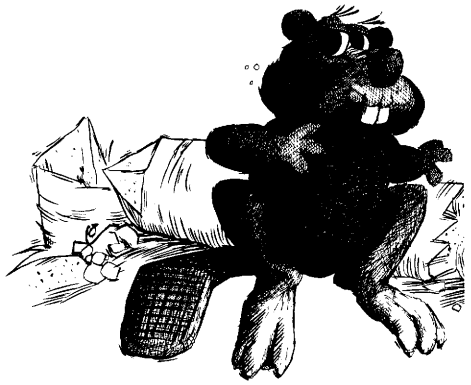
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



The Equivalence Problem For Singled-Valued Two-Way Transducers Is Decidable

Karel Culik II
Juhani Karhumäki

CS-85-24

August, 1985

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

THE EQUIVALENCE PROBLEM FOR SINGLE-VALUED TWO-WAY TRANSDUCERS IS DECIDABLE*

Karel Culik II

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Juhani Karhumäki†

Department of Mathematics
University of Turku
Turku, Finland

ABSTRACT

The equivalence problem for deterministic two-way transducers was a longstanding problem finally shown decidable by E.M. Gurari. We show a generalization of this result, namely that it is decidable whether two single-valued two-way transducers are equivalent on an NPDTOL language. We obtain this result by showing that it is even decidable whether a two-way transducer is single-valued on an NPDTOL language. Our solution is based on the compactness property of the systems of equations over a finitely generated free monoid shown recently by J. Lawrence and M. Albert. This was a longstanding open problem known as the Ehrenfeucht Conjecture. Our approach also shows that for one-way transducers their single-valuedness on a given HDTOL language can be tested, and thus the equivalence of two single-valued one-way transducers on an HDTOL language is decidable, too.

Our results imply the decidability of the HDOL and HDTOL sequence equivalence problems, also longstanding open problems solved recently by K. Ruohonen and the authors, respectively.

Key words: equivalence problem, two-way transducer, single-valued transducer, unambiguous transducer, Ehrenfeucht Conjecture, DTOL language.

Abbreviated title: Single-valued two-way transducers.

---

* This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. A-7403.

† This work was done during the second author's visit at the University of Waterloo.

1. Introduction

The equivalence problem for deterministic two-way transducers was a long-standing open problem that was finally shown decidable in [8]. Our main result is a generalization of this problem. It is generalized in two directions, single-valued rather than deterministic two-way transducers are considered and their comparison is restricted to NPDTOL languages. Especially the latter extension is substantial since even a much simpler problem, namely that of testing equivalence of morphisms on PDOL languages, which is only a special case of the HDOL sequence equivalence problem, was open until recently [15], [5]. Also the decidability of the HDTOL sequence equivalence problem, shown recently in [5], follows as a special case of our results.

We demonstrate the decidability of the equivalence problem by showing that it is even decidable whether a given two-way transducer is single-valued on an NPDTOL language. Our proof uses some properties of NPDTOL languages and most importantly a recent deep algebraic result, the validity of the Ehrenfeucht Conjecture [1], a problem which originated in formal language theory [6] and [11].

Let L be an arbitrary language over Σ. We say that a finite subset F of L is a *test set* (with respect to morphisms) for L if whenever two morphisms agree on F they agree on L as well. The *Ehrenfeucht Conjecture* states that each language possesses a test set. In [4] it was shown that the Ehrenfeucht Conjecture can be equivalently stated in algebraic terms as follows. Every infinite system of equations, with a finite number of unknowns, over a free monoid has an equivalent finite subsystem. The validity of the Ehrenfeucht Conjecture, in the latter form, was established in [1]. We will use this algebraic form to demonstrate the validity of a generalization of the former version, namely that each language possesses a finite test set with respect to two-way single-valued transducers with a bounded number of states (Theorem 4).

Then we show that a finite test set exists effectively if L is an NPDTOL language [14]. This step relies on the authors' earlier result that the equivalence problem for systems of equations, with a finite number of unknowns, over a free monoid is decidable, which, in turn, is based on the deep result of Makanin [12] that the solvability of such systems is decidable.

Clearly, the effective existence of a test set for NPDTOL languages, with respect to single-valued two-way transducers with a bounded number of states, implies that it is decidable whether single-valued two-way transducers agree on an NPDTOL language. Since the domain of every two-way transducer is a regular set and since the family of regular sets is included in the family of NPDTOL languages we have as a corollary the result in the title of this paper.

