When Is a Functional Tree Transduction Deterministic?

Helmut Seidl

Fachbereich Informatik
Universität des Saarlandes
D–6600 Saarbrücken
Germany

Abstract. We give a decision procedure to determine whether or not the transduction of a functional transducer can be realized by a deterministic (resp. reduced deterministic) transducer. In case this is possible we exhibit a general construction to build this transducer.

0 Introduction

A (finite–state bottom–up) tree transducer is a finite state device. While traversing an input tree it produces an output tree in a bottom–up fashion. It is called functional iff there is at most one output tree for every input.

In practice, the concept of tree transducers is used, e.g., by tools for generating code selectors from formal descriptions of target machine languages (cf. [4]). Given a (regular) tree grammar which specifies the semantics of machine instructions in terms of operators of the intermediate representation language, a (typically non–deterministic) tree transducer is generated. The accepting computations describe all sequences of target machine code possibly generated for a given piece of program. Taking into account that different instructions may have different costs (e.g., need more or less processor cycles) a functional transducer is constructed which produces a cheapest instruction sequence.

Our main concern in this paper is the implementation of such functional transducers. Usually, this is done in two passes. The first one traverses the input tree to compute an overview over all possible computations from which the second one constructs an accepting computation and produces the output for it. We would like to avoid this second pass and produce output immediately. This can be done provided the corresponding tree transducer is deterministic. However, simple examples show that it is not always possible to construct an equivalent deterministic transducer. Therefore, the natural question arises: *When can the transduction of a functional transducer be realized by a deterministic one?*

Also, one would like to produce output only for subcomputations which finally is part of the resulting output. This can be guaranteed provided the tree transducer is *reduced deterministic*. Again, it is not always possible to construct even for a deterministic transducer an equivalent reduced deterministic transducer. Therefore, we also have to deal with the question: *When can the* *transduction of a functional transducer be realized by a reduced deterministic one?*

Corresponding results for word transducers are known at least since [2]. Explicit constructions are described in [7] and [10].

So far, for tree transducers no related results are known. Several papers deal with deterministic or functional transducers, but they are mainly concerned with the equivalence problem. In [11] Zachar showed that for deterministic transducers equivalence is decidable. In [3] Engelfriet showed that it is decidable whether or not a tree transducer is functional. In [9] it was shown by the author that this problem is decidable even in polynomial time. As a corollary it follows that also equivalence of functional tree transducers is decidable in polynomial time provided the transducers agree in their domains. Especially, equivalence of unambiguous or deterministic transducers is decidable in polynomial time.

The present paper answers the question whether or not the transduction of a functional transducer can be realized by a deterministic or even reduced deterministic transducer. It is organized as follows. In Section 1, we provide some basic definitions. In Section 2, we introduce tree transducers. Especially, we allow these transducers to produce a final pattern after having reached a final state at the root of the input tree. This extension does not enlarge the power of a non–deterministic transducer; in case of a deterministic transducer this notion corresponds to the well–known class or "subsequential transducers" [2, 7, 10] for words. Section 3 shows that, for every functional transducer we can construct an unambiguous transducer realizing the same transduction. Then we investigate deterministic transducers. Sections 4 and 5 present necessary conditions (D0), (D1) and (D2) (resp. (wD0), (D1) and (D2)) on a transducer for its transduction to be reduced deterministic (resp. deterministic). Our Main Theorems are presented in Section 6. Provided Properties (D0), (D1) and (D2) (resp. (wD0), (D1) and (D2)) hold for a functional reduced tree transducer M we construct an equivalent reduced deterministic (resp. deterministic) transducer.

Because of space limitations most details of the proofs are omitted. They can be found in the full paper.

1 Trees

In this section we give basic definitions and state some fundamental properties about trees. Moreover, we present some technical propositions which will be used in the sequel.

A *ranked alphabet* or *signature* is a pair (Σ, ρ) where Σ is a finite alphabet and $\rho : \Sigma \to \mathbb{N}_0$ is a function mapping symbols to their rank. Usually, if ρ is understood we write Σ for short and define $\Sigma_j = \rho^{-1}(j)$. T_Σ denotes the free Σ-algebra of (finite ordered Σ–labeled) *trees*, i.e., T_Σ is the smallest set T satisfying (i) $\Sigma_0 \subseteq T$, and (ii) if $a \in \Sigma_m$ and $t_1, \ldots, t_m \in T$, then $a(t_1, \ldots, t_m) \in T$. Note: (i) can be viewed as the subcase of (ii) where $m = 0$.

Let $N \subseteq \Sigma_0$. The *N-depth* of a tree $t \in T_\Sigma$, $\text{depth}_N(t)$, is defined by $\mathrm{depth}_N(t) = \begin{cases} -\infty \text{ if } t \in \Sigma_0 \backslash N \\ 0 \quad \text{ if } t \in N \end{cases}$ if $t \in \Sigma_0$, and $\mathrm{depth}_N(t) = 1 + \max\{\mathrm{depth}_N(t_1), \ldots, \mathrm{depth}_N(t_m)\}$ if $t = a(t_1, \ldots, t_m)$ for some $a \in \Sigma_m, m > 0$. If $N = \Sigma_0$ we omit the index N; also, if $N = \{x\}$ contains just one element we also write $\mathrm{depth}_x(\_)$ instead of $\mathrm{depth}_{\{x\}}(\_)$.

The *size* of t , $|t|$, is defined by $|t| = 1$ if $t \in \Sigma_0$, and $|t| = 1 + \sum_{j=1}^{m} |t_j|$ if $t = a(t_1, \ldots, t_m)$ for some $a \in \Sigma_m, m > 0$.

Let X denote a set of variables of rank 0. Define $T_\Sigma(X) = T_{\Sigma \cup X}$. We use this different notation in order to indicate which variables are to be substituted. (Clearly, $T_\Sigma \subseteq T_\Sigma(X)$.) $t \in T_\Sigma(X)$ is called *X–proper* iff every $x \in X$ occurs in t exactly once. If $X = \{x\}$ we write x–proper instead of $\{x\}$–proper, and if X is understood we skip the prefix X.

Every map $\theta : X \to T_\Sigma(X)$ can be extended to a map $\theta : T_\Sigma(X) \to T_\Sigma(X)$ by $t\theta = x\theta$ if $t = x$, and $t\theta = a(t_1\theta, \ldots, t_m\theta)$ if $t = a(t_1, \ldots, t_m)$ with $a \in \Sigma$. θ is called *X–substitution* or simply substitution if X is understood. If $X = \{x_1, \ldots, x_m\}$ and $x_i\theta = t_i$, we denote $t\theta$ also by $t[t_1, \ldots, t_m]$. Of special importance is the case where the set X of variables which are to be substituted consists of just one element x. Assume $x\theta = t_2$ and $t_1 \in T_\Sigma(x) = T_\Sigma(\{x\})$. Then we write $t_1\theta = t_1 t_2$. The set $T_\Sigma(x)$ is a monoid w.r.t. x–substitution. (The neutral element is x). $T_\Sigma(x)$ is not a free monoid. Especially, $t_1 t_2 = t_1$ if t_1 does not contain an occurrence of x.

