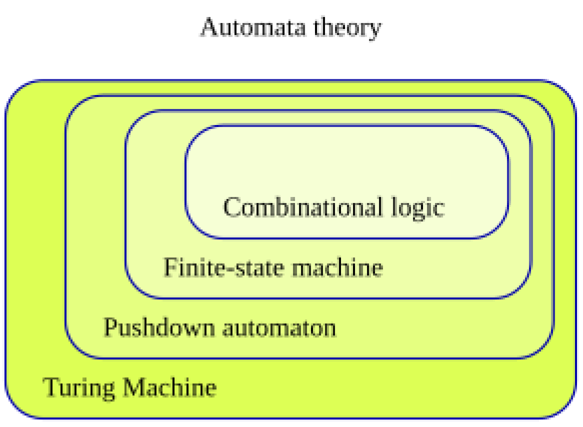
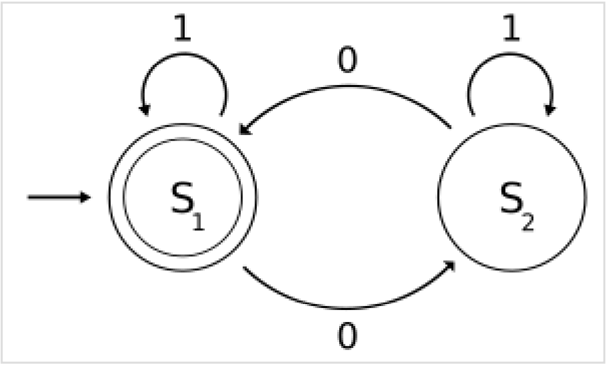
Wikipedia
The Free Encyclopedia

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 with close connections to mathematical logic. The word *automata* comes from the Greek word αὐτόματος, which means "self-acting, self-willed, self-moving". An automaton (automata in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically. An automaton with a finite number of states is called a finite automaton (FA) or finite-state machine (FSM). The figure on the right illustrates a finite-state machine, which is 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 previous state and current input symbol as its arguments.



Classes of automata



The automaton described by this state diagram starts in state S_1, and changes states following the arrows marked 0 or 1 according to the input symbols as they arrive. The double circle marks S_1 as an accepting state. Since all paths from S_1 to itself contain an even number of arrows marked 0, this automaton accepts strings containing even numbers of 0s.

Automata theory is closely related to formal language theory. In this context, automata are used as finite representations of formal languages that may be infinite. Automata are often classified by the class of formal languages they can recognize, as in the Chomsky hierarchy, which describes a nesting relationship between major classes of automata. Automata play a major role in the theory of computation, compiler construction, artificial intelligence, parsing and formal verification.

History

The theory of abstract automata was developed in the mid-20th century in connection with finite automata.[1] Automata theory was initially considered a branch of mathematical systems theory, studying the behavior of discrete-parameter systems. Early work in automata theory differed from previous work on systems by using abstract algebra to describe information systems rather than differential calculus to describe material systems.[2] The theory of the finite-state transducer was developed under different names by different research communities.[3] The earlier concept of Turing machine was also included in the discipline along with new forms of infinite-state automata, such as pushdown automata.

1956 saw the publication of *Automata Studies*, which collected work by scientists including Claude Shannon, W. Ross Ashby, John von Neumann, Marvin Minsky, Edward F. Moore, and Stephen Cole Kleene.[4] With the publication of this volume, "automata theory emerged as a relatively autonomous discipline".[5] The book included Kleene's description of the set of regular events, or regular languages, and a relatively stable measure of complexity in Turing machine programs by Shannon.[6] In the same year, Noam Chomsky described the Chomsky hierarchy, a correspondence between automata and formal grammars,[7] and Ross Ashby published *An Introduction to Cybernetics*, an accessible textbook explaining automata and information using basic set theory.