Now, a natural question is whether the property of single-valuedness (on an NPDTOL language) itself is effectively testable. The answer is affirmative and to demonstrate it we use a minor modification of the arguments outlined above for the solution of the equivalence problem. Actually, it is easy to see that the decidability of the single-valuedness (on an NPDTOL language) implies the decidability of the equivalence of single-valued transducers (on an NPDTOL language). Hence, in the following we prove the decidability of the former. However, here in the introduction we outlined the direct proof of the decidability of the equivalence problem because it seemed more understandable.

A two-way transducer is called unambiguous if it is single-valued and its underlying acceptor is unambiguous. The unambiguous two-way transducers are properly between the deterministic and the single-valued ones. We show that it is decidable whether a two-way transducer is unambiguous. Our test for unambiguity is much simpler than that for single-valuedness.

The decidability of the equivalence and single-valuedness problems for one-way transducers (on the whole domain) was shown in [16] and later independently in [2]. A generalization of these results, namely the decidability of the equivalence or single-valuedness for one-way transducer on a context-free

language was shown in [3]. The same problems for HDTOL languages, or even DOL languages, were open, however their decidabilities follow from our results. Indeed, if instead of two-way transducers one-way transducers are considered, then our above decidability results hold also for the family of HDTOL languages, and not only for the family of NPDTOL languages. For example, it is decidable whether a given one-way transducer is single-valued on a given HDTOL language (Theorem 11).

We note that the equivalence problem for the nondeterministic transducers is undecidable, this holds even if only ϵ-free one-way transducers (gsm's) with one-letter input (or output) alphabet are considered [10]. We also note that the single-valuedness problem for push-down transducers is undecidable. This problem can be easily reduced to the problem of deciding the emptiness of the intersection of two context-free languages.

2. Preliminaries

A *two-way* (*sequential*) *transducer* (*twt*) is a 5-tuple $M = (K, \Sigma, \Delta, H, s_0)$ where K, Σ, and Δ are finite nonempty sets of states, inputs and outputs, respectively, s_0 in K is the initial state and H is a finite subset of $K \times \Sigma^2 \times \Delta^* \times K \times \{-1,0,1\}$.

Our definition differs from the usual one (cf. [7]) in the sense that our reading head sees simultaneously two symbols of the input tape. This modification is introduced to facilitate the proof of Lemma 3 but does not effect the decidability results since for a *twt* in the usual form it is easy to construct an equivalent one in our form, and vice versa (ignoring the short words). As in [7] we do not have final states or endmarkers, however, as far as our problems are concerned the model with final states and/or endmarkers is clearly equivalent. We also assume, without loss of generality, that the initial state s_0 is not entered in any

computation. Informally a *twt* functions as follows. It starts by reading the leftmost two tape symbols while in state s_0, and a transducer terminates when the *twt* moves right from the rightmost tape symbol to the blank tape where no further moves are defined, i.e. where the reading head sees a blank on the right.

We will describe a *configuration* of a *twt* by a string $xapby$ where $x, y \in \Sigma^*$, $a, b \in \Sigma$, and $p \in K$, indicating that the finite control is in state p and the reading head above symbols a and b (to avoid defining H on "blanks" we assume that the input word is at least of length two). When $(p, ab, w, q, k) \in H$ then M in state p may read ab, write w, transit to state q and move k squares right on the input tape. Thus M may go from configuration $xapby$ into configuration $xqaby$, $xaqby$ or $xabqy$ for $k = -1, 0, 1$, respectively.

The input-output relation realized by a *twt* M is called two-way transduction and is denoted by $\tau(M)$. Two *twt*'s are *equivalent* if they define the same transduction. A *twt* M is *single-valued* if $(x, y) \in \tau(M)$ and $(x, z) \in \tau(M)$ implies $y = z$.

A *DTOL system* (resp. *PDTOL system*) is a tuple $S = (\Sigma, h_1, \ldots, h_n, w)$ where Σ is a finite alphabet, $h_i : \Sigma^* \to \Sigma^*$ is a morphism (resp. ϵ-free morphism) for $i = 1, \ldots, n$ and $w \in \Sigma^+$. The language generated by S is denoted by $L(S)$ and defined as the closure of $\{w\}$ under morphisms $h_1, \ldots, h_n$. An *HDTOL* (resp. *NPDTOL*) *language* is a morphic image of a DTOL language (resp. a morphic image of a PDTOL language under an ϵ-free morphism). For properties of these and related languages see [14]. As a general reference to the theory of formal languages we give [9].

The following deep algebraic result has been recently proved in [1].

