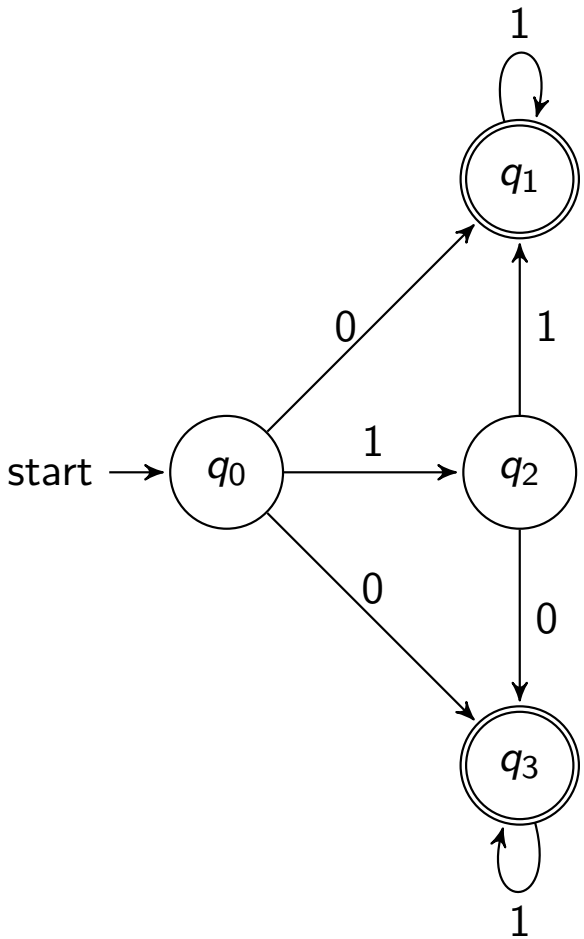
Converting an NFA to a DFA

Each state in the DFA corresponds to a set of NFA states.

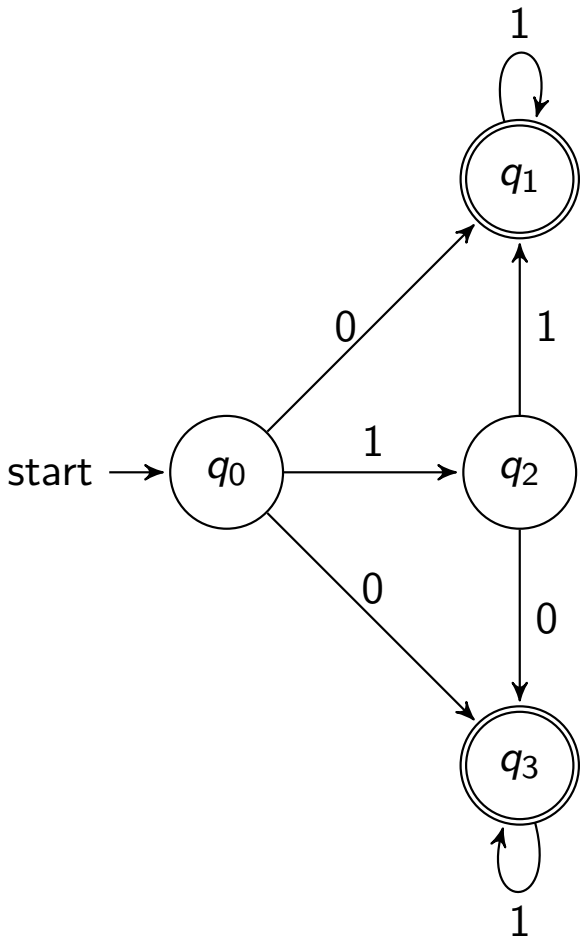
- Merge the initial states into one.
- For each transition create a state that merges all the reachable states.
- Repeat the procedure for the newly created states.
- States that contain an accepting state are accepting.