Let $\tilde{T}_\Sigma(x)$ denote the submonoid of $T_\Sigma(x)$ consisting of all trees t which contain *at least* one occurrence of x. Note that trees in $\tilde{T}_\Sigma(x)$ may contain not only one occurrence of x but also two occurrences or more. Call a tree $t \in \tilde{T}_\Sigma(x)$ *x–irreducible* iff $t \neq x$ and $t = uv$ implies either $u = x$ or $v = x$. If x is understood we also skip the prefix x. So for example, $t = a(x, b(x))$ is irreducible whereas $t' = a(b(x), b(x)) = a(x, x)b(x)$ is not. Also, trees $a(x, t)$ or $a(t, x)$ for all trees $t \in T_\Sigma$ are irreducible. Let $I_\Sigma(x)$ denote the set of irreducible trees in $\tilde{T}_\Sigma(x)$. Note that $I_\Sigma(x)$ is infinite whenever $\Sigma_j \neq \emptyset$ for some $j > 1$.

In [9] the following theorem was proven.

Theorem 1. *As a monoid, $\tilde{T}_\Sigma(x)$ is freely generated from $I_\Sigma(x)$, i.e., $\tilde{T}_\Sigma(x) = I_\Sigma(x)^*$.* □

Consider for example the tree $t = a(a(b(x), b(x)), c)$ for $a, b, c \in \Sigma$. Then $t = u_1 u_2 u_3$ for irreducible trees u_i where $u_1 = a(x, c)$; $u_2 = a(x, x)$ and $u_3 = b(x)$.

Theorem 1 allows to define the *x–length* $|t|_x$ of a tree $t \in \tilde{T}_\Sigma(x)$: $|t|_x = n$ iff $t = u_1 \ldots u_n$ for irreducible trees u_j. Observe that $|t|_x \leq \mathrm{depth}_x(t)$, and $|t|_x = \mathrm{depth}_x(t)$ whenever t is x–proper. If $t = u_1 u_2 \in \tilde{T}_\Sigma(x)$ then u_1 is called an *x–prefix* of t. Accordingly, u_2 is called *x–suffix* of t.

2 Bottom–up Finite State Tree Transducers

In this section we introduce finite tree automata (FTAs for short) and bottom–up finite state tree transducers (FSTs for short). Similar to [9] we define an

FST M as a pair (A,T) where A is the finite tree automaton underlying M and T is the output function. Moreover, we recall the notion of reducedness of a transducer from [8, 9] which turns out to be useful in this context as well.

In the sequel, X denotes the fix denumerable set $\{x_i | i \in \mathbb{N}\}$ of variables and for $m \geq 0$, $X_m = \{x_1, \ldots, x_m\}$.

A *finite tree automaton* (FTA for short) is a 4–tuple $A = (Q, \Sigma, \delta, Q_F)$ where Q is a finite set of *states*, $Q_F \subseteq Q$ is the set of *final* states, Σ is the signature of *input trees*, $\delta \subseteq \bigcup_{m \geq 0} Q \times \Sigma_m \times Q^m$ is the set of *transitions* of A; the transitions in $\delta \cap \bigcup_{m \geq 0} \{q\} \times \Sigma_m \times Q^m$ are also called q–*transitions*.

Let $t = a(t_1, \ldots, t_m) \in T_\Sigma(X_k)$ and $q, q_1, \ldots, q_k \in Q$. A $(q, q_1 \ldots q_k)$–*computation* ϕ of A for t starts at variables x_j in states q_j and consists of $(p_j, q_1 \ldots q_k)$-computations of A for the subtrees t_j, $j = 1, \ldots, m$, together with a transition $(q, a, p_1 \ldots p_m) \in \delta$ for the root. We write the state at the root to the left of the states at the variable leafs. This convention is chosen in accordance with our prefix notation of trees and the left–to–right order of substitutions. Formally, we represent ϕ as a tree over signature δ and set of variables X_k as follows. If $t = x_j$ and $q = q_j$ then $\phi = x_j$. If $t = a(t_1, \ldots, t_m)$ then $\phi = \tau(\phi_1, \ldots, \phi_m)$ where $\tau = (q, a, p_1 \ldots p_m) \in \delta$ for suitable states $p_1, \ldots, p_m \in Q$ and ϕ_j is a $(p_j, q_1 \ldots q_k)$–computation for t_j, $j = 1, \ldots, m$. Overloading the symbol δ, we write $(q, t, q_1 \ldots q_k) \in \delta$ iff there exists a $(q, q_1 \ldots q_k)$–computation of A for t.

Assume $t \in T_\Sigma(X_k)$ and $t = t_0[t_1, \ldots, t_k]$. Assume ϕ_0 is a $(q, p_1 \ldots p_k)$–computation for t_0, and ϕ_i are $(p_i, q_1 \ldots q_m)$–computations for t_i , $i = 1, \ldots, k$. Then $\phi_0[\phi_1, \ldots, \phi_k]$ is a $(q, q_1 \ldots q_m)$–computation ϕ of A for t. Conversely, if t_0 contains exactly one occurrence of any x_j , $j = 1, \ldots, k$ (i.e., is X_k-proper), then every $(q, q_1 \ldots q_m)$–computation ϕ for t uniquely can be decomposed into a $(q, p_1 \ldots p_k)$–computation ϕ_0 for t_0, and $(p_i, q_1 \ldots q_m)$–computations ϕ_i for t_i (for suitable states p_i) such that $\phi = \phi_0[\phi_1, \ldots, \phi_k]$. ϕ_i is called *subcomputation* of ϕ on t_i.

A (q, ϵ)–computation is also called q–computation. A q–computation is called *accepting*, iff $q \in Q_F$. $\mathcal{L}(A) = \{t \in T_\Sigma |$ there is an accepting computation of A for $t\}$ is the *language* accepted by A.

A is called *unambiguous* iff there is at most one accepting computation for every input tree $t \in T_\Sigma$. A is called *deterministic* iff for every input symbol $a \in \Sigma$ and sequence $q_1 \ldots q_k \in Q^*$, there is at most one state $q \in Q$ such that $(q, a, q_1 \ldots q_k) \in \delta$. Clearly, every deterministic FTA is also unambiguous.