Theorem 1. *For each infinite system of equations, with a finite number of unknowns, over a finitely generated free monoid there exists (noneffectively) an equivalent finite subsystem.*

This settles the famous Ehrenfeucht Conjecture that was originally formulated in the language form below. The algebraic form was proposed and shown equivalent to the language form in [4].

Let L be any language over a finite alphabet Σ and F a finite subset of L. We say that F is a *test set* for L (with respect to morphisms) if for any pair of morphisms (g,h), $g(x) = h(x)$ holds for all x in L if and only if $g(x) = h(x)$ holds for all x in F. The language form of the Ehrenfeucht Conjecture is the following.

Theorem 1'. *Every language over a finite alphabet possesses a test set.*

3. Single-Valuedness Test Sets

In this section we establish our main result (Theorem 3) which implies that it is decidable whether a *twt* is single-valued on an NPDTOL language. Some applications of this result are shown in Section 4. Our proof goes via several lemmas and theorems some of which, we believe, are of interest on their own.

We start by fixing our specific terminology. Let $T_n(\Sigma,\Delta)$ (resp. $SVT_n(\Sigma,\Delta)$) denote the family of (resp. single-valued) *twt*'s with input alphabet Σ, output alphabet Δ and at most n states. A *transducer schema* over Ω is a *twt* with the following properties:

(i) if $(p,ab,u,q,k) \in H$ and $(p,ab,v,q,k) \in H$, then $u = v$ and $u \in \Omega$.

(ii) if $(p_1,a_1b_1,u,q_1,k_1) \in H$ and $(p_2,a_2b_2,v,q_2,k_2) \in H$ with $(p_1,a_1b_1,q_1,k_1) \neq (p_2,a_2b_2,q_2,k_2)$, then $u \neq v$.

Clearly, for a fixed alphabet Σ and fixed number of states, there exists only a finite number of distinct transducer schemata (up to the renaming of outputs), so under these assumptions Ω can be fixed. For our purposes it is illustrative to call Ω the *set of unknowns*.

Let $S_n(\Sigma,\Omega) = \{M \in T_n(\Sigma,\Omega) \mid M$ is a transducer schema$\}$. For a mapping $i : \Omega \rightarrow \Delta^*$ and S in $S_n(\Sigma,\Omega)$, we denote by $i(S)$ the *twt* obtained from S by replacing in H each output (unknown) c by $i(c)$. We say that $i(S)$ is an *interpretation* of S via i. Let $I(S)$ be the set of all the interpretations of S and $I(S_n(\Sigma,\Omega)) = \bigcup_{S \in S_n(\Sigma,\Omega)} I(S)$. Clearly, $I(S_n(\Sigma,\Omega)) \subsetneq T_n(\Sigma,\Delta)$. The inclusion is proper because of condition (i) in the definition of a transducer schema. However, $I(S_n(\Sigma,\Omega))$ contains $SVT_n(\Sigma,\Delta)$, assuming that we identify the transducers which differ from each other only because of useless transition rules. Consequently, $S_n(\Sigma,\Omega)$ serves as a "finite base" for the family $SVT_n(\Sigma,\Delta)$ in the sense that each *twt* in this family is obtained from a transducer schema in $S_n(\Sigma,\Omega)$ via an interpretation. Observe, however, that transducer schemata are not necessarily single-valued and hence $SVT_n(\Sigma,\Delta) \subsetneq I(S_n(\Sigma,\Omega))$.

Finally, we define the notion of the single-valuedness test set for a language L with respect to a class θ of *twt*'s. We say that a finite subset F of L is a *single-valuedness test set for L with respect to θ* if any *twt* in θ is single-valued on L if and only if it is single-valued on F.

The terminology and notation introduced above is used throughout the paper. Now, we are ready for our first auxiliary result.

Lemma 1. *For any language $L \subseteq \Sigma^*$ and for a transducer schema S there exists a single-valuedness test set F for L with respect to $I(S)$.*

Proof: Consider a word w in $L \cap dom(S)$. Let $comp(w)$ denote the set of all computations of w in S. We say that a computation u contains a loop if it

contains a subcomputation from a configuration into itself. Let the set of loop-free computations of w be denoted by $comp_{lf}(w)$. Furthermore, let $comp_l(w)$ denote the set of all subcomputations of computations of w which are loops but do not contain any loops as proper subcomputations. Clearly, both $comp_{lf}(w)$ and $comp_l(w)$ are finite. The output of a computation u in S is, of course, a unique word in Ω^*, let us denote it by $output(u)$.