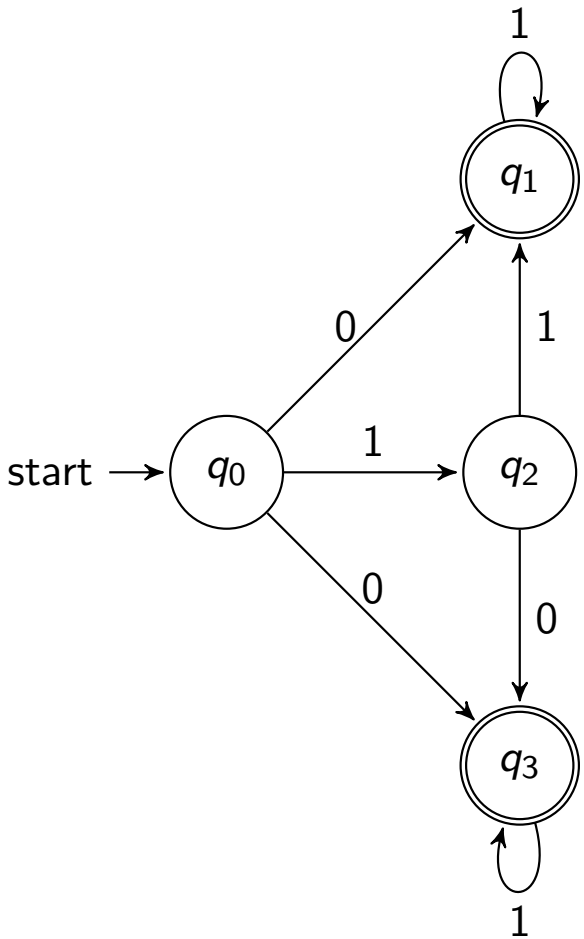
Type	New State	0	1
	q_0		



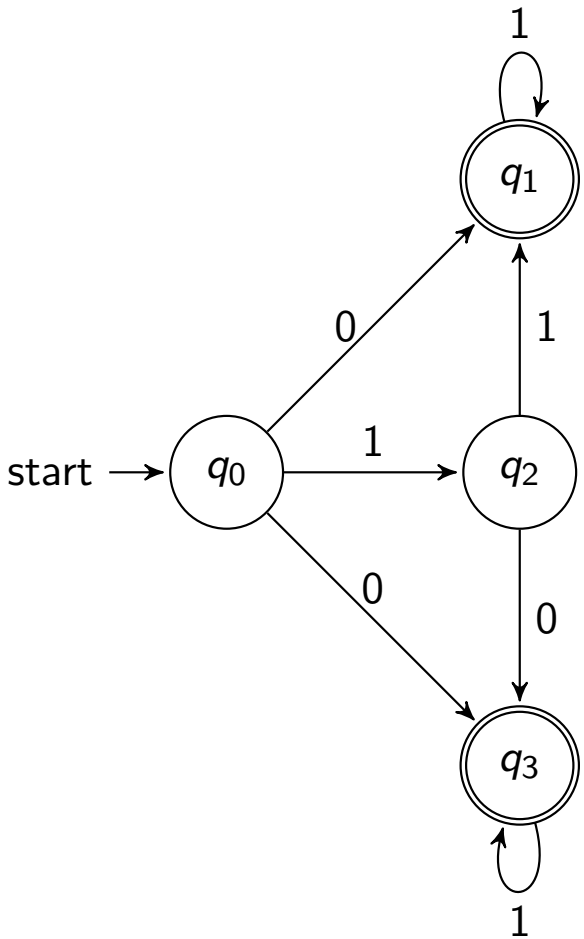
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2



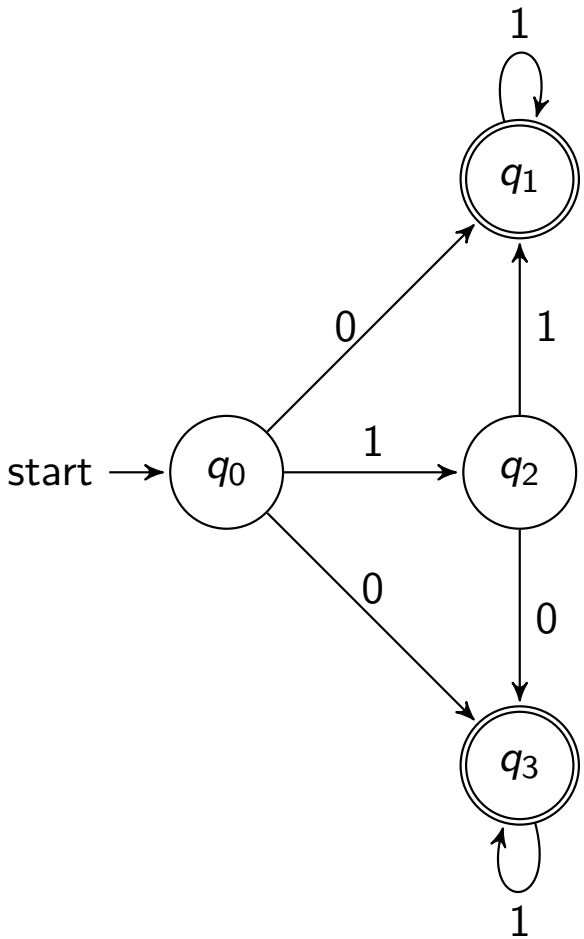
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
	q_1, q_3		



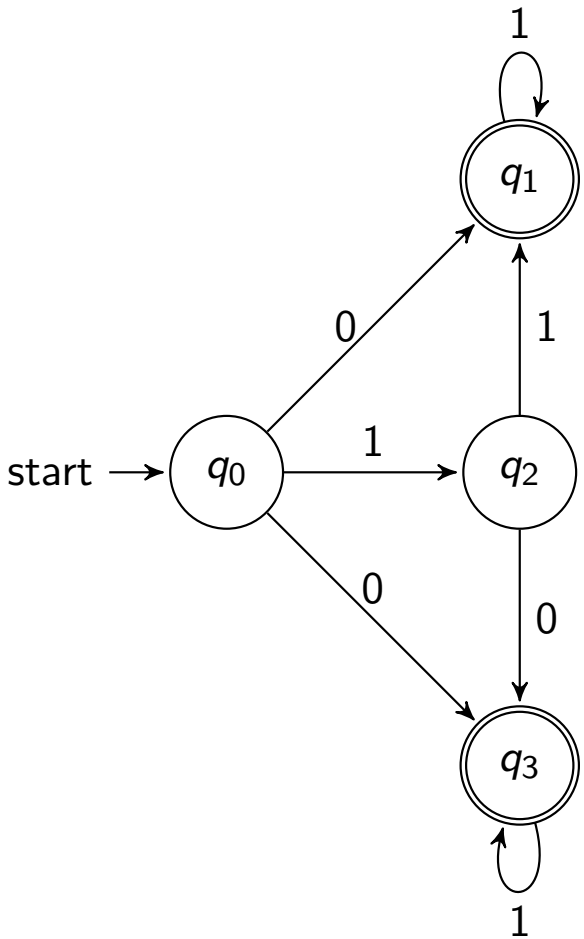
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
$\circledcirc$	q_1, q_3	none	q_1, q_3