The *size* of A, $|A|$, is defined by

$$|A| = \sum_{(q,a,q_1 \ldots q_m) \in \delta} (m+2)$$

A *bottom-up finite state tree transducer* (FST for short) is a pair $M = (A, T)$. $A = (Q, \Sigma, \delta, Q_F)$ is the FTA *underlying* M whereas $T : \delta \cup Q_F \rightarrow T_\Delta(X)$, the *output function* of M, maps transitions and accepting states to their *output patterns*. Transitions $\tau = (q, a, q_1 \ldots q_m)$ are mapped to trees $T(\tau) \in T_\Delta(X_m)$, whereas states $f \in Q_F$ are mapped to trees $T(f) \in T_\Delta(x_1)$.

Note that an FST according to the definition in [9] is not allowed to produce output depending on the final state after having processed the whole input tree. For non–deterministic transducers, the additional feature does not increase the power of the corresponding device. For deterministic transducers it does. Here, it corresponds to the notion of "subsequential transducers" for words [2, 7, 10].

T is extended to computations as follows. Assume ϕ is a $(q, q_1 \dots q_k)$-computation of A. If $\phi = x_j$ for some j then $T(\phi) = x_j$. If $\phi = \tau(\phi_1, \dots, \phi_m)$ then $T(\phi) = T(\tau)[T(\phi_1), \dots, T(\phi_m)]$. $T(\phi)$ is also called the *output* produced by ϕ. By this definition, $T(\phi[\phi_1, \dots, \phi_k]) = T(\phi)[T(\phi_1), \dots, T(\phi_k)]$. Overloading the symbol T, we also write $(q, t, s, q_1 \dots q_k) \in T$ iff there exists a $(q, q_1 \dots q_k)$–computation ϕ of A for t with $T(\phi) = s$.

For some tree $t \in T_\Sigma$, $\mathcal{T}_M(t) = \{T(f)T(\phi) | \phi$ accepting f–computation of A for $t\}$ denotes the set of outputs of M for t. $\mathcal{T}(M) = \{(t, s) | t \in \mathcal{L}(A), s \in \mathcal{T}_M(t)\}$ is the *transduction* realized by M. M is called *functional* iff $\#\mathcal{T}_M(t) \leq 1$ for every input tree $t \in T_\Sigma$. M is called *unambiguous* or *deterministic* iff the underlying FTA is *unambiguous* or *deterministic* respectively. Clearly, every deterministic FST is unambiguous; and every unambiguous FST is functional.

The *size* $|M|$ of M is defined by

$$|M| = |A| + \sum_{\tau \in \delta}(|T(\tau)| + 1) + \sum_{f \in Q_F}(|T(f)| + 1)$$

A state $q \in Q$ is called *useful* iff an accepting computation ϕ exists that contains a q–transition. If this is not the case, q is called *useless*. Useless states can be removed without changing the "behavior" of A (and hence also M). An FTA A is called *reduced* iff A has no useless states.

The rest of this section is concerned with a "reduction" of transducers. Outputs for subcomputations which are not parts of the final output uniformly should equal a special output tree, namely $\perp$. $\perp$ is a new symbol (i.e., $\perp \notin \Delta$) of rank 0. Accordingly, we consider FSTs $M = (A, T)$ where the range of T is contained in $T_\Delta(X) \cup \{\perp\}$. However, we consider in fact only FSTs (A, T) where an output tree $\perp$ is always substituted for a variable x_j which does not occur in the corresponding output pattern. Therefore, $\perp$ does not occur as the leaf of an output tree $s \neq \perp$, i.e., the output of every $(q, q_1 \dots q_k)$–computation ϕ of A either equals $\perp$ or is in $T_\Delta(X_k)$.

The FST $M = (A, T)$ is called *reduced*, iff A is reduced and there is some subset $U(M)$ of states such that (1) and (2) hold:

(1) For every $\tau = (q, a, q_1 \dots q_m) \in \delta$,
- $q \in U(M)$ iff $T(\tau) = \perp$; and
- $q_j \in U(M)$ iff x_j does not occur in $T(\tau)$.

(2) $f \in Q_F \cap U(M)$ iff $T(f) \in T_\Delta$.

The states in $U(M)$ are exactly those which are used by subcomputations ϕ which produce $\perp$. If a subcomputation has reached some state not in $U(M)$ we can be sure that the output for the corresponding subcomputation is part of the final output. The proof of a corresponding proposition in [8] can be carried over to prove:

Proposition 2. *Assume M is an FST. Then a reduced FST M_r can be constructed in polynomial time such that*

- $\mathcal{T}(M) = \mathcal{T}(M_r)$;
- $|M_r| \leq 2 \cdot |M|$.

Proof. We repeat the construction of M_r since we will refer to it in Section 6.

Define $\bar{M} = (\bar{A}, \bar{T})$ where $\bar{A} = (\bar{Q}, \Sigma, \bar{\delta}, \bar{Q}_F)$ where $\bar{Q} = Q \times \{0,1\}$; $\bar{Q}_F$ consists of all pairs $\langle f, i\rangle$ where $f \in Q_F$ and $i = 0$ iff x_1 does not occur in $T(f)$. For such a final state $\langle f, i\rangle$ define $\bar{T}(\langle f, i\rangle) = T(f)$. Finally, $\bar{\delta}$ and $\bar{T}|_{\bar{\delta}}$ are defined as follows. Assume $\tau = (q, a, q_1 \dots q_k) \in \delta$.

Then $\langle \tau, 0\rangle = (\langle q, 0\rangle, a, \langle q_1, 0\rangle \dots \langle q_m, 0\rangle)$, $\langle \tau, 1\rangle = (\langle q, 1\rangle, a, \langle q_1, \epsilon_1\rangle \dots \langle q_m, \epsilon_m\rangle)$ are in $\bar{\delta}$ where $\epsilon_j = 1$ if x_j occurs in $T(\tau)$, and $\epsilon_j = 0$ otherwise. $\langle \tau, 1\rangle$ and $\langle \tau, 0\rangle$ are the versions of τ which are used in subcomputations whose output will be part resp. will not be part of the final output. Therefore, we put $\bar{T}(\langle \tau, 1\rangle) = T(\tau)$ and $\bar{T}(\langle \tau, 0\rangle) = \bot$.