Now, we associate the word w with the following system of equations over the free monoid Δ^* with Ω as the set of unknowns:

$$\begin{cases} output(u) = output(u'), & \text{for all } u, u' \in comp_{lf}(w) \\ output(v) = \epsilon, & \text{for all } v \in comp_l(w) \end{cases}$$

Clearly, this system is finite and let us denote it by E_w. It is straightforward to see that E_w characterizes these transducers in $I(S)$ which are single-valued on w. That is to say, each solution of E_w defines such a *twt* (or a class of *twt*'s), and vice versa.

Similarly, the (possibly infinite) system

$$E_L = \bigcup_{w \in L \cap dom(S)} E_w$$

of equations characterizes these *twt*'s which are single-valued on L. But, by Theorem 1, E_L has a finite equivalent subsystem. Hence, there exists a finite subset F of $L \cap dom(S)$ such that $E_F = \bigcup_{w \in F} E_w$ is equivalent to E_L. It follows that any *twt* in $I(S)$ which is single-valued on F is so also on L, proving the lemma.

□

The following result generalizes Lemma 1.

Theorem 2. *Given a language $L \subseteq \Sigma^*$ and a natural number n, there exists*

(noneffectively) a finite subset F of L which is single-valuedness test set for L with respect to $T_n(\Sigma,\Delta)$.

Proof: By Lemma 1, for each transducer schema S there exists a finite subset $F(S)$ of L which tests the single-valuedness for *twt*'s in $I(S)$. Since there exists only a finite number of different transducer schemata it follows that the set $F' = \bigcup_{S \in S_n(\Sigma,\Omega)} F(S)$ is a single-valuedness test set for transducers in $I(S_n(\Sigma,\Omega))$.

Let T be a *twt* in $T_n(\Sigma,\Delta)-I(S_n(\Sigma,\Omega))$. Then T is of the form $(K,\Sigma,\Delta,H_1 \cup H_2,s_0)$ where $(K,\Sigma,\Delta,H_1,s_0)$ is in $I(S_n(\Sigma,\Omega))$ and H_2 is a finite union of sets of transitions of the form

$$\{(p,ab,w_j,q,k) \mid j = 1, \ldots, t\} \tag{1}$$

where $a,b \in \Sigma$, $p,q \in K$, $k \in \{-1,0,1\}$, $w_j \in \Delta^*$ and $t \geq 2$. Identifying those T's which have the same domain and, as well as, the sets in (1) with the vectors (p,ab,q,k) we have only a finite number of possibilities for T. For each such possibility we choose, for each set of rules in (1), a word (if such exists) from $L \cap dom(T)$ such that these rules can be used in a computation on this word. Let a finite set of words thus obtained be F''. Then it is straightforward to see that F can be chosen to be $F' \cup F''$.

□

In order to prove that Theorem 2 holds effectively in certain special cases we need a few more lemmas.

Lemma 2. *For two finite languages L_1 and L_2, with $L_1 \subseteq L_2 \subseteq \Sigma^*$, and for an integer n, it is decidable whether or not L_1 is a single-valuedness test set for L_2 with respect to $T_n(\Sigma,\Delta)$.*

Proof: We first show that the lemma holds with respect to $I(S_n(\Sigma,\Omega))$. To prove this it is clearly enough to prove that the lemma holds with respect to all interpretations of a fixed transducer schema.

Let S be a fixed transducer schema in $S_n(\Sigma,\Omega)$. As in the proof of Lemma 1 we associate the languages L_1 and L_2 with systems of equations E_{L_1} and E_{L_2} such that E_{L_i} characterizes those transducers in $I(S)$ which are single-valued on L_i. It was shown in [4] (using a deep decidability result of [12]) that the equivalence of two finite systems of equations over a free monoid is decidable. Hence, we can decide whether L_1 is a single-valuedness test set for L_2 with respect to $I(S)$.

In order to finish the proof we recall the considerations at the end of the proof of Theorem 2. It follows from these and from the first part of this proof that the problem of Lemma 2 is reduced to the problem of deciding whether or not a given transition rule of a given *twt* can be used in any computation on a given word. Of course, this problem is decidable, and hence the lemma follows.

□

Our next lemma resembles the known fact that an inverse morphism does not increase the complexity of a regular language (measured as the number of states of an automaton needed to recognize the language).