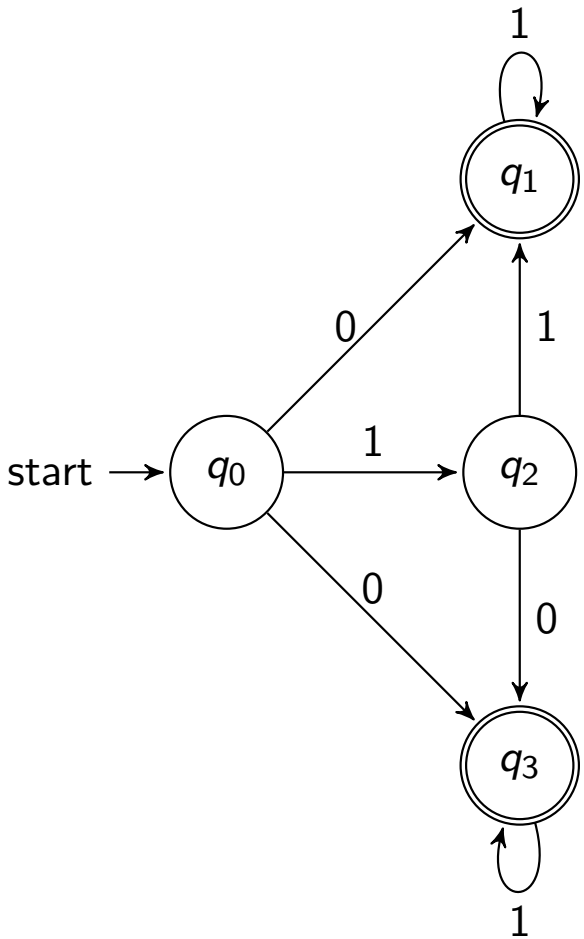
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
$\circledcirc$	q_1, q_3	none	q_1, q_3
	q_2		



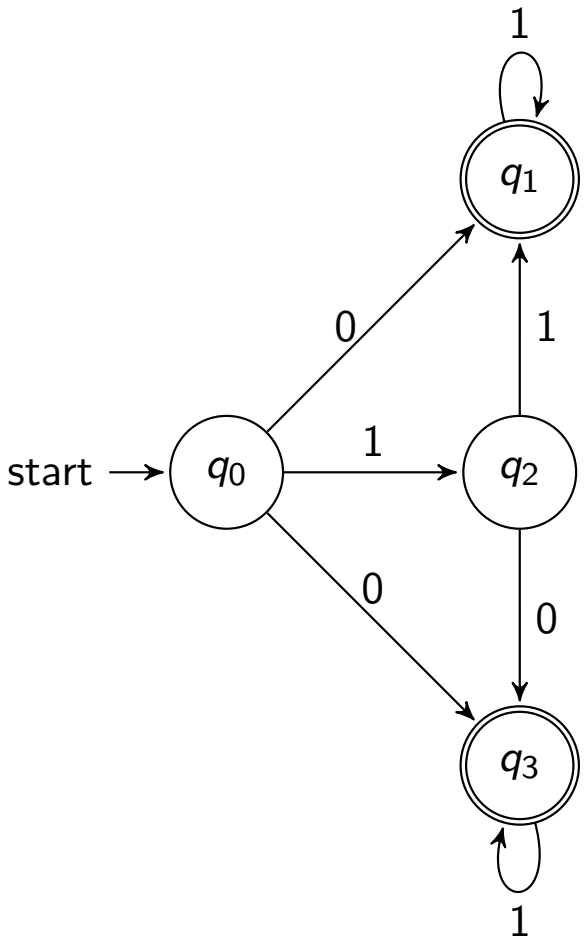
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
◎	q_1, q_3	none	q_1, q_3
	q_2	q_3	q_1



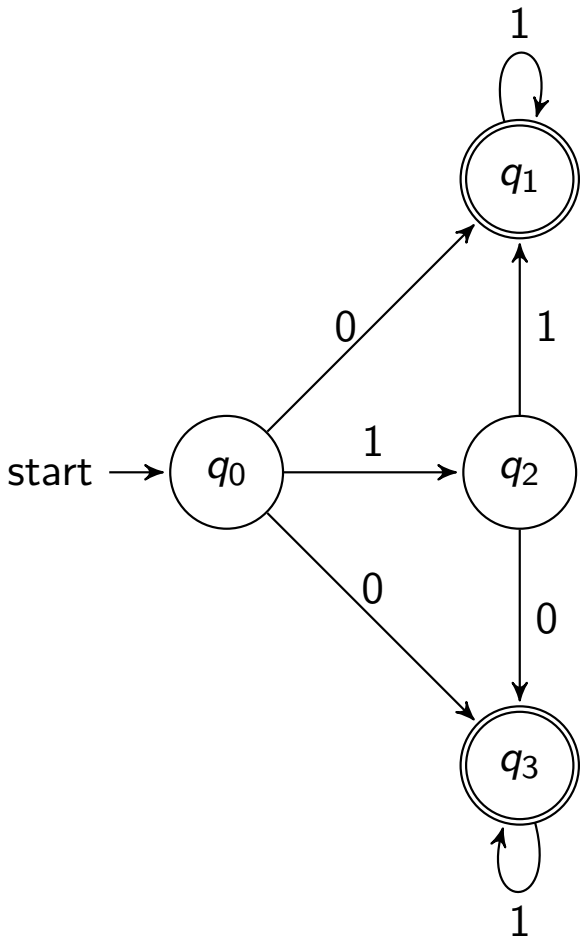
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
$\circledcirc$	q_1, q_3	none	q_1, q_3
	q_2	q_3	q_1
	q_3		



