LANGUAGE OPERATIONS PERFORMED BY FINITE TRANSDUCERS

Ing. Martin VÍTEK, Doctoral Degree Programme (3)
Dept. of Information Systems, FIT, BUT
E–mail: vitek@fit.vutbr.cz

Supervised by: Prof. Alexander Meduna

ABSTRACT

Stringology represents a modern part of the formal language theory, which deals with strings, languages and operations on them. It introduces many new language operations, which can be divided into two groups — insertion and deletion operations. Some of these operations are described in [1]. This paper presents these operations and some their properties. The main contribution of this paper are algorithms for construction of finite transducers which translate input regular language by selected operation to output language.

1 INTRODUCTION

The language operations that change strings by shuffling or inserting some substrings fulfill an important role in several modern computer science fields, ranging from cryptography through various text algorithms and stringology to DNA computation. Therefore, it comes as no surprise that the formal language theory has recently played a special attention to their investigation (see [1]). The present paper introduces and discusses some more operations of this kind. Specifically, it discusses operations sequential insertion, parallel insertion, scattered sequential insertion, sequential deletion, parallel deletion and scattered sequential deletion. For these operations there have been designed algorithms for construction of finite transducers performing these operations on given languages.

2 NEW LANGUAGE OPERATIONS

2.1 SEQUENTIAL INSERTION

The result of sequential insertion of string v into string u is a set of strings u, which have in any place inserted the string v. This operation can be generalized to sequential insertion on languages. We obtain the result of sequential insertion of language L_2 into language L_1 by sequentially inserting every string from L_2 into every string in L_1.

EXAMPLE:

$u = cd$, $v = a$

$u \leftarrow v = \{acd, cad, cda\}$

2.2 PARALLEL INSERTION

The parallel insertion of a language L_2 into a string u is a set of strings obtained after inserting strings from L_2 between all symbols of u, before the first symbol and after the last symbol of u. Parallel insertion of language L_2 into language L_1 is the union of sets obtained after parallel inserting L_2 into all strings from L_1.

EXAMPLE:

$L_1 = \{cd\}$, $L_2 = \{a, b\}$

$L_1 \Leftarrow L_2 = \{acada, acadb, acbda, acbdb, bcada, bcadb, bcbda, bcbdb\}$

2.3 SCATTERED SEQUENTIAL INSERTION

Both previous operations have the same property that the inserted string iss inserted in the compact way on one place. But we can also insert the string scattered, so not the whole string but its separate symbols are sparsely inserted. The result of scattered sequential insertion of string v into string u is string u having inserted all symbols of v on arbitrary places respecting their order in v. Scattered sequential insertion of language L_2 into language L_1 is the union of scattered sequential insertion of all strings from L_2 into all strings from L_1.

EXAMPLE:

$L_1 = \{abb\}$, $L_2 = \{cd\}$

$L_1 \leftarrow_s L_2 = \{cdabb, cadbb, cabdb, cabbd, acdbb, acbdb, acbbd, abcdb, abcbd, abbcd\}$

2.4 SEQUENTIAL DELETION

The result of sequential deletion of string v from string u is a set of strings v, from which we have extracted an arbitrary occurrence of the string u. Sequential deletion of language L_2 from language L_1 is the union of sequential deletions of strings from language L_2 from strings from language L_1.

EXAMPLE:

$L_1 = \{abababa, ab, ba^2, aba\}$, $L_2 = \{aba\}$

$L_1 \rightarrow L_2 = \{baba, abba, abab, \varepsilon\}$

We obtain this result as union of the following sets:

$abababa \rightarrow aba = \{baba, abba, abab\}$

$ab \rightarrow aba = \emptyset$

$ba^2 \rightarrow aba = \emptyset$

$aba \rightarrow aba = \{\varepsilon\}$

2.5 PARALLEL DELETION

Parallel deletion of language L_2 from string u erases all the non–overlapping occurrences of strings in L_2 from u. No nonempty string from L_2 can appear between any two occurrences of strings from L_2 to be erased. The result can still contain a string from L_2 as the result of catenation of the remaining pieces. Parallel deletion of language L_2 from language L_1 is obtained by parallel deletion of L_2 from all strings in L_1.