Lemma 3. *Let* $h : \Gamma^* \to \Sigma^*$ *be an* ϵ*-free morphism and* T *a twt in* $T_n(\Sigma,\Delta)$. *There exists a twt* T_h *in* $T_n(\Gamma,\Delta)$ *satisfying the following conditions:*

(i) $dom(T_h) = h^{-1}(dom(T))$,

(ii) $T_h(w) \subseteq T(h(w))$ *for each word* w *in* Γ^*, *and*

(iii) *T_h is single-valued on $L \subseteq \Gamma^*$ if and only if T is single-valued on $h(L)$.*

In particular, if T is single-valued so is T_h and in this case $T_h(w) = T(h(w))$ for all w in Γ^*.

Proof: Clearly, the last sentence of the lemma follows from the previous ones.

In order to define T_h we need some notation and preliminary considerations. For letters a and b in Γ and for a state q in the state set Q of T we say that a computation u of T is (a,q,b)-*minimal* if

(a) it starts from the configuration $\cdots h(a)qh(b)\cdots$,

(b) it ends in a configuration either of the form $\cdots h(a)h(b)q'\cdots$ or of the form $\cdots q'h(a)h(b)\cdots$ for some q' in K, and

(c) no configuration with the exception of the last one is of the form (b).

Clearly, for a triple (a,q,b) the set of (a,q,b)-minimal computations might be infinite. In order to keep it finite we define the notions of loop-free and single-loop computations: computation u above is *loop-free* (resp. *single-loop*) (a,q,b)-*minimal* if in addition to (a)-(c) it satisfies:

(d) no configuration is repeated (resp. repeated but once) in the computation.

Recalling that the initial state s_0 is never entered in any computation of T we can modify, in an obvious way, the notions of loop-free and single-loop computations for computations starting from a configuration of the form $a's_0\beta\cdots$, with a' in Σ, β in Σ^+ and $a'\beta = h(a)$ for some a in Γ, and ending in a configuration of the form $h(a)q'\cdots$ for some q' in Q.

Using the above notation we now construct the transitions H_h of the transducer T_h. For each loop-free or single-loop (a,q,b)-minimal computation ending in the configuration of the form $\cdots h(a)h(b)q'\cdots$ (resp. $\cdots q'h(a)h(b)\cdots$) H_h contains a transition $(q,ab,w,q',1)$ (resp. $(q,ab,w,q',-1)$) where w is the output produced by T in the considered

computation. For the state s_0 the transitions are defined analogously using above modifications. The set of states of T_h is the set of these states encountered when the above procedure is carried over all loop-free and single-loop (a,q,b)-minimal computations. Moreover, the initial and final states of T_h are those of T (belonging to the state set of T_h).

Clearly, T_h is a well defined transducer in $T_n(\Gamma,\Delta)$. It is also a straightforward consequence of the construction that T_h satisfies the requirements (i)-(iii).

□

As an application of Lemma 3 we derive the following

Lemma 4. *Let $h : \Gamma^* \rightarrow \Sigma^*$ be an ϵ-free morphism and F a single-valuedness test set for a language $L \subseteq \Sigma^*$ with respect to $T_n(\Gamma,\Delta)$. Then $h(F)$ is a single-valuedness test set for $h(L)$ with respect to $T_n(\Sigma,\Delta)$.*

Proof: It follows readily from Lemma 3. Indeed, if Lemma 4 would not hold then there exists a transducer T in $T_n(\Sigma,\Delta)$ such that it is single-valued on $h(F)$ but not on $h(L)$. Then the transducer T_h of Lemma 3 is in $T_n(\Gamma,\Delta)$ and is single-valued on F but not on L, a contradiction.

□

Now, finally we are ready for an effective subcase of Theorem 2.

Lemma 5. *Given a PDTOL language $L \subseteq \Sigma^*$ and a natural number n, there effectively exists a single-valuedness test set F for L with respect to $T_n(\Sigma,\Delta)$.*

Proof: Let L be generated by a PDTOL system $(\Sigma,h_1, \ldots, h_n,w)$. Define,

for each $i \geq 0$,

$$L_0 = \{w\}$$

$$L_{i+1} = L_i \cup h_1(L_i) \cup \cdots \cup h_n(L_i).$$

Clearly, each L_i is finite so that we can, by Lemma 2 and Theorem 2, find effectively an integer i_0 such that L_{i_0} is a single-valuedness test set for L_{i_0+1}. Now, an obvious modification of the argument originally presented in the proof of Theorem 3.2 in [4], cf. also the proof of Theorem 4 in [5], shows that L_{i_0} is a single-valuedness test set for L with respect to $T_n(\Sigma,\Delta)$, too. In this concluding argument a crucial role is played by Lemma 3 and its consequence Lemma 4.

□

Now we are ready for our main result which is not only interesting on its own, but has, as we shall see in the next section, several important consequences.