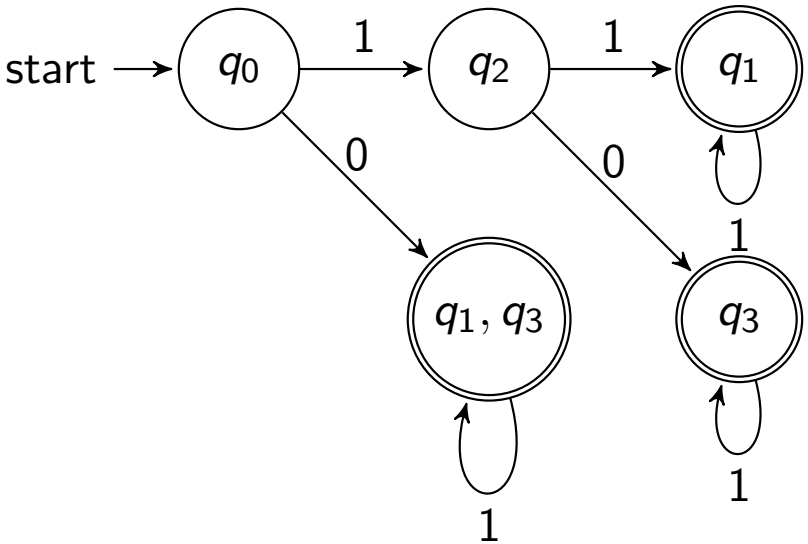
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
◎	q_1, q_3	none	q_1, q_3
	q_2	q_3	q_1
◎	q_3	none	q_3

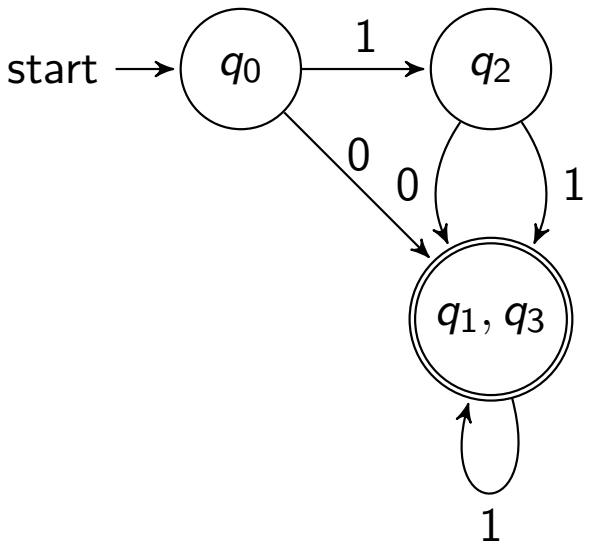


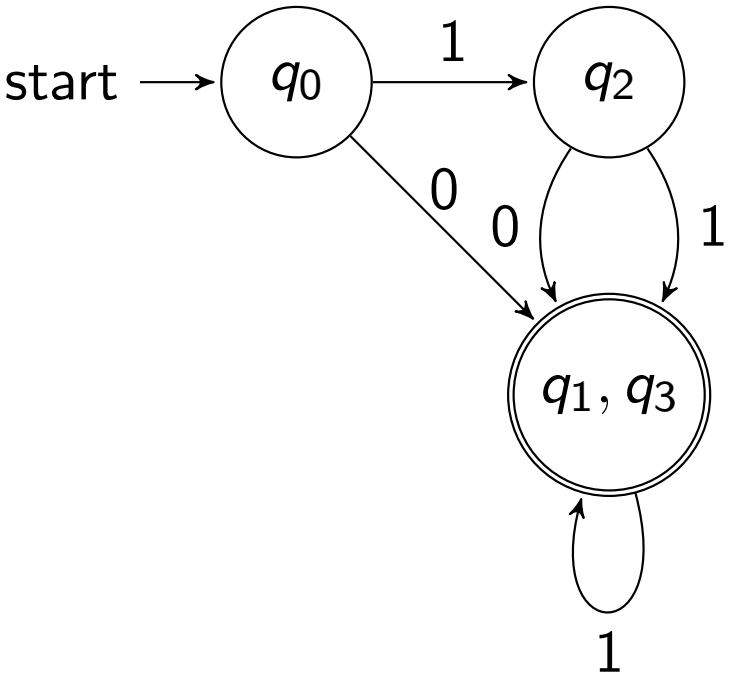
Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
◎	q_1, q_3	none	q_1, q_3
	q_2	q_3	q_1
◎	q_3	none	q_3
	q_1		



Type	New State	0	1
$\rightarrow$	q_0	q_1, q_3	q_2
◎	q_1, q_3	none	q_1, q_3
	q_2	q_3	q_1
◎	q_3	none	q_3
◎	q_1	none	q_1


start
q_0
1
q_2
1
q_1
0
0
1
q_1, q_3
q_3
1
1


start
q_0
1
q_2
0
0
1
q_1, q_3
1



This automaton accepts strings such as 0, 01, 011, 01111, 11, 111111, 10 and 101111.

It recognizes the language $L = (01^*) \mid (111^*) \mid (101^*)$

Exponential

The DFA can have exponentially many more states than the NFA.

Load the dfa into a browser (additional input: slide26)

Regular Languages – Further Examples

- Set of strings extending a fixed word;
- Set of strings lexicographic before a fixed word;
- Set of decimal numbers being a multiple of a fixed number p;
- Set of decimal numbers where the digit sum is at least 5;
- Set of decimal numbers where the digit sum is a multiple of a fixed number p;
- Set of decimal numbers containing exactly 8 times a 1 and three times a 2 in their representation;
- Any union or intersection or concatenation of some sets of the above form.

Here "decimal number" means natural number written with digits from "0" to "9".

Regular Languages

The languages recognized by finite state automata are regular languages.

