Kleene's algorithm

In theoretical computer science, in particular in formal language theory, **Kleene's algorithm** transforms a given nondeterministic finite automaton (NFA) into a regular expression. Together with other conversion algorithms, it establishes the equivalence of several description formats for regular languages. Alternative presentations of the same method include the "elimination method" attributed to Brzozowski and McCluskey, the algorithm of McNaughton and Yamada,[1] and the use of Arden's lemma.

Algorithm description

According to Gross and Yellen (2004),[2] the algorithm can be traced back to Kleene (1956).[3] A presentation of the algorithm in the case of deterministic finite automata (DFAs) is given in Hopcroft and Ullman (1979).[4] The presentation of the algorithm for NFAs below follows Gross and Yellen (2004).[2]

Given a nondeterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, with $Q = \{ q_0,...,q_n \}$ its set of states, the algorithm computes

> the sets R_{ij}^{k} of all strings that take M from state q_i to q_j without going through any state numbered higher than k.

Here, "going through a state" means entering *and* leaving it, so both i and j may be higher than k, but no intermediate state may. Each set R_{ij}^{k} is represented by a regular expression; the algorithm computes them step by step for $k = -1, 0, ..., n$. Since there is no state numbered higher than n, the regular expression R_{0j}^{n} represents the set of all strings that take M from its start state q_0 to q_j. If $F = \{ q_1,...,q_f \}$ is the set of accept states, the regular expression $R_{01}^{n} \mid ... \mid R_{0f}^{n}$ represents the language accepted by M.

The initial regular expressions, for $k = -1$, are computed as follows for $i \neq j$:

$$R_{ij}^{-1} = a_1 \mid ... \mid a_m \qquad \text{where } q_j \in \delta(q_i,a_1), ..., q_j \in \delta(q_i,a_m)$$

and as follows for $i=j$:

$$R_{ii}^{-1} = a_1 \mid ... \mid a_m \mid \varepsilon \qquad \text{where } q_i \in \delta(q_i,a_1), ..., q_i \in \delta(q_i,a_m)$$

In other words, R_{ij}^{-1} mentions all letters that label a transition from i to j, and we also include ε in the case where $i=j$.

After that, in each step the expressions R_{ij}^{k} are computed from the previous ones by

$$R_{ij}^{k} = R_{ik}^{k-1} (R_{kk}^{k-1})^{*} R_{kj}^{k-1} \mid R_{ij}^{k-1}$$

Another way to understand the operation of the algorithm is as an "elimination method", where the states from 0 to n are successively removed: when state k is removed, the regular expression R_{ij}^{k-1}, which describes the words that label a path from state $i>k$ to state $j>k$, is rewritten into R_{ij}^{k} so as to take into account the

possibility of going via the "eliminated" state *k*.

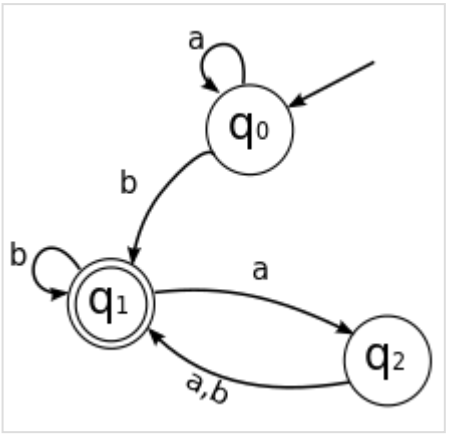
By induction on *k*, it can be shown that the length[5] of each expression R_{ij}^{k} is at most $\frac{1}{3}(4^{k+1}(6s+7) - 4)$ symbols, where *s* denotes the number of characters in Σ. Therefore, the length of the regular expression representing the language accepted by *M* is at most $\frac{1}{3}(4^{n+1}(6s+7)f - f - 3)$ symbols, where *f* denotes the number of final states. This exponential blowup is inevitable, because there exist families of DFAs for which any equivalent regular expression must be of exponential size.[6]

In practice, the size of the regular expression obtained by running the algorithm can be very different depending on the order in which the states are considered by the procedure, i.e., the order in which they are numbered from 0 to *n*.