The study of linear bounded automata led to the Myhill–Nerode theorem,[8] which gives a necessary and sufficient condition for a formal language to be regular, and an exact count of the number of states in a minimal machine for the language. The pumping lemma for regular languages, also useful in regularity proofs, was proven in this period by Michael O. Rabin and Dana Scott, along with the computational equivalence of deterministic and nondeterministic finite automata.[9]

In the 1960s, a body of algebraic results known as "structure theory" or "algebraic decomposition theory" emerged, which dealt with the realization of sequential machines from smaller machines by interconnection.[10] While any finite automaton can be simulated using a universal gate set, this requires that the simulating circuit contain loops of arbitrary complexity. Structure theory deals with the "loop-free" realizability of machines.[5] The theory of computational complexity also took shape in the 1960s.[11][12] By the end of the decade, automata theory came to be seen as "the pure mathematics of computer science".[5]

Automata

What follows is a general definition of an automaton, which restricts a broader definition of a system to one viewed as acting in discrete time-steps, with its state behavior and outputs defined at each step by unchanging functions of only its state and input.[5]

Informal description

An automaton *runs* when it is given some sequence of *inputs* in discrete (individual) *time steps* (or just *steps*). An automaton processes one input picked from a set of *symbols* or *letters*, which is called an *input alphabet*. The symbols received by the automaton as input at any step are a sequence of symbols called *words*. An automaton has a 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 *transition function* that takes the previous state and current input symbol as parameters. At the same time, another function called the *output function* produces symbols from the *output alphabet*, also according to the previous state and current input symbol. The automaton reads the symbols of the input word and transitions between states until the word is read completely, if it is finite in length, at which point the automaton *halts*. A state at which the automaton halts is called the *final state*.

To investigate the possible state/input/output sequences in an automaton using formal language theory, a machine can be assigned a *starting state* and a set of *accepting states*. Then, depending on whether a run starting from the starting state ends in an accepting state, the automaton can be said to *accept* or *reject* an input sequence. The set of all the words accepted by an automaton is called the *language recognized by the automaton*. A familiar example of a machine recognizing a language is an electronic lock, which accepts or rejects attempts to enter the correct code.

Formal definition

Automaton

An automaton can be represented formally by a quintuple $M = \langle \Sigma, \Gamma, Q, \delta, \lambda \rangle$, where:

- Σ is a finite set of *symbols*, called the *input alphabet* of the automaton,
- Γ is another finite set of symbols, called the *output alphabet* of the automaton,
- Q is a set of *states*,
- δ is the *next-state function* or *transition function* $\delta : Q \times \Sigma \to Q$ mapping state-input pairs to successor states,
- λ is the *next-output function* $\lambda : Q \times \Sigma \to \Gamma$ mapping state-input pairs to outputs.

If Q is finite, then M is a finite automaton.[5]

Input word

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

Run

A sequence of states $q_0, q_1, \ldots, q_n$, where $q_i \in Q$ such that $q_i = \delta(q_{i-1}, a_i)$ for $0 < i \leq n$, is a *run* of the automaton on an input $a_1 a_2 ... a_n \in \Sigma^*$ starting from state q_0. In other words, at first the automaton is at the start state q_0, and receives input a_1. For a_1 and every following a_i in the input string, the automaton picks the next state q_i according to the transition function $\delta(q_{i-1}, a_i)$, until the last symbol a_n has been read, leaving the machine in the *final state* of the run, q_n. Similarly, at each step, the automaton emits an output symbol according to the output function $\lambda(q_{i-1}, a_i)$.

The transition function δ is extended inductively into $\overline{\delta} : Q \times \Sigma^* \to Q$ to describe the machine's behavior when fed whole input words. For the empty string ε, $\overline{\delta}(q, \varepsilon) = q$ for all states q, and for strings wa where a is the last symbol and w is the (possibly empty) rest of the string, $\overline{\delta}(q, wa) = \delta(\overline{\delta}(q, w), a)$.[10] The output function λ may be extended similarly into $\overline{\lambda}(q, w)$, which gives the complete output of the machine when run on word w from state q.

Acceptor

In order to study an automaton with the theory of formal languages, an automaton may be considered as an *acceptor*, replacing the output alphabet and function Γ and λ with

- $q_0 \in Q$, a designated *start state*, and
- F, a set of states of Q (i.e. $F \subseteq Q$) called *accept states*.