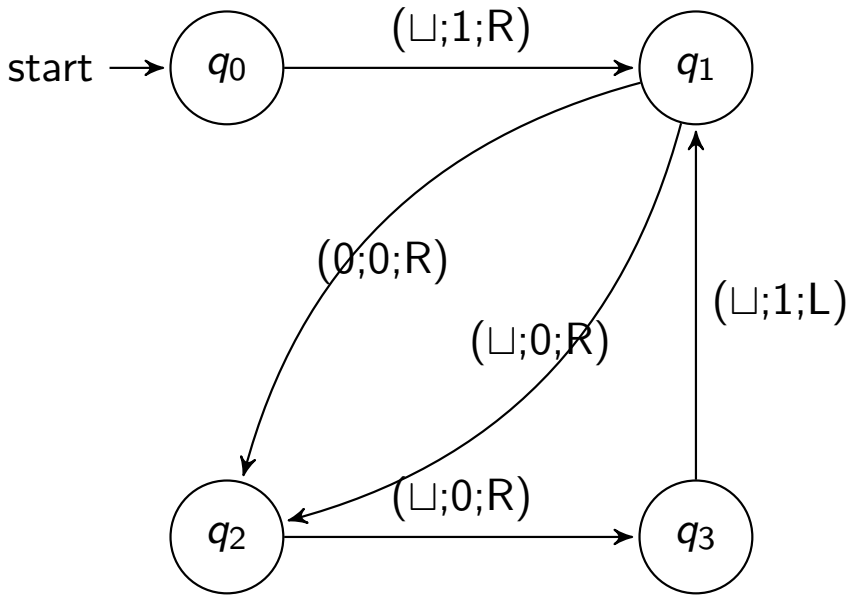
Regular Languages

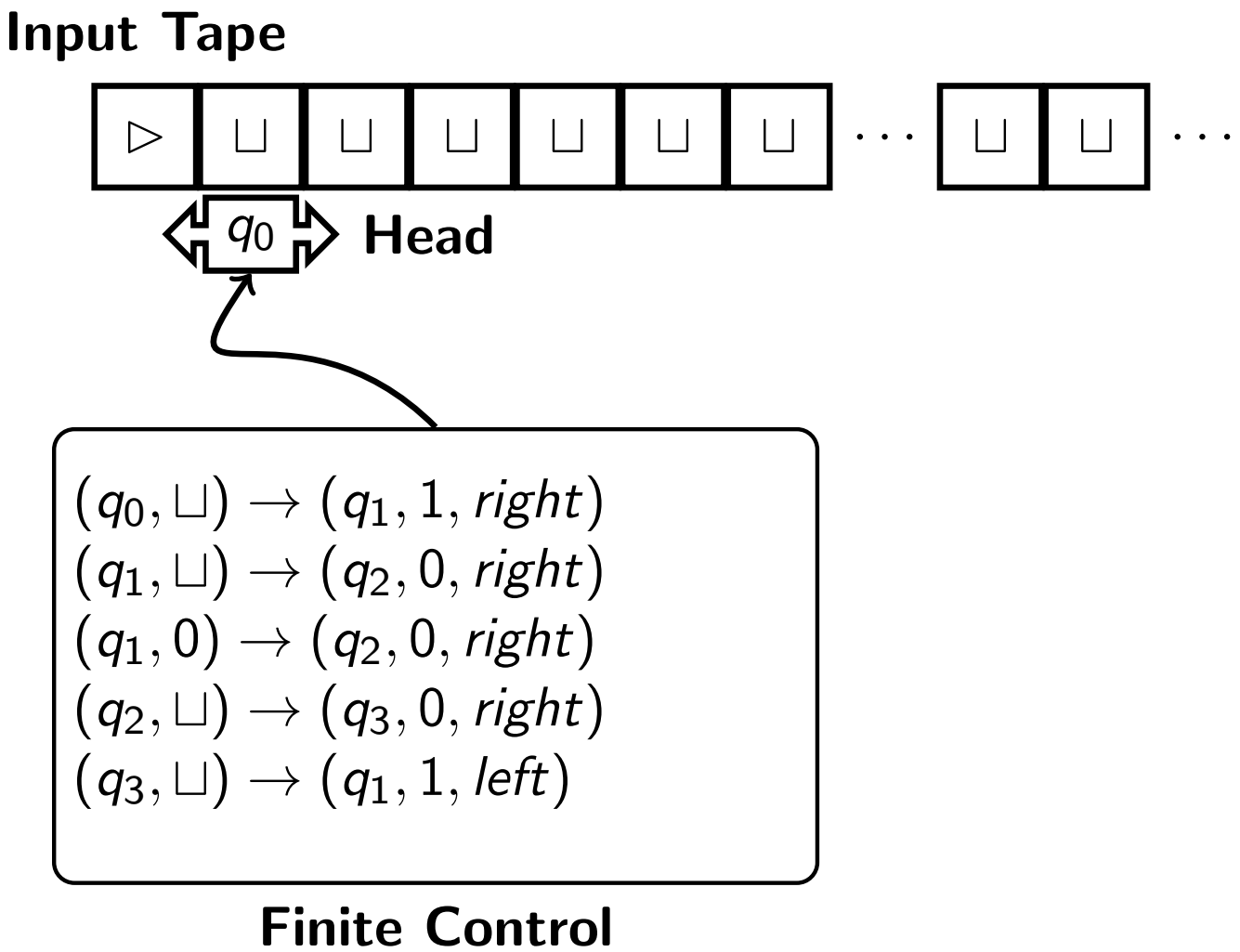
Regular languages over an alphabet Σ is the smallest set of languages that contains the empty ($\emptyset$) and the singleton languages, and is closed under union, concatenation and Kleene star.

- Singleton: `a`
- Union: `a | b`
- Concatenation: `ab`
- Kleene star: `a*`

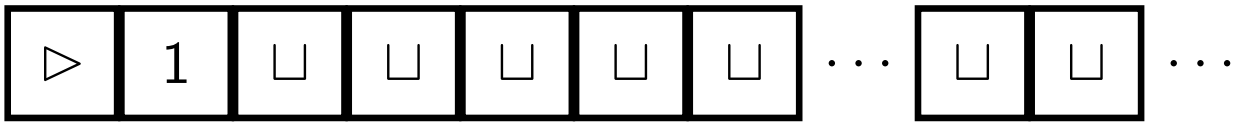
Turing Machine

A Turing machine is a sextuple $(Q, \Sigma, b, \delta, q_0, F)$ where Q is set of states, Σ is a finite alphabet or set of symbols, b is a blank symbol, δ is a transition function $(Q \times \Sigma \to Q \times \Sigma \times \{Right, Left\})$, q_0 is the initial state and F is a set of accepting states.