Example

The automaton shown in the picture can be described as $M = (Q, \Sigma, \delta, q_0, F)$ with

- the set of states $Q = \{ q_0, q_1, q_2 \}$,
- the input alphabet $\Sigma = \{ a, b \}$,
- the transition function δ with $\delta(q_0,a)=q_0$, $\delta(q_0,b)=q_1$, $\delta(q_1,a)=q_2$, $\delta(q_1,b)=q_1$, $\delta(q_2,a)=q_1$, and $\delta(q_2,b)=q_1$,
- the start state q_0, and
- set of accept states $F = \{ q_1 \}$.



Example DFA given to Kleene's algorithm

Kleene's algorithm computes the initial regular expressions as

$$R_{00}^{-1} = a \mid \varepsilon$$
$$R_{01}^{-1} = b$$
$$R_{02}^{-1} = \varnothing$$
$$R_{10}^{-1} = \varnothing$$
$$R_{11}^{-1} = b \mid \varepsilon$$
$$R_{12}^{-1} = a$$
$$R_{20}^{-1} = \varnothing$$
$$R_{21}^{-1} = a \mid b$$
$$R_{22}^{-1} = \varepsilon$$

After that, the R_{ij}^{k} are computed from the R_{ij}^{k-1} step by step for $k = 0, 1, 2$. Kleene algebra equalities are used to simplify the regular expressions as much as possible.

Step 0

$$R_{00}^{0} = R_{00}^{-1} (R_{00}^{-1})^{*} R_{00}^{-1} \mid R_{00}^{-1} = (a \mid \varepsilon) (a \mid \varepsilon)^{*} (a \mid \varepsilon) \mid a \mid \varepsilon = a^{*}$$

$$
\begin{aligned}
R^0_{01} &= R^{-1}_{00}\,(R^{-1}_{00})^*\,R^{-1}_{01} \mid R^{-1}_{01} &&= (a \mid \varepsilon)\,(a \mid \varepsilon)^*\,b && \mid b && = a^*\,b\\
R^0_{02} &= R^{-1}_{00}\,(R^{-1}_{00})^*\,R^{-1}_{02} \mid R^{-1}_{02} &&= (a \mid \varepsilon)\,(a \mid \varepsilon)^*\,\varnothing && \mid \varnothing && = \varnothing\\
R^0_{10} &= R^{-1}_{10}\,(R^{-1}_{00})^*\,R^{-1}_{00} \mid R^{-1}_{10} &&= \varnothing && (a \mid \varepsilon)^*\,(a \mid \varepsilon) \mid \varnothing && = \varnothing\\
R^0_{11} &= R^{-1}_{10}\,(R^{-1}_{00})^*\,R^{-1}_{01} \mid R^{-1}_{11} &&= \varnothing && (a \mid \varepsilon)^*\,b && \mid b \mid \varepsilon && = b \mid \varepsilon\\
R^0_{12} &= R^{-1}_{10}\,(R^{-1}_{00})^*\,R^{-1}_{02} \mid R^{-1}_{12} &&= \varnothing && (a \mid \varepsilon)^*\,\varnothing && \mid a && = a\\
R^0_{20} &= R^{-1}_{20}\,(R^{-1}_{00})^*\,R^{-1}_{00} \mid R^{-1}_{20} &&= \varnothing && (a \mid \varepsilon)^*\,(a \mid \varepsilon) \mid \varnothing && = \varnothing\\
R^0_{21} &= R^{-1}_{20}\,(R^{-1}_{00})^*\,R^{-1}_{01} \mid R^{-1}_{21} &&= \varnothing && (a \mid \varepsilon)^*\,b && \mid a \mid b && = a \mid b\\
R^0_{22} &= R^{-1}_{20}\,(R^{-1}_{00})^*\,R^{-1}_{02} \mid R^{-1}_{22} &&= \varnothing && (a \mid \varepsilon)^*\,\varnothing && \mid \varepsilon && = \varepsilon
\end{aligned}
$$

