Automata Theory (2A)

Young Won Lim
6/9/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Automata

The word **automata** (the plural of **automaton**) comes from the Greek word **αὐτόματα**, which means "**self-acting**".

https://en.wikipedia.org/wiki/Automata_theory

Automata Theory

Automata theory is the study of
abstract machines and **automata**,
as well as the computational problems
that can be solved using them.

It is a theory in theoretical computer science
and discrete mathematics.

https://en.wikipedia.org/wiki/Automata_theory

Automata Informal description (1) – Inputs

An **automaton** runs
when it is given some sequence of **inputs**
in discrete (individual) time steps or steps.

word	1000100
alphabet	{0,1}

An automaton processes one input
picked from a set of **symbols** or **letters**,
which is called an **alphabet**.

The symbols received by the automaton
as **input** at any step are
a finite sequence of **symbols** called **words**.

https://en.wikipedia.org/wiki/Automata_theory

Automata informal description (2) – States

An automaton has a finite set of **states**.

At each moment during a run of the automaton,
the automaton is in one of its **states**.

When the automaton receives new **input**
it moves to another **state** (or **transitions**)
based on a **function** that takes
the **current state** and **input symbol** as parameters.

This function is called the **transition function**.

https://en.wikipedia.org/wiki/Automata_theory

Automata informal description (3) – Stop

The **automaton**
reads the symbols of the **input word**
one after another and
transitions from **state** to **state**
according to the **transition function**
until the **word** is read completely.

word 1000100

Once the input **word** has been read,
the automaton is said to have stopped.

The state at which the automaton **stops**
is called the **final state**.

https://en.wikipedia.org/wiki/Automata_theory

Automata informal description (4) – Accept / Reject

Depending on the **final state**,
it's said that the automaton
either **accepts** or **rejects** an **input word**.

word 1000100

There is a **subset** of **states** of the automaton,
which is defined as the set of **accepting states**.

If the **final state** is an **accepting state**,
then the automaton **accepts** the **word**.

Otherwise, the **word** is **rejected**.

https://en.wikipedia.org/wiki/Automata_theory

Automata informal description (5) – Language

The set of **all the words accepted**
by an automaton is called the
"**language** of that automaton".

Any **subset** of the **language** of an automaton is
a language **recognized** by that automaton.

https://en.wikipedia.org/wiki/Automata_theory

Automata informal description (6) – Decision on inputs

an **automaton** is a mathematical object
that takes a word as **input**
and **decides** whether to **accept** it or **reject** it.

Since all computational problems are
reducible into the **accept/reject question** on **inputs**,
(all problem instances can be represented
in a finite length of symbols),
automata theory plays a crucial role
in computational theory.

https://en.wikipedia.org/wiki/Automata_theory

Automata Applications

Automata theory is closely related to **formal language** theory.

An automaton is a **finite representation** of a **formal language** that may be an **infinite set**.

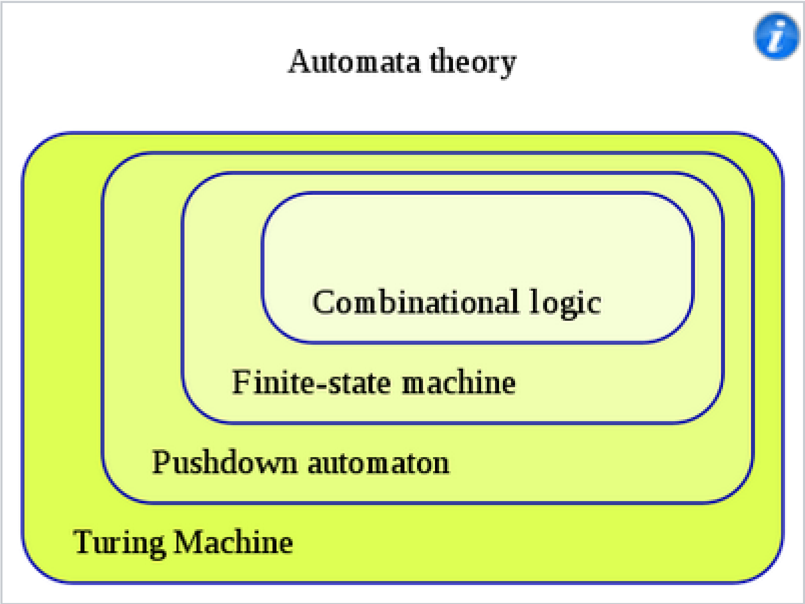
Automata are often classified by the **class** of **formal languages** they can **recognize**, typically illustrated by the **Chomsky hierarchy**, which describes the relations between various **languages** and kinds of formalized **logic**.

