On the equivalence problem for deterministic multitape automata and transducers *

Karel Culik II
Dept.of Computer Science
University of South Carolina
Columbia, SC 29208, USA

Juhani Karhumäki
Department of Mathematics
University of Turku
Turku, Finland

Abstract

We discuss the technique for testing the equivalence of two deterministic automata by constructing a language that matches the computations of two equivalent automata on the same input word. Specifically, we propose to use HDTOL languages that are powerful enough to match computations of many equivalent deterministic multitape automata, and at the same time, have nice decidable properties. Using this new technique of HDTOL matching, we show that the inclusion problem between an arbitrary deterministic multitape automaton and a simple one is decidable in both directions. Further, we show that the computations of two simple automata can be HDTOL matched on their common domain. This implies that the equivalence problem for transducers based on simple automata is decidable. The latter result is the best possible since the problem is undecidable even for transducers based on automata with parallel loops.

1 Introduction

One of the basic techniques to attack the equivalence problem of deterministic automata (of certain type) is to try to match equivalent computations, i.e. computations of the same words, of given two automata via another, potentially more powerful, device. In other words, one tries to find a language such that its words simultaneously encode the

*Reported research was supported by the National Sciences Foundation Grant CCR-8702752.

computations of words of these given two automata. If this is achieved, then the test for equivalence is reduced to some decidability problem, most frequently to the emptiness problem, of the family used in the matching. A typical example of this approach is Valiant's solution of the equivalence problem for deterministic finite turn push-down automata; his matching device is a nondeterministic push-down automaton, for which the emptiness is decidable while the equivalence is not [15].

Actually, the matching used by Valiant is not as strict as we required above, since in his approach the equivalence of computations is defined by a coarser equivalence relation than the equality. If the matching is as strict as we require, then not only the equivalence problem for the considered automata, but also that for the transducers based on these automata, becomes decidable in many cases. Indeed, the latter becomes a problem of testing the equivalence of morphisms (word by word) on the matching language. This problem was introduced in [9] and has been studied extensively thereafter. Here we need to test morphisms on HDTOL languages. This problem was open for a number of years and finally shown decidable in [5]. Morphisms can be tested, although it is not easy at all, on HDTOL languages.

The above approach works extremely nicely for deterministic one-tape finite automata. Indeed, given two such automata $\mathcal{A}_1$ and $\mathcal{A}_2$, there exists a third one, say $\mathcal{A}$, which matches the computations of $\mathcal{A}_1$ and $\mathcal{A}_2$ for words in $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Consequently, the matching device is not more powerful than the original ones and therefore this method provides an elegant proof of the decidability of the equivalence problem for deterministic finite transducers. However, the situation changes drastically when we move from one-tape case to multitape case. It is well known that the intersection of relations defined by deterministic multitape automata need not be definable by such an automaton, and in fact the emptiness of such intersections is undecidable. On the other hand, since our basic interest is in the equivalence problem for deterministic multitape automata, it would be enough if accepting computations of two equivalent automata could be matched by some simple enough device. However, even this seems to be difficult since two equivalent deterministic multitape automata may carry out computations of the same n-tuple in different order and hence standard methods of "sequential formal language" theory seems to be useless. This motivated us to use the theory of parallel rewriting systems, that is L systems, cf. [14], since such systems have a power to operate simultaneously in different parts of a word under rewriting.

We shall be dealing with n-tape finite automata introduced in [13]. We call such an automaton simple if all of its nodes are involved in at most one loop, and show that for two simple automata the intersection of the relations defined by them can be described effectively by an HDTOL language, cf. [14], or more precisely the intersection is an HDTOL relation. This certainly does not hold for n-tape automata in general. Indeed, even for automata with only parallel, but not nested loops the emptiness of the intersection is undecidable.

An important notion for this paper is that of an HDTOL relation, introduced implicitly in [4] and later used in [6]. A relation $R \subseteq \Sigma^\star \times \Delta^\star$ is an HDTOL relation iff it can be expressed in the form $R = \{(h(x), g(x)) \mid x \in L\}$ for some HDTOL language L and two morphisms h and g. For our current purposes we generalize this notion to n dimensions. This notion is strongly motivated by Nivat Theorem, cf. [1], characterizing rational relations.

