A Package TESTAS for Checking Some Kinds of Testability

A.N. Trahtman

Bar-Ilan University, Dep. of Math. and St.,
52900,Ramat Gan,Israel
trakht@macs.biu.ac.il

Abstract. We implement a set of procedures for deciding whether or not a language given by its minimal automaton or by its syntactic semigroup is locally testable, right or left locally testable, threshold locally testable, strictly locally testable, or piecewise testable. The bounds on order of local testability of transition graph and order of local testability of transition semigroup are also found. For given k, the k-testability of transition graph is verified. Some new effective polynomial time algorithms are used. These algorithms have been implemented as a C/C^{++} package.

1 Introduction

Locally testable and piecewise testable languages with generalizations are the best known subclasses of star-free languages with wide spectrum of applications.

Membership of a long text in a locally testable language just depends on a scan of short subpatterns of the text. It is best understood in terms of a kind of computational procedure used to classify a two-dimensional image: a window of relatively small size is moved around on the image and a record is made of the various attributes of the image that are detected by what is observed through the window. No record is kept of the order in which the attributes are observed, where each attribute occurs, or how many times it occurs. We say that a class of images is locally testable if a decision about whether a given image belongs to the class can be made simply on the basis of the set of attributes that occur.

Kim, McNaughton and McCloskey have found necessary and sufficient conditions of local testability for the state transition graph Γ of deterministic finite automaton [9]. By considering the cartesian product $\Gamma \times \Gamma$, we modify these necessary and sufficient conditions and the algorithms used in the package are based on this approach.

The locally threshold testable languages were introduced by Beauquier and Pin [1]. These languages generalize the concept of locally testable language and have been studied extensively in recent years.

Right [left] local testability was introduced and studied by König [11] and by Garcia and Ruiz [8]. These papers use different definitions of the conception and we follow here [8]:

A finite semigroup S is right [left] locally testable iff it is locally idempotent and locally satisfies the identity $xyx = xy$ [$xyx = yx$].

J.-M. Champarnaud and D. Maurel (Eds.): CIAA 2002, LNCS 2608, pp. 228–232, 2003.

We introduced polynomial time algorithms for the right [left] local testability problem for transition graph and transition semigroup of the deterministic finite automaton. Polynomial time algorithm verifies transition graph of automaton with locally idempotent transition semigroup.

There are several systems for manipulating automata and semigroups. The list of these systems is following [7] and preprint of [3]:

REGPACK [12] AUTOMATE [5] AMoRE [13] Grail [17] The FIRE Engine [27] LANGAGE [3]. APL package [6]. Froidure and Pin package [7]. Sutner package [20]. Whale Calf [16].

Some algorithms concerning distinct kinds of testability of finite automata were implemented by Caron [3], [4]. His programs verify piecewise testable, locally testable, strictly and strongly locally testable languages.

In our package TESTAS (testability of automata and semigroups), the area of implemented algorithms was essentially extended. We consider important and highly complicated case of locally threshold testable languages [25]. The transition semigroups of automata are studied in our package at the first time [22]. Some algorithms (polynomial and even in some way non-polynomial) check the order of local testability [24]. We implement a new efficient algorithm for piecewise testability improving the time complexity from $O(n^5)$ [3], [19] to $O(n^2)$ [25]. We consider algorithms for right local testability ($O(n^2)$ time and space complexity), for left local testability ($O(n^3)$ time and space complexity) and the corresponding algorithms for transition semigroups ($O(n^2)$ time and space complexity). The graphs of automata with locally idempotent transition semigroup are checked too ($O(n^3)$ time complexity). All algorithms dealing with transition semigroup of automaton have $O(n^2)$ space complexity.

2 Algorithms Used in the Package

Let the integer a denote the size of alphabet and let g be the number of nodes. By n let us denote here the size of the semigroup.

The syntactic characterization of locally threshold testable languages was given by Beauquier and Pin [1]. From their result follow necessary and sufficient conditions of local threshold testability for transition graph of DFA [25] and used in our package a polynomial time algorithm for the local threshold testability problem for transition graph and for transition semigroup of the language.

Let us notice here that the algorithm for transition graph from [25] ([26]) is valid only for complete graph. Of course, the general case can be reduced to the case of complete graph by adding of a sink state. Let us notice also another error from [25] ([26]): in the Theorem 16 (17) in the list of the conditions of local threshold testability, the property that any T_{SCC} is well defined is missed.

