Theoretical Computer Science 106 (1992) 135–181
Elsevier

Single-valuedness of tree transducers is decidable in polynomial time*

Helmut Seidl**

Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald 15, D-6600 Saarbrücken 11, Germany

Abstract

Seidl, H., Single-valuedness of tree transducers is decidable in polynomial time, Theoretical Computer Science 106 (1992) 135–181.

A bottom-up finite-state tree transducer (FST) A is called single-valued iff for every input tree there is at most one output tree.

We give a polynomial-time algorithm which decides whether or not a given FST is single-valued. The algorithm is based on:

- the freedom of the submonoid of trees which contain at least one occurrence of one variable $*$;
- the succinct representation of trees by graphs;
- a sequence of normalizing transformations of the given transducer; and
- a polynomially decidable characterization of pairs of equivalent output functions.

We apply these methods to show that finite-valuedness is decidable in polynomial time as well.

0. Introduction

A bottom-up finite-state tree transducer (FST) is a finite-state device which produces its output tree while consuming a given input tree in a bottom-up fashion. Since multiple occurrences of variables in patterns are allowed, an FST is able to generate several identical copies of images of subtrees. Since some variables can be missing, the image of a correctly parsed subtree may be skipped again.

In compiler construction finite-state transducers are an important tool for manipulating abstract syntax trees [7]. A good survey on tree automata theory and its applications is found in [6]. Formally, FSTs can be viewed as one possible

*The present paper is a long version of the talk [14] presented at the Toyohashi Symposium on Theoretical Computer Science.

**Partially supported by Deutsche Forschungsgemeinschaft SFB 124 and ASMICS.

generalization of generalized sequential machines (GSMs). Therefore, the natural question arises whether or not results about GSMs can be extended to FSTs.

Until recently, the knowledge about finite-valued finite tree transducers was comparatively poor. In 1978 Zachar showed that equivalence is decidable for deterministic FSTs [20]. In 1980 Engelfriet exhibited a nice generalization (T1) of a word lemma by Schützenberger [10] to trees which allows to decide whether or not a given FST is single-valued [4]. Note that any algorithm which decides single-valuedness can be used to decide equivalence of single-valued FSTs. Both Zachar's and Engelfriet's papers are not concerned with algorithmic complexity. In [13] a theory of finite-valued FSTs is developed. Especially, it is proved that for every $k \geqslant 1$ it is decidable in nondeterministic polynomial time whether a given FST is *not* k-valued. Two necessary and sufficient conditions (F1) and (F2) are exhibited for an FST to be finite-valued and it is shown that it also can be decided in nondeterministic polynomial time whether they do *not* hold. It remained open whether or not these questions can be decided even in deterministic polynomial time. Our interest in finite-valued FSTs in [13] is due to the fact that equivalence of finite-valued FSTs is decidable [13], although equivalence is undecidable in general.

In this paper we construct a *deterministic* polynomial-time algorithm which decides whether or not a given FST is single-valued. Especially, this implies that the equivalence of deterministic FSTs can be decided in deterministic polynomial time as well. This result is obtained by a rather involved investigation of the structural properties of single-valued FSTs. We also succeed in constructing a deterministic polynomial-time algorithm deciding finite-valuedness of FSTs. It remains open whether or not k-valuedness for $k > 1$ is solvable in deterministic polynomial time as well.

The paper is organized as follows. As in [13] we start by an investigation of the combinatorics of trees. Especially, we prove that the submonoid of trees containing *at least one* occurrence of a variable x is free.

The efficiency of our algorithms is based on a succinct representation of trees of possibly exponential size by graphs. Therefore, in Section 2 we formally introduce and study (finite ordered acyclic rooted labeled) graphs together with their relation to (tuples of) trees. We describe the basic algorithms for them. This section may be skipped at first reading and only be consulted when wondering about the implementation of the algorithmic ideas.

In Section 3 we introduce bottom-up finite-state tree transducers (FSTs). We recall the notion of reducedness from [13]. Additionally, we introduce an even stronger normal form which says that an output is "delayed" as long as possible. FSTs with this property are called strongly reduced. We show that for every reduced FST we can find in linear time an equivalent strongly reduced FST.

Instead of considering one output function and two accepting computations it is more convenient to study one accepting computation and two output functions. The corresponding formal device $\Pi = (A, T_1, T_2)$ consisting of one finite tree automaton together with two output functions is called pairing. Section 4 studies pairings. Especially, we exhibit necessary Properties (U1) and (U2) of a strongly reduced pairing

Π such that the outputs for every accepting computation w.r.t. T_1 and T_2 are equal.

In Section 5 we analyze the behavior of a strongly reduced pairing having Properties (U1) and (U2) on paths of an input tree. The outputs now can be viewed as elements in some finitely generated free tree monoid! This allows to give a third necessary condition (U3′) for a strongly reduced pairing to have equivalent output functions. It turns out that Properties (U1), (U2) and (U3′) together are not only necessary but also sufficient for the output functions T_i to be equivalent. Finally, we give an equivalent formulation (U3) of (U3′) as a graph property. Property (U3) is the (by no means trivial) generalization of the usual equivalence of outputs of GSMs. Since all three Properties (U1), (U2) and (U3) can be decided in deterministic polynomial time we obtain a deterministic polynomial-time algorithm deciding whether or not an FST is single-valued. We apply this method to derive for every $m \geqslant 1$ a deterministic polynomial-time algorithm which decides whether or not two single-valued FSTs are equivalent provided the underlying tree automata are "m-ambiguous". Since every deterministic FST is unambiguous, this result especially holds for deterministic FSTs.

Finally, in Section 6 we apply the methods of Section 5 to prove that it can be decided in deterministic polynomial time whether or not an FST is finite-valued. This is done by successively considering a sequence of sets of properties each of which characterizes finite-valuedness.

1. Trees

In this section we give basic definitions and state some fundamental properties about trees. We prove that the monoid $\tilde{T}_\Sigma(x)$ of trees containing at least one occurrence of the variable x is free (Theorem 1.3). 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 $m = 0$.