This allows the following to be defined:

Accepting word

A word $w = a_1 a_2 \ldots a_n \in \Sigma^*$ is an *accepting word* for the automaton if $\bar{\delta}(q_0, w) \in F$, that is, if after consuming the whole string w the machine is in an accept state.

Recognized language

The language $L \subseteq \Sigma^*$ *recognized* by an automaton is the set of all the words that are accepted by the automaton, $L = \{w \in \Sigma^* \mid \bar{\delta}(q_0, w) \in F\}$.[13]

Recognizable languages

The recognizable languages are the set of languages that are recognized by some automaton. For *finite automata* the recognizable languages are regular languages. For different types of automata, the recognizable languages are different.

Variant definitions 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" that we want to model using the automaton. People have studied many variations of automata. The following are some popular variations in the definition of different components of automata.

Input

- *Finite input*: An automaton that accepts only finite sequences of symbols. The above introductory definition only encompasses finite words.
- *Infinite input*: An automaton that accepts infinite words (ω-words). Such automata are called *ω-automata*.
- *Tree 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 symbols from the state according to the transition relation of the automaton. Such an automaton is called a tree automaton.
- *Infinite tree input* : The two extensions above can be combined, so the automaton reads a tree structure with (in)finite branches. Such an automaton is called an infinite tree automaton.

States

- *Single state*: An automaton with one state, also called a *combinational circuit*, performs a transformation which may implement combinational logic.[10]
- *Finite states*: An automaton that contains only a finite number of states.
- *Infinite states*: An automaton that may not have a finite number of states, or even a countable number of states. Different kinds of abstract memory may be used to give such machines finite descriptions.
- *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*.
- *Queue memory*: An automaton may have memory in the form of a queue. Such a machine is called *queue machine* and is Turing-complete.
- *Tape memory*: The inputs and outputs of automata are often described as input and output *tapes*. Some machines have additional *working tapes*, including the Turing machine, linear bounded automaton, and log-space transducer.

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. The term transition function is replaced by transition relation: The automaton *non-deterministically* decides to jump into one of the allowed choices. Such automata are called *nondeterministic automata*.
- *Alternation*: This idea is quite similar to tree automata but orthogonal. The automaton may run its *multiple copies* on the *same* next read symbol. Such automata are called *alternating automata*. The acceptance condition must be satisfied on all runs of such *copies* to accept the input.
- *Two-wayness*: Automata may read their input from left to right, or they may be allowed to move back-and-forth on the input, in a way similar to a Turing machine. Automata which can move back-and-forth on the input are called two-way finite automata.

Acceptance condition

- *Acceptance of finite words*: Same as described in the informal definition above.
- *Acceptance of infinite words*: an *ω-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 automata, geometric automata and metric automata have probabilistic acceptance.

Different combinations of the above variations produce many classes of automata.

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

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

Automata theory also studies the existence or nonexistence of any effective algorithms to solve problems similar to the following list:

- Does an automaton accept at least one input word? (Emptiness checking)
- Is it possible to transform a given non-deterministic automaton into a deterministic automaton without changing the language recognized? (Determinization)
- For a given formal language, what is the smallest automaton that recognizes it? (Minimization)

Types of automata

The following is an incomplete list of types of automata.

Automaton	Recognizable languages
Nondeterministic/Deterministic finite-state machine (FSM)	regular languages
Deterministic pushdown automaton (DPDA)	deterministic context-free languages
Pushdown automaton (PDA)	context-free languages
Linear bounded automaton (LBA)	context-sensitive languages
Turing machine	recursively enumerable languages
Deterministic Büchi automaton	ω-limit languages
Nondeterministic Büchi automaton	ω-regular languages
Rabin automaton, Streett automaton, Parity automaton, Muller automaton	
Weighted automaton	

Discrete, continuous, and hybrid automata

Normally automata theory describes the states of abstract machines but there are discrete automata, analog automata or continuous automata, or hybrid discrete-continuous automata, which use digital data, analog data or continuous time, or digital *and* analog data, respectively.