Step 1

$$
\begin{aligned}
R^1_{00} &= R^0_{01}\,(R^0_{11})^*\,R^0_{10} \mid R^0_{00} &&= a^*b && (b \mid \varepsilon)^*\,\varnothing && \mid a^* && = a^*\\
R^1_{01} &= R^0_{01}\,(R^0_{11})^*\,R^0_{11} \mid R^0_{01} &&= a^*b && (b \mid \varepsilon)^*\,(b \mid \varepsilon) && \mid a^*\,b && = a^*\,b^*\,b\\
R^1_{02} &= R^0_{01}\,(R^0_{11})^*\,R^0_{12} \mid R^0_{02} &&= a^*b && (b \mid \varepsilon)^*\,a && \mid \varnothing && = a^*\,b^*\,ba\\
R^1_{10} &= R^0_{11}\,(R^0_{11})^*\,R^0_{10} \mid R^0_{10} &&= (b \mid \varepsilon)\,(b \mid \varepsilon)^*\,\varnothing && \mid \varnothing && = \varnothing\\
R^1_{11} &= R^0_{11}\,(R^0_{11})^*\,R^0_{11} \mid R^0_{11} &&= (b \mid \varepsilon)\,(b \mid \varepsilon)^*\,(b \mid \varepsilon) && \mid b \mid \varepsilon && = b^*\\
R^1_{12} &= R^0_{11}\,(R^0_{11})^*\,R^0_{12} \mid R^0_{12} &&= (b \mid \varepsilon)\,(b \mid \varepsilon)^*\,a && \mid a && = b^*\,a\\
R^1_{20} &= R^0_{21}\,(R^0_{11})^*\,R^0_{10} \mid R^0_{20} &&= (a \mid b)\,(b \mid \varepsilon)^*\,\varnothing && \mid \varnothing && = \varnothing\\
R^1_{21} &= R^0_{21}\,(R^0_{11})^*\,R^0_{11} \mid R^0_{21} &&= (a \mid b)\,(b \mid \varepsilon)^*\,(b \mid \varepsilon) && \mid a \mid b && = (a \mid b)\,b^*\\
R^1_{22} &= R^0_{21}\,(R^0_{11})^*\,R^0_{12} \mid R^0_{22} &&= (a \mid b)\,(b \mid \varepsilon)^*\,a && \mid \varepsilon && = (a \mid b)\,b^*\,a \mid \varepsilon
\end{aligned}
$$

Step 2

$$
\begin{aligned}
R^2_{00} &= R^1_{02}\,(R^1_{22})^*\,R^1_{20} \mid R^1_{00} &&= a^*b^*ba && ((a|b)b^*a \mid \varepsilon)^* && \varnothing && \mid a^* && = a^*\\
R^2_{01} &= R^1_{02}\,(R^1_{22})^*\,R^1_{21} \mid R^1_{01} &&= a^*b^*ba && ((a|b)b^*a \mid \varepsilon)^* && (a|b)b^* && \mid a^*\,b^*\,b && = a^*\,b\,(a\,(a \mid b) \mid b)^*\\
R^2_{02} &= R^1_{02}\,(R^1_{22})^*\,R^1_{22} \mid R^1_{02} &&= a^*b^*ba && ((a|b)b^*a \mid \varepsilon)^* && ((a|b)b^*a \mid \varepsilon) && \mid a^*\,b^*\,ba && = a^*\,b^*\,b\,(a\,(a \mid b)\,b^*)^*\,a\\
R^2_{10} &= R^1_{12}\,(R^1_{22})^*\,R^1_{20} \mid R^1_{10} &&= b^*\,a && ((a|b)b^*a \mid \varepsilon)^* && \varnothing && \mid \varnothing && = \varnothing\\
R^2_{11} &= R^1_{12}\,(R^1_{22})^*\,R^1_{21} \mid R^1_{11} &&= b^*\,a && ((a|b)b^*a \mid \varepsilon)^* && (a|b)b^* && \mid b^* && = (a\,(a \mid b) \mid b)^*\\
R^2_{12} &= R^1_{12}\,(R^1_{22})^*\,R^1_{22} \mid R^1_{12} &&= b^*\,a && ((a|b)b^*a \mid \varepsilon)^* && ((a|b)b^*a \mid \varepsilon) && \mid b^*\,a && = (a\,(a \mid b) \mid b)^*\,a
\end{aligned}
$$