The *depth* of a tree $t \in T_\Sigma$, depth(t), is defined by $\text{depth}(t) = 0$ if $t \in \Sigma_0$, and $\text{depth}(t) = 1 + \max\{\text{depth}(t_1), \ldots, \text{depth}(t_m)\}$ if $t = a(t_1, \ldots, t_m)$ for some $a \in \Sigma_m$, $m > 0$. 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 $a \in \Sigma_m$, $m > 0$.

The set of *nodes of* t, $O(t)$ is the subset of $\mathbb{N}^*$ defined by $O(t) = \{\varepsilon\} \cup \bigcup_{j=1}^{m} j \cdot O(t_j)$, where $t = a(t_1, \ldots, t_m)$ for some $a \in \Sigma_m$, $m \geqslant 0$. Note that the cardinality of $O(t)$, $\# O(t)$, equals $|t|$.

t defines maps $t(\_) : O(t) \to \Sigma$ and $t/\_ : O(t) \to T_\Sigma$ mapping the nodes o of t to their labels or the subtree of t with root o, respectively. We have

$$t(o) = \begin{cases} a & \text{if } o = \varepsilon, \\ t_j(o') & \text{if } o = j \cdot o', \end{cases} \quad \text{and} \quad t/o = \begin{cases} t & \text{if } o = \varepsilon, \\ t_j/o' & \text{if } o = j \cdot o'. \end{cases}$$

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, $T_\Sigma\subseteq T_\Sigma(X)$.) Assume $t\in T_\Sigma(X)$. For $x\in X$ the set $O_x(t)$ of *occurrences* of x is the set $\{o\in O(t)\mid t(o)=x\}$. t is called *X-proper* iff every $x\in X$ occurs in t exactly once, i.e. $\# O_x(t)=1$ for every x. 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 $t\theta=a(t_1\theta,\dots,t_m\theta)$ if $t=a(t_1,\dots,t_m)$ with $a\in\Sigma$. θ is called *X-substitution* or simply substitution if X is understood. If $X=\{x_j,\dots,x_m\}$ (i.e. the variables are indexed by some interval of natural numbers) and $x_i\theta=t_i$, we denote $t\theta$ also by $t[t_j,\dots,t_m]$. 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 $t_1\theta=t_1t_2$. 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, $t_1t_2=t_1$ if t_1 does not contain an occurrence of x.

Proposition 1.1 (originating from [4] and cited from [13]) states basic properties of $T_\Sigma(x)$.

Proposition 1.1. *Assume* $s_1,s_2,t_1,t_2,t_1',t_2'\in T_\Sigma(x)$.

(i) *Bottom Cancellation: Assume* $t_1\neq t_1'$. *Then*

$$s_1t_1=s_2t_1 \textit{ and } s_1t_1'=s_2t_1' \textit{ implies } s_1=s_2.$$

(ii) *Top Cancellation: Assume x occurs in s_1. Then*

$$s_1t_1=s_1t_1' \textit{ implies } t_1=t_1'.$$

(iii) *Factorization: Assume $t_i\neq t_i'$ for $i=1,2$. Then*

$$s_1t_1=s_2t_2 \textit{ and } s_1t_1'=s_2t_2' \textit{ implies } \exists r: s_1r=s_2 \textit{ or } s_1=s_2r.$$

In case the second factors contain variables as well, both bottom cancellation and factorization have a much simpler form. 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. We obtain:

Corollary 1.2. *Assume* $s_1,s_2,t_1,t_2\in\tilde{T}_\Sigma(x)$.

(i) *Bottom Cancellation:* $s_1t_1=s_2t_1$ *implies* $s_1=s_2$.

(ii) *Top Cancellation:* $s_1t_1=s_1t_2$ *implies* $t_1=t_2$.

(iii) *Factorization:* $s_1t_1=s_2t_2$ *implies* $\exists r\in\tilde{T}_\Sigma(x)$: $s_1r=s_2$ *or* $s_1=s_2r$.

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 $j > 1$.

Employing Corollary 1.2 we find the main result of this section:

Theorem 1.3. (i) *Every tree t in $\tilde{T}_\Sigma(x)$ can be written as a (possibly empty) product $t = u_1 \ldots u_k$ for x-irreducible trees $u_1, \ldots, u_k \in I_\Sigma(x)$.*

(ii) *If $t = u_1 \ldots u_k$ and $t = v_1 \ldots v_{k'}$ for $u_1, \ldots, u_k, v_1, \ldots, v_{k'} \in I_\Sigma(x)$, then $k = k'$ and $u_\kappa = v_\kappa$ for all κ.*

(iii) *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 $u_3 = b(x)$ (see Fig. 1).

Theorem 1.3 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 $|t|_x \leqslant \text{depth}(t)$ and $|t|_x = |o|$ if t contains exactly one occurrence of x and o is the unique leaf with this label. If $t = u_1 u_2 \in \tilde{T}_\Sigma(x) = I_\Sigma(x)^*$ then u_1 is called an *x-prefix* of t. Accordingly, u_2 is called *x-suffix* of t.

The rest of this section is concerned with trees (possibly) containing occurrences of more than one variable. It is for this case where we have to distinguish irreducibilities corresponding to different variables. Define $X_k = \{x_1, \ldots, x_k\}$ and assume $*$ is a variable not in X_k. By the independence of substitutions into different variables we have:

Proposition 1.4. *Assume $v \in \tilde{T}_{\Sigma \cup X_k}(*)$ and $u_j \in \tilde{T}_\Sigma(x_j)$ for $j = 1, \ldots, k$. Then the following holds:*

(i) *If $v[u_1, \ldots, u_k] = v_1 v_2$ then $v = s_1 s_2$ for suitable trees $s_i \in T_{\Sigma \cup X_k}(*)$ such that $v_1 = s_1[u_1, \ldots, u_k]$ and $v_2 = s_2[u_1, \ldots, u_k]$.*