Then, $\mathcal{T}(M) = \mathcal{T}(\bar{M})$ and $\bar{M}$ satisfies (1) and (2) of the definition of reducedness with $U(\bar{M}) = Q \times \{0\}$. Now eliminate superfluous states and transitions. Note here that by definition, every state of $\bar{M}$ is derivable; however, not every state is also accessible from $Q_{r,F}$. □

Prop. 2 is the justification that we always w.l.o.g. may assume that a functional tree transducer is reduced. However, if the tree transducer M was deterministic it is not necessarily the case that the reduced tree transducer M_r constructed according to Prop. 2 is deterministic as well!

3 Unambiguous Transducers

As for word transducers, for every functional transducer we can construct a reduced unambiguous transducer realizing the same transduction. We have:

Theorem 3. *For every functional transducer M an unambiguous transducer M_u can be constructed such that*

1. $\mathcal{T}(M) = \mathcal{T}(M_u)$;
2. $|M_u| \leq |M| \cdot 3^{|M|}$.

Proof. The idea is the following. First, we construct an FTA A_0 whose states are pairs $\langle D, B\rangle$ of sets of states of A where D is the set of derivable states and $B \subseteq D$ is the set of accessible states in D. Then we assume a natural ordering on the states of A. This ordering induces a lexicographical ordering on every subset of Q^k, $k \geq 0$. We want to produce output according to an accepting computation of A for the given input tree t which is determined by the following strategy: for every node of t we select a unique accessible and derivable state together with a transition in a topdown fashion as follows. We start at the root by selecting the smallest accepting state which is derivable. Now assume transition $\tau = (\langle D, B\rangle, a, \langle D_1, B_1\rangle \dots \langle D_k, B_k\rangle)$ has been chosen at some node

o of the accepting computation ϕ of A_0 for the input tree t, and the selected state for node o is q. By construction, $q \in B$. Let $trans(\tau, q)$ denote the set of q-transitions $(q, a, q'_1 \dots q'_k)$ of A such that $q'_j \in D_j$ for all j. By construction of A_0, $trans(\tau, q)$ is not empty. Then the transition of ϕ for o is that element of $trans(\tau, q)$ for which the sequence $q_1 \dots q_k$ of successor states is minimal, and select for $j = 1, \dots, k$, state q_j for the j-th successor node of o in t. By construction, ϕ is an accepting computation of A for t. Finally, the output we produce for t is the output of A for ϕ.

To implement this idea we construct an FST $M_1 = (A_1, T_1)$ where A_1 is obtained by adding a third component to the states of A_0 holding the corresponding selected states. Formally, $A_1 = (Q_1, \Sigma, \delta_1, Q_{1,F})$ where $Q_1 = \{\langle D, B, q\rangle | q \in B \subseteq D\}$ with $Q_{1,F} = \{\langle D, B, q\rangle \in Q_1 | B = Q_F \cap D, q \in B \text{ minimal}\}$ where $T_1(\langle D, B, q\rangle) = T(q)$. The transition relation δ_1 is defined as follows.

$\tau = (\langle D, B, q\rangle, a, \langle D_1, B_1, q_1\rangle \dots \langle D_k, B_k, q_k\rangle) \in \delta_1$ iff

- $D = \{p \in D | \forall j \exists p_j \in D_j : (p, a, p_1 \dots p_k) \in \delta\}$;
- $B_j = \{p_j \in D_j | \exists p \in B, \forall j' \neq j \exists p_{j'} \in D_{j'} : (p, a, p_1 \dots p_k) \in \delta\}$;
- and $q_1 \dots q_k \in B_1 \times \dots \times B_k$ is minimal such that $\tau_0 = (q, a, q_1 \dots q_k) \in \delta$.

Moreover, $T_1(\tau) = T(\tau_0)$.

Finally, M_u is obtained from M_1 by removing useless states and transitions from A_1 and restricting T_1 accordingly. □

4 Cyclic Computations

In this section we recall the notion of a pairing. The notion of a pairing was introduced in [9] to allow for an elegant description of pairs of computations of a transducer for the same input tree. We use it to state the necessary conditions (D0) (resp. (wD0)) and (D1) for an FST such that $\mathcal{T}(M)$ can be realized by a reduced deterministic (resp. deterministic) transducer.

Given an FTA $A = (Q, \Sigma, \delta, Q_F)$ we define the FTA $A^{(2)}$ as follows. Let $\bar{A} = (\bar{Q}, \Sigma, \bar{\delta}, \bar{Q}_F)$ be the FTA with $\bar{Q} = Q^2$ and $\bar{Q}_F = Q^2$ where $\tau = (\langle p, q\rangle, a, \langle p_1, q_1\rangle \dots \langle p_k, q_k\rangle) \in \bar{\delta}$ iff $\tau_1 = (p, a, p_1 \dots p_k) \in \delta$ and $\tau_2 = (q, a, q_1 \dots q_k) \in \delta$. In this case, we write $\tau = \langle \tau_1, \tau_2\rangle$. Accordingly, computations of $\bar{A}$ are viewed as pairs of computations of A for the same input tree.

$\bar{A}$ need not be reduced. Thus, $A^{(2)}$ is obtained from $\bar{A}$ by removing useless states and transitions. In $A^{(2)}$ every pair of states is accepting which jointly are derivable.

A triple $\Pi = (A, T_1, T_2)$ is called *pairing* iff A is an FTA and both T_1 and T_2 are output functions for A. In our applications here (A, T_i) always will be functional.

For M we define the *pairing* $M^{(2)} = (A^{(2)}, T_1, T_2)$ where T_i produce the output according to the i-th components of the transitions of δ^2 with $T_i(z') = x_1$ for all z'.

A proper (z', z')–computation ϕ is called *cyclic* iff both $T_1(\phi) \notin \{\perp, x_1\}$ and $T_2(\phi) \notin \{\perp, x_1\}$. A state z' of $A^{(2)}$ is called *cyclic* iff a cyclic (z', z')–computation exists. Cyclic computations play an essential role in our characterization of (reduced) deterministic transductions. Property (D0) says that a (z, z)–computation already is cyclic provided either the first or the second output for it is non–trivial:

M has Property (D0) iff

(D0) $T_1(\psi) \in \{\perp, x_1\}$ iff $T_2(\psi) \in \{\perp, x_1\}$ for every proper (z, z)–computation ψ of $A^{(2)}$.