Input Tape
▷
⊔
⊔
⊔
⊔
⊔
⊔
⊔
⊔
q_0
Head
$(q_0, \sqcup) \rightarrow (q_1, 1, right)$
$(q_1, \sqcup) \rightarrow (q_2, 0, right)$
$(q_1, 0) \rightarrow (q_2, 0, right)$
$(q_2, \sqcup) \rightarrow (q_3, 0, right)$
$(q_3, \sqcup) \rightarrow (q_1, 1, left)$
Finite Control

Input Tape

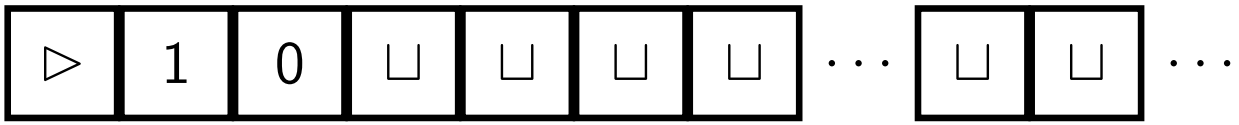


Head

$(\mathbf{q_0}, \sqcup) \rightarrow (\mathbf{q_1}, \mathbf{1}, \mathbf{right})$
$(q_1, \sqcup) \rightarrow (q_2, 0, right)$
$(q_1, 0) \rightarrow (q_2, 0, right)$
$(q_2, \sqcup) \rightarrow (q_3, 0, right)$
$(q_3, \sqcup) \rightarrow (q_1, 1, left)$

Finite Control

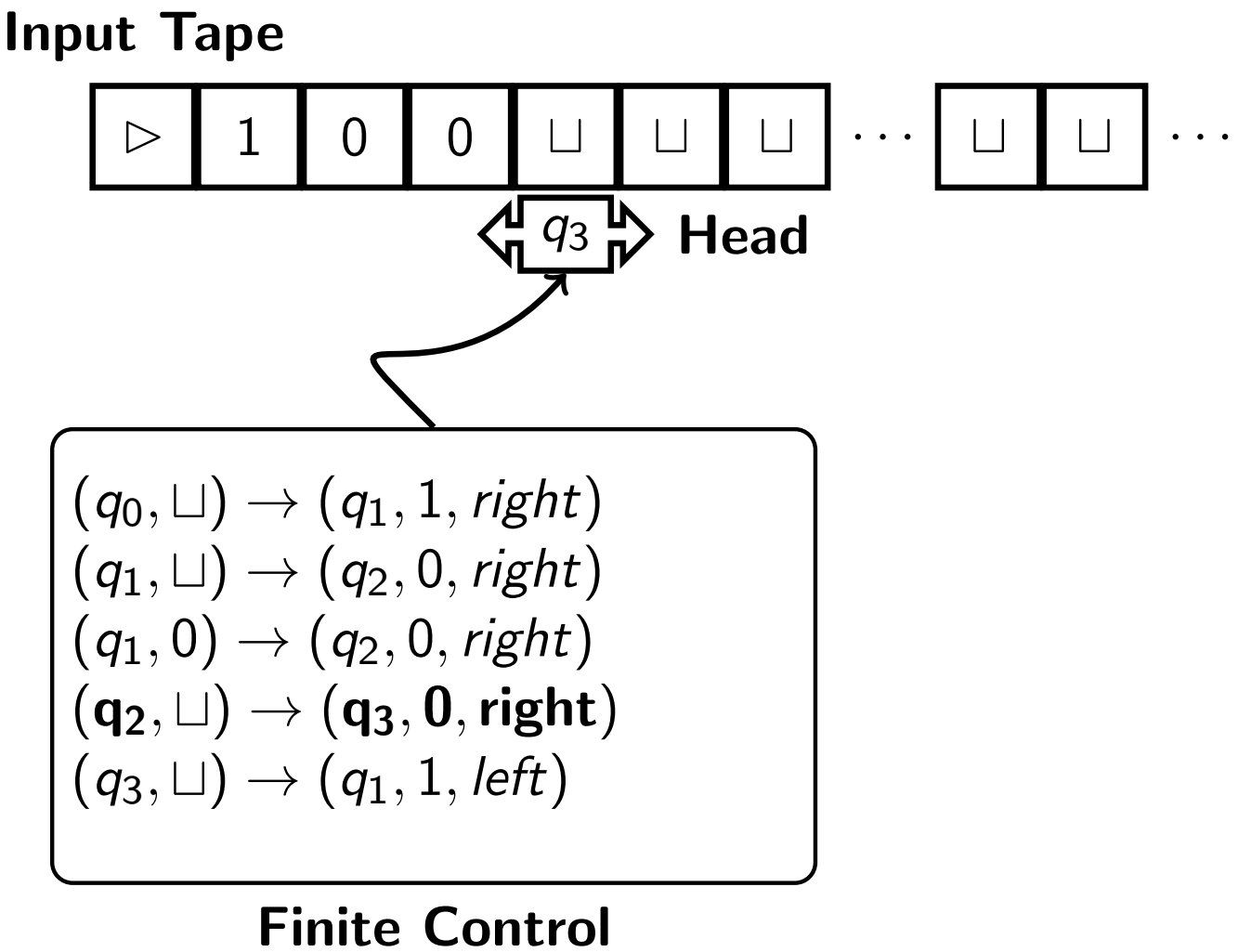
Input Tape



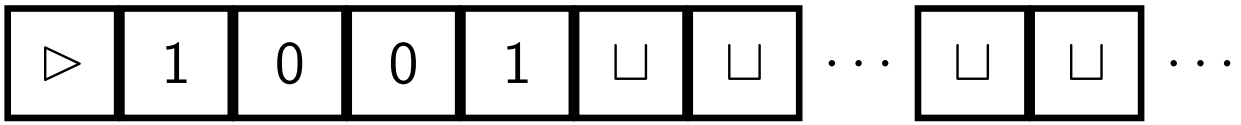
Head

$(q_0, \sqcup) \rightarrow (q_1, 1, \mathit{right})$
$(\mathbf{q_1}, \sqcup) \rightarrow (\mathbf{q_2}, \mathbf{0}, \mathbf{right})$
$(q_1, 0) \rightarrow (q_2, 0, \mathit{right})$
$(q_2, \sqcup) \rightarrow (q_3, 0, \mathit{right})$
$(q_3, \sqcup) \rightarrow (q_1, 1, \mathit{left})$