(ii) *v is $*$-irreducible iff $v[u_1, \ldots, u_k]$ is $*$-irreducible.*

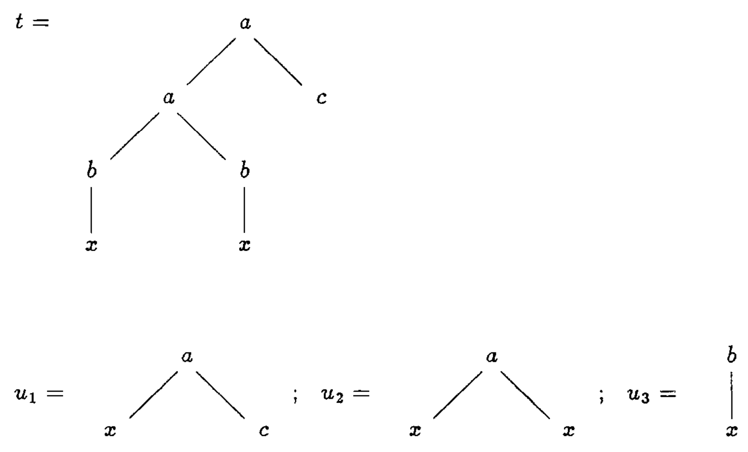

Fig. 1.

Proof. First consider statement (i). Define $t=v[u_1,\ldots,u_k]$. Clearly, $O_*(v_1)\subseteq O(t)$ and $v_2=t(o)$ for every $o\in O_*(v_1)$. Since v_2 contains at least one occurrence of $*$, v_2 cannot be a subtree of any of the u_j. Hence, in fact,

$$O_*(v_1)\subseteq O(v).$$

Moreover, $v_2=s_2[u_1,\ldots,u_k]$, where $s_2=v/o$ for some $o\in O_*(v)$. Let $s'=v/o'$ for some other node o' in $O_*(v_1)$. Then

$$s'[u_1,\ldots,u_k]=v_2=s_2[u_1,\ldots,u_k].$$

Since u_j contains at least one occurrence of x_j, x_j occurs in v_2 iff x_j occurs in s_2 iff x_j occurs in s'. Therefore, s and s' contain occurrences of the same variables. Without loss of generality (w.l.o.g.) these are all variables $x_1,\ldots,x_k$ and $k>0$. Applying bottom cancellation according to Corollary 1.2 iteratively w.r.t. variables $x_1,\ldots,x_k$ we deduce:

(1) $\quad s_2=v/o \quad$ for all $o\in O_*(v_1)$.

v_1 is trivially obtained from t by replacing all subtrees at nodes in $O_*(v_1)$ with $*$. Therefore, define s_1 as the tree obtained from v by replacing all subtrees at nodes in $O_*(v_1)$ with $*$. It follows that $s_1[u_1,\ldots,u_k]=v_1$ and, hence, by (1)

$$v[u_1,\ldots,u_k]=t=(s_1[u_1,\ldots,u_k])s_2[u_1,\ldots,u_k]=(s_1s_2)[u_1,\ldots,u_k].$$

Again, v and s_1s_2 contain occurrences of the same variables. Hence, by bottom cancellation, $v=s_1s_2$, which proves statement (i). Statement (ii) follows from (i) and the observation that for every $s\in\tilde{T}_{\Sigma\cup X_k}(*)$, $s=*$ iff $s[u_1,\ldots,u_k]=*$. □

Proposition 1.5. *Assume $t\in T_\Sigma(X_k)$ contains at least one occurrence of some variable x_j. Then $t=t_0t'$ for some $t_0\in\tilde{T}_\Sigma(*)$ such that the following hold.*

(1) *If $t=s_0s'$ for some $s_0\in\tilde{T}_\Sigma(*)$ then $t_0=s_0r$ for some $r\in\tilde{T}_\Sigma(*)$, i.e. t_0 is the uniquely determined maximal prefix of t in $\tilde{T}_\Sigma(*)$.*

(2) *Assume t contains also an occurrence of some variable $x_{j'}$ with $j'\neq j$ as well, $u_\kappa\in\tilde{T}_\Sigma(x_\kappa)$ for $\kappa=1,\ldots,k$, then t_0 is the maximal prefix of $t[u_1,\ldots,u_k]$ in $\tilde{T}_\Sigma(*)$ as well.*

Proof. Define $S=\{u_0\in\tilde{T}_\Sigma(*)\mid t=u_0u'\}$. Then $S\neq\emptyset$ since $*\in S$. Moreover, s is finite since $|u|\leqslant|t|$ for every $u\in S$. For $u_1,u_2\in S$, $t=u_1t_1$ and $t=u_2t_2$ for some $t_1,t_2\in T_\Sigma(X_k)$. Since t contains an occurrence of x_j, both t_1 and t_2 contain occurrences of x_j. Therefore, we may apply factorization and obtain $u_1=u_2r$ or $u_1r=u_2$ for some $r\in T_\Sigma(*)$. Since u_i contain occurrences of $*$, r also contains an occurrence of $*$. Hence, S is totally ordered by the $*$-prefix relation on $\tilde{T}_\Sigma(*)=I_\Sigma(*)^*$. Consequently, S contains exactly one maximal element. This proves (1).

For a proof of (2) assume $t=t_0v$, where t_0 is the maximal prefix according to statement (1). Then $t[u_1,\ldots,u_k]=t_0v[u_1,\ldots,u_k]$. Hence, t_0 is a prefix of $t[u_1,\ldots,u_k]$ in $\tilde{T}_\Sigma(*)$. For a contradiction assume that t_0 is not maximal. Then $v[u_1,\ldots,u_k]=v_1v_2$ for some v_1 in $\tilde{T}_\Sigma(*)$ and $v_2\in\tilde{T}_\Sigma(X_k)$. By Proposition 1.4(i), $v=s_1s_2$, where $s_1[u_1,\ldots,u_k]=v_1$. Since v_1 does not contain variables x_j, s_1 in fact equals v_1. Hence, t_0v_1 is a prefix of t in $\tilde{T}_\Sigma(*)$, contradicting the maximality of t_0. □

Similar to prefixes we can determine maximal x_j-suffixes of a tree $t \in T_\Sigma(X_k)$. The proof of the next proposition follows from the freedom of the monoids $\tilde{T}_{\Sigma \cup X_k \setminus \{x_j\}}(x_j)$, $j=1,\dots,k$.