The time complexity of the graph algorithm for local threshold testability is $O(ag^5)$. The algorithm is based on consideration of the graphs Γ^2 and Γ^3 and therefore has $O(ag^3)$ space complexity. The time complexity of the semigroup algorithm is $O(n^3)$.

Polynomial time algorithms for the local testability problem for semigroups [22] of order $O(n^2)$ and for graphs [25] of order $O(ag^2)$ are implemented in the package too. We use in our package a polynomial time algorithm of worst case asymptotic cost $O(ag^2)$ for finding the bounds on order of local testability for a given transition graph of the automaton [24] and a polynomial time algorithm of worst case asymptotic cost $O(ag^3)$ for checking the 2-testability [24]. Checking the k-testability for fixed k is polynomial but growing with k. For checking the k-testability [24], we use an algorithm of worst case asymptotic cost $O(g^3a^{k-2})$. The order of the last algorithm is growing with k and so we have non-polynomial algorithm for finding the order of local testability. The algorithms are based on consideration of the graph Γ^2 and have $O(ag^2)$ space complexity. The 1-testability is verified by help of algorithm of cost $O(a^2g)$.

The situation in semigroups is more favorable than in graphs. We implement in our package a polynomial time algorithm of worst case asymptotic cost $O(n^2)$ for finding the order of local testability for a given semigroup [22]. The class of locally testable semigroups coincides with the class of strictly locally testable semigroups [23], whence the same algorithm of cost $O(n^2)$ checks strictly locally testable semigroups.

Stern [19] modified necessary and sufficient conditions of piecewise testability of DFA (Simon [18]) and described a polynomial time algorithm to verify piecewise testability.

We use in our package a polynomial time algorithm to verify piecewise testability of deterministic finite automaton of worst case asymptotic cost $O(ag^2)$ [25]. In comparison, the complexity of Stern's algorithm [19] is $O(ag^5)$. Our algorithm uses $O(ag^2)$ space. We implement also an algorithm to verify piecewise testability of a finite semigroup of cost $O(n^2)$

3 Description of the Package TESTAS

The package includes programs that analyze:

1) an automaton of the language presented as oriented labeled graph;

2) an automaton of the language presented by its syntactic semigroup,

and find

3) the direct product of two semigroups or of two graphs,

4) the syntactic semigroup of an automaton presented by its transition graph.

First two programs are written in C/C^{++} and can by used in WINDOWS environment. The input file may be ordinary txt file. We open source file with transition graph or transition semigroup of the automaton in the standard way and then check different properties of automaton from menu bar. Both graph and semigroup are presented on display by help of rectangular table.

First two numbers in input graph file are the size of alphabet and the number of nodes. Transition graph of the automaton is presented by the matrix:

nodes X labels

where the nodes are presented by integers from 0 to n-1. i-th line of the matrix is a list of successors of i-th node according the label in row. The (i,j) cell contains number of the node from the end of the edge with label from the j-th row and beginning in i-th node. There exists opportunity to define the number of nodes, size of alphabet of edge labels and to change values in the matrix.

The input of semigroup algorithms is Cayley graph of the semigroup presented by the matrix:

elements X generators

where the elements of the semigroup are presented by integers from 0 to $n-1$ with semigroup generators in the beginning. i-th line of the matrix is a list of products of i-th element on all generators.

Set of generators is not necessarily minimal, therefore the multiplication table of the semigroup (Cayley table) is acceptable too. Comments without numerals may be placed in the input file as well.

The program checks local testability, local threshold testability and piecewise testability of syntactic semigroup of the language. Strictly locally testable and strongly locally testable semigroups are verified as well. The level of local testability of syntactic semigroup is also found. Aperiodicity and associative low can be checked too. There exists possibility to change values of products in the matrix of the Cayley graph.

The checking of the algorithms is based in particular on the fact that the considered objects belong to variety and therefore are closed under direct product. Two auxiliary programs written in C that find direct product of two semigroups and of two graphs belong to the package. The input of semigroup program consists of two semigroup presented by their Cayley graph with generators in the beginning of the element list. The result is presented in the same form and the set of generators of the result is placed in the beginning of the list of elements. The number of generators of the result is $n_1g_2+n_2g_1-g_1g_2$ where n_i is the size of the i-th semigroup and g_i is the number of its generators. The components of direct product of graphs are considered as graphs with common alphabet of edge labels. The labels of both graphs are identified according their order. The number of labels is not necessary the same for both graphs, but the result alphabet used only common labels from the beginning of both alphabets. Big size semigroups and graphs can be obtained by help of these programs.