From the point of view of decision problems HDTOL relations became a particularly powerful tool when it was shown that morphisms can be tested on HDTOL languages, cf. [5]. An example of the use of this tool is shown here. As an application of our "intersection theorem" we generalize the known result that the equivalence of two simple deterministic n-tape automata is decidable, cf. [12] or [11], to the following: It is decidable whether two n-tape finite transducers based on simple deterministic n-tape automata are equivalent on their common domain. The proof uses an idea of "HDTOL matching" of computations of two n-tape automata developed in [7].

Further, using a different proof technique, we show that the inclusion problem in both directions is decidable between arbitrary deterministic n-tape automaton and a simple one. Recall that the inclusion problem for deterministic simple n-tape automata has been shown decidable (in a slightly more general form) in [11], while the equivalence problem for simple (not necessarily deterministic) n-tape automata has been solved already in [12]. Regarding general 2-tape automata the equivalence is decidable, as first shown by [2], while the inclusion is undecidable, cf. [1].

2 Preliminaries

We assume that the reader is familiar with basics of automata theory and the theory of semilinear sets of vectors, cf. [10]. In addition we list [14] as the reference dealing with HDTOL languages.

An *n-tape automaton* over disjoint alphabets Σ_i, for $i = 1, \ldots, n$, is a quadruple

$$\mathcal{A} = (Q, T, q_0, F), \tag{1}$$

where Q is a finite set of states, $T \subseteq Q \times \bigcup_{i=1}^{n}(\Sigma_i \cup \#_i) \times Q$ is a finite transition relation, q_0 is the initial state, and $F \subseteq Q$ is the set of final states. (Here $\#_i$'s denote the endmarker of a word in $\Sigma_i^\star$). The automaton $\mathcal{A}$ *accepts* an n-tuple $(w_1, \ldots, w_n)$ of words in $\Sigma_1^\star \times \ldots \times \Sigma_n^\star$ iff $\mathcal{A}$ as a one-tape automaton accepts at least one word from the set $w_1\#_1 ⧢ w_2\#_2 \ldots ⧢ w_n\#_n$ where ⧢ denotes the shuffle operation. The set of all n-tuples accepted by $\mathcal{A}$ is denoted by $R(\mathcal{A})$. An n-tuple automaton $\mathcal{A}$ of (1) is *deterministic* iff Q and T can be partitioned as follows:

$$Q = \bigcup_{i=1}^{n} Q_i \text{ with } Q_i \cap Q_j = \emptyset \text{ for } i \neq j \tag{2}$$

and

$$T = \bigcup_{i=1}^{n} T_i, \text{ where } T_i = T \cap Q_i \times (\Sigma_i \cup \{\#_i\}) \times Q, \tag{3}$$

and $\text{card}(T_i \cap \{q\} \times \{a\} \times Q) \leq 1$ for $i = 1, \ldots, n, q \in T_i$ and $a \in \Sigma_i \cup \{\#_i\}$. Conditions (2) and (3) mean that the state of $\mathcal{A}$ uniquely determines on which tape automaton can next read. In addition, if not only the current state but also the symbol to be read is fixed, then the next state is uniquely determined. It follows that if $\mathcal{A}$ is deterministic, then for each n-tuple $(w_1, \ldots, w_n) \in \Sigma_1^\star \times \Sigma_2^\star \times \ldots \times \Sigma_n^\star$ there exists at most one accepting computation of $\mathcal{A}$. It also follows that the requirement that Σ_i's are disjoint becomes superfluous in the case of deterministic n-tape automata.

An n-tape automaton $\mathcal{A}$ is *simple* if each of its states is involved in at most one loop of $\mathcal{A}$ (viewed as a one-tape automaton). Obviously, final states of a simple automaton can be specified as states having no exists. A *simple branch* is a simple n-state automaton having only one final state from which there are no transitions. We say that an automaton contains *only parallel* loops if in any loop there exists at most one state which is involved in another loop.

By associating outputs to transitions of an n-tape automaton we can talk about *n-tape transducers* (of certain type). Here the outputs are over a single tape, but this is not an important restriction.