Proposition 1.6. *Assume* $t \in T_\Sigma(X_k)$, *where* $k>1$ *and* t *contains at least one occurrence of every variable* x_j. *Then* $t=s[t_1,\dots,t_k]$ *with* $t_j \in \tilde{T}_\Sigma(x_j)$ *for* $j=1,\dots,k$ *such that for every decomposition* $t=s'[t'_1,\dots,t'_k]$, *where* $t'_j \in \tilde{T}_\Sigma(x_j)$, t'_j *is a* x_j*-suffix of* t_j *for* $j=1,\dots,k$.

Thus, every tree $t \in T_\Sigma(X_k)$, $k>1$, which contains at least one occurrence of every variable x_j can be decomposed uniquely as $t=t_0 s[t_1,\dots,t_k]$, where t_0 is the maximal prefix of t in $\tilde{T}_\Sigma(*)$ and t_j are the maximal x_j-suffixes of t. The tree $s \in T_\Sigma(X_k)$ is called *kernel* of t and the decomposition *kernel decomposition*. In case $k=1$ we define the kernel decomposition of t by $t=t_0 s[t_1]$, where $t_0=t[*]$, $s=x_1$ and $t_1=x_1$. For an example consider tree $t=b\,d(c,a(a(x_2,x_2),bx_1),bx_1)$. Then the kernel decomposition of t is $t=t_0 s[t_1,t_2]$, where $s=d(c,a(x_2,x_1),x_1)$ and $t_0=b*$, $t_1=bx_1$, $t_2=a(x_2,x_2)$ (see Fig. 2).

Trees $t,t' \in T_\Sigma(X_k)$ are called *comparable* ($t \approx t'$) iff

(C) trees $v_j, v'_j \in \tilde{T}_\Sigma(x_j)$, $j=1,\dots,k$ exist with $t[v_1,\dots,v_k]=t'[v'_1,\dots,v'_k]$.

Apparently, condition (C) is equivalent to

(C′) $t=s[u_1,\dots,u_k]$ and $t'=s[u'_1,\dots,u'_k]$ for some $s \in T_\Sigma(X_k)$ and $u_j, u'_j \in \tilde{T}_\Sigma(x_j)$, where for every j either $u_j=x_j$ or $u'_j=x_j$.

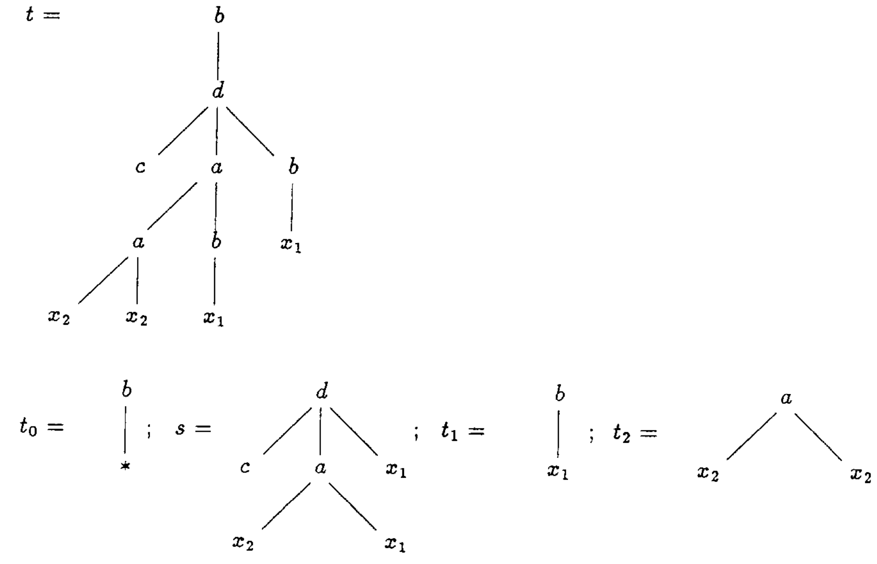

Fig. 2.

Thus, e.g., trees $a(x_1,bx_2)$ and $a(a(x_1,x_1),x_2)$ are comparable but the trees $a(a(x_1,x_1),x_2)$ and $a(a(x_1,x_2),x_2)$ are not.

Especially, observe that $t\approx t'$ implies that $x\in X_k$ occurs in t iff x occurs in t' iff x occurs in s. We have:

Fact 1.7. *Assume $t,t'\in T_\Sigma(X_k)$ are comparable and $1\leqslant j<j'\leqslant k$. Then no occurrence $o\in O_{x_j}(t)$ is a prefix of an occurrence $o'\in O_{x_{j'}}(t')$ and vice versa.*

Fact 1.8. *Assume $t_1,t_2\in T_\Sigma(X)$ contain occurrences of every $x\in X$, and θ_i, $i=1,2$, are X-substitutions with $x\theta_i\in \tilde{T}_\Sigma(x)$ and either $x\theta_1=x$ or $x\theta_2=x$ for every x. Then the following three statements are equivalent:*

(1) *$t_i=s\theta_i$ for some $s\in T_\Sigma(X)$;*

(2) *$t_1\theta_2=t_2\theta_1$.*

(3) *If $t_i=u_is_i\theta'_i$ is the kernel decomposition then $u_1=u_2$, $s_1=s_2$ and $x\theta'_1\theta_2=x\theta'_2\theta_1$ for every $x\in X$.*

Proof. Certainly, $\theta_1\theta_2=\theta_2\theta_1$. Therefore, assuming (1) we can conclude:

$$t_1\theta_2=(s\theta_1)\theta_2=s(\theta_1\theta_2)=s(\theta_2\theta_1)=(s\theta_2)\theta_1=t_2\theta_1.$$

Hence, (1) implies (2). The reverse implication follows with induction on the cardinality of X by bottom cancellation. Finally, the equivalence of (2) and (3) follows from Propositions 1.5 and 1.6. □

