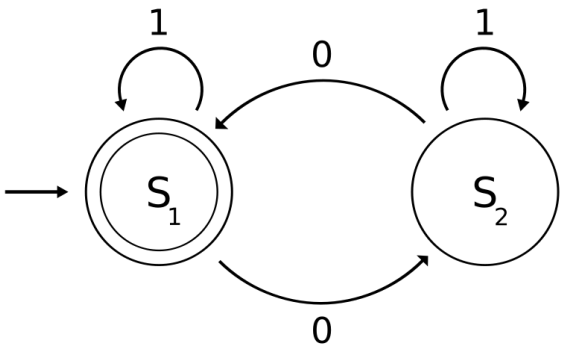
Automata theory

In theoretical computer science, **automata theory** is the study of abstract machines and the computational problems that can be solved using these abstract machines. These abstract machines are called automata.

As suggested in figure, an automaton takes a symbol as input and "*jumps*" or *transitions*, from one state to another according to a transition function (which is expressed as arrows). This transition function tells the automaton which state to go to next given a *current state* and a *current symbol*. Automata are similar to finite state machines (FSM).



An example of automata and study of mathematical properties of such automata is automata theory

Automata theory is closely related to formal language theory as the automata are often classified by the class of formal languages they are able to recognize. An automaton is considered to be a finite representation of a formal language that may be an infinite set.

Automata play a major role in compiler design and parsing.

Automata

Following is an introductory definition of one type of automata, which attempts to help one grasp the essential concepts involved in automata theory.

Informal description

An automaton is supposed to *run* on some given sequence or string of *inputs* in discrete time steps. At each time step, an automaton gets one input that is picked up from a set of *symbols* or *letters*, which is called an *alphabet*. At any time, the symbols so far fed to the automaton as input form a finite sequence of symbols, which is called a *word*. An automaton contains a finite set of states. At each instance in time of some run, automaton is *in* one of its states. At each time step when the automaton reads a symbol, it *jumps* or *transits* to next state depending on its current state and on the symbol currently read. This function in terms of the current state and input symbol is called *transition function*. The automaton *reads* the input word one symbol after another in the sequence and transits from state to state according to the transition function, until the word is read completely. Once the input word has been read, the automaton is said to have been *stopped* and the state at which automaton has stopped is called *final state*. Depending on the final state, it's said that the automaton either *accepts* or *rejects* an input word. 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*. The set of all the words accepted by an automaton is called the *language recognized by the automaton*.

Formal definition

Automaton

An **automaton** is represented formally by the 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, $\delta: Q \times \Sigma \to Q$.
- q_0 is the *start state*, that is, the state which the automaton is *in* when no input has been processed yet, where $q_0 \in Q$.
- F is a set of states of Q (i.e. $F \subseteq Q$) called **accept states**.

Input word

An automaton reads a finite string of symbols $a_1,a_2,....., a_n$, where $a_i \in \Sigma$, which is called a *input word*. Set of all words is denoted by Σ*.

Run

A *run* of the automaton on an input word $w = a_1,a_2,....., a_n \in \Sigma^*$, is a sequence of states $q_0,q_1,q_2,....., q_n$, where $q_i \in Q$ such that q_0 is a start state and $q_i = \delta(q_{i-1},a_i)$ for $0 < i \leq n$. In words, at first the automaton is at the start state q_0 and then automaton reads symbols of the input word in sequence. When automaton reads symbol a_i then it jumps to state $q_i = \delta(q_{i-1},a_i)$. q_n said to be the *final state* of the run.

Accepting word

A word $w \in \Sigma^*$ is accepted by the automaton if $q_n \in F$.

Recognized language

An automaton can recognize a formal language. The recognized language $L \subset \Sigma^*$ by an automaton is the set of all the words that are accepted by the automaton.