Next we define our central notion of an HDTOL relations, cf. [4] and [6]. An *HDTOL* *language* L over Σ is a language of the form

$$L = g(\bigcup_{i=1}^{\infty} L_i)$$

where

$$L_0 = \{w\}, \; L_{i+1} = \bigcup_{j=1}^{t} h_j(L_i) \quad (i \geq 1)$$

for a word $w \in \Delta^\star$, morphisms $h_j : \Delta^\star \to \Delta^\star (j = 1, \ldots, t)$, $g : \Delta^\star \to \Sigma^\star$ and an integer $t \geq 1$. Let $\Sigma_1, \ldots, \Sigma_n$ be finite (not necessarily disjoint) alphabets. A relation $\rho \subseteq \Sigma_1^\star \times \ldots \times \Sigma_n^\star$ is called an *HDTOL relation* iff there exists an alphabet Γ, an HDTOL language L over Γ and morphisms $g_i : \Gamma^\star \to \Sigma_i^\star$ such that

$$\rho = \{(g_1(w), \ldots, g_n(w)) \mid w \in L\}. \tag{4}$$

The definition of an HDTOL relations is motivated by Nivat representation of rational relations, cf. [1]. They also provide a generalization of rational relations since, as it is easy to see, each regular language is an HDTOL language. HDTOL relations have a couple of interesting properties which we believe make them important and useful family of relations. First of all, they are purely morphically defined. Secondly, the underlying family of languages has desirable properties, such as that the emptiness is decidable and that morphisms can be tested on these languages, cf. [14] and [6], respectively.

Finally, we recall the definition of a semilinear set. Let $I\!N$ denote the set of nonnegative integers. A set K is *linear* iff it is a subset of $I\!N^k$, for some $k \geq 1$, and of the form

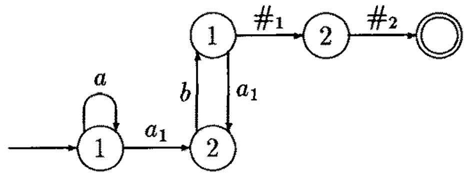
$$K = \{x_0 + \Sigma_{i=1}^{N} n_i x_i \mid n_i \geq 0\} \text{ for } N \geq 0 \text{ and } x_0, x_1, \ldots, x_N \text{ in } I\!N^k.$$

Further, K is *semilinear* iff it is a finite union of linear sets.

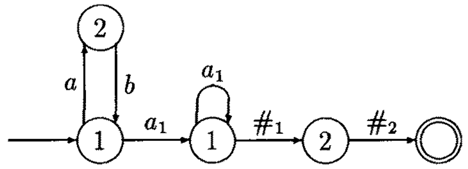
3 Examples and results

In this section we consider simple n-tape automata and show that they possess some desirable properties general n-tape automata do not have. First we recall that the intersection of two relations defined by deterministic n-tape automata need not to be realized by such an automaton. This is seen, for example, by considering the following automata:

$\mathcal{A}_1$:



$\mathcal{A}_2$:



Here a and a_1 are read from the first tape and b from the second. Clearly,

$$R = R(\mathcal{A}_1) \cap R(\mathcal{A}_2) = \{(a^n a_1^n, b^n) \mid n \geq 1\}$$

and hence it cannot be accepted by a 2-tape automaton.

In the above example the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are simple, and still the intersection is not a rational relation. However, we are going to show that such a intersection is always an HDTOL relation.

Our proof is based on the fact that for simple automata a computation is uniquely specified by stating the number of repetitions of each cycle. This allows us to show that there is a finitary matching of cycles for the computations in the common domain and therefore an HDTOL matching on this domain. For a detail proof of the following theorem see [7].

Theorem 1 *For two simple n-tape automata $\mathcal{A}_1$ and $\mathcal{A}_2$ the intersection $R(\mathcal{A}_1) \cap R(\mathcal{A}_2)$ is an HDTOL relation. Moreover, its HDTOL representation can effectively be found.*

Theorem 1 and its proof techniques provide a few interesting corollaries. First we obtain new proofs for the equivalence and inclusion problems for simple not necessarily deterministic n-tape automata, cf. [12] or [11]. For detailed proofs see [7].