Fact 1.9. *Assume $t_1,t_2\in \tilde{T}_{\Sigma\cup X}(*)$, where $t_1=u_1\ldots u_k$ and $t_2=v_1\ldots v_kv_{k+1}\ldots v_{k+r}$ are the decompositions of t_i into $*$-irreducible factors. If $t_1\approx t_2$ then:*

(i) *$u_j\approx v_j$ for $j=1,\ldots,k$;*

(ii) *$v_{k+\rho}\in I_\Sigma(*)$ for $\rho=1,\ldots,r$.*

Proof. Induction on k. If $k=0$ then statements (i) and (ii) trivially hold. Therefore, assume $k>0$. For the inductive step we consider $t_1=u_1t'_1$ and $t_2=v_1t'_2$, where u_1 and v_1 are $*$-irreducible. It suffices to prove that

$$(*)\qquad u_1\approx v_1 \quad \textit{and} \quad t'_1\approx t'_2.$$

Since $t_1\approx t_2$ trees $r_i\in\tilde{T}_\Sigma(*)$ and X-substitutions τ_i, $i=1,2$, exist such that $t_1\theta_1r_1=t_2\theta_2r_2$, where for every $x\in X$, $x\theta_i\in\tilde{T}_\Sigma(x)$. We have

$$t_1\theta_1r_1=(u_1t'_1)\theta_1r_1=(u_1\theta_1)(t'_1\theta_1r_2)$$

and, likewise,

$$t_2\theta_2r_2=(v_1t'_2)\theta_2r_2=(v_1\theta_2)(t'_2\theta_2r_2)$$

By Proposition 1.4(i), both $u_1\theta_1$ and $v_2\theta_2$ are $*$-irreducible. Since $\tilde{T}_{\Sigma\cup X}(*)$ is free, it follows that $u_1\theta_1=v_1\theta_2$ and $t'_1\theta_1r_1=t'_2\theta_2r_2$. This implies statement $(*)$. □

2. Graphs

The efficiency of the algorithms to be explained in the following sections is based on the succinct representation of trees of (possibly) exponential size by graphs. In this section we give the theoretical justification for doing so. We introduce graphs and give algorithms for basic tests and operations on trees which work on representations of trees by graphs instead of trees themselves. Finally, in Propositions 2.2–2.5 we describe the essential graph algorithms which serve as subroutines in our deterministic polynomial-time algorithms deciding single-valuedness and finite-valuedness.

Assume Σ is a ranked alphabet. A (Σ-labeled, ordered, rooted) *graph* g is a 4-tuple (V, r, λ, E), where V is the set of *vertices* or *nodes*, $r \in V^*$ is the *root word*, $\lambda : V \to \Sigma$ is the *labeling*, and $E : V \to V^*$ is the *successor function* of g, where $E(v) \in V^m$ iff $\lambda(v) \in \Sigma_m$. If $E(v) = v_1 \ldots v_m$ for $v_j \in V$ then $E_j(v) = v_j$ is the jth successor of v. The *size* of g, $|g|$, is just the number of its nodes plus the length of r, i.e. $|g| = \# V + |r|$. A *graph homomorphism* $h : g \to g'$ for graphs $g = (V, r, \lambda, E)$ and $g' = (V', r', \lambda', E')$ is a mapping $h : V \to V'$ such that $h(r) = h(r')$; $\lambda'(h(v)) = \lambda(v)$ and $h(E(v)) = E'(h(v))$ for every $v \in V$ (i.e. h is compatible with roots, labeling and successors). Two graphs g, g' are *isomorphic* iff there is a bijective graph homomorphism $h : g \to g'$. We write (in abuse of the equality sign) $g = g'$.

Let $\Sigma_B = \{(a, j) \mid a \in \Sigma,\ 1 \leqslant j \leqslant \rho(a)\}$. A triple $\langle v, w, v' \rangle \in V \times \Sigma_B^* \times V$ is called *path* in g from v to v' iff either $w = \varepsilon$ and $v = v'$ or $w = w'(a, j)$ for some $(a, j) \in \Sigma_B$ such that $\langle v, w', v'' \rangle$ is a path from v to some node v'' labeled with a and $E_j(v'') = v'$. Assume $r = r_1 \ldots r_m$. g is called (*root*) *connected* iff for every node v there is a path from some root r_j to v; g is called *acyclic* iff for every node v the only path from v to v is $\langle v, \varepsilon, v \rangle$.

If not stated otherwise, we henceforth assume that our graphs are Σ-labeled, ordered, rooted, connected and acyclic. Let G_Σ^m denote the set of all graphs $g = (V, r, \lambda, E)$, where $r \in V^m$. Define $G_\Sigma^* = \bigcup_{m \geqslant 0} G_\Sigma^m$ and $G_\Sigma = G_\Sigma^1$. Graphs in G_Σ are used for succinct representations of trees in T_Σ, whereas graphs from G_Σ^m are used to represent sequences of trees of length m.

The *subgraph* of a graph $g \in G_\Sigma^*$ with root word $w = w_1 \ldots w_k$ is the graph $g_w = (V_w, w, \lambda_w, E_w) \in G_\Sigma^k$, where $V_w = \{w_1, \ldots, w_k\} \cup \bigcup_{j=1}^k V_{E(w_j)}$. λ_w and E_w are the restrictions of λ and E to V_w, respectively. Since g is assumed to be acyclic, this is in fact a definition.

For $g = (V, r, \lambda, E) \in G_\Sigma$ with $r \in V$ the tree $t(g)$ *represented* by g is recursively defined as follows. If $V = \{r\}$ and $\lambda(r) = a$ then $t(g) = a$. If $V \neq \{r\}$, $\lambda(r) = a$ and $E(r) = v_1 \ldots v_m$ then $t(g) = a(t(g_{v_1}), \ldots, t(g_{v_m}))$. Accordingly, every graph $g = (V, r_1 \ldots r_m, \lambda, E) \in G_\Sigma^m$ with $r_i \in V$ represents the m-tuple $t(g) = \langle t(g_{r_1}), \ldots, t(g_{r_m}) \rangle \in (T_\Sigma)^m$.