It turns out that although Property (D0) is always satisfied when $\mathcal{T}(M)$ can be realized by a *reduced* deterministic transducer (cf. Prop. 4 (1)), it may not be satisfied when $\mathcal{T}(M)$ can be realized by a deterministic transducer only. Therefore, a weaker property (wD0) is needed. (wD0) limits the possibilities how M may violate (D0):

M has Property (wD0) iff

(wD0) The set of states of $A^{(2)}$ is the (disjoint) union of sets Q_0, Q_1, Q_2 and Q_3 where

- For every $z' \in Q_0$ and every proper (z', z')–computation ψ of $A^{(2)}$, $T_i(\psi) \in \{\perp, x_1\}$ for $i = 1, 2$.
- For every $z' \in Q_1$ the first output of every proper (z', z')–computation of $A^{(2)}$ is in $\{\perp, x_1\}$, and some (z', z')–computation ψ of $A^{(2)}$ exists such that $T_2(\psi) \notin \{\perp, x_1\}$.
- For every $z' \in Q_2$ the second output of every proper (z', z')–computation of $A^{(2)}$ is in $\{\perp, x_1\}$, and some (z', z')–computation ψ of $A^{(2)}$ exists such that $T_1(\psi) \notin \{\perp, x_1\}$.
- For every $z' \in Q_3$ and every proper (z', z')–computation ψ of $A^{(2)}$, $T_1(\psi) \in \{\perp, x_1\}$ iff $T_2(\psi) \in \{\perp, x_1\}$, and some proper (z', z')–computation ψ' of $A^{(2)}$ exists such that $T_i(\psi') \notin \{\perp, x_1\}$.

For these sets the following holds:

1. If for some $i \in \{1, 2, 3\}$, some state z_2 of $A^{(2)}$ is reachable from some $z_1 \in Q_i$ then $z_2 \in Q_0 \cup Q_i$.
2. If for some $i \in \{1, 2, 3\}$, some state $z_2 \in Q_i$ is reachable from some z_1 of $A^{(2)}$ then $z_1 \in Q_0 \cup Q_i$.
3. Assume $(z, a, z_1 \dots z_k)$ is a transition $A^{(2)}$ and for some $i \in \{1, 2, 3\}$ and j, some state of Q_i is reachable from z_j. Then every state reachable from some $z_{j'}$, $j' = 1, \dots, k$, is in $Q_0 \cup Q_i$.

Proposition 4. *Assume $M = (A, T)$ is a reduced FST.*

1. *If $\mathcal{T}(M)$ can be realized by a reduced deterministic transducer then M has Property (D0).*
2. *If $\mathcal{T}(M)$ can be realized by a deterministic transducer then M has Property (wD0).*

It can be decided in polynomial time whether or not M has Property (D0) or (wD0). □

Property (D1) speaks on the difference between the produced outputs when reaching a cyclic state z.

M has Property (D1) iff

(D1) For every state z of $A^{(2)}$, every z–computation ϕ and every cyclic (z, z)–computation ψ,

(1) $r_1, r_2 \in \tilde{T}_\Delta(x_1)$ exist with $x_1 \in \{r_1, r_2\}$ with $r_1T_1(\psi^k\phi) = r_2T_2(\psi^k\phi)$ for every $k \geq 0$.

For words, Property (D0) together with Property (D1) hold iff all pairs of states are *twinned* (cf. [2, 10]). Property (wD0) is without analogue in the word case. Assume M has Property (D1). Then also the following Properties hold:

(2) r_i is a prefix of $T_{3-i}(\psi^l)$ for some $l \geq 0$ with $\text{depth}(r_i) \leq \text{depth}(T_{3-i}(\phi))$.

(3) $|T_1(\psi)|_{x_1} = |T_2(\psi)|_{x_1}$.

(4) Assume ψ' is another (z, z)–computation of $A^{(2)}$. Then also $r_1T_1((\psi')^k\phi') = r_2T_2((\psi')^k\phi)$ for every $k \geq 0$.

(5) Assume ϕ' is another z–computation of $A^{(2)}$, and $r'_1T_1(\psi^k\phi') = r'_2T_2(\psi^k\phi')$ for every $k \geq 0$. Then for $i = 1, 2$, $r'_is'_i = r_is_i$ for $s_i, s'_i \in \tilde{T}_\Delta(x_1)$ with $x_1 \in \{s'_i, s_i\}$.

Especially by (5), r_1 and r_2 are uniquely determined.

Proposition 5. *Assume for $j = 1, 2$, ψ_j is a cyclic (z_j, z_j)–computation of $A^{(2)}$, ϕ_1 is a (z_1, z_2)–computation and ϕ_2 is a z_2–computation.*

If M has Property (D1) then for every $k \geq 0$,

$$r_1T_1(\psi_2^k\phi_2) = r_1T_1(\psi_2^k\phi_2)$$

and also

$$s_1T_1(\psi_1^k\phi_1)r_2 = s_2T_2(\psi_1^k\phi_1)r_1$$

for some $s_i, r_i \in \tilde{T}_\Delta(x_1)$ where both $x_1 \in \{s_1, s_2\}$ and $x_1 \in \{r_1, r_2\}$. □

Proposition 6. *Let M be a reduced FST. If $T(M)$ can be realized by a deterministic transducer then M has Property (D1).* □

It turns out that Property (D1) together with Property (D0) (or (wD0)) neither characterizes determinism nor reduced determinism.

5 Bounded Difference

In this section we introduce Property (D2) which completes the set of properties necessary to characterize deterministic resp. reduced deterministic transductions. Property (D2) has no counterpart in the word case. Together Properties (D0), (D1) and (D2) (resp. (wD0), (D1) and (D2)) allow to prove that M is of bounded difference (bd) (resp. weakly bounded difference (wbd)). In the next section we will construct an equivalent reduced deterministic transducer for M whenever M is of bounded difference thus showing that Property (bd) (and hence jointly Properties (D0), (D1) and (D2)) is also sufficient for reduced determinism. A corresponding result holds for determinism as well where Properties (D0) and (bd) are replaced by Properties (wD0) and (wbd) respectively.

M has Property (D2) iff for every transition $\tau = (z, a, z_1 \dots z_k)$ and z_j–computations ϕ_j of $A^{(2)}$, $j = 1, \dots, k$, the following holds.

(D2) Let J denote the set of all x_j where $\phi_j = \psi_{j1}\psi_{j2}\psi_{j3}$ for proper computations ψ_{ji} where ψ_{j2} is cyclic. Let $y_i = T_i(\tau)[v_{i1}, \dots, v_{ik}]$ where $v_{ij} = x_j$ whenever $x_j \in J$ and $v_{ij} = T_i(\phi_j)$ otherwise. Assume $\#J > 1$. Consider the decompositions $y_i = r_i y'_i$ where r_i is the maximal x_1–prefix of y_i. Then J–substitutions θ_i exit such that for every $j \in J$,

1. $x_j \in \{x_j\theta_1, x_j\theta_2\}$;
2. $y'_1\theta_2 = y'_2\theta_1$;
3. $(x_j\theta_1)T_1(\psi_{j1}\psi_{j2}^m\psi_{j3}) = (x_j\theta_2)T_2(\psi_{j1}\psi_{j2}^m\psi_{j3})$ for every $m \geq 0$.