EXAMPLE:

$L_1 = \{ababab a, aababa, abaabaaba\}$, $L_2 = \{aba\}$

$L_1 \Rightarrow L_2 = \{b, abba, aba, aab, \varepsilon\}$

We obtain this result as the union of the following sets:

$$ababab a \Rightarrow \{aba\} = \{b, abba\}$$
$$aababa \Rightarrow \{aba\} = \{aba, aab\}$$
$$abaabaaba \Rightarrow \{aba\} = \{\varepsilon\}$$

2.6 SCATTERED SEQUENTIAL DELETION

Similarly as scattered sequential insertion we can define sequential deletion in a scattered sense. We do not delete the whole substring *v* but all its individual symbols in their order in *v*. Generalized to languages, the result is the union of scattered sequential deletion of all strings from one language from strings of the second language.

EXAMPLE:

$L_1 = \{a^n b^n c^n \mid n \geq 1\}$, $L_2 = \{ab^2c^3\}$

$L_1 \rightarrow_s L_2 = \{a^{n+2} b^{n+1} c^n \mid n \geq 0\}$

3 FINITE TRANSDUCERS

The main contribution of this paper are algorithms constructing finite transducers, which can perform selected operation on strings from a given regular language and therefore translate that language into another language, result of that operation. These algorithms accept two finite automata as their inputs. The first automaton describes language which this operation will be performed on. The second automaton represents regular language which will be either inserted or deleted in a given way according to the selected operation. As the result these algorithms produce the finite transducer translating the first language according to the selected operation and the second language.

All following algorithms expect two deterministic finite automata: $M_1 = (Q_1, \Sigma, P_1, s_1, F_1)$ accepting language L_1, the language, which we will perform the selected operation on, and $M_2 = (Q_2, \Sigma, P_2, s_2, F_2)$ accepting L_2, the language which will be either inserted to or deleted from L_1 depending on the selected operation. The output is always finite transducer $M = (Q, \Sigma, P, s, F)$ translating L_1 to (L_1 operation L_2).

3.1 COPYING TRANSDUCER

If we have finite automaton and want to make a copying transducer from it, we simply change description of each transition in it. If the previous description was *x*, new description will be *x*|*x*. These transducers simply accept the same language as the original automaton and copy input to its output tape.

3.2 GENERATING TRANSDUCER

Making generating transducer from a given finite automaton is also simple. For each transition we change its description *x* to ε|*x*. So this transducer doesn't read any symbol from its input tape but nondeterministically generates strings from the language described by original automaton to its output tape.

3.3 DELETING TRANSDUCER

Deleting transducer only accepts strings from the input tape and doesn't write to output tape. Making a deleting transducer from a given finite automaton lies in changing the descriptions of transitions. Each description in the form *x* we will substitute with *x*|ε.