Recognizable languages

The recognizable languages is the set of languages that are recognized by some automaton. For above definition of automata the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

Variations in definition of automata

Automata are defined to study useful machines under mathematical formalism. So, the definition of an automaton is open to variations according to the "real world machine", which we want to model using the automaton. People have studied many variations of automata. Above, the most standard variant is described, which is called deterministic finite automaton. The following are some popular variations in the definition of different components of automata.

Input

- *Finite input*: An automaton that accepts only finite sequence of words. The above introductory definition only accepts finite words.
- *Infinite input*: An automaton that accepts infinite words (ω-words). Such automata are called *ω-automata*.
- *Tree word input*: The input may be a *tree of symbols* instead of sequence of symbols. In this case after reading each symbol, the automaton *reads* all the successor symbols in the input tree. It is said that the automaton *makes one copy* of itself for each successor and each such copy starts running on one of the successor symbol from the state according to the transition relation of the automaton. Such an automaton is called tree automaton.

States

- *Finite states*: An automaton that contains only a finite number of states. The above introductory definition describes automata with finite numbers of states.

- *Infinite states*: An automaton that may not have a finite number of states, or even a countable number of states. For example, the quantum finite automaton or topological automaton has uncountable infinity of states.
- *Stack memory*: An automaton may also contain some extra memory in the form of a stack in which symbols can be pushed and popped. This kind of automaton is called a *pushdown automaton*

Transition function

- *Deterministic*: For a given current state and an input symbol, if an automaton can only jump to one and only one state then it is a *deterministic automaton*.
- *Nondeterministic*: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation. Notice that the term transition function is replaced by transition relation: The automaton *non-deterministically* decides to jump into one of the allowed choices. Such automaton are called *nondeterministic automaton*.
- *Alternation*: This idea is quite similar to tree automaton, but orthogonal. The automaton may run its *multiple copies* on the *same* next read symbol. Such automata are called *alternating automaton*. Acceptance condition must satisfy all runs of such *copies* to accept the input.

Acceptance condition

- *Acceptance of finite words*: Same as described in the informal definition above.
- *Acceptance of infinite words*: an *omega automaton* cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
- *Probabilistic acceptance*: An automaton need not strictly accept or reject an input. It may accept the input with some probability between zero and one. For example, quantum finite automaton, geometric automaton and *metric automaton* has probabilistic acceptance.

Different combinations of the above variations produce many variety of automaton.

Automata theory

Automata theory is a subject matter which studies properties of various types of automata. For example, following questions are studied about a given type of automata.

- Which class of formal languages is recognizable by some type of automata?(Recognizable languages)
- Is certain automata *closed* under union, intersection, or complementation of formal languages?(Closure properties)
- How much is a type of automata expressive in terms of recognizing class of formal languages? And, their relative expressive power?(Language Hierarchy)

Automata theory also studies if there exist any effective algorithm or not to solve problems similar to following list.

- Does an automaton accept any input word?(emptiness checking)
- Is it possible to transform a given non-deterministic automaton into deterministic automaton without changing the recognizing language?(Determinization)
- For a given formal language, what is the smallest automaton that recognize it?(Minimization).

Classes of automata

Following is an incomplete list of some types of automata.

Automata	Recognizable language
Deterministic finite automata (DFA)	regular languages
Nondeterministic finite automata (NFA)	regular languages
Nondeterministic finite automata with ε transitions (FND-ε or ε-NFA)	regular languages
Pushdown automata (PDA)	context-free languages
Linear bounded automata (LBA)	context-sensitive language
Turing machines	recursively enumerable languages
Timed automata	
Deterministic Büchi automata	omega limit languages
Nondeterministic Büchi automata	omega regular languages
Nondeterministic/Deterministic Rabin automata	omega regular languages
Nondeterministic/Deterministic Streett automata	omega regular languages
Nondeterministic/Deterministic parity automata	omega regular languages
Nondeterministic/Deterministic Muller automata	omega regular languages

Discrete, continuous, and hybrid automata

Normally automata theory describes the states of abstract machines but there are analog automata or continuous automata or hybrid discrete-continuous automata, using analog data, continuous time, or both.