Analogously to Prop. 5 we have:

Proposition 7. *Let M be a reduced FST satisfying properties (D1) and (D2). Assume the assumptions of Property (D2) hold, and additionally a cyclic (z', z')–computation ψ' exists for some state z' together with a proper (z', z)–computation ϕ'. Then some $s_i \in \tilde{T}_\Delta(x_1)$ exist with $x_1 \in \{s_1, s_2\}$ such that $s_1T_1((\psi')^m\phi')r_1 = s_2T_2((\psi')^m\phi')r_2$ for every $m \geq 0$.*

Note especially, that in the situation of Prop. 7, either r_1 is a suffix of r_2 or vice versa.

Proposition 8. *Let M be a reduced FST. If $\mathcal{T}(M)$ can be realized by a deterministic transducer then M has Property (D2).* □

For the following always assume that A has $n > 0$ states. The reduced FST M is of *bounded difference (bd)* iff the following holds:

(1) For every z–computation ϕ of $A^{(2)}$ one of the following statements hold:
  1. depth $T_i(\phi) < n \cdot |M|$ for both $i = 1$ and $i = 2$;
  2. $T_i(\phi) = v_i u$ for some $u \in T_\Delta$ and $v_i \in \tilde{T}_\Delta(x_1)$ such that for $i = 1, 2$, $\text{depth}_{x_1}(v_i) < n \cdot |M|$ and v_i does not contain subtrees in T_Δ of depth at least $(n+1) \cdot |M|$.

(2) For every transition $\tau = (z, a, z_1 \dots z_k)$ and z_j–computations ϕ_j of $A^{(2)}$ the following holds. Let J_i denote the set of all x_j where $\text{depth}(T_i(\phi_j)) \geq n \cdot |M|$ for $i = 1, 2$, and $J = J_1 \cup J_2$. Let $y_i = T_i(\tau)[v_{1i}, \dots, v_{ki}]$ where $v_{ji} = x_j$ whenever $x_j \in J$ and $v_{ji} = T_i(\phi_j)$ otherwise. Assume $\#J > 1$. Consider the decompositions $y_i = z_i y_i'$ where z_i is the maximal x_1–prefix of y_i. Then J–substitutions θ_i, $i = 1, 2$, exist such that for every $j \in J$,
 1. $x_j \in \{x_j\theta_1, x_j\theta_2\} \subseteq \tilde{T}_\Delta(x_j)$;
 2. $y_1'\theta_2 = y_2'\theta_1$;
 3. $(x_j\theta_1)T_1(\phi_j) = (x_j\theta_2)T_2(\psi_j)$.

M is of *weakly bounded difference (wbd)* iff for every z–computation ϕ of $A^{(2)}$, statement (1) must only hold provided depth $T_i(\phi) \geq n \cdot |M|$ for both $i \in \{1, 2\}$; and statement (2) only provided $J_1 \neq \emptyset \neq J_2$.

6 Characterizations

This section presents the main theorems of this paper. Theorem 9 states that the transduction of a reduced functional transducer M can be realized by a reduced deterministic one iff M has Properties (D0), (D1) and (D2). Theorem 10 is the corresponding version for realization by deterministic transducers. Here, the characterization only differs in that Property (D0) is replaced by Property (wD0).

Theorem 9. *Assume M is a reduced functional FST. Then the following three statements are equivalent:*

(1) $\mathcal{T}(M)$ can be realized by a reduced deterministic transducer.
(2) M has Properties (D0), (D1) and (D2).
(3) M is of bounded difference.

It can be decided whether or not the transduction of an FST can be realized by a reduced deterministic one.

Proof. Let $M = (A, T)$ be a reduced functional FST where $A = (Q, \Sigma, \delta, Q_F)$ and $n = \#Q > 0$. We only give the construction showing that (3) implies (1). Assume that M is of bounded difference. We construct a reduced deterministic FST M_d realizing $\mathcal{T}(M)$.

Let $A_0 = (Q_0, \Sigma, \delta_0, Q_{0,F})$ be the standard subset automaton for A, i.e., $Q_0 = 2^Q$; $Q_{0,F} = \{B \subseteq Q | B \cap Q_F \neq \emptyset\}$; and δ_0 consists of all transitions $(B, a, B_1 \dots B_k)$ where $B = \{q \in Q | \forall j \exists q_j \in B_j : (q, a, q_1 \dots q_k) \in \delta\}$.

The idea of the construction is the following. When starting the computation for some input tree t at the leaves, M_d behaves like A_0 but records for every accessible state q that part of the output of some q–computation of which it is not yet certain whether it will be part of the final output or not. This is similar as in the word case. The new difficulty that arises is to handle the start–up phase. Having accumulated sufficiently large possible output trees we have to find the right "end", i.e., that subtree which safely can be produced and leave variables at the right places. This choice is easy if for more than one of the subtrees at the present node large outputs can be produced (cf. Case 3). Otherwise, the size of the subtree's output must be large enough such that it can be uniquely detected (cf. Case 1).

Formally, we construct $M_d = (A_d, T_d)$ with $A_d = (Q_d, \Sigma, \delta_d, Q_{d,F})$ as follows.

Q_d is some set of mappings $\mu : B \to T_\Delta(x_1) \cup \{\bot\}$, $B \in Q_0$ such that $\mu(q)$ is a tree of depth less than $(2n+1) \cdot |M|$. $Q_{d,F}$ consists of all mappings $\mu \in Q_d$ which are defined for some $q \in Q_F$ where $T_d(\mu) = T(q)\mu(q)$, and $U(M_d) = \{\mu \in Q_d | \text{range}(\mu) \subseteq T_\Delta \cup \{\bot\}\}$.

The sets Q_d and δ_d are iteratively determined as the union of their "approximations", i.e., by

$$Q_d = \bigcup_{\nu \geq 0} Q^{(\nu)} \text{ and } \delta_d = \bigcup_{\nu \geq 0} \delta^{(\nu)}$$

where $Q^{(0)} = \emptyset$ and $\delta^{(0)} = \emptyset$. The construction will be done such that the following invariant holds:

(0) Assume $(\mu, u, s, \epsilon) \in T_d$, and B is the domain of μ. Then $(B, u, \epsilon) \in \delta_0$, and for every $q \in B$,
- $(q, u, \mu(q)s, \epsilon) \in T$;
- x_1 occurs in $\mu(q)$ iff x_1 occurs in $\mu(q')$ for every $q' \in B$;
- If x_1 occurs in $\mu(q)$ then $\text{depth}(s) \geq (n+1) \cdot |M|$. Otherwise, $s = \bot$.