Corollary 1 *The equivalence and inclusion problems for simple n-tape automata are decidable.*

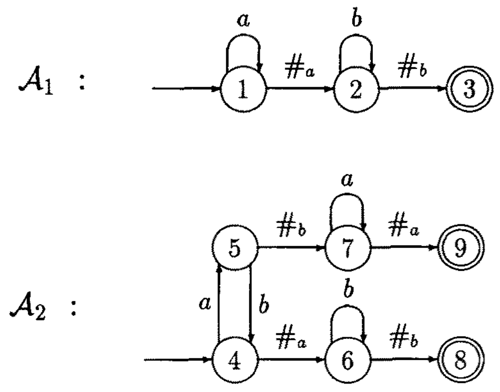


Figure 1:

Corollary 2 *It is decidable whether two simple n-tape transducers are equivalent on their common domain.*

It is easy to see that Theorem 1 cannot be extended to arbitrary n-tape automata. In fact the problem becomes undecidable even when parallel, but not nested loops are allowed, see [7].

Example 1. Consider the 2-tape automata defined by the state diagrams shown in Figure 1 where a and b are over different tapes. Clearly, $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent and accept every pair of words in $a^\star \times b^\star$. Let $E_1 = \{a\#_a, b, \#_b\}$ denote transitions of $\mathcal{A}_1$ and $E_2 = \{\bar{a}_1, \bar{b}_1, \bar{a}_2, \bar{b}_2, \bar{\#}_{a,1}, \bar{\#}_{b,1}, \bar{\#}_{a,2}, \bar{\#}_{b,2}\}$ those of $\mathcal{A}_2$, where smaller index of a letter corresponds to transition labeled by this letter and connected to states with smaller numbers. Now, the computations of $\mathcal{A}_1$ and $\mathcal{A}_2$ are HDTOL matched as follows. Consider a DTOL system G having S as the initial word and containing tables (morphisms)

$$\begin{array}{lll}
T_1 : & S \to X_1X_2X_4 & \\
T_2 : & X_1 \to aX_1, & X_4 \to \bar{a}_1X_5 \\
T_3 : & X_1 \to \#_a, & X_4 \to \bar{\#}_{a,1}X_6 \\
T_4 : & X_2 \to bX_2, & X_5 \to \bar{b}_1X_4 \\
T_5 : & X_2 \to \#_b, & X_5 \to \bar{\#}_{b,1}X_2 \\
T_6 : & X_2 \to bX_2, & X_6 \to \bar{b}_2X_6 \\
T_7 : & X_2 \to \#_b, & X_6 \to \bar{\#}_{b,2} \\
T_8 : & X_1 \to aX_1, & X_7 \to \bar{a}_2X_7 \\
T_9 : & X_1 \to \#_a, & X_7 \to \bar{\#}_{a,2}
\end{array}$$

Observe that we have specified only nonindentity productions in each table. Clearly,

$$L(G) \cap (E_1 \cup E_2)^\star.$$

matches the computations of $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same pair of inputs, and it is an HDTOL language by the closure properties of this family.

In the above matching we followed computations of $\mathcal{A}_2$ from left to right and matched always a transition with a suitable one in $\mathcal{A}_1$. Of course, these suitable ones are not from left to right order in $\mathcal{A}_2$: This is exactly where DTOL systems come into the play.

The above HDTOL matching is by no means the only possible. Another is obtained by matching "corresponding cycles", cf. T_3, T_4 and T_5 below, instead of corresponding transitions. More precisely, define a DTOL system G_1 having S_1 as the initial word and containing tables

$$\begin{array}{ll}
T_1 : & S_1 \to aX_1\#_aX_2\#_bX_3\bar{a}_1\bar{\#}_{b,1}X_4\bar{\#}_{a,2} \\
T_2 : & S_1 \to X_1\#_aX_2\#_bX_3\bar{\#}_{a,1}X_5\bar{\#}_{b,2} \\
T_3 : & X_1 \to aX_1,\ X_2 \to bX_2,\ X_3 \to \bar{a}_1\bar{b}_1X_3 \\
T_4 : & X_1 \to aX_1,\ X_4 \to \bar{a}_2X_4 \\
T_5 : & X_2 \to bX_2,\ X_5 \to \bar{b}_2X_5 \\
T_6 : & X_1 \to 1,\ X_2 \to 1,\ X_3 \to 1,\ X_4 \to 1,\ X_5 \to 1
\end{array}$$

Again it is clear that

$$L(G_1) \cap (E_1 \cup E_2)^\star$$

provides the required HDTOL matching.