Automata play a major role in
- theory of computation,
- compiler construction,
- artificial intelligence,
- parsing and
- formal verification.

https://en.wikipedia.org/wiki/Automata_theory

Class of Automata

- Combinational Logic
- Finite State Automaton (FSA)
- Pushdown Automaton (PDA)
- Turing Machine



Classes of automata
(Clicking on each layer will take you to an article on that subject)

https://en.wikipedia.org/wiki/Automata_theory

Class of Automata

Finite State Automaton (FSA)	Regular Language
Pushdown Automaton (PDA)	Context-Free Language
Turing Machine	Recursively Enumerable Language
Automaton	Formal Languages

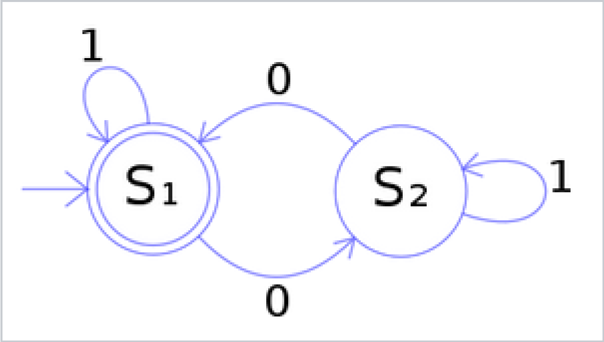
https://en.wikipedia.org/wiki/Automata_theory

Finite State Automaton

The figure at right illustrates a **finite-state machine**, which belongs to a well-known type of **automaton**.

This automaton consists of **states** (represented in the figure by circles) and **transitions** (represented by arrows).

As the automaton sees a **symbol** of **input**, it makes a **transition** (or jump) to another **state**, according to its **transition function**, which takes the **current state** and the recent **symbol** as its **inputs**.



The study of the mathematical properties of such automata is automata theory. The picture is a visualization of an automaton that recognizes strings containing an even number of *0*s. The automaton starts in state *S1*, and transitions to the non-accepting state *S2* upon reading the symbol *0*. Reading another *0* causes the automaton to transition back to the accepting state *S1*. In both states the symbol *1* is ignored by making a transition to the current state.

https://en.wikipedia.org/wiki/Automata_theory

Pushdown Automaton (1)

a type of automaton that employs a **stack**.

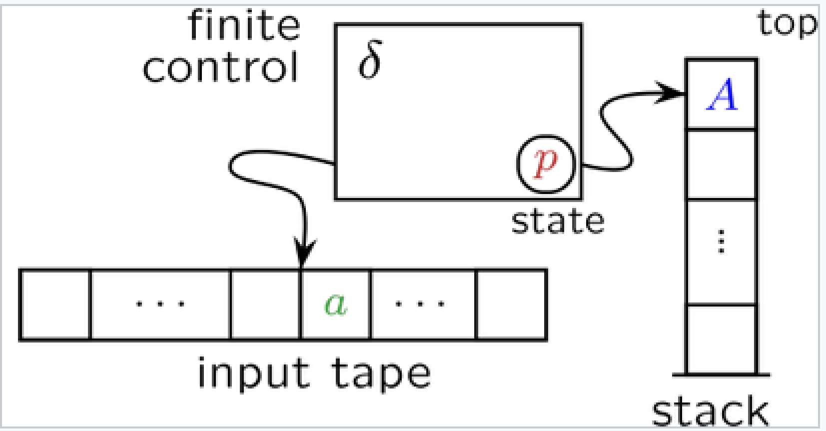
The term "pushdown" refers to the fact that
the stack can be regarded as being "pushed down"
like a tray dispenser at a cafeteria,
since the operations never work on elements
other than the **top element**.