R^2_{20}	$= R^1_{22} (R^1_{22})^* R^1_{20} \mid R^1_{20}$	$= ((a\mid b)b^*a \mid \varepsilon)$	$((a\mid b)b^*a \mid \varepsilon)^*$	$\varnothing$	$\mid \varnothing$	$= \varnothing$
R^2_{21}	$= R^1_{22} (R^1_{22})^* R^1_{21} \mid R^1_{21}$	$= ((a\mid b)b^*a \mid \varepsilon)$	$((a\mid b)b^*a \mid \varepsilon)^*$	$(a\mid b)b^*$	$\mid (a \mid b)\, b^*$	$= (a \mid b)\, (a\, (a \mid b) \mid b)^*$
R^2_{22}	$= R^1_{22} (R^1_{22})^* R^1_{22} \mid R^1_{22}$	$= ((a\mid b)b^*a \mid \varepsilon)$	$((a\mid b)b^*a \mid \varepsilon)^*$	$((a\mid b)b^*a \mid \varepsilon)$	$\mid (a \mid b)\, b^*\, a \mid \varepsilon$	$= ((a \mid b)\, b^*\, a)^*$

Since q_0 is the start state and q_1 is the only accept state, the regular expression R^2_{01} denotes the set of all strings accepted by the automaton.

See also

- Floyd–Warshall algorithm — an algorithm on weighted graphs that can be implemented by Kleene's algorithm using a particular Kleene algebra
- Star height problem — what is the minimum stars' nesting depth of all regular expressions corresponding to a given DFA?
- Generalized star height problem — if a complement operator is allowed additionally in regular expressions, can the stars' nesting depth of Kleene's algorithm's output be limited to a fixed bound?
- Thompson's construction algorithm — transforms a regular expression to a finite automaton

References

1. McNaughton, R.; Yamada, H. (March 1960). "Regular Expressions and State Graphs for Automata". *IRE Transactions on Electronic Computers*. **EC-9** (1): 39–47. doi:10.1109/TEC.1960.5221603 (https://doi.org/10.1109%2FTEC.1960.5221603). ISSN 0367-9950 (https://www.worldcat.org/issn/0367-9950).
2. Jonathan L. Gross and Jay Yellen, ed. (2004). *Handbook of Graph Theory*. Discrete Mathematics and it Applications. CRC Press. ISBN 1-58488-090-2. Here: sect.2.1, remark R13 on p.65
3. Kleene, Stephen C. (1956). "Representation of Events in Nerve Nets and Finite Automata" (http://www.dlsi.ua.es/~mlf/nnafmc/papers/kleene56representation.pdf) (PDF). *Automata Studies, Annals of Math. Studies*. **34**. Princeton Univ. Press. Here: sect.9, p.37-40
4. John E. Hopcroft, Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation* (https://archive.org/details/introductiontoau00hopc). Addison-Wesley. ISBN 0-201-02988-X. Here: Section 3.2.1 pages 91-96
5. More precisely, the number of regular-expression symbols, "a_i", "ε", "|", "*", "·"; not counting parentheses.
6. Gruber, Hermann; Holzer, Markus (2008). "Finite Automata, Digraph Connectivity, and Regular Expression Size". In Aceto, Luca; Damgård, Ivan; Goldberg, Leslie Ann; Halldórsson, Magnús M.; Ingólfsdóttir, Anna; Walukiewicz, Igor (eds.). *Automata, Languages and Programming*. Lecture Notes in Computer Science. Vol. 5126. Springer Berlin Heidelberg. pp. 39–50. doi:10.1007/978-3-540-70583-3_4 (https://doi.org/10.1007%2F978-3-540-70583-3_4). ISBN 9783540705833. S2CID 10975422 (https://api.semanticscholar.org/CorpusID:10975422).. Theorem 16.