Theorem 3. *Given an NPDTOL language $L \subseteq \Sigma^*$ and a natural number n, there effectively exists a single-valuedness test set F for L with respect to $T_n(\Sigma,\Delta)$.*

Proof: Assume that $L = h(L')$ for a PDOL language L' and an ϵ-free morphism $h : \Gamma^* \rightarrow \Sigma^*$. Let F' be, according to lemma 5, a single-valuedness test set for L' with respect to $T_n(\Gamma,\Delta)$. Then, by Lemma 4, we can choose $F = h(F')$.

□

4. Applications of The Main Result

In this section we apply our main result as well as Theorem 2 to other problems. Our first applications deal with test sets for testing equivalence of single-valued *twt*'s. Let $L \subseteq \Sigma^*$ be a language and θ a family of transducers on Σ^*. We say that a finite subset F of L is a *test set for L with respect to θ* if for any two transducers in θ they are equivalent on L if and only if they are equivalent on F. Recalling that $SVT_n(\Sigma,\Delta)$ denotes the family of single-valued *twt*'s, with input alphabet Σ, output alphabet Δ and at most n states, we have:

Theorem 4. *For each language $L \subseteq \Sigma^*$ and a natural number n, there exists (noneffectively) a test set F with respect to $SVT_n(\Sigma,\Delta)$.*

Proof: Consider two single-valued *twt*'s T_1 and T_2 with at most n states. Let $T_1 \cup T_2$ be a transducer satisfying $(T_1 \cup T_2)(w) = T_1(w) \cup T_2(w)$ for all w. Clearly, such a *twt* exists and can be chosen to contain no more than $2n$ states. It is also obvious that T_1 and T_2 are equivalent on L if and only if the following two conditions are satisfied:

(i) $L \cap dom(T_1) = L \cap dom(T_2)$,

(ii) $T_1 \cup T_2$ is single-valued on L.

To test (i) for T_1 and T_2 it is enough to include in F one word (if such exists) from $(L \cap (dom(T_1) - dom(T_2))) \cup (L \cap (dom(T_2) - dom(T_1)))$. Let us call this word $w(dom(T_1),dom(T_2))$. Since the set of languages $\{dom(T) \mid T \in SVT_n(\Sigma,\Delta)\}$ is finite, the set of words thus obtained when T_1 and T_2 ranges over $SVT_n(\Sigma,\Delta)$ can be chosen finite, too. Let the set of words thus obtained be F'. Thus the theorem follows if we choose $F = F' \cup F''$, where F'' is a single-valuedness test set for L with respect $T_{2n}(\Sigma,\Delta)$.

□

We also have the following effective subcase of Theorem 4.

Theorem 5. *For an NPDTOL language L and a natural number n, there effectively exists a test set F with respect to $SVT_n(\Sigma,\Delta)$.*

Proof: The result follows directly from Theorem 3 and the proof of Theorem 4 providing we can show that the words $w(dom(T_1),dom(T_2))$ in this proof can be found effectively. To show this let T_1 and T_2 be (single-valued) two-way transducers. Then $dom(T_1)$ and $dom(T_2)$ are regular languages, and hence by the closure properties of HDTOL (or equivalently EDTOL) languages, cf. [14], $(L \cap (dom(T_1) - dom(T_2))) \cup (L \cap (dom(T_2) - dom(T_2)))$ is an EDTOL language. It can be effectively found and since the emptiness for EDTOL language is decidable we can actually find effectively the word $w(dom(T_1), dom(T_2))$. Hence, the proof is complete.

□

Theorems 3 and 5 have the following two interesting corollaries.

Theorem 6. *Given an NPDTOL language L and a two-way transducer T, it is decidable whether or not T is single-valued on L.*

Theorem 7. *Given an NPDTOL language L and single-valued two-way transducers T_1 and T_2, it is decidable whether or not T_1 and T_2 are equivalent on L.*

Next we apply our results to more classical problems of formal languages. We first generalize a decidability result of Schützenberger [16], cf. also [2], which says that it is decidable whether or not a given one-way transducer is single-valued.

Theorem 8. *It is decidable whether or not a given two-way transducer T is single-valued.*

Proof: Follows immediately from Theorem 6. Indeed, since $dom(T)$ is regular, we can, as is straightforward to see, cf. also [13], find effectively an NPDTOL language L such that $L = dom(T)$.

□

Similarly to above we deduce from Theorem 7 a proper strengthening of the main result in [8].