A **stack automaton**, by contrast, does allow access to and operations on deeper elements.

https://en.wikipedia.org/wiki/Automata_theory

Pushdown Automaton (2)

a pushdown automaton (PDA) is
a type of automaton that employs a stack



A diagram of a pushdown automaton

https://en.wikipedia.org/wiki/Pushdown_automaton

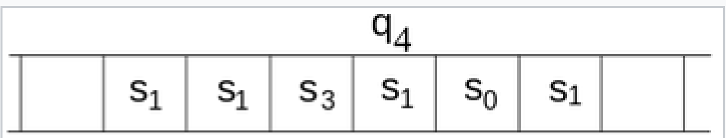
Turing Machine (1)

A **Turing machine** is
a mathematical **model** of computation
that defines an **abstract machine**,
which manipulates **symbols** on a strip of **tape**
according to a **table** of **rules**.

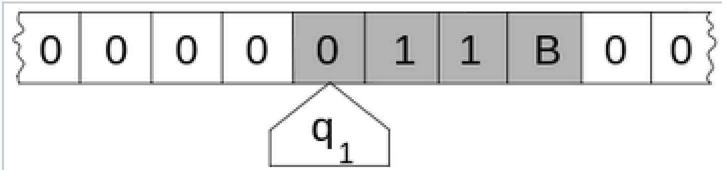
Despite the model's simplicity,
given any computer **algorithm**,
a **Turing machine** capable of **simulating**
that algorithm's logic can be constructed.

https://en.wikipedia.org/wiki/Automata_theory

Turing Machine (2)



The head is always over a particular square of the tape; only a finite stretch of squares is shown. The instruction to be performed (q_4) is shown over the scanned square. (Drawing after Kleene (1952) p. 375.)



Here, the internal state (q_1) is shown inside the head, and the illustration describes the tape as being infinite and pre-filled with "0", the symbol serving as blank. The system's full state (its *complete configuration*) consists of the internal state, any non-blank symbols on the tape (in this illustration "11B"), and the position of the head relative to those symbols including blanks, i.e. "011B". (Drawing after Minsky (1967) p. 121.)

https://en.wikipedia.org/wiki/Turing_machine

1. Definition of Finite State Automata

A deterministic finite automaton is represented formally by a 5-tuple <**Q**, **Σ**, **δ**, q_0, F>, where:

- **Q** is a finite set of **states**.
- **Σ** is a finite set of **symbols**, called the **alphabet** of the automaton.
- **δ** is the **transition function**, that is, **δ: Q × Σ → Q**.
- q_0 is the **start state**, that is, the state of the automaton before any input has been processed, where $q_0 \in$ **Q**.
- **F** is a set of **states** of **Q** (i.e. **F**⊆**Q**) called **accept states**.

https://en.wikipedia.org/wiki/Automata_theory

2. Deterministic Pushdown Automaton

A **PDA** is formally defined as a 7-tuple:

$M = (\mathbf{Q}, \mathbf{\Sigma}, \mathbf{\Gamma}, \boldsymbol{\delta}, \mathbf{q_0}, \mathbf{Z}, \mathbf{F})$ where

- **Q** is a finite set of **states**
- **Σ** is a finite set which is called the **input alphabet**
- **Γ** is a finite set which is called the **stack alphabet**
- **δ** is a finite subset of $\mathbf{Q} \times (\mathbf{\Sigma} \cup \{\varepsilon\}) \times \mathbf{\Gamma} \times \mathbf{Q} \times \mathbf{\Gamma}^*$, the **transition relation**.
- $\mathbf{q_0} \in \mathbf{Q}$ is the **start state**
- $\mathbf{Z} \in \mathbf{\Gamma}$ is the **initial stack symbol**
- $\mathbf{F} \subseteq \mathbf{Q}$ is the set of **accepting states**

https://en.wikipedia.org/wiki/Pushdown_automaton

3. Turing Machine

Turing machine as a 7-tuple $M = (\mathbf{Q}, \Gamma, b, \Sigma, \delta, q_0, F)$ where   \ set minus