Any sequence of trees $s = \langle s_1, \ldots, s_m \rangle$ in T_Σ^m can also be viewed as a graph $\tilde{s} = (V, r, \lambda, E)$, where $V = \{\langle j, o \rangle \mid o \in O(s_j)\}$, $r = \langle 1, \varepsilon \rangle \cdots \langle m, \varepsilon \rangle$, $\lambda(\langle j, o \rangle) = s_j(o)$ and $E(\langle j, o \rangle) = \langle j, o \cdot 1 \rangle \cdots \langle j, o \cdot m \rangle$ provided $s_j(o) \in \Sigma_m$. Clearly, $s = t(\tilde{s})$, i.e. $\tilde{s}$ is just another description of s. Therefore, we will not distinguish between s and $\tilde{s}$ and view T_Σ^m as a subset of G_Σ^m.

Define a partial ordering $\geqslant$ on G_Σ^m as follows. Assume $g = (V, r, \lambda, E)$ and $g' = (V', r', \lambda', E')$ are graphs. Then $g \geqslant g'$ iff there is a surjective graph homomorphism

$h: g \to g'$. The maximal elements w.r.t. $\geqslant$ are the sequences of trees in T_Σ^m and $\boldsymbol{t}(g) \geqslant g$ for every graph g. However, there are also unique minimal elements w.r.t. this ordering.

Proposition 2.1. (i) *There is a unique minimal element* $\boldsymbol{m}(g)$ *in the set* $\{g' \mid g \geqslant g'\}$ *for every graph* g. $\boldsymbol{m}(g)$ *can be computed from* g *in linear time (on a RAM with uniform cost measure).*

(ii) *For every two graphs* g *and* g', $\boldsymbol{t}(g) = \boldsymbol{t}(g')$ *iff* $\boldsymbol{m}(g) = \boldsymbol{m}(g')$.

Proof. For a proof of (i) observe that $\boldsymbol{m}(g)$ is obtained from g by "collapsing" isomorphic subgraphs of g as much as possible. A detailed description of the algorithm can be found in [3]. It works as follows.

(1) It levels the nodes according to the maximal distance to a leaf i.e. a node v with $E(v) = \varepsilon$.

(2) For every level it collects all nodes v having the same label $\lambda(v)$ and isomorphic subgraphs $g_{E_j(v)}$ and identifies them.

Step (1) can be executed by a RAM in linear time. Some kind of bucket sort can be used to implement step (2). Provided $\max\{\rho(a) \mid a \in \Sigma\}$ is bounded by some constant (which always can be assumed in our context) and $\# \Sigma$ is polynomial in $\# V$, step (2) takes time $O(\# V)$. This proves (i).

To prove one direction of (ii) assume $\boldsymbol{t}(g) = \boldsymbol{t}(g') = t$. Both $t \geqslant g$ and $t \geqslant g'$. Therefore, $\boldsymbol{m}(t) = \boldsymbol{m}(g) = \boldsymbol{m}(g')$. For the opposite direction simply observe that $g \geqslant g'$ implies $\boldsymbol{t}(g) = \boldsymbol{t}(g')$. □

From Proposition 2.1 we conclude that any graph g with $\boldsymbol{m}(t) \leqslant g \leqslant t$ can be used to represent the sequence t of trees. The minimal representative $\boldsymbol{m}(t)$ is also called *subtree graph* of t.

Proposition 2.2. *Assume* $t_1, \ldots, t_m, s_1, \ldots, s_k \in T_\Sigma(X_k)$, *where* $g = \boldsymbol{m}(\langle t_1, \ldots, t_m \rangle)$ *and* $g' = \boldsymbol{m}(\langle s_1, \ldots, s_k \rangle)$. *Then* $\bar{g} = \boldsymbol{m}(\langle t_1[s_1, \ldots, s_k], \ldots, t_m[s_1, \ldots, s_k] \rangle)$ *can be computed from* g *and* g' *in linear time.*

Proof. Assume $r = r_1 \ldots r_m$ is the root word of g and $r' = r'_1 \ldots r'_k$ is the root word of g'. Define $g^{(1)}$ as the graph obtained from g and g' as follows. For $j = 1, \ldots, m$, identify the node in g labeled with x_j with r'_j. Choose (the equivalence classes of) $r_1 \ldots r_m$ as the new root word and remove all nodes no longer reachable from these. Finally, define $\bar{g} = \boldsymbol{m}(g^{(1)})$. $g^{(1)}$ can be computed in linear time. Hence by Proposition 2.1, $\bar{g}$ can be computed in linear time as well. □

For this algorithm to work it is not necessary that g and g' are disjoint graphs. In fact, we may consider the case where g and g' are identical. In this case we compute "repeated substitution".

Assume $t=\langle t_1,\ldots,t_m\rangle\in T_\Sigma(X_m)$ and θ_t is the X_m-substitution defined by $x_j\theta_t=t_j$. Let G_t denote the digraph (V,E) with $V=\{1,\ldots,m\}$ and $E=\{\langle i,j\rangle \mid x_j \text{ occurs in } t_i\}$. For $n\geqslant 1$, define t^n inductively by $t^1=t$ and $t^n=(t^{n-1})\theta_t$ for $n>1$. If G_t is acyclic then $t^n=t^m$ for every $n\geqslant m$. The next proposition shows that this limit can be computed in polynomial time as well.

Proposition 2.3. *Assume* $t\in T_\Sigma(X_m)^m$ *and* $g=\boldsymbol{m}(t)$. *If* G_t *is acyclic then a graph* g^m *can be computed in linear time with* $g^m=\boldsymbol{m}(t^m)$. □

The next proposition allows to factor a representation of trees $t\in\tilde{T}_\Sigma(x)$ into representations of x-irreducible trees.