Hierarchy in terms of powers

The following is an incomplete hierarchy in terms of powers of different types of virtual machines. The hierarchy reflects the nested categories of languages the machines are able to accept.[14]

Automaton
Deterministic Finite Automaton (DFA) -- Lowest Power
(same power) || (same power)
Nondeterministic Finite Automaton (NFA)
(above is weaker) ∩ (below is stronger)
Deterministic Push Down Automaton (DPDA-I) with 1 push-down store
∩
Nondeterministic Push Down Automaton (NPDA-I) with 1 push-down store
∩
Linear Bounded Automaton (LBA)
∩
Deterministic Push Down Automaton (DPDA-II) with 2 push-down stores
||
Nondeterministic Push Down Automaton (NPDA-II) with 2 push-down stores
||
Deterministic Turing Machine (DTM)
||
Nondeterministic Turing Machine (NTM)
||
Probabilistic Turing Machine (PTM)
||
Multitape Turing Machine (MTM)
||
Multidimensional Turing Machine

Applications

Each model in automata theory plays important roles in several applied areas. Finite automata are used in text processing, compilers, and hardware design. Context-free grammar (CFGs) are used in programming languages and artificial intelligence. Originally, CFGs were used in the study of human languages. Cellular automata are used in the field of artificial life, the most famous example being John Conway's Game of Life. Some other examples which could be explained using automata theory in biology include mollusk and pine cone growth and pigmentation patterns. Going further, a theory suggesting that the whole universe is computed by some sort of a discrete automaton, is advocated by some scientists. The idea originated in the work of Konrad Zuse, and was popularized in America by Edward Fredkin. Automata also appear in the theory of finite fields: the set of irreducible polynomials that can be written as composition of degree two polynomials is in fact a regular language.[15] Another problem for which automata can be used is the induction of regular languages.

Automata simulators

Automata simulators are pedagogical tools used to teach, learn and research automata theory. An automata simulator takes as input the description of an automaton and then simulates its working for an arbitrary input string. The description of the automaton can be entered in several ways. An automaton can be defined in a symbolic language or its specification may be entered in a predesigned form or its transition diagram may be drawn by clicking and dragging the mouse. Well known automata simulators include Turing's World, JFLAP, VAS, TAGS and SimStudio.[16]

Category-theoretic models

One can define several distinct categories of automata[17] following the automata classification into different types described in the previous section. The mathematical category of deterministic automata, sequential machines or *sequential automata*, and Turing machines with *automata homomorphisms* defining the arrows between automata is a Cartesian closed category,[18] it has both categorical limits and colimits. An automata homomorphism maps a quintuple of an automaton A_i onto the quintuple of another automaton A_j. Automata homomorphisms can also be considered as *automata transformations* or as semigroup homomorphisms, when the state space, $\boldsymbol{S}$, of the automaton is defined as a semigroup $\mathbf{S}_g$. Monoids are also considered as a suitable setting for automata in monoidal categories.[19][20][21]

Categories of variable automata

One could also define a *variable automaton*, in the sense of Norbert Wiener in his book on *The Human Use of Human Beings via* the endomorphisms $A_i \to A_i$. Then one can show that such variable automata homomorphisms form a mathematical group. In the case of non-deterministic, or other complex kinds of automata, the latter set of endomorphisms may become, however, a *variable automaton groupoid*. Therefore, in the most general case, categories of variable automata of any kind are categories of groupoids or groupoid categories. Moreover, the category of reversible automata is then a 2-category, and also a subcategory of the 2-category of groupoids, or the groupoid category.

See also

- Boolean differential calculus
- Petri net

References

1. Mahoney, Michael S. "The Structures of Computation and the Mathematical Structure of Nature" (http://www.rutherfordjournal.org/article030107.html). The Rutherford Journal. Retrieved 2020-06-07.