In this paper we are mainly using the first approach of the above example, that is we are matching computations transition by transition. In [7] it has been shown that the common accepting computations of two simple n-tape automata can be HDTOL matched using our second approach of matching the corresponding cycles. Either of these approaches is flexible to number of variations. For example, as it is done below, instead of matching computations one might want to match the labels of the computations, i.e. the corresponding input words.

Example 2. Consider 3-tape automata $\mathcal{A}_1$ and $\mathcal{A}_2$ defined by the state diagrams of Figure 2.

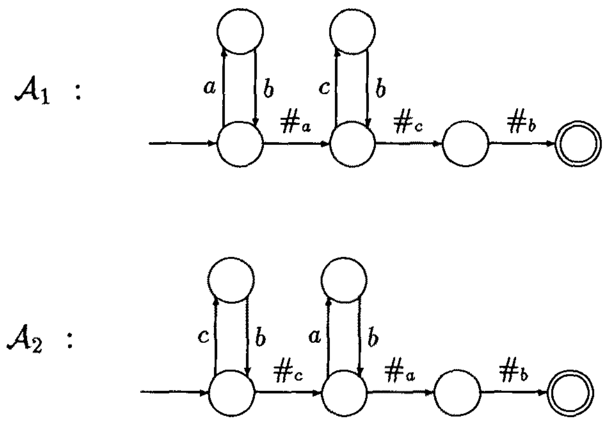


Figure 2:

Here each of a, b and c is on a different tape. Clearly,

$$R(\mathcal{A}_1) = R(\mathcal{A}_2) = \{(a^n, b^m, c^k) \mid n + k = m\},$$

so that $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent. Moreover, there exists an obvious correspondence between loops in $\mathcal{A}_1$ and $\mathcal{A}_2$: The first loop in $\mathcal{A}_1$ corresponds to the second of $\mathcal{A}_2$ and vice versa. This provides a guidance how computations must be matched cycle by cycle. However, here we have to match labels of computations instead of transitions. In order to do this define a DTOL system G with S as the starting word and with the tables

$$\begin{aligned} T_1 &: \quad S \to A\#_a C\#_c\#_b \bar{C}\bar{\#}_c \bar{A}\bar{\#}_a\bar{\#}_b \\ T_2 &: \quad A \to abA,\ \bar{A} \to \bar{a}\bar{b}\bar{A} \\ T_3 &: \quad C \to cbC,\ \bar{C} \to \bar{c}\bar{b}\bar{C} \\ T_4 &: \quad A \to 1,\ \bar{A} \to 1,\ C \to 1,\ \bar{C} \to 1 \end{aligned}$$

Now

$$L(G_1) \cap \{a, b, c, \#_a, \#_b, \#_c, \bar{a}, \bar{b}, \bar{c}, \bar{\#}_a, \bar{\#}_b, \bar{\#}_c\}^*$$

HDTOL matches labels of corresponding computations of $\mathcal{A}_1$ and $\mathcal{A}_2$. We note that Example 2. can be used to show that Bird's algorithm for testing the equivalence of two-tape deterministic automata, cf. [2], cannot be extended in a natural way to higher dimensions.

The following result is shown using a different proof technique based again on HDTOL matching. For a detailed proof of the following result see [8].

Theorem 2 *Given a deterministic n-tape automaton $\mathcal{A}$ and deterministic simple n-tape automaton $\mathcal{B}$ it is decidable whether*

(i) $R(\mathcal{A}) \subseteq R(\mathcal{B})$ *and*

(ii) $R(\mathcal{B}) \subseteq R(\mathcal{A})$.

Proof idea. Because we use both accepting and nonaccepting computations we need the completeness of automata. Clearly, we can assume that $\mathcal{A}$ is complete. In order to make the simple automaton $\mathcal{B}$ complete we have to relax a bit our requirements put for simple automata. We say that a branch (i.e. n-tape automata with one final state) is *almost simple* if it is of the form a simple branch followed by a finite number of parallel-loop diagrams of the form shown in Figure 3. Clearly, each simple n-tape automaton

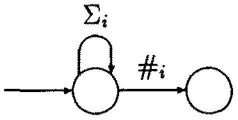


Figure 3:

can be completed in such a way that it is composed of a finite number of almost simple branches.