Theorem 9. *The equivalence problem for single-valued two-way transducers is decidable.*

Note that, if we replace the condition (i) in the proof of Theorem 4 by the condition (ii) $L \cap dom(T_1) \subseteq L \cap dom(T_2)$, then we can easily modify Theorems 7 and 9 to test the inclusion of two *twt*'s instead of their equivalence. Therefore, we also proved the following.

Theorem 10. *The inclusion problem for single-valued two-way transducers is decidable.*

We conclude this section with the following remarks. Even the special cases of Theorems 6 and 7, where the transducers are one-way transducers, are new results. Moreover, for one-way transducers we can state these results in a stronger form as follows.

Theorem 11. *Given an HDTOL language L and a one-way transducer T, it is decidable whether or not T is single-valued on L.*

Proof: In the case of one-way transducer Lemma 3 holds for all morphisms, and not only for ϵ-free ones. This is essentially the known result that if a language L is recognized by a finite automaton with n states then $h^{-1}(L)$, where h is an arbitrary morphism, is recognized by a finite automaton with no more than n states. After this observation the proof of Theorem 11 is a straightforward modification of that of Theorem 6.

□

Similarly we have

Theorem 12. *Given an HDTOL language L and single-valued one-way transducers T_1 and T_2, it is decidable whether or not T_1 and T_2 are equivalent on L.*

Note that Theorem 12 is a proper generalization of both the HDOL and the HDTOL sequence equivalence problems, which were for a long time open until shown decidable in [15] and [5], respectively. It also solves, as a special case, the (open) problem of deciding whether or not two deterministic one-way transducers are equivalent on a given HDTOL language.

5. Test for Unambiguity

Our algorithm for testing the single-valuedness of a *twt* is complicated. If instead of the single-valuedness the stricter requirement of unambiguity is considered, then the situation is much easier: not only the algorithm but also the proof of its existence is simpler as we now show.

We say that a *twt* is *unambiguous* if it is single-valued and its underlying acceptor is unambiguous. Thus we can restrict our consideration to two-way acceptors rather than *twt*'s since the testing of single-valuedness of a *twt* with unambiguous underlying acceptor is easy. We recall that a two-way acceptor is *unambiguous* if for each word there is at most one successful computation.

Theorem 13. *It is decidable whether or not a given two-way acceptor is unambiguous.*

Proof: Let $A = (K, \Sigma, E, q_0)$ where $E \subseteq K \times \Sigma^2 \times K \times \{-1, 0, 1\}$, be a two-way acceptor. We will write the transition rules in E in the form $(p, ab) \rightarrow (q, k)$. We say that two transition rules $(p, ab) \rightarrow (q, k)$ and $(p, ab) \rightarrow (r, j)$ create a *fork* if $(q, k) \neq (r, j)$. Clearly, A is ambiguous if and only if there exists at least one such fork in E which causes the ambiguity, or more precisely, there exist two distinct computations on input $uabv$ that both reach the same configuration $uapbv$ and then split along the two distinct branches of the fork.

There is only a finite number of forks, and they can be easily listed. So the theorem holds if we can decide whether a given fork causes ambiguity.

Consider the above fork. We construct three modifications of acceptor A. Let $A_i = (K_i, \Sigma \cup \{\hat{a}, \hat{b}\}, E_i, q_i)$ for $i = 1, 2, 3$ be two-way acceptors defined as follows. Acceptor A_1 simulates A (identifying $\hat{a}$ with a and $\hat{b}$ with b) as long as the reading head is in between $\hat{a}$ and $\hat{b}$ in state p. Then it moves into

5. Test for Unambiguity

Our algorithm for testing the single-valuedness of a *twt* is complicated. If instead of the single-valuedness the stricter requirement of unambiguity is considered, then the situation is much easier: not only the algorithm but also the proof of its existence is simpler as we now show.

We say that a *twt* is *unambiguous* if it is single-valued and its underlying acceptor is unambiguous. Thus we can restrict our consideration to two-way acceptors rather than *twt*'s since the testing of single-valuedness of a *twt* with unambiguous underlying acceptor is easy. We recall that a two-way acceptor is *unambiguous* if for each word there is at most one successful computation.

Theorem 13. *It is decidable whether or not a given two-way acceptor is unambiguous.*

Proof: Let $A = (K, \Sigma, E, q_0)$ where $E \subseteq K \times \Sigma^2 \times K \times \{-1, 0, 1\}$, be a two-way acceptor. We will write the transition rules in E in the form $(p, ab) \rightarrow (q, k)$. We say that two transition rules $(p, ab) \rightarrow (q, k)$ and $(p, ab) \rightarrow (r, j)$ create a *fork* if $(q, k) \neq (r, j)$. Clearly, A is ambiguous if and only if there exists at least one such fork in E which causes the ambiguity, or more precisely, there exist two distinct computations on input $uabv$ that both reach the same configuration $uapbv$ and then split along the two distinct branches of the fork.