Proposition 2.4. (i) *Assume* $t\in\tilde{T}_\Sigma(*)$, $g=\boldsymbol{m}(t)$ *and* $t=u_1\ldots u_k$ *is the factorization of* t *into* x-*irreducible factors* $u_j\in\tilde{T}_\Sigma(*)$. *Then* $\boldsymbol{m}(\langle u_1,\ldots,u_k\rangle)$ *can be computed from* g *in linear time.*

(ii) *Assume* $t=\langle t_1,\ldots,t_m\rangle$, *where* $t_\mu\in\tilde{T}_\Sigma(x_\mu)$ *for every* μ, $g=\boldsymbol{m}(t)$, *and* $t_\mu=u_{\mu,1}\ldots u_{\mu,k_\mu}$ *is the decomposition of* t_μ *into* x-*irreducible factors* $u_{\mu,j}\in\tilde{T}_\Sigma(x_\mu)$. *Then* $\boldsymbol{m}(\langle u_{1,1},\ldots,u_{1,k_1},\ldots,u_{m,1},\ldots,u_{m,k_m}\rangle)$ *can be computed from* g *in linear time.*

Proof. We only prove statement (i). Since $g=\boldsymbol{m}(t)$, g contains a unique leaf v labeled by $*$. Let r be the root of g. A node v' of g *factors* v iff every path $\langle r,w,v\rangle$ can be factored into a path $\langle r,w_1,v'\rangle$ and a path $\langle v',w_2,v\rangle$ with $w_1w_2=w$. We first consider the following.

(1) If $t=t_1t_2$ then $O_*(t_1)=\boldsymbol{m}^{-1}(v')$ for some node v' factoring v.

(2) For a node v' in g define $g^c_{v'}$ as the graph obtained from g by replacing node v' with a node labeled $*$ and removing all nodes which are no longer reachable from the root. Then

$$t=\boldsymbol{t}(g^c_{v'})\boldsymbol{t}(g_{v'}) \text{ iff } v' \text{ factors } v.$$

(1): Since $t/o=t/o'$ for every $o,o'\in O_*(t_1)$, all nodes in $O_*(t_1)$ are mapped to the same node v' in g. To show that v' factors v consider a path $\langle r,w,v\rangle$ in g. Since $\boldsymbol{m}$ is a surjective graph homomorphism a path $\langle\varepsilon,w,l\rangle$ in t exists, where l is a leaf labeled with $*$. Since $t=t_1t_2$, this path can be factored into paths $\langle\varepsilon,w_1,o\rangle$ and $\langle o,w_2,l\rangle$ for some node $o\in O_*(t_1)$ with $w_1w_2=w$. Clearly, $\boldsymbol{m}(o)=v'$ and $\boldsymbol{m}(l)=v$. Therefore, applying $\boldsymbol{m}$ to these paths we obtain paths $\langle r,w_1,v'\rangle$ and $\langle v',w_2,v\rangle$ in g with $w_1w_2=w$. Since the path $\langle r,w,v\rangle$ was arbitrary, v' factors v.

(2): Consider graphs $g_1=g^c_{v'}$ and $g_2=g_{v'}$. First assume $t=\boldsymbol{t}(g_1)\boldsymbol{t}(g_2)$ but v' does not factor v. Then there is a path $\langle r,w,v\rangle$ in g which does not pass through v'. Hence, $\boldsymbol{t}(g_1)$ contains a leaf $l\in\boldsymbol{m}^{-1}(v)$. Since $t/l=*$ but $\boldsymbol{t}(g_1)\boldsymbol{t}(g_2)/l=\boldsymbol{t}(g_2)$, it follows that $\boldsymbol{t}(g_2)=*$. We conclude that $v'=v$. Hence, v' factors v, in contradiction to our assumption.

Conversely, assume v' factors v. Since $g=\boldsymbol{m}(t)$, this implies that

$$(*)\qquad \forall o\in\boldsymbol{m}^{-1}(v)\,\exists o'\in\boldsymbol{m}^{-1}(v'){:}\ o' \text{ is a prefix of } o.$$

By definition of $\boldsymbol{m}$, all subtrees t/o with $o \in \boldsymbol{m}^{-1}(v')$ are isomorphic. Therefore, $(*)$ implies that $t = t_1 t_2$, where $t_2 = t/o$ for some $o \in \boldsymbol{m}^{-1}(v')$ and t_1 is obtained from t by replacing all subtrees t/o, $o \in \boldsymbol{m}^{-1}(v')$, with $*$. Since $\boldsymbol{m}(t_1) = g_1$ and $\boldsymbol{m}(t_2) = g_2$, claim (2) follows.

Now, let $(v_0, \ldots, v_k)$ be a maximal sequence of nodes of g factoring v on some path from the root of g to v. Especially, v_0 is the root of g itself and $v_k = v$. Define the graph g' as the graph obtained from g as follows:

Step 1: Add new nodes $s_1, \ldots, s_{k-1}$ labeled by $*$;

Step 2: Redirect all edges in g to node v_j to node s_j for $j = 1, \ldots, k-1$; and

Step 3: Choose $v_0 v_1 \ldots v_{k-1}$ as new root word.

Then, by (1) and (2), $\boldsymbol{t}(g) = \boldsymbol{t}(g_{v_1}) \ldots \boldsymbol{t}(g_{v_k})$ is a factorization of t into irreducible factors. It remains to show that $(v_0, \ldots, v_k)$ can be computed in linear time. Define the $*$-level of a node v' as the maximal length of a path from v' to v if such a path exists and otherwise as ∞. We have:

(3) Node v' factors v iff the $*$-level of v' is finite, and there is no other node in g with the same $*$-level.