Now, we decompose the simple automaton $\mathcal{B}$ into accepting and nonaccepting (almost) simple branches. Then in order to test inclusion (i) we test whether or not

$$R(\mathcal{A}) \cap R(\beta) = \emptyset \tag{5}$$

for each fixed nonaccepting branch of $\mathcal{B}$.

Now, we use again our technique of HDTOL matching. We use it essentially as in the first part of Example 1, that is we follow the computations of $\mathcal{A}$ and HDTOL match the transitions of $\mathcal{A}$ with "corresponding" ones of β. However, we are able to do the matching only in such a way that we obtain not actual but "potential computations" of β, and it turns out that some of these potential computations is an actual one iff the relation (5) fails. This, in turn, is checked by testing the emptiness of two semilinear sets.

The proof of (ii) is completely analogous when we consider nonaccepting computations of $\mathcal{A}$ and accepting computations of $\mathcal{B}$. □

Our Theorem 2 should be compared to that of [11] where it is shown that inclusion problem is decidable for a class of deterministic n-tape automata. This class of automata which are generalizations of simple automata in the sense if loops are only over one tape then they can be arbitrary, otherwise the simplicity is required.

It is also of interest to note that in the above proof of (i) we do not need the determinism of $\mathcal{A}$. Consequently, we have

Corollary 3 *Given an n-tape automaton $\mathcal{A}$ and deterministic simple automaton $\mathcal{B}$ it is decidable whether $R(\mathcal{A}) \subseteq R(\mathcal{B})$.*

On the other hand, in part(ii) of Theorem 2 we need the determinism of $\mathcal{A}$. Indeed, here we deal with nonaccepting computations of $\mathcal{A}$, and in a nondeterministic automaton the existence of a nonaccepting computation of an n-tuple does not guarantee that the n-tuple is not accepted. In fact, for an arbitrary $\mathcal{A}$, the universality problem which is undecidable, cf. [1], is a special case of the question whether

$$R(\mathcal{B}) \subseteq R(\mathcal{A})$$

for an almost simple $\mathcal{B}$.

References

[1] J. Berstel, *Transductions and Context-Free Languages*, Teubner, Stuttgart, 1979.

[2] M. Bird, The equivalence problem for deterministic two-tape automata, *J. Comput. Sci.* **7** (1973) 218 - 236.

[3] K. Culik II, A purely homomorphic characterization of recursively enumerable sets, *J. Assoc. Comput. Mach.* **6** (1979) 345 - 350.

[4] K. Culik II and J. Karhumäki, Systems of equations over a free monoid and Ehrenfeucht's conjecture, *Discrete Math.* **43** (1983) 139 - 153.

[5] K. Culik II and J. Karhumäki, The equivalence of finite valued transducers (on HDTOL languages) is decidable, *Theoret. Comput. Sci.* **47** (1986) 71 - 84.

[6] K. Culik II and J. Karhumäki, Systems of equations over a finitely generated free monoid having an effectively findable equivalent finite subsystem, in: *H.Jurgensen, G.Lallement and H.Weinert (eds), Semigroups Theory and Applications*, Oberwolfach 1986, *Lecture Notes in Mathematics* **1320**, pp. 18-27 (Springer, 1988).

[7] K. Culik II and J. Karhumäki, Loops in automata and HDTOL relations, Technical Report TR88003, USC (1988).

[8] K. Culik II and J. Karhumäki, HDTOL matching of computations of multitape automata, Technical Report TR88004, USC (1988).

[9] K. Culik II and A. Salomaa, On the decidability of morphic equivalence for languages, *J. Comput. System Sci.* **17** (1978) 163 - 175.

[10] M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, 1978.

[11] E. Kinber, The inclusion problem for some classes of deterministic multitape automata, *Theoret. Comput. Sci.* **26** (1983) 1 - 24.

[12] H. R. Lewis, A new decidability problem with applications, *Proceedings of 18th FOCS Conference* (1979) 62 - 73.

[13] M. Rabin and D. Scott, Finite automata and their decision problems, *IBM J. Res. Develop.* **3** (1959) 114 - 125.

[14] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[15] L. G. Valiant, The equivalence problem for deterministic finite-turn pushdown automata, *Inform. Control* **25** (1974) 123 - 133.