There is only a finite number of forks, and they can be easily listed. So the theorem holds if we can decide whether a given fork causes ambiguity.

Consider the above fork. We construct three modifications of acceptor A. Let $A_i = (K_i, \Sigma \cup \{\hat{a}, \hat{b}\}, E_i, q_i)$ for $i = 1, 2, 3$ be two-way acceptors defined as follows. Acceptor A_1 simulates A (identifying $\hat{a}$ with a and $\hat{b}$ with b) as long as the reading head is in between $\hat{a}$ and $\hat{b}$ in state p. Then it moves into

a new state that moves the reading head over any word in Σ^* to the right and out of the tape.

Acceptor A_2 starts in a new initial state and moves its reading head to the right until it hits $\hat{a}\hat{b}$. Then it simulates the first branch of the fork and continues as A, interpreting $\hat{a}$ as a and $\hat{b}$ as b. Acceptor A_3 is like A_2 except it follows the second branch of the fork.

Clearly, $ua\hat{b}v \in L(A_1)$ if and only if there exists a computation of A from the initial configuration to the configuration $uapbv$. Further, $ua\hat{b}v \in L(A_2)$ (resp. $ua\hat{b}v \in L(A_3)$) if and only if there is a computation of A_2 (resp. A_3) from the configuration $uapbv$ to an accepting configuration with the first atomic move following the first (resp. the second) branch of the fork. Therefore, the considered fork causes ambiguity if and only if $L(A_1) \cap L(A_2) \cap L(A_3) \neq \emptyset$. But each $L(A_i)$, $i = 1, 2, 3$, is an effectively given regular language, see [9], hence the validity of the above inequality can easily be tested.

□

Theorem 13 deserves one final comment. Although we were able to use the decidability of the single-valuedness to solve also the equivalence problem for single-valued transducers, it seems not to be possible to have the same reduction for the case of unambiguity. So the simple proof of Theorem 13, does not give a simple proof of the decidability of the equivalence problem for unambiguous two-way transducers.

References

[1] M.H. Albert, and J. Lawrence, A proof of Ehrenfeucht's conjecture, *Theoret. Comput. Sci.*, to appear.

[2] M. Blattner, and T. Head, Single valued a-transducers, *J. Comput. System Sci., 15* (1977), pp. 310-327.

[3] K. Culik II, Some decidability results about regular pushdown translations, *Inform. Process. Lett., 8* (1979), pp. 5-8.

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

[5] K. Culik II, and J. Karhumäki, The decidability of the DTOL sequence equivalence problem and related decision problems, University of Waterloo, Department of Computer Science, Research Report CS-85-05, (1985).

[6] K. Culik II, and A. Salomaa, Test sets and checking words for homomorphisms equivalence, *J. Comput. Systems Sci., 19* (1980), pp. 379-395.

[7] R. Ehrich, and S. Yau, Two-way sequential transductions and stack automata, *Inform. and Control, 18* (1971), pp. 404-446.

[8] E.M. Gurari, The equivalence problem for deterministic two-way transducers is decidable, *SIAM J. Comput., 11* (1982), pp. 448-452.

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

[10] O. Ibarra, The unsolvability of the equivalence problem for ϵ-free NGSM's with unary input (output) alphabet and applications, *SIAM J. Comput., 4* (1978), pp. 524-532.

[11] J. Karhumäki, The Ehrenfeucht Conjecture: A compactness claim for finitely generated free monoids, *Theoret. Comput. Sci., 29* (1984), pp. 285-308.

[12] G.S. Makanin, The problem of solvability of equations in a free semigroup, *Mat. Sb., 103* (1977), pp. 147-236. (English Trasl. in: *Math. USSR Sb., 32* (1977), pp. 129-198.)

[13] M. Nielsen, G.Rozenberg, A. Salomaa, and S. Skyum, Nonterminals, homomorphisms and codings in different variations of OL systems. I and II, *Acta Inform., 3* (1974), pp. 357-364; 4 (1974), pp. 87-106.

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

[15] K. Ruohonen, Test sets for iterated morphisms, manuscript (1984).

[16] M.P. Schützenberger, Sur les relations rationnelles, *Lecture Notes in Computer Science, 33*, Springer, Berlin, 1975, pp. 209-213.