The $*$-levels of all nodes in g can be computed by post-order traversal through g (before computing the $*$-level of a node v' compute the $*$-levels of the successors of v') in polynomial (even linear) time. Therefore, $(v_0, \ldots, v_k)$ can be computed in polynomial (even linear) time as well. $\square$

We use the above techniques for factorizations of trees to obtain:

Proposition 2.5. (i) *Let k be a fixed constant, and assume $t \in T_\Sigma(X_k)$ contains at least one occurrence of every variable in X_k, $g = \boldsymbol{m}(t)$ and $t = u_0 s[u_1, \ldots, u_k]$ is the kernel decomposition of t. Then $\boldsymbol{m}(\langle v, s, u_1, \ldots, u_k \rangle)$ can be computed from g in polynomial (even linear) time.*

(ii) *Assume $k_0 < k_1 < \cdots < k_m$ with $k_0 = 0$, where $k_\mu - k_{\mu-1}$ is bounded by some constant. Assume $t = \langle t_1, \ldots, t_m \rangle$, where $t_\mu \in T_\Sigma(\{x_{k_{\mu-1}+1}, \ldots, x_{k_\mu}\})$ is proper for $\mu = 1, \ldots, m$ and $g = \boldsymbol{m}(\langle t_1, \ldots, t_m \rangle)$. Assume $t_\mu = v_\mu s_\mu [u_{k_{\mu-1}+1}, \ldots, u_{k_\mu}]$ is the kernel decomposition of t_μ. Then $\boldsymbol{m}(\langle v_1, \ldots, v_m, s_1, \ldots, s_m, u_1, \ldots, u_{k_m} \rangle)$ can be computed from g in polynomial (even linear) time.*

Proof. For simplicity, we again consider only a proof of statement (i). For $j = 1, \ldots, k$ let v_j denote the node in g labeled with x_k. For $j = 1, \ldots, k$, define S_j as the set of nodes v that factor v_j. Since it contains the root r of g, S_j is not empty. S_j is totally ordered by the reachability relation $\rightarrow$ on g. Therefore, S_j contains a minimal node w.r.t. this ordering that is *not* contained in any of the sets $S_{j'}$ with $j' \neq j$. Call this node v_j'. Finally, let S_0 denote the set of nodes v in g that factor *all* nodes $v_j, j = 1, \ldots, k$, i.e. $S_0 = \bigcap_{j=1}^{k} S_j$. Since $r \in S_0$, S_0 is not empty. As a subset, e.g., of S_1, S_0 is totally ordered by $\rightarrow$ as well. Therefore, S_0 contains a maximal node. Call this node v_0'. Define the graph g' as the graph obtained from g as follows:

Step 1: Add new nodes $s_0, \ldots, s_k$, where s_0 is labeled by $*$ and s_j is labeled with x_j for $j = 1, \ldots, k$;

Step 2: Redirect all edges in g to node v'_j to node s_j for $j = 0, \ldots, k$; and

Step 3: Choose $rv'_0 v'_1 \ldots v'_k$ as new root word.

Using claims (1) and (2) of the proof of Proposition 2.4, one can prove that g' indeed represents $\langle v, s, u_1, \ldots, u_k \rangle$. Also according to this proof, we observe that the sets S_j, $j = 1, \ldots, k$, together with the ordering, can be constructed in linear time. Therefore, since k is constant, S_0 can be constructed in polynomial (even linear) time as well. Thus, nodes $v'_0, \ldots, v'_k$ can be computed in polynomial (even linear) time. For remaining steps (1)–(3) of the construction it is not difficult to construct polynomial- (even linear-) time algorithms. Therefore, assertion (i) follows. $\square$

3. 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). Different to [13] 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. Similar to [13] we define the notion of a computation quite carefully in order to fix our terminology for the composition and decomposition of subcomputations. Outputs for subtrees which are not part of the final output are irrelevant. Therefore, we consider tree transducers that are only allowed to skip "empty trees". We recall from [13] that this restriction can be imposed onto our tree transducers without loss of generality (Theorem 3.1 (i)). As a new normal form we also want our transducers to "delay" output as long as possible. These are called strongly reduced. We show that for every reduced FST an equivalent strongly reduced FST can be computed in linear time (Theorem 3.1 (ii)).

For Sections 3 and 4, X denotes the fixed denumerable set $\{x_i \mid i \in \mathbb{N}\}$ of variables and $X_m = \{x_1, \ldots, x_m\}$.

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*, and $\delta \subseteq \bigcup_{m \geqslant 0} Q \times \Sigma_m \times Q^m$ is the set of *transitions* of A; the transitions in $\delta \cap \bigcup_{m \geqslant 0} \{q\} \times \Sigma_m \times Q^m$ 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 leaves. 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, $j = 1, \ldots, m$. If $\phi(o) \notin X$, the transition $\phi(o)$ also is called the transition *chosen* at o.

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 $q \in Q_F$. $L(A) = \{t \in T_\Sigma \mid \text{there is an accepting computation of } A \text{ for } t\}$ is the *language* accepted by A. The *size* of A, $|A|$, is defined by $|A| = \sum_{(q,a,q_1 \ldots q_m) \in \delta} (m+2)$.

For estimating the complexity of our algorithms we always assume that the input signature Σ is *fixed*. Only the sets of states and transitions vary. Thus, the rank of Σ is viewed as a *constant*.

A *bottom-up finite-state tree transducer* (FST) is a pair $M = (A, T)$. $A = (Q, \Sigma, \delta, Q_F)$ is the FTA *underlying* M, whereas $T: \delta \to T_\Delta(X)$, the *output function* of M, maps every transition $\tau = (q, a, q_1 \ldots q_m)$ to the *output pattern* $T(\tau) \in T_\Delta(X_m)$ for τ. Note that an FST according to the definition in [13] is obtained by allowing several transitions $(q, a, q_1 \ldots q_m)$ of the underlying finite tree automaton, which are distinguished by the different outputs they produce. The extension of the techniques explained in the present paper to this slightly more general situation is straightforward and, therefore, omitted.

T is extended to computations as follows. Assume ϕ is a $(q, q_1 \ldots q_k)$-computation of A. If $\phi = x_j$ for some j then $T(\phi) = x_j$. If $\phi = \tau(\phi_1, \ldots, \phi_m)$ then $T(\phi) = T(\tau)[T(\phi_1), \ldots, T(\phi_m)]$. $T(\phi)$ is also called the *output* produced by ϕ. By this definition, $T(\phi[\phi_1, \ldots, \phi_k]) = T(\phi)[T(\phi_1), \ldots, T(\phi_k)]$.

For some tree $t \in T_\Sigma$, $T_M(t) = \{T(\phi) \mid \phi \text{ accepting computation of } A \text{ for } t\}$ denotes the set of outputs of M for t; $\text{val}_M(t) = \# T_M(t)$ denotes the number of different outputs of M for t. $T(M) = \{(t, s