ELSEVIER Theoretical Computer Science 155 (1996) 439–446

Theoretical Computer Science

Note

Local languages and the Berry–Sethi algorithm

Jean Berstel, Jean-Eric Pin*

LITP, Institut Blaise Pascal, 4 place Jussieu, 75252 Paris, France

Received February 1995
Communicated by M. Nivat

Abstract

One of the basic tasks in compiler construction, document processing, hypertext software and similar projects is the efficient construction of a finite automaton from a given rational (regular) expression. The aim of the present paper is to give an exposition and a formal proof of the background for the algorithm of Berry and Sethi relating the computation involved to a well-known family of recognizable languages, the local languages.

1. Introduction

One of the basic tasks in compiler construction, document processing, hypertext software and similar projects is the efficient construction of a finite automaton from a given rational (regular) expression. There exist a great variety of algorithms for this. An impressive account has been given recently by Watson [11]. For several reasons, the algorithm of Berry and Sethi [2] is of particular interest (see [4,5] for a discussion). The aim of the present paper is to give an exposition and formal proof of the background for this algorithm by relating the computation involved to a well-known family of recognizable languages, the local languages.

Local languages were studied in some detail in [10], see also [7]. These languages are very easy to define, and they are exactly the languages recognized by a special family of automata also called Glushkov automata. The main result used in the Berry–Sethi algorithm is that every language denoted by a linear rational expression can be recognized by a Glushkov automaton. We give a short proof of this, by showing that every language denoted by a linear rational expression is local. Observe however that the inclusion is strict.

*Corresponding author. Email: pin@litp.ibp.fr

SSDI 0304-3975(95)00104-2

The development of efficient algorithms is an important issue (see [8, 5, 13]) but we are not concerned with this problem in this paper. Our goal is rather to provide a simple formal proof of the correctness of the algorithm.

In the topic of transducing a regular expression to an automaton, the terminology is not yet uniform. Thus, linear expressions are called restricted in [11]. Also, what we denote by P and S is frequently written *First* and *Last*. The set of factors of length 2 of a language (or of the language denoted by an expression) that we write F for short is sometimes written *Follow*.

A first presentation of the relation between the Berry–Sethi algorithm and local languages appeared in [3].

2. Local languages

Given a language $L \subset A^*$ define

$$P(L) = \{a \in A \mid aA^* \cap L \neq \emptyset\}, \qquad S(L) = \{a \in A \mid A^*a \cap L \neq \emptyset\},$$

$$F(L) = \{x \in A^2 \mid A^*xA^* \cap L \neq \emptyset\}, \qquad N(L) = A^2 \backslash F(L).$$

By definition, $P(L)$ is the set of first letters of words in L and $F(L)$ is the set of factors (subwords) of length 2 of words in L. Clearly, for every language, one has

$$L \backslash \{1\} \subset (P(L)A^* \cap A^*S(L)) \backslash A^*N(L)A^*.$$

A language L is called *local* if equality holds. More precisely, a language $L \subset A^*$ is said to be *local* if there exist two subsets P and S of A and a subset N of A^2 such that[1]

$$L \backslash \{1\} = (PA^* \cap A^*S) \backslash A^*NA^*.$$

For example, if $A = \{a, b, c\}$, the language

$$(abc)^* = \{1\} \cup [(aA^* \cap A^*c) \backslash A^* \{aa, ac, ba, bb, cb, cc\} A^*]$$

is local. The terminology "local" can be explained as follows: in order to know whether a given word is in L, it suffices to verify that its first letter is in P, its last letter is S, and all its factors of length 2 are not in N. Thus, membership in L can be checked by scanning the word through a window of size 2. Conversely, if a language L is local, it is easy to recover the parameters P, S and N. Indeed P (respectively S) is the set of all first (last) letter of the words of L and N is the set of words of length 2 that are not factors of any word in L.