An important verification tool of the package is the possibility to study both transition graph and semigroup of an automaton. The program written in C finds syntactic semigroup from the transition graph of the automaton.

Maximal size of semigroups we consider on standard PC was about some thousands elements. Maximal size of considered graphs was about some hundreds nodes. The program used in such case memory on hard disc and works some minutes.

References

1. D. Beauquier, J.E. Pin, Factors of words, Lect. Notes in Comp. Sci., 372(1989), 63–79.
2. J.A. Brzozowski, I. Simon, Characterizations of locally testable events, Discrete Math. 4(1973), 243–271.

3. P. Caron, LANGAGE: A Maple package for automaton characterization of regular languages, Springer, Lect. Notes in Comp. Sci., 1436(1998), 46–55.
4. P. Caron, Families of locally testable languages, Theoret. Comput. Sci., 242(2000), 361–376.
5. J.M. Camparnaud, G. Hansel, Automate, a computing package for automata and finite semigroups, J. of Symbolic Comput., 12(1991), 197–220.
6. G. Cousineau, J.F. Perrot, J.M. Rifflet, APL programs for direct computation of a finite semigroup, APL Congress 73, Amsterdam, North Holl. Publ., (1973) 67–74.
7. V.Froidure,J.-E. Pin, Algorithms for computing finite semigroups. F. Cucker and M. Shub eds., Foundations of Comp. Math. (1997), 112–126.
8. P. Garcia, Jose Ruiz, Right and left locally testable languages, *Theoret. Comput. Sci.*, **246**(2000), 253–264.
9. S. Kim, R. McNaughton, R. McCloskey, A polynomial time algorithm for the local testability problem of deterministic finite automata, IEEE Trans. Comput., N10, 40(1991) 1087–1093.
10. S. Kim, R. McNaughton, Computing the order of a locally testable automaton, Lect. Notes in Comp. Sci., 560(1991) 186–211.
11. R. König, Reduction algorithm for some classes of aperiodic monoids, R.A.I.R.O. Theor.Inform., 19, **3**(1985), 233–260.
12. E. Leiss, Regpack, an interactive package for regular languages and finite automata, Research report CS-77-32, Univ. of Waterloo, 1977.
13. O. Matz, A. Miller, A. Potthoff, W. Thomas, E.Valkema, Report on the program AMoRE, Inst. inf. und pract. math., Christian-Albrecht Univ. Kiel, 1995.
14. R. McNaughton, Algebraic decision procedure for local testability, Math. Syst. Theory, 8(1974), 60–76.
15. R. McNaughton, S. Papert, Counter-free automata, M.I.T. Press Mass., (1971).
16. A. Okhotin, Whale Calf, a parser generator for conjunctive grammars. 7-th Int. Conf. on Impl. and Appl. of Automata, CIAA2002, Tours, 2002, 211–216.
17. D. Raymond, D. Wood, Grail, a $C^{+}+$ library for automata and expressions, J. of Symb. Comp., 17(1994) 341–350.
18. I. Simon, Piecewise testable events, Lect. Notes in Comp. Sci., 33(1975), 214–222.
19. J. Stern, Complexity of some problems from the theory of automata. Inf. and Control, 66(1985), 163–176.
20. K. Sutner, Finite State Mashines and Syntactic Semigroups, The Mathematica J., 2(1991), 78–87.
21. A.N. Trahtman, The varieties of testable semigroups. Semigroup Forum, 27, (1983), 309–318.
22. A.N. Trahtman, A polynomial time algorithm for local testability and its level. Int. J. of Algebra and Comp., vol. 9, 1(1998), 31–39.
23. A.N. Trahtman, Identities of locally testable semigroups. Comm. in Algebra, v. 27, 11(1999), 5405–5412.
24. A.N. Trahtman, Algorithms finding the order of local testability of deterministic finite automaton and estimation of the order, Th. Comp. Sci., 235(2000), 183–204.
25. A.N. Trahtman, Piecewise and local threshold testability of DFA. Lect. Notes in Comp. Sci., 2138(2001), 347–358.
26. A.N. Trahtman, An algorithm to verify local threshold testability of deterministic finite automata. Lect. Notes in Comp. Sci., 2214(2001), 164–173.
27. B.W. Watson, The design and implementation of the FIRE Engine: A $C^{+}+$ toolkit for Finite Automata and Regular Expressions, Comp. Sci. Rep. 94722, Endhoven Univ. of Techn. 1994.