Assume $\nu > 0$ and the sets $Q^{(\nu-1)}$ and $\delta^{(\nu-1)}$ already have been defined. Let $\mu_1, \ldots, \mu_k \in Q^{(\nu-1)}$, B_j be the domains of μ_j, $j = 1, \ldots, k$, and $(B, a, B_1 \ldots B_k)$ a transition of the subset automaton A_0. We construct one map $\mu : B \to T_\Delta(x_1) \cup \{\bot\}$ such that $\mu \in Q^{(j)}$ and $\tau = (\mu, a, \mu_1 \ldots \mu_k) \in \delta^{(j)}$ and define $T_d(\tau)$.

By definition of δ_0, for every $q \in B$ some $\tau_q = (q, a, p_{q,1} \cdots p_{q,k}) \in \delta$ exists with $p_{q,j} \in B_j$ for all j. For $q \in B$ let $u_q = T(\tau_q)[\mu_1(p_{q,1})x_1, \ldots, \mu_k(p_{q,k})x_k]$, and let J_q denote the set of all j such that either $\mu_j(p_{q,j}) \in \tilde{T}_\Delta(x_1)$ or has depth $\geq n \cdot |M|$.

Case 1: $u_r \in T_\Delta$ for some $r \in B$.

According to claim (0) for $Q^{(\nu-1)}$, $u_q \in T_\Delta$ for all $q \in B$. If $\text{depth}(u_q) < (2n+1) \cdot |M|$ for all $q \in B$ then we define $\mu(q) = u_q$ and $T_d(\tau) = \bot$.

Otherwise, some $r \in B$ exists with $\text{depth}(u_r) \geq (2n+1) \cdot |M|$. We claim that

(1) $v \in T_\Delta$ and $w_q \in \tilde{T}_\Delta(x_1)$ exist such that for all $q \in B$,
- $u_q = w_q v$;
- $\text{depth}_{x_1}(w_q) < n \cdot |M|$;
- w_q does not contain subtrees from T_Δ of depth $\geq (n+1) \cdot |M|$;
- the maximal common x_1–suffix of $w_q, q \in B$, is x_1.

Under this assumption, we put $\mu(q) = w_q$ and $T_d(\tau) = v$.

Case 2: $\exists j : \mu_j(p_{r,j}) \in \tilde{T}_\Delta(x_1)$ for some $r \in Q$ and $\forall q \in B : \#J_q \leq 1$.

Then by claim (0) for $Q^{(\nu-1)}$, $\mu_j(p_{q,j}) \in \tilde{T}_\Delta(x_1)$ for every $q \in B$. Hence, $J_q = \{j\}$, and $\mu_{j'}(p_{q,j'}) \in T_\Delta$ for all $j' \neq j$. Therefore, $u_q \in \tilde{T}_\Delta(x_j)$ for every

$q \in B$. Moreover, u_q does not contain subtrees in T_Δ of depth $\geq (n+1) \cdot |M|$. Let $u_q = w_q y$ where y is the maximal common x_j–suffix of all u_q. We claim that

(2) For every $q \in B$, $\text{depth}_{x_1}(w_q) < n \cdot |M|$.

Provided Claim (2) holds, we put $\mu(q) = w_q x_1$ and $T_d(\tau) = y$.
Case 3: $\exists j : \mu_j(p_{r,j}) \in \tilde{T}_\Delta(x_1)$ and $\#J_r > 1$ for some $r \in B$.
Again by Claim (0) for $Q^{(\nu-1)}$, $\mu_j(p_{q,j}) \in \tilde{T}_\Delta(x_1)$ for all $q \in B$. We claim:

(3) There exists exactly one $s \in T_\Delta(X_k)$ such that the following holds:
- for all $q \in B$, $u_q = v_q s$ where $v_q \in \tilde{T}_\Delta(x_1)$ with $\text{depth}_{x_1}(v_q) < n \cdot |M|$ such that v_q does not contain subtrees from T_Δ of depth $\geq (n+1) \cdot |M|$;
- s is maximal with this property.

Provided Claim (3) holds, we define $\mu(q) = v_q$ for every $q \in B$, and put $T_d(\tau) = s$. Observe that x_j occurs in s iff $\mu_j(p_{q,j}) \notin T_\Delta$ for some q. Therefore, (provided claim (0) for $Q^{(\nu-1)}$ holds) x_j occurs in $T_d(\tau)$ iff $\mu_j \notin U(M_d)$.

This finishes the construction. In order to prove its correctness we claim:

(4) Assume $(B, u, \epsilon) \in \delta_0$. Then $(\mu, u, s, \epsilon) \in T_d$ for some $\mu \in Q_d$ whose domain is B and where for every $q \in B$, $(q, u, \mu(q)s, \epsilon) \in T$.

This claim together with claim (0) and the definition of $T_d|_{Q_{d,F}}$ show that $\mathcal{T}(M_d) = \mathcal{T}(M)$. Since M_d is reduced deterministic by construction we are done provided Claims (0) through (4) hold.

Claims (0) through (4) must be proven together by induction on the depth of an input tree u. Note that termination of the construction is due to the second parts of claims (1) and (2). □

Analogously to Theorem 9 we find:

Theorem 10. *Assume M is a reduced functional FST. Then the following three statements are equivalent:*

(1) $\mathcal{T}(M)$ can be realized by a deterministic transducer.
(2) M has Properties (wD0), (D1) and (D2).
(3) M is of weakly bounded difference.

It can be decided whether or not the transduction of an FST can be realized by a deterministic one. □

From Theorems 9 and Theorems 10 we conclude that the transduction of a deterministic transducer can be realized by a reduced deterministic one iff the corresponding reduced transducer has Property (D0). We have:

Corollary 11. *Assume M is a deterministic FST. Let M_r be a reduced functional transducer with $\mathcal{T}(M) = \mathcal{T}(M_r)$. Then the following two statements are equivalent:*

(1) $\mathcal{T}(M)$ can be realized by a reduced deterministic transducer.
(2) M_r has Property (D0).

It can be decided in polynomial time whether or not the transduction of a deterministic FST can be realized by a reduced deterministic one. □

Proof. If $\mathcal{T}(M_r)$ can be realized by a reduced deterministic transducer then by Prop. 4, M_r has Property (D0). Conversely, assume M_r has Property (D0). Since $\mathcal{T}(M_r)$ can be realized by a deterministic transducer (namely M), M_r has Properties (D1) and (D2) by Prop. 6 and 8. But then by Theorem 9, $\mathcal{T}(M_r)$ can be realized by a reduced deterministic transducer.