- **Q** is a finite, non-empty set of **states**;
- **Γ** is a finite, non-empty set of **tape alphabet symbols**;
- **b** $\in$ **Γ** is the **blank symbol**
- $\Sigma \subseteq \Gamma \setminus \{ b \}$ is the set of **input symbols** in the initial tape contents;
- $q_0 \in Q$ is the **initial state**;
- $F \subseteq Q$ is the set of **final states** or **accepting states**.
- $\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{ L , R \}$ is **transition function**, where **L** is **left shift**, **R** is **right shift**.

The initial tape contents is said to be accepted by M if it eventually halts in a state from F .

https://en.wikipedia.org/wiki/Turing_machine

FSA, PDA, Turing Machine

Deterministic Finite State Automaton	(**Q**, **Σ**, **δ**, q_0, F)
Deterministic Pushdown Automaton	(**Q**, **Σ**, **Γ**, **δ**, q_0, **Z**, F)
Turing machine	(**Q**, **Γ**, **b**, **Σ**, **δ**, q_0, F)

Σ is the input alphabet (a finite non-empty set of symbols).
Q is a finite, non-empty set of states.
δ is the state-transition function: **δ : S × Σ → S**
s_0 is the initial state, an element of S.
F is the set of final states, a (possibly empty) subset of S.
Γ is a finite set which is called the **stack alphabet**
Z ∈ **Γ** is the **initial stack symbol**
Γ is a finite, non-empty set of **tape alphabet symbols**;
b ∈ **Γ** is the **blank symbol**

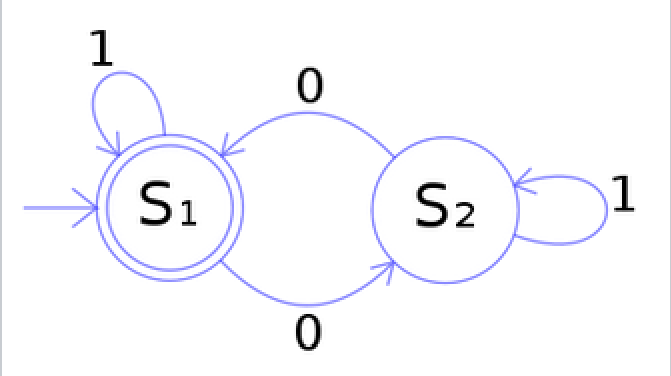
Deterministic Finite State Automaton (FSA)

Deterministic Finite Automaton Example (1)

The following example is of a DFA M, with a binary alphabet, which requires that the input contains an even number of 0s.

M = (Q, Σ, δ, q0, F) where
- Q = {S1, S2},
- Σ = {0, 1},
- q0 = S1,
- F = {S1}, and
- δ is defined by the following state transition table:

	0	**1**
S_1	S_2	S_1
S_2	S_1	S_2



The state diagram for *M*

(Σ, S, s_0, δ, F)