Finite Control


Input Tape
▷
1
0
0
⊔
⊔
⊔
⊔
⊔
q_3
Head
$(q_0, \sqcup) \to (q_1, 1, right)$
$(q_1, \sqcup) \to (q_2, 0, right)$
$(q_1, 0) \to (q_2, 0, right)$
$(\mathbf{q_2}, \sqcup) \to (\mathbf{q_3}, \mathbf{0}, \mathbf{right})$
$(q_3, \sqcup) \to (q_1, 1, left)$
Finite Control

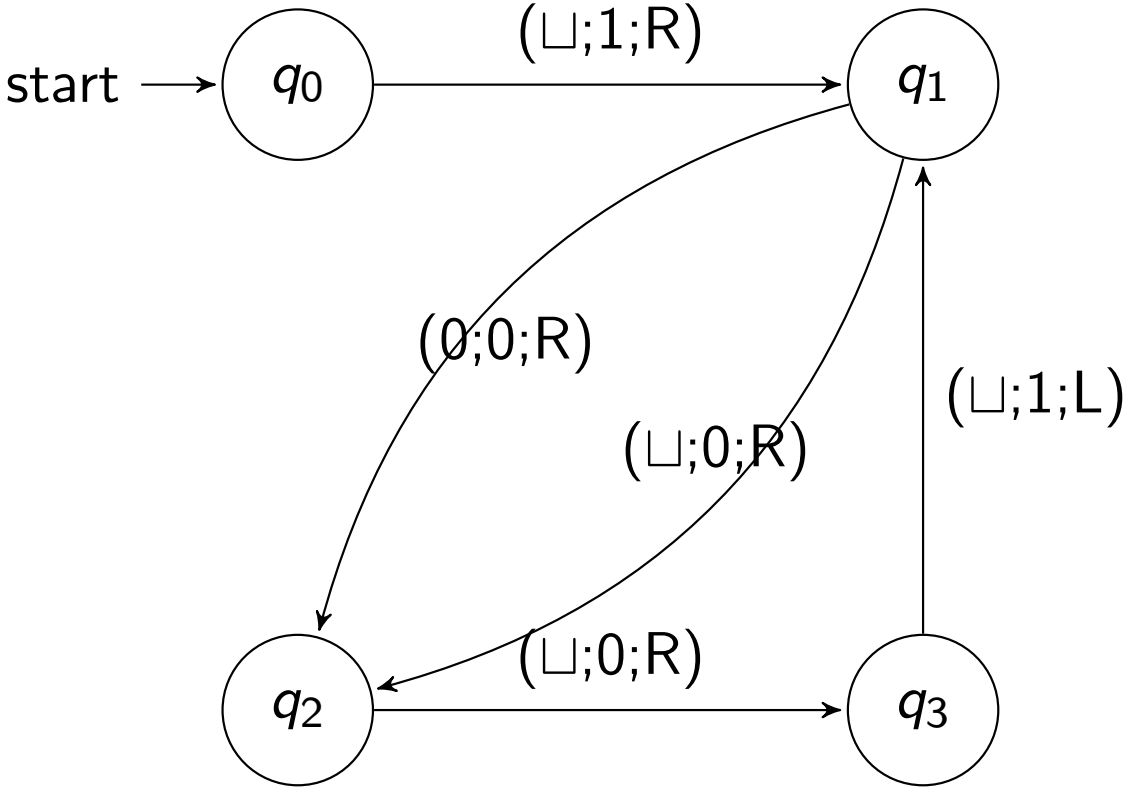
Input Tape



Head

$(q_0, \sqcup) \rightarrow (q_1, 1, right)$
$(q_1, \sqcup) \rightarrow (q_2, 0, right)$
$(q_1, 0) \rightarrow (q_2, 0, right)$
$(q_2, \sqcup) \rightarrow (q_3, 0, right)$
$(\mathbf{q_3}, \sqcup) \rightarrow (\mathbf{q_1}, \mathbf{1}, \mathbf{left})$

Finite Control



Let us try it with JFLAP

Recursively Enumerable Languages

The languages recognized by Turing machine are recursively enumerable languages.

Universal Turing Machine

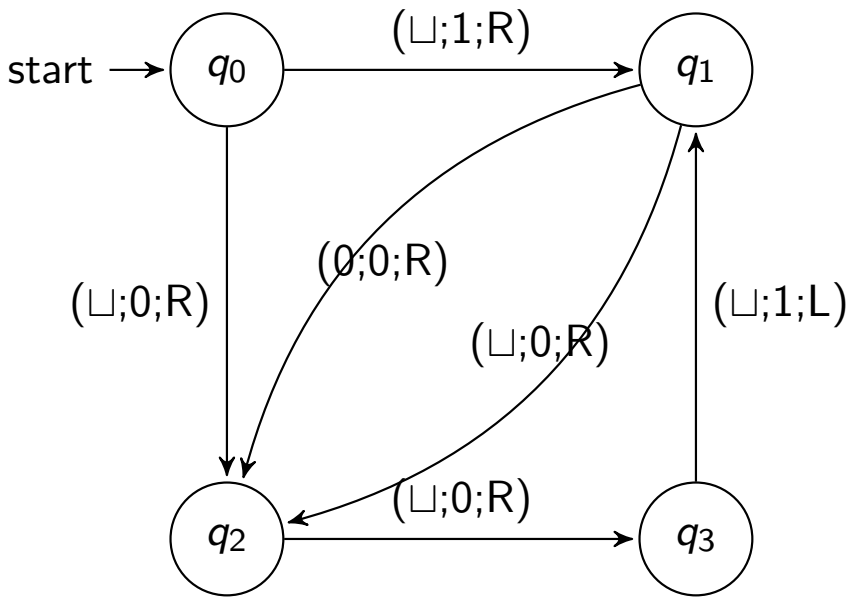
A Turing machine can simulate another Turing machine.
Both the program and the input are input of this universal Turing machine.