Applications

Most implementations of automatons are used to make a software recognize a certain language, common examples are compilers and regular expression engines. There are most commocn applications of automata are: 1.Cellular automata 2.Weightef automata 3.image processing 4.Graphics animation 5.Compiling and NLS

References

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Pearson Education. ISBN 0-201-44124-1.
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X. Part One: Automata and Languages, chapters 1–2, pp.29–122. Section 4.1: Decidable Languages, pp.152–159. Section 5.1: Undecidable Problems from Language Theory, pp.172–183.
- James P. Schmeiser, David T. Barnard (1995). *Producing a top-down parse order with bottom-up parsing*. Elsevier North-Holland.

External links

- Visual Automata Simulator (http://www.cs.usfca.edu/~jbovet/vas.html), A tool for simulating, visualizing and transforming finite state automata and Turing Machines, by Jean Bovet
- JFLAP (http://www.jflap.org)
- dk.brics.automaton (http://www.brics.dk/automaton)
- libfa (http://www.augeas.net/libfa/index.html)
- Proyecto SEPa (in Spanish) (http://www.ucse.edu.ar/fma/sepa/)
- Exorciser (in German) (http://www.swisseduc.ch/informatik/exorciser/index.html)
- Automata Made it on Java, with the Sources so you can see (http://torturo.com/programas-hechos-en-java/)

Article Sources and Contributors

Automata theory *Source*: http://en.wikipedia.org/w/index.php?oldid=394537603 *Contributors*: Aaron Schulz, Ahmad.shahwan, Ahoerstemeier, Alansohn, Allan McInnes, Andrew Eisenberg, Ankog, Arslan asghar, Ashutosh y0078, AxelBoldt, Begoon, Bethnim, Bookandcoffee, Bzier, CRGreathouse, ChuckHG, Cominging, Crystallina, Dcoetzee, Deldotvee, DerHexer, Dudesleeper, Déjà Vu, Ehn, Ericszhao1, Eslip17, FoxLogick, Fudo, Gaius Cornelius, Giftlite, GlasGhost, Gonzonoir, HRV, Helix84, Hermel, Hjfreyer, Ilyaroz, IvanAndreevich, JK the unwise, Jackson, Jagged 85, Jcarroll, Jdoe87, Jeff G., Jeffrey Mall, Jibarraj, Jimbryho, Jitse Niesen, Jjovanw, Johndbritton, Jokes Free4Me, Joseph Solis in Australia, Jpbowen, Jpceayene, Jpvinall, Juniuswikia, KSlayer, KSmrq, Knverma, Konradek, Linas, Ling.Nut, Lobner, Magmi, Mangledorf, Maple.writes, Mark lee stillwell, MarkSweep, Marudubshinki, MathMartin, MatthewUND, Mct mht, Mean as custard, Mhhwang2002, Michael Devore, Msoos, Musiphil, Nunquam Dormio, Nuttycoconut, Oddity-, Oleg Alexandrov, Quoth, Qwertyus, Rjwilmsi, Ruud Koot, Ryanli (usurped), S h i v a (Visnu), SOMNIVM, Saber girl08, Saforrest, Salix alba, Schwarzbichler, Sgkay, Sharaar22, Sietse Snel, Silvonen, Snowolf, Spoon!, Stephen Shaw, The Thing That Should Not Be, Thunderboltz, TimBentley, Tobias Bergemann, Toolnut, Ubermonkey, Unbitwise, Valodzka, Vegpuff, Vento, Vojta, Wjmallard, Yosef Berman, Ze miguel, Zero sharp, عیس, 163 anonymous edits

Image Sources, Licenses and Contributors

Image:DFAexample.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DFAexample.svg *License*: Public Domain *Contributors*: User:Cepheus

License