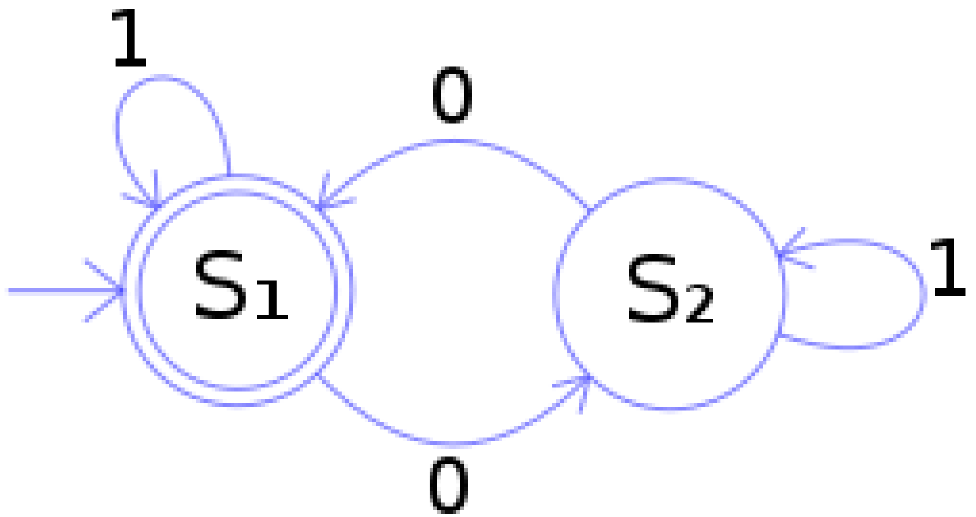
(Q, Σ, δ, q_0, F)

https://en.wikipedia.org/wiki/Deterministic_finite_automaton

Deterministic Finite Automaton Example (2)

State Transition Table

Input / State	1	0
S_1	S_1	S_2
S_2	S_2	S_1



$$\{S_1, S_2\} \times 0,1 \rightarrow \{S_1, S_2\}$$

https://en.wikipedia.org/wiki/State_transition_table

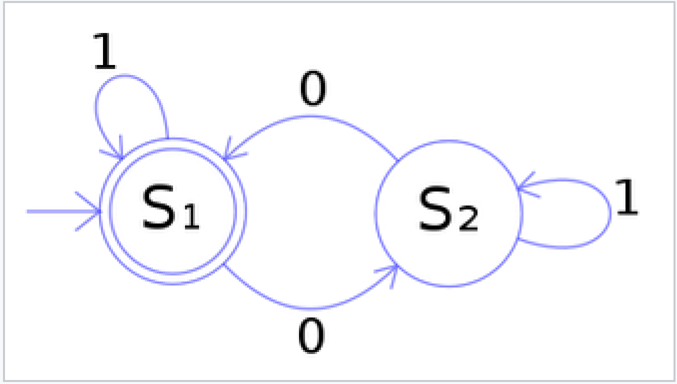
Deterministic Finite Automaton Example (3)

The **state S1** represents that there has been an even number of 0s in the input so far, while **S2** signifies an odd number.

A **1** in the input does not change the state of the automaton.

When the input ends, the state will show whether the input contained an even number of **0**s or not.

If the input did contain an even number of **0**s, M will finish in **state S1**, an accepting state, so the input string will be accepted.



The state diagram for *M*

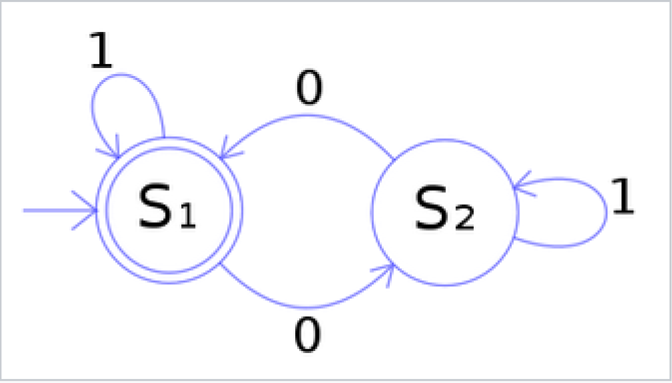
https://en.wikipedia.org/wiki/Deterministic_finite_automaton

Deterministic Finite Automaton Example (4)

The language recognized by M is
the regular language given
by the regular expression
((1*) 0 (1*) 0 (1*))*,

where "*" is the Kleene star,
e.g., **1*** denotes any number
(possibly zero) of consecutive **ones**.

zero or more



The state diagram for *M*

https://en.wikipedia.org/wiki/Deterministic_finite_automaton

References

[1] http://en.wikipedia.org/
[2]