Turing Machine = Algorithm?

There is much more to algorithms than to Turing machines.
For instance we can consider two different algorithms for Euclidean algorithm (GCD by subtraction versus division).

Non Deterministic Turing Machine

A Turing machine is non deterministic if, instead of a transition function, it uses a transition relation.



Push-down Automaton

A push-down automaton uses a stack.

Context Free Languages

The languages recognized by push-down automata are context free languages.

Typical Example

$$\{0^n 1^n \mid n \in \mathbb{N} \wedge n > 1\}$$

Linear Bounded Automaton

A linear bounded automaton is a Turing machine restricted to work between two bounds.

Context Sensitive Languages

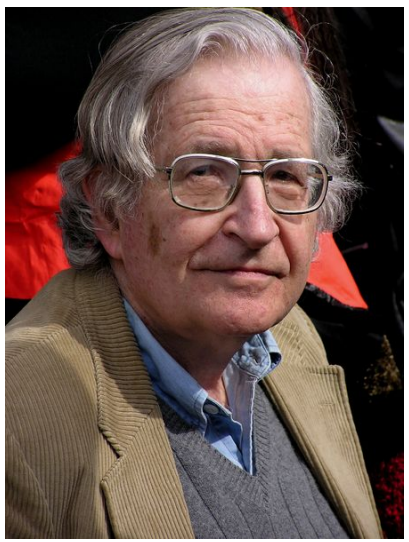
The languages recognized by Turing machine are context sensitive languages.

Typical Example

$$\{0^n 1^n 2^n \mid n \in \mathbb{N} \wedge n > 1\}$$

Chomsky Hierarchy

Grammar	Language	Machine
Type-0	Recursively enumerable	Turing machine
Type-1	Context-sensitive	Linear-bounded Turing machine
Type-2	Context-free	Pushdown automaton
Type-3	Regular	Finite state automaton



Regular Expressions

We describe here the regular expressions of `java.util.regex.Pattern`. They are used in JavaScript and in JEdit (for search and replace), for instance.

Character Classes

- a character matches itself.
- `[abc]` matches any character in the set of `a`, `b` and `c`.
- `[^abc]` matches any character not in the set of `a`, `b` and `c`.
- `[a-z]` matches any character in the range `a` to `z`, inclusive.

Concatenation and Branching

Let `R` and `S` be regular expressions.

- `RS` matches whatever the expression `R` concatenated with `S` matches; for example `[a-z][aeiou]` matches a letter followed by a vowel.
- `R|S` matches whatever the expression `R` or the expression `S` matches; for example `aa|ee|ii|oo|uu` matches all identical double vowels.

Repetitions

- * matches the empty string or any number of repetitions of the preceding expression.
- + matches one or more repetitions of the preceding expression.
- ? matches the preceding expression or the null string.
- {m} matches exactly m repetitions of the one-character expression.

Stingy (Minimal) Matching

If a repeating operator (above) is immediately followed by a ?, the repeating operator will stop at the smallest number of repetitions that can complete the rest of the match.

One-Character Operators

- `.` matches any single character.
- `\d` matches any decimal digit.
- `\D` matches any non-digit.
- `\n` matches the newline character.
- `\s` matches any whitespace character.
- `\S` matches any non-whitespace character.
- `\t` matches a horizontal tab character.
- `\w` matches any word (alphanumeric) character.
- `\W` matches any non-word (alphanumeric) character.
- `\\` matches the backslash (“\”) character.

Positional Operators

- ^ matches at the beginning of a line.
- $ matches at the end of a line.
- \B matches at a non-word break.
- \b matches at a word boundary.

Irregular Subexpressions

`(abc)` and `(?:...)` are used for grouping.

- `(abc)` matches whatever the expression abc would match, and saves it as a subexpression.
- `$n` where $0 < \texttt{n} < 10$, substituted with the text matched by the nth subexpression.

Examples

- "Gaul" The substring "Gaul"
- "a | b" "a" or "b"
- "e+" One or more "e"
- "(\s[a-z]{3}\s)" Three lowercase letters between spaces
- "a[a-z]*e" An "a" followed by letters followed by an "e"
- "a[z-z]*?e" An "a" followed by letters followed by an "e"

```
1 var patt=new RegExp("(a | b)b*","mg");
2 var patt=/"(a | b)b*"/"mg";
3 ...
4 result1=patt.test(text);
5 result2=text.replace(patt, "<B>$1</B>");
```

Load this HTML file into a browser

Alan Turing's Legacy

" Turing's importance extends far beyond Turing Machines. His work deciphering secret codes drastically shortened World War II and pioneered early computer technology. He was also an early innovator in the field of artificial intelligence, and came up with a way to test if computers could think – now known as the Turing Test. Besides this abstract work, he was down to earth; he designed and built real machines, even making his own relays and wiring up circuits. This combination of pure math and computing machines was the foundation of computer science.
As a human being, Turing was also extraordinary and original. He was eccentric, witty, charming and loyal. He was a marathon runner with world class time. He was also openly gay in a time and place where this was not accepted. While in many ways the world was not ready for Alan Turing, and lost him too soon, his legacy lives on in modern computing."

Attribution