2. Booth, Taylor (1967). *Sequential Machines and Automata Theory*. New York: John Wiley & Sons. p. 1-13. ISBN 0-471-08848-X.
3. Ashby, William Ross (1967-01-15). "The Place of the Brain in the Natural World" (https://web.archive.org/web/20230604002636/http://rossashby.info/Ashby-Mechanisms_of_intelligence.pdf#16) (PDF). *Currents in Modern Biology*. **1** (2): 95–104. doi:10.1016/0303-2647(67)90021-4 (https://doi.org/10.1016%2F0303-2647%2867%2990021-4). PMID 6060865 (https://pubmed.ncbi.nlm.nih.gov/6060865). Archived from the original (http://www.rossashby.info/Ashby-Mechanisms_of_intelligence.pdf#16) (PDF) on 2023-06-04. Retrieved 2021-03-29.: "The theories, now well developed, of the "finite-state machine" (Gill, 1962), of the "noiseless transducer" (Shannon and Weaver, 1949), of the "state-determined system" (Ashby, 1952), and of the "sequential circuit", are essentially homologous."
4. Ashby, W. R.; et al. (1956). C.E. Shannon; J. McCarthy (eds.). *Automata Studies*. Princeton, N.J.: Princeton University Press.
5. Arbib, Michael (1969). *Theories of Abstract Automata*. Englewood Cliffs, N.J.: Prentice-Hall.
6. Li, Ming; Paul, Vitanyi (1997). *An Introduction to Kolmogorov Complexity and its Applications*. New York: Springer-Verlag. p. 84.
7. Chomsky, Noam (1956). "Three models for the description of language" (https://chomsky.info/wp-content/uploads/195609-.pdf) (PDF). *IRE Transactions on Information Theory*. **2** (3): 113–124. doi:10.1109/TIT.1956.1056813 (https://doi.org/10.1109%2FTIT.1956.1056813). S2CID 19519474 (https://api.semanticscholar.org/CorpusID:19519474). Archived (https://web.archive.org/web/20160307035549/https://chomsky.info/wp-content/uploads/195609-.pdf) (PDF) from the original on 2016-03-07.
8. Nerode, A. (1958). "Linear Automaton Transformations" (https://www.ams.org/journals/proc/1958-009-04/S0002-9939-1958-0135681-9/). *Proceedings of the American Mathematical Society*. **9** (4): 541. doi:10.1090/S0002-9939-1958-0135681-9 (https://doi.org/10.1090%2FS0002-9939-1958-0135681-9).
9. Rabin, Michael; Scott, Dana (Apr 1959). "Finite Automata and Their Decision Problems" (https://web.archive.org/web/20101214122150/http://www.cse.chalmers.se/~coquand/AUTOMATA/rs.pdf) (PDF). *IBM Journal of Research and Development*. **3** (2): 114–125. doi:10.1147/rd.32.0114 (https://doi.org/10.1147%2Frd.32.0114). Archived from the original on 2010-12-14.
10. Hartmanis, J.; Stearns, R.E. (1966). *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, N.J.: Prentice-Hall.
11. Hartmanis, J.; Stearns, R. E. (1964). "Computational complexity of recursive sequences" (http://www.cs.albany.edu/~res/complexity_recursive_seq_1964.pdf) (PDF).
12. Fortnow, Lance; Homer, Steve (2002). "A Short History of Computational Complexity" (https://people.cs.uchicago.edu/~fortnow/papers/history.pdf) (PDF).
13. Moore, Cristopher (2019-07-31). "Automata, languages, and grammars". arXiv:1907.12713 (https://arxiv.org/abs/1907.12713) [cs.CC (https://arxiv.org/archive/cs.CC)].
14. Yan, Song Y. (1998). *An Introduction to Formal Languages and Machine Computation* (https://books.google.com/books?id=ySOwQgAACAAJ). Singapore: World Scientific Publishing Co. Pte. Ltd. pp. 155–156. ISBN 978-981-02-3422-5.
15. Ferraguti, A.; Micheli, G.; Schnyder, R. (2018), *Irreducible compositions of degree two polynomials over finite fields have regular structure*, The Quarterly Journal of Mathematics, vol. 69, Oxford University Press, pp. 1089–1099, arXiv:1701.06040 (https://arxiv.org/abs/1701.06040), doi:10.1093/qmath/hay015 (https://doi.org/10.1093%2Fqmath%2Fhay015), S2CID 3962424 (https://api.semanticscholar.org/CorpusID:3962424)
16. Chakraborty, P.; Saxena, P. C.; Katti, C. P. (2011). "Fifty Years of Automata Simulation: A Review" (http://dl.acm.org/citation.cfm?id=2038893&dl=ACM&coll=DL&CFID=65021406&CFTOKEN=86634854). *ACM Inroads*. **2** (4): 59–70. doi:10.1145/2038876.2038893 (https://doi.org/10.1145%2F2038876.2038893). S2CID 6446749 (https://api.semanticscholar.org/CorpusID:6446749).