3.4 EXAMPLES OF FINITE TRANSDUCERS

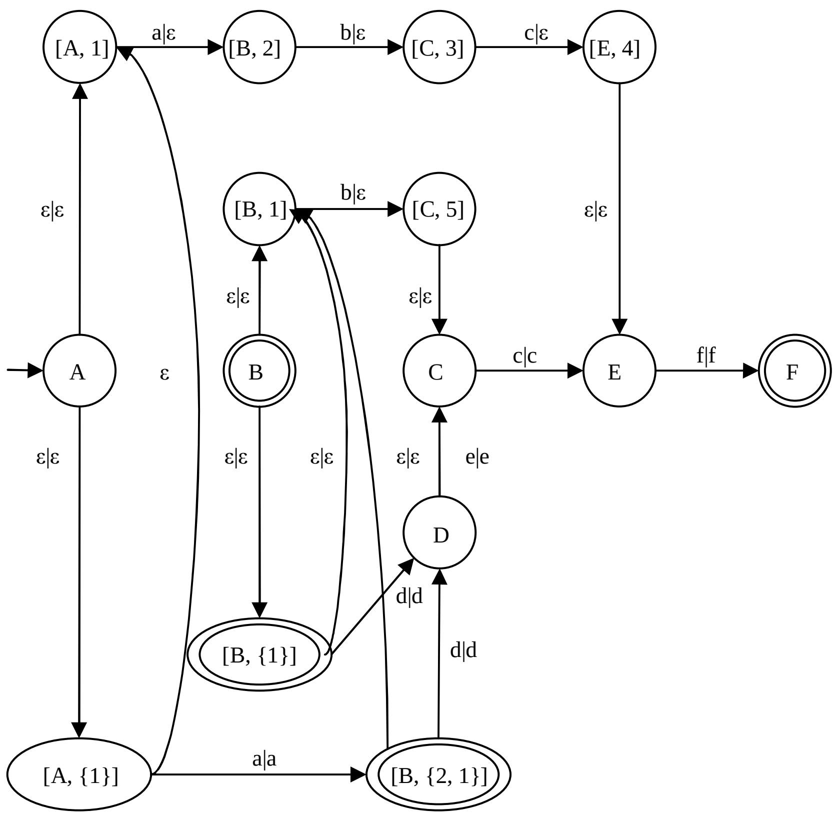

Fig. 1: *Illustration of transducer for parallel deletion*

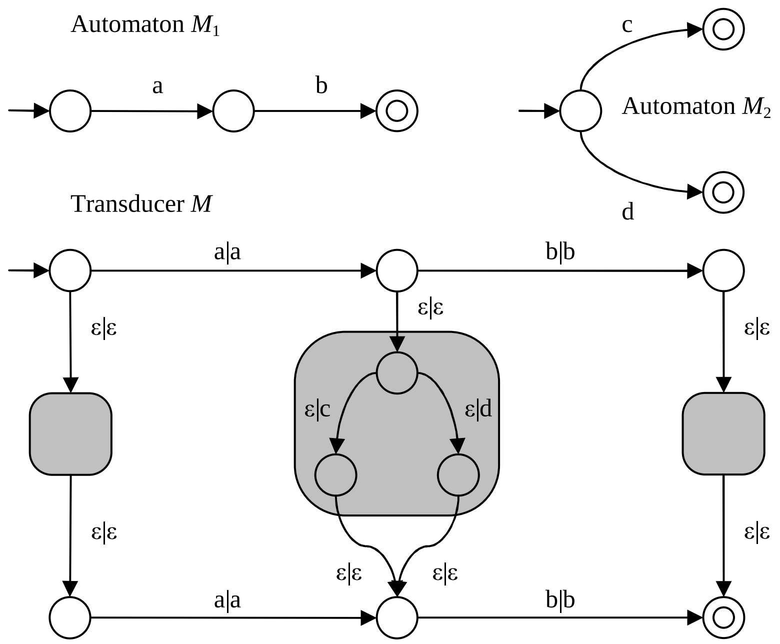

Fig. 2: *Illustration of transducer for sequential insertion*

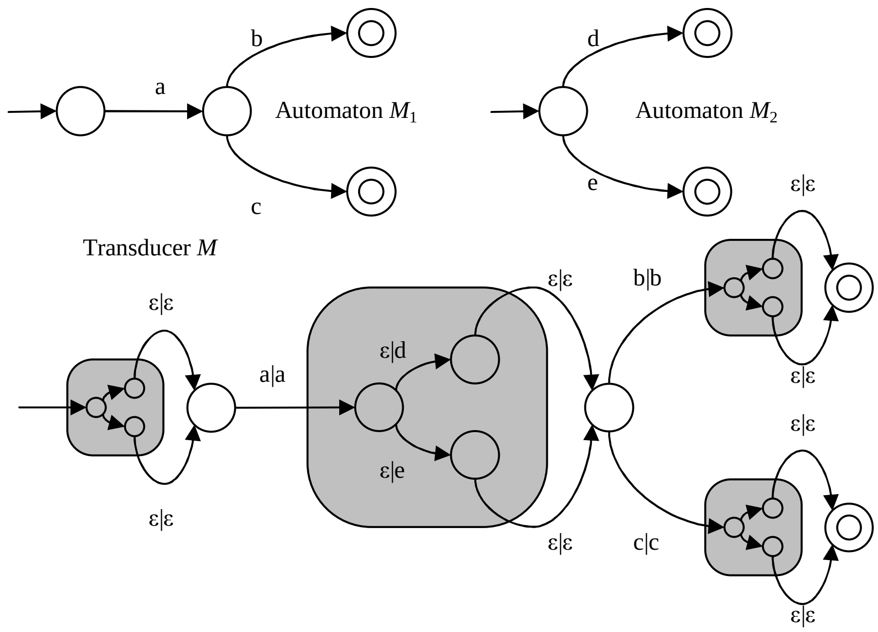

Fig. 3: *Illustration of transducer for parallel insertion*

REFERENCES

[1] Kari, L.: On insertion and deletion in formal languages, Turku, Finland, 1991

[2] Meduna, A., Vítek, M.: New language operations in formal language theory, Schedae Informaticae, vol. 13/2004, Kraków, Poland, ISSN 0860–0295