INFORMATICA, 2000, Vol. 11, No. 1, 65–70
© 2000 *Institute of Mathematics and Informatics, Vilnius*

An Algorithm for Transformation of Finite Automata to Regular Expressions

Larisa STANEVICHENE, Aleksei VYLITOK

Department of Computational Mathematics and Cybernetics
Moscow State University, Moscow, Russia
e-mail: stanev@ccas.ru, vytolik@cs.msu.su

Received: January 2000

Abstract. An original algorithm for transformation of finite automata to regular expressions is presented. This algorithm is based on effective graph algorithms and gives a transparent new proof of equivalence of regular expressions and finite automata.

Key words: regular sets, finite automata, construction of regular expressions.

1. Introduction

Regular sets play an important role in mathematics and applications. There exist many papers devoted to research and application of regular sets. We note here only (Martynenko, 1998). The last treatise presents a technique of syntax controlled data processing and uses notions of regular expression and finite automaton.

The work (Martynenko, 1998) along with some others demonstrates an importance of constructions, which prove Klenee's theorem on equivalence of regular expressions and finite automata. There exist many proofs of Klenee's theorem; one of the latest of them is contained in (Melnikov, 1998). The present paper gives a construction of a regular expression from finite automaton based on effective graph algorithms. For example, an algorithm of strongly connected components selection (Reingold, 1977) is used as important part of the construction.

Our construction is recursive as Klenee's one. It interprets an automaton as quite natural system of subautomata. This system indicates a method of automaton partition and allows to reduce automaton processing to a processing of its smaller parts. Presented here algorithm implies the idea of processing of D-graphs (Stanevichene, 1997; Gomozov, 1999) by parts.

In Section 2 of this paper some useful notions and denotations are introduced. Section 3 defines a subautomata hierarchy for a finite automaton. In Section 4 the algorithm for transformation of finite automata to regular expressions is formulated.

2. Preliminaries

Let us introduce notations and terms that are used in the next sections of the paper.

DEFINITION 1. A finite automaton (over the alphabet Σ) is a quintuple

$$\mathcal{A} = (K, \Sigma, \delta, p_0, F),$$

where
K is a finite set of states, or vertices;
Σ is an input alphabet (finite, also);
$\delta \subseteq K \times (\Sigma \cup \{\Lambda\}) \times K$ is a set of commands, or edges (the second element of an edge is its label);
$p_0 \in K$ is an initial state; $F \subseteq K$ is a set of final states.

So, a computation of a finite automaton $\mathcal{A}$ over a word x may be regarded as $\mathcal{A}$'s path with the label x. The label of a path is defined as the natural sequence of its edge labels.

DEFINITION 2. Let T be a path of a finite automaton. Then its label is noted by $\omega(T)$, its initial vertex is noted by $beg(T)$, and its terminal vertex is noted by $end(T)$.

DEFINITION 3. Let T be a path of a finite automaton $\mathcal{A}$ =(K, Σ, δ, p_0, F). Let $beg(T) = p_0$, $end(T) \in F$. Then T is called a sentence.

DEFINITION 4. $Sentences(\mathcal{A})$ is defined as the set of all sentences of the finite automaton $\mathcal{A}$.

Note that the language $L(\mathcal{A})$ may be defined by: $L(\mathcal{A}) = \{\omega(T) \mid T \in Sentenses(\mathcal{A})\}$.

DEFINITION 5. Let Σ be an alphabet. Let $\Sigma\cap$ $\{*, +, \epsilon, \emptyset$, (,)$\}$=$\emptyset$. Let us define recursively a regular expression γ over Σ (as a specific word over the alphabet $Alph = \Sigma\cup\{*, +, \epsilon, \emptyset$, (,)$\}$) and the regular language $L(\gamma)$:

1) $a \in \Sigma\cup\{\epsilon, \emptyset\}$ is a regular expression; $L(a) = \{a\}$ for $a \in \Sigma$; $L(\epsilon) = \{\Lambda\}$, where Λ denotes the empty word; $L(\emptyset) = \emptyset$;
2) if α and β are regular expressions, then:
   a) $\alpha + \beta$ is a regular expression; the subexpressions α and β are called addends of this expression, the expression itself is called a sum; $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$;
   b) $(\alpha)(\beta)$ is a regular expression; the subexpressions α and β are called factors, the expression itself is called a product; if a factor is not a sum, then parentheses are not required; $L((\alpha)(\beta)) = L(\alpha)L(\beta)$;
   c) $(\beta)^*$ is a regular expression that is called an iteration of β; $L((\beta)^*) = (L(\beta))^*$;
3) all regular expressions over Σ are constructed by 1–2.

3. An Hierarchy in a Finite Automaton

DEFINITION 6. An oriented graph is strongly connected iff, for every its vertices u and v there exist paths from u to v and from v to u.

DEFINITION 7. Let $\hat{G} = (\hat{V}, \hat{E})$ be an oriented graph, $G = (V, E)$ be a strongly connected subgraph of $\hat{G}$. Let $V \subseteq V' \subseteq \hat{V}$, $E \subseteq E' \subseteq \hat{E}$, $(V, E) \neq (V', E')$ imply that (V', E') is not strongly connected graph. Then G is called a strongly connected component of $\hat{G}$.

DEFINITION 8. Define a trivial graph as one that has a unique vertex and empty set of edges.

DEFINITION 9. Let $\mathcal{B}$ be a subgraph of a finite automaton over Σ. Let $\mathcal{C} = (V, E)$ be a nontrivial strongly connected component of the graph $\mathcal{B}$, $\pi \in E$. Then denote a finite automaton $(V, \Sigma, E - \{\pi\}, end(\pi), \{beg(\pi)\})$ by $Automaton(\mathcal{B}, \pi)$.

DEFINITION 10. Let $\mathcal{C}$ be a connected subgraph of a finite automaton. Let us define recursively a rank $rank(\mathcal{C})$ of this subgraph:

1) $rank(\mathcal{C})$=0, if $\mathcal{C}$ is a trivial strongly connected component;
2) $rank(\mathcal{C}) = 1 + min\{rank(Automaton(\mathcal{C}, \pi)) \mid \pi$ is an edge of $\mathcal{C}\}$, if $\mathcal{C}$ coincides with its nontrivial strongly connected component;
3) $rank(\mathcal{C}) = max\{rank(\mathcal{B}) \mid \mathcal{B}$ is a strongly connected component of $\mathcal{C}\}$.

4. A Transformation of a Finite Automaton to a Regular Expression

Let us denote by $Lines(\mathcal{A})$ a set of $\mathcal{A}$'s sentences having no cycles: $Lines(\mathcal{A}) = \{T \in Sentences(\mathcal{A}) \mid \forall (T_1, T_2, T_3) \quad T = T_1T_2T_3 \Rightarrow (T_2$ is not a cycle$)\}$.

DEFINITION 11. Let us define recursively a regular expression $\mathcal{E}(\mathcal{A})$ over the alphabet of edges of the automaton $\mathcal{A}$; below every empty path is regarded as a synonym of the notation Λ of empty word, every nonempty path is regarded as a word over the alphabet of $\mathcal{A}$'s edges:

$$\mathcal{E}(\mathcal{A}) = \sum_{T \in Lines(\mathcal{A})} Addend(T),$$

$$Addend(T) = \begin{cases} Cycles(beg(T))\,\epsilon, \quad T \text{ is empty}, \\ Cycles(beg(T))\pi_1 Cycles(end(\pi_1))...\pi_k Cycles(end(\pi_k)), \\ \qquad k \geqslant 1, \ \ T = \pi_1...\pi_k, \quad \pi_i \text{ is an edge of } \mathcal{A} \text{ for } i = 1, ..., k, \end{cases}$$

where auxiliary expression $Cycles(p),\ p \in K$, is defined below.

If p is a vertex of a nontrivial strongly connected component, then

$$Cycles(p) = \Bigg(\sum_{\substack{\pi \text{ is an edge of a strongly} \\ \text{connected component,} \\ beg(\pi) = p}} Bound(\pi) \Bigg)^*,$$

else $Cycles(p) = p$; $Bound(\pi) = \pi(\mathcal{E}(Automaton(\mathcal{B}, \pi)))$ for an edge π of a strongly connected component $\mathcal{B}$ of an automaton $\mathcal{A}$.

Theorem 1. $L(\mathcal{E}(\mathcal{A})) = Sentences(\mathcal{A})$.

Proof. By induction on $r = rank(\mathcal{A})$. For $r = 0$ it holds the equality $Lines(\mathcal{A}) = Sentences(\mathcal{A})$. Consequently,

$$\mathcal{E}(\mathcal{A}) = \sum_{T \in Sentences(\mathcal{A})} Addend(T),$$

where

$$Addend(T) = \begin{cases} \epsilon, \quad T \text{ is empty}, \\ \pi_1 \ldots \pi_k, \;\; k \geqslant 1, \\ \qquad T = \pi_1 ... \pi_k, \; \pi_i \text{ is an edge of } \mathcal{A} \text{ for } i = 1, ..., k, \end{cases}$$

and the theorem holds in this case.

Let $r > 0$. Assume the assertion for every automaton the rank of which is not greater than $r - 1$. Consider an automaton $\mathcal{A}$ having the rank r.

For every strongly connected component $\mathcal{B}$ of the automaton $\mathcal{A}$ and every vertex p of this component, the regular expression $Cycles(p)$ defines all the cycles with the terminal vertex p. Indeed, let π be an edge of the strongly connected component and $beg(\pi) = p$. Consider the addend $Bound(\pi)$ from the definition of the expression $Cycles(p)$. By induction hypothesis its subexpression $\mathcal{E}(Automaton(\mathcal{B}, \pi))$ defines all the paths from $end(\pi)$ to $beg(\pi) = p$ not containing the edge π. Thus, the expression $Bound(\pi)$ defines all the cycles of the form πT, where $end(T) = p$ and p does not appear within T. Consequently, the sum

$$\sum_{\substack{\pi \text{ is an edge of a strongly} \\ \text{connected component,} \\ beg(\pi) = p}} Bound(\pi)$$

defines all the cycles the first edge of which does not recur and have the initial vertex p. Observe that every cycle with the end p may be presented by a sequence of cycles of the form being indicated above. Hence, the iteration of the considered sum defines all the paths from p to p.

This fact implies that the sum $\mathcal{E}(\mathcal{A})$ defines all the paths the first vertex of which is p_0 and the last one is a final vertex of the automaton $\mathcal{A}$.

Define a morphism φ by: $\varphi(\pi) = \omega(\pi),\ \pi \in \delta,\ \omega(\pi) \in \Sigma$; $\varphi(\pi) = \epsilon,\ \pi \in \delta,\ \omega(\pi) = \Lambda$; $\varphi(a) = a,\ a \notin \delta$. Then the regular expression $\varphi(\mathcal{E}(\mathcal{A}))$ describes the language $L(\mathcal{A})$.

The definition of a regular expression $\varphi(\mathcal{E}(\mathcal{A}))$ implies a recursive algorithm of a transformation of a finite automaton to an equivalent regular expression. The details of the algorithm are omitted in this paper.

References

Gomozov, A.L., L.I. Stanevichene (1999). A generalization of regular expressions. *Informatica*, **10**(1), 27–44.

Martynenko, B.K. (1998). A syntax controlled data processing. *Doctoral thesis*. St Petersburg State University (in Russian).

Melnikov, B.F., A.A. Vakhitova (1998). Some more on the finite automata. *Korean Journal of Computational and Applied Mathematics*, **5**(3), 495–505.

Reingold, E.N., J. Nievergelt, N. Deo (1977). *Combinatorial Algorithms. Theory and Practice.* Englewood Cliffs, N.Y., Prentice Hall.

Stanevichene, L.I. (1997). D-graphs in context-free language theory. *Informatica*, **8**(1), 43–56.

L. Stanevichene received the degree of candidate of Physical and Mathematical Sciences from the Computer Centre of the USSR Academy of Sciences in 1972. She is a senior tutor of the Computational Mathematics and Cybernetics Department of the Moscow State University. Her research interests include formal language theory and its applications.

A. Vylitok received the degree of candidate of Physical and Mathematical Sciences from the Moscow State University in 1998. His candidate thesis was prepared under the supervision of L. Stanevichene. He is a research worker of the Computational Mathematics and Cybernetics Department of the Moscow State University. His research interests include the theory of parsing, formal language theory and its applications.

Algoritmas baigtiniams automatams išreikšti reguliariaisiais reiškiniais

Larisa STANEVICHENE, Aleksei VYLITOK

Straipsnyje pateiktas originalus algoritmas, skirtas baigtiniams automatams išreikšti reguliariaisiais reiškiniais. Algoritmas grindžiamas efektyviais grafų apdorojimo algoritmais. Jis traktuotinas kaip naujas konstryktyvus teoremos apie baigtinių automatų ir reguliariųjų reiškinių ekvivalentumą įrodymo būdas.