However, the general construction of the equivalent reduced deterministic transducer given in the proof of Theorem 9 can be replaced by a simpler construction here. Let $M_r = (A_r, T_r)$ be the reduced functional transducer with $\mathcal{T}(M_r) = \mathcal{T}(M)$ as constructed in the proof of 2. Since M was deterministic the set $D_r(u)$ of states derivable by some input tree u (w.r.t. A_r) always is a subset of $\{\langle q, 0\rangle, \langle q, 1\rangle\}$ where $(q, u, \epsilon) \in \delta$. Therefore, if $\langle q, 0\rangle \notin D_r(u)$ we safely can produce output whereas as long as $\langle q, 0\rangle \in D_r(u)$ we only produce $\perp$ and memorize a possible output provided $\langle q, 1\rangle \in D_r(u)$.

To implement this idea assume $M = (A, T)$ with $A = (Q, \Sigma, \delta, Q_F)$. We construct $M_d = (A_d, T_d)$ with $A_d = (Q_d, \Sigma, \delta_d, Q_{d,F})$ as follows. Q_d is some set of pairs $Q \times (\{\perp\} \cup T_\Delta \cup \{x_1\})$ such that

1. $\langle q, \perp\rangle \in Q_d$ iff $\langle q, 0\rangle \in Q_r$ but $\langle q, 1\rangle \notin Q_r$; i.e., the output for the presently processed subtree is irrelevant in all cases;
2. $\langle q, y\rangle \in Q_d$ with $y \in T_\Delta$ iff both $\langle q, 0\rangle \in Q_r$ and $\langle q, 1\rangle \in Q_r$; i.e., the output t produced so far possibly is relevant; whereas
3. $\langle q, x_1\rangle \in Q_d$ iff $\langle q, 0\rangle \notin Q_r$ but $\langle q, 1\rangle \in Q_r$; i.e., the output is definitively relevant and already has been produced.

$Q_{d,F}$ consists of all pairs $\langle q, y\rangle \in Q_d$ where $q \in Q_F$, and $U(M_d) = Q_d \cap (Q \times (\{\perp\} \cup T_\Delta))$.

The sets Q_d and δ_d are again iteratively defined as the union of their "approximations", i.e., by

$$Q_d = \bigcup_{\nu \geq 0} Q^{(\nu)} \text{ and } \delta_d = \bigcup_{\nu \geq 0} \delta^{(\nu)}$$

where $Q^{(0)} = \emptyset$ and $\delta^{(0)} = \emptyset$.

Assume $\nu > 0$ and $\langle q_1, y_1\rangle, \ldots, \langle q_k, y_k\rangle \in Q^{(\nu-1)}$. Let $\tau = (q, a, q_1 \ldots q_k) \in \delta$. We construct some $y \in \{\perp\} \cup T_\Delta \cup \{x_1\}$ such that $\langle q, y\rangle \in Q^{(\nu)}$ and $\tau_d = (\langle q, y\rangle, a, \langle q_1, y_1\rangle \ldots \langle q_k, y_k\rangle) \in \delta^{(\nu)}$, and define $T_d(\tau_d)$.

Case 1: $\langle q, 0\rangle \in Q_r$ and $\langle q, 1\rangle \notin Q_r$.

Note that then also for every j, $\langle q_j, 0\rangle \in Q_r$ and $\langle q_j, 1\rangle \notin Q_r$. Hence, all $y_j \in \{\perp\} \cup T_\Delta$. Therefore, we put $y = \perp$ and $T_d(\tau_d) = \perp$.

Case 2: $\langle q, 0\rangle \in Q_r$ and $\langle q, 1\rangle \in Q_r$.

Again, $y_j \in \{\perp\} \cup T_\Delta$ for all j. Moreover, $y_j \neq \perp$ whenever x_j occurs in $T(\tau)$. Therefore, $y = T(\tau)[y_1, \ldots, y_k]$ in deed is in T_Δ. Finally, we put $T_d(\tau_d) = \perp$.

Case 3: $\langle q,0\rangle \notin Q_r$ and $\langle q,1\rangle \in Q_r$.

This is the only case where we have to produce an output pattern different from $\perp$. Define $y'_j = x_j$ if $y_j = x_1$ and $y'_j = y_j$ otherwise. Then put $y = x_1$ and $T_d(\tau_d) = T(\tau)[y'_1, \ldots, y'_k]$.

It is easy to verify that the construction terminates, and that this construction produces a reduced deterministic transducer which is equivalent to M (or M_r). Especially for every $\langle q, y\rangle \in Q_d$, $\text{depth}(y) < \#Q \cdot |M|$.

7 Conclusion

We gave a characterization of functional tree transducers whose transduction can be realized by deterministic or even reduced deterministic transducers. It remained open whether or not these two characterizations can be decided even in polynomial time. At least we were able to prove that it can be decided in polynomial time whether or not the transduction of a deterministic transducer can be realized by a reduced deterministic one.

8 References

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman: *The design and analysis of computer algorithms.* Addison–Wesley, 1974
2. J. Berstel: *Transductions and Context–Free Languages.* Teubner, Stuttgart, 1979
3. J. Engelfriet: Some open questions and recent results on tree transducers and tree languages. In: Formal Language Theory, ed. by R.V. Book, Academic Press 1980, pp. 241–286
4. Chr. Ferdinand, H. Seidl, R. Wilhelm: Tree automata for code selection. In: R. Giegerich, S.L. Graham (eds): *Code Generation – Concepts, Tools, Techniques.* Springer, Workshops in Computing, pp. 31–50, 1992
5. F. Gecseg, M. Steinby: Tree automata. Akademiai Kiado, Budapest, 1984
6. R. Giegerich, K. Schmal: Code selection techniques: pattern matching, tree parsing and inversion of derivors. Proc. of ESOP 1988, LNCS 300 pp. 245–268
7. C. Reutenauer: Subsequential Functions: Characterizations, Minimization, Examples. Proc. IMYCS 1990, LNCS 464, pp. 62–79
8. H. Seidl: Equivalence of finite–valued bottom–up finite state tree transducers is decidable. Proc. CAAP '90, LNCS 431, pp. 269–284, 1990; extended version to appear in: Math. Syst. Theory
9. H. Seidl: Single–valuedness of tree transducers is decidable in polynomial time. To appear in: TCS, special issue of CAAP 90
10. A. Weber, R. Klemm:: Economy of Description for Single–valued Transducers. Preprint J.W.Goethe–Universität, Frankfurt/Main, 1991
11. Z. Zachar: The solvability of the equivalence problem for deterministic frontier–to–root tree transducers. Acta Cybernetica 4 (1978), pp. 167–177