17. Jirí Adámek and Věra Trnková. 1990. *Automata and Algebras in Categories*. Kluwer Academic Publishers:Dordrecht and Prague
18. Mac Lane, Saunders (1971). *Categories for the Working Mathematician*. New York: Springer. ISBN 978-0-387-90036-0.
19. http://www.math.cornell.edu/~worthing/asl2010.pdf James Worthington.2010.Determinizing, Forgetting, and Automata in Monoidal Categories. ASL North American Annual Meeting, 17 March 2010
20. Aguiar, M. and Mahajan, S.2010. *"Monoidal Functors, Species, and Hopf Algebras"*.
21. Meseguer, J., Montanari, U.: 1990 Petri nets are monoids. *Information and Computation* **88**:105–155

Further reading

- John E. Hopcroft; Rajeev Motwani; Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Pearson Education. ISBN 978-0-201-44124-6.
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 978-0-534-94728-6. 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.
- Elaine Rich (2008). *Automata, Computability and Complexity: Theory and Applications*. Pearson. ISBN 978-0-13-228806-4.
- Salomaa, Arto (1985). *Computation and automata* (https://archive.org/details/computationautom0000salo). Encyclopedia of Mathematics and Its Applications. Vol. 25. Cambridge University Press. ISBN 978-0-521-30245-6. Zbl 0565.68046 (https://zbmath.org/?format=complete&q=an:0565.68046).
- Anderson, James A. (2006). *Automata theory with modern applications*. With contributions by Tom Head. Cambridge: Cambridge University Press. ISBN 978-0-521-61324-8. Zbl 1127.68049 (https://zbmath.org/?format=complete&q=an:1127.68049).
- Conway, J.H. (1971). *Regular algebra and finite machines*. Chapman and Hall Mathematics Series. London: Chapman & Hall. Zbl 0231.94041 (https://zbmath.org/?format=complete&q=an:0231.94041).
- John M. Howie (1991) *Automata and Languages*, Clarendon Press ISBN 0-19-853424-8 MR1254435 (https://mathscinet.ams.org/mathscinet-getitem?mr=1254435)
- Sakarovitch, Jacques (2009). *Elements of automata theory*. Translated from the French by Reuben Thomas. Cambridge University Press. ISBN 978-0-521-84425-3. Zbl 1188.68177 (https://zbmath.org/?format=complete&q=an:1188.68177).
- James P. Schmeiser; David T. Barnard (1995). *Producing a top-down parse order with bottom-up parsing*. Elsevier North-Holland.
- Igor Aleksander; F. Keith Hanna (1975). *Automata Theory: An Engineering Approach*. New York: Crane Russak. ISBN 978-0-8448-0657-0.
- Marvin Minsky (1967). *Computation: Finite and infinite machines* (https://archive.org/details/computationfinit0000mins). Princeton, N.J.: Prentice Hall.
- John C. Martin (2011). *Introduction to Languages and The Theory of Computation*. New York: McGraw Hill. ISBN 978-0-07-319146-1.

External links

- dk.brics.automaton (http://www.brics.dk/automaton)

- libfa (http://www.augeas.net/libfa/index.html)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Automata_theory&oldid=1222578099"

-