One can easily find a deterministic automaton recognizing a local language given the parameters P, S and N. We consider the following type of automata which, as we shall see, characterize local languages: a deterministic (but not necessarily complete)

[1] P stands for prefix, S for suffix, and N for non-factor.

automaton $\mathscr{A} = (Q, A, ., i, T)$ is said to be *local* if, for every letter a, the set $\{q.a \mid q \in Q\}$ contains at most one element. A deterministic automaton is said to be *standard* if it contains no transition arriving on the initial state.

Proposition 2.1. *Let $L = (PA^* \cap A^*S)\backslash A^*NA^*$ be a local language. Then L is recognized by the standard local automaton $\mathscr{A}$ having $A \cup \{1\}$ as set of states, 1 as initial state, S as set of final states and whose transitions are given by the rules $1.a = a$ if $a \in P$ and $a.b = b$ if $ab \notin N$.*

Proof. Let indeed $u = a_1 \cdots a_n$ be a word accepted by $\mathscr{A}$. Then there is a successful path

$$1 \xrightarrow{a_1} a_1 \xrightarrow{a_2} a_2 \cdots a_{n-1} \xrightarrow{a_n} a_n .$$

Consequently, the end of the path, a_n, is a final state and thus $a_n \in S$. Similarly, since there is a transition $1 \xrightarrow{a_1} a_1$, one has necessarily $a_1 \in P$. Finally, for $1 \leqslant j \leqslant n-1$, there is a transition $a_j \xrightarrow{a_{j+1}} a_{j+1}$, and thus $a_j a_{j+1} \notin N$. It follows that $u \in L$.

Conversely, if $u = a_1 \cdots a_n \in L$, it follows that $a_1 \in P$, $a_n \in S$ and, for $1 \leqslant j \leqslant n$, $a_j a_{j+1} \notin N$. Therefore $1 \xrightarrow{a_1} a_1 \xrightarrow{a_2} a_2 \cdots a_{n-1} \xrightarrow{a_n} a_n$ is a successful path of $\mathscr{A}$ and $\mathscr{A}$ accepts w. Consequently the language recognized by $\mathscr{A}$ is L.

If the local language contains the empty word, the previous construction can be applied, by taking $S \cup \{1\}$ as set of final states. This completes the proof. □

Proposition 2.2. *Let $L \subset A^*$ be a rational language. The following conditions are equivalent:*

(1) *L is a local language.*
(2) *L is recognized by a local automaton.*
(3) *L is recognized by a standard local automaton.*

Proof. (1) implies (3) by Proposition 2.1. (3) implies (2) is trivial.

(2) implies (1): Let $\mathscr{A} = (Q, A, ., i, T)$ be a local automaton that recognizes a language L. Set

$$P = \{a \in A \mid i.a \text{ is defined}\},$$

$$S = \{a \in A \mid \text{there exists } q \in Q \text{ such that } q.a \in T\},$$

$$N = \{x \in A^2 \mid x \text{ is the label of no path in } \mathscr{A}\},$$

$$K = (PA^* \cap A^*S)\backslash A^*NA^*.$$

Let $u = a_1 \cdots a_n$ be a non-empty word of L. Then u is the label of a successful path

$$c: \quad i = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$$

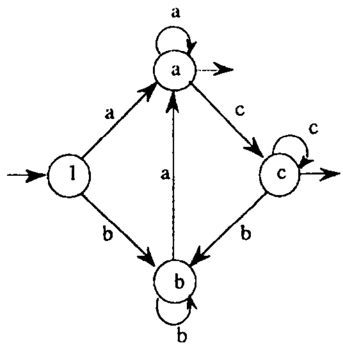

Fig. 1.

In particular, $a_1 \in P$, $q_n \in T$ and thus $a_n \in S$, and for $1 \leqslant j \leqslant n-1$, one has $a_j a_{j+1} \notin N$. Consequently $u \in K$, and thus $L \setminus \{1\}$ is contained in K.

Conversely, let $u = a_1 \cdots a_n$ be a non-empty word of K and set $q_0 = i$. By assumption, $a_1 \in P$, $a_n \in S$ and, for $1 \leqslant j \leqslant n-1$, $a_j a_{j+1} \notin N$. Since $a_1 \in P$, $q_0 . a_1$ is defined. Set $q_0 . a_1 = q_1$. We show by induction that there exists a sequence of states q_j $(0 \leqslant j \leqslant n)$ such that $a_1 \cdots a_j$ is the label of a path $q_0 \to q_1 \to \cdots \to q_j$ of $\mathscr{A}$. Indeed, since $a_j a_{j+1} \notin N$, $a_j a_{j+1}$ is the label of some path $p \xrightarrow{a_j} q \xrightarrow{a_{j+1}} r$. But since the automaton $\mathscr{A}$ is local, $q_{j-1} . a_j = p . a_j$, that is $q = q_j$ and thus q_{j+1} is defined as $q_{j+1} = r$. Finally since $a_n \in S$, it follows that $q_n \in T$. Consequently $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$ is a successful path of $\mathscr{A}$ and its label u is recognized by $\mathscr{A}$. $\square$

Example 2.1. Let $A = \{a, b, c\}$, $P = \{a, b\}$, $S = \{a, c\}$ and $N = \{ab, bc, ca\}$. Then the language $L = (PA^* \cap A^*S) \setminus A^*NA^*$ is recognized by the automaton represented in Fig. 1.

Local language are stable under various operations:

Proposition 2.3. *Let A_1 and A_2 be two disjoint subsets of the alphabet A, and let $L_1 \subset A_1^*$ and $L_2 \subset A_2^*$ be two local languages. Then the languages $L_1 \cup L_2$ and $L_1 L_2$ are also local languages.*

Proof. Let $\mathscr{A}_1 = (Q_1, A_1, E_1, i_1, T_1)$ and $\mathscr{A}_2 = (Q_2, A_2, E_2, i_2, T_2)$ be standard local automata recognizing L_1 and L_2 respectively. Then $L_1 \cup L_2$ is recognized by the local automaton (Q, A, E, i, T) where

$$Q = (Q_1 \setminus \{i_1\}) \cup (Q_2 \setminus \{i_2\}) \cup \{i\} \quad (i \text{ is a new state})$$

$$E = \{(q, a, q') \mid (q, a, q') \in E_1 \cup E_2, q \neq i_1, q \neq i_2\}$$

$$\cup \{(i, a, q) \mid (i_1, a, q) \in E_1 \text{ or } (i_2, a, q) \in E_2\}$$

$$T = \begin{cases} T_1 \cup T_2 & \text{if } i_1 \notin T_1 \text{ and } i_2 \notin T_2 \\ (T_1 \setminus \{i_1\}) \cup (T_2 \setminus \{i_2\}) \cup \{i\} & \text{otherwise.} \end{cases}$$

For the product, set $\mathscr{A} = (Q, A, E, I, T)$, with

$$Q = (Q_1 \cup Q_2) \setminus \{i_2\}$$

$$E = E_1 \cup \{(q, a, q') \in E_2 \mid q \neq i_2\} \cup \{(q_1, a, q_2) \mid q_1 \in T_1 \text{ and } (i_2, a, q_2) \in E_2\}$$

$$I = I_1$$

$$T = \begin{cases} T_2 & \text{if } i_2 \notin T_2. \\ T_1 \cup (T_2 \setminus \{i\}) & \text{if } i_2 \in T_2 \text{ (that is if } 1 \in L_2). \end{cases}$$

By construction, $\mathscr{A}$ is a local automaton and it is easy to verify that it recognizes $L_1 L_2$. □

Proposition 2.4. *Let L be a local language. Then the language L^* is also a local language.*

Proof. Let $\mathscr{A} = (Q, A, E, i, T)$ be a standard local automaton recognizing L. Consider the automaton $\mathscr{A}' = (Q, A, E', i, T \cup \{i\})$, with

$$E' = E \cup \{(q, a, q') \mid q \in T \text{ and } (i, a, q') \in E\}$$

Then $\mathscr{A}'$ is local and recognizes L^*. □

3. Berry–Sethi Algorithm

Berry and Sethi proposed an algorithm to find a non-deterministic automaton recognizing a given rational expression. For any rational expression e, we denote by $L(e)$ the language that e represents.

We say that a rational expression is *linear* if every letter a has at most one occurrence in the expression (in Watson [11], it is called *restricted*). For example, the expression $[a_1 a_2 (a_3 a_4)^* \cup (a_5 a_6)^* a_7]^*$ is linear. One can linearize any rational expression by replacing all the letters that occur in it by distinct symbols. For example, the above expression is a linearization of the expression $e = [ab(ba)^* \cup (ac)^* b]^*$. Now, given an automaton that recognizes the language $L(e')$ of a linearized version e' of a rational expression e, it is easy to obtain an automaton for the language $L(e)$, by replacing letters of e' by the corresponding letters of e. For instance, if $\mathscr{A}$ is the automaton represented in Fig. 2 (which recognizes the language $[(a_1 a_2)^* a_3]^*$), one obtains, by replacing a_1 and a_3 by a and a_2 by b, the (non-deterministic) automaton $\mathscr{A}'$, which recognizes $[(ab)^* a]^*$.

Therefore it suffices to be able to compute an automaton for each linear expression.

Proposition 3.1. *For every linear expression e, the language $L(e)$ is local.*

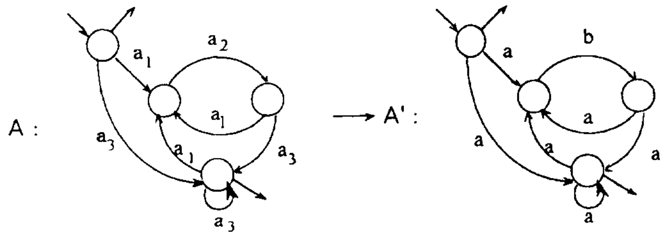

Fig. 2.

Proof. The proof is by induction on the formation rules of linear expressions. First, the languages represented by 0, 1 and a, for $a \in A$, are local languages. Next, by Proposition 2.4, if e represents a local language, then e^* represents also a local language. Let now e and e' be two linear expressions and suppose that the expression $(e \cup e')$ is linear. Let B (respectively B') be the set of letters occurring in e (e'). Since $(e \cup e')$ is linear, the sets B and B' are disjoint, and the local language $L(e)$ ($L(e')$) is contained in B^* (B'^*). By Proposition 2.3, the languages $L(e \cup e')$ and $L(ee')$ are also local. □

Observe that the converse does not hold: for instance, the language $(ab)^*a$ is local but is not denoted by a linear expression.

We have seen in the previous section an algorithm to compute a deterministic automaton recognizing a given local language L. It suffices to test whether the empty word belongs to L and to compute the sets

$$P(L) = \{a \in A \mid aA^* \cap L \neq \emptyset\}.$$

$$S(L) = \{a \in A \mid A^*a \cap L \neq \emptyset\},$$

$$F(L) = \{x \in A^2 \mid A^*xA^* \cap L \neq \emptyset\}.$$

But this can be easily done given a rational expression (linear or not) representing the language, by making use of the following well-known recursive procedures. First, we compute $\Lambda(e) = \{1\} \cap L(e)$ as follows:

$$\Lambda(0) = \emptyset;$$

$$\Lambda(1) = \{1\};$$

$$\Lambda(a) = \emptyset \quad \text{for all } a \in A;$$

$$\Lambda(e \cup e') = \Lambda(e) \cup \Lambda(e');$$

$$\Lambda(e.e') = \Lambda(e) \cap \Lambda(e');$$

$$\Lambda(e^*) = \{1\};$$

Next,

$$P(0) = \emptyset; \qquad S(0) = \emptyset;$$
$$P(1) = \emptyset; \qquad S(1) = \emptyset;$$
$$P(a) = \{a\} \quad \text{for all } a \in A; \qquad S(a) = \{a\} \quad \text{for all } a \in A;$$
$$P(e \cup e') = P(e) \cup P(e'); \qquad S(e \cup e') = S(e) \cup S(e');$$
$$P(e.e') = P(e) \cup \Lambda(e)P(e'); \qquad S(e.e') = S(e') \cup S(e)\Lambda(e');$$
$$P(e^*) = P(e); \qquad S(e^*) = S(e);$$

$$F(0) = \emptyset;$$
$$F(1) = \emptyset;$$
$$F(a) = \emptyset \quad \text{for all } a \in A;$$
$$F(e \cup e') = F(e) \cup F(e');$$
$$F(e.e') = F(e) \cup F(e') \cup S(e)P(e');$$
$$F(e^*) = F(e) \cup S(e)P(e).$$

To sum up, given a rational expression e, Berry–Sethi algorithm produces a non-deterministic automaton as follows:

(1) Compute a linear version e' of e and memorize the encoding of letters.
(2) Compute recursively the sets $P(e')$, $S(e')$ and $F(e')$.
(3) Compute a deterministic automaton $\mathscr{A}'$ recognizing e'.
(4) Decode the letters of e' to compute a non-deterministic automaton recognizing e.

4. Final remark

Observe that Berry and Sethi have given an unusual proof of a well-known result, namely that every rational language is the homomorphic image of a local language.

Added in proof. B.W. Watson's thesis is: Taxonomies and Toolkits of Regular Language Algorithms, Eindhoven University of Technology, Sept. 1995.

References

[1] A.V. Aho, R. Sethi and J. Ullman, *Compilers: Principles, Techniques and Tools. Addison-Wesley Series in Computer Science* (Addison–Wesley, Reading, Mass., 1986).
[2] G. Berry and R. Sethi, From regular expressions to deterministic automata, *Theoret. Comput. Sci.* **48** (1986) 117–126.
[3] J. Berstel, Finite automata and rational languages, an introduction, in: J.E. Pin, ed., *Formal Properties of Finite Automata and Applications*, Lecture Notes in Computer Science, Vol. 386 (1987) 2–14.

[4] A. Brüggemann-Klein, Regular expressions into finite automata, in: LATIN'92, Lecture Notes in Computer Science, Vol. 583 (1992) 87–98.
[5] A. Brüggemam-Klein and D. Wood, Deterministic regular languages, in: STACS 92, Lecture Notes in Computer Science, Vol. 577 (1992) 173–184.
[6] J.M. Champarnaud, From a regular expression to an automaton, *Inform. Process. Lett.*, to appear.
[7] S. Eilenberg, *Automata, Languages and Machines*, Vol A (Academic Press, New York, 1974).
[8] V.M. Glushkov, The abstract theory of automata, *Russian Math. Surveys* **16** (1961) 1–53.
[9] R. McNaughton and H. Yamada, Regular expressions and state graphs for automata, *IEEE Trans, Electronic Comput.* **9** (1960) 39–47.
[10] M. Nivat, Transductions des langages de Chomsky, *Ann. Fourier* **18** (1968) 339–455.
[11] B.W. Watson, A taxonomy of finite automata construction algorithms, Computing Science Note 93-43, Eindhoven University of Technology. The Netherlands, 1993.
[12] B.W. Watson, A taxonomy of finite automata minimization algorithms, Computing Science Note 93-44, Eindhoven University of Technology, The Netherlands, 1993.
[13] D. Ziadi, J.L. Ponty and J.-M. Champarnaud, Passage d'une expression rationnelle à un automate fini non-déterministe, *Bull. Belgian Math. Soc. Simon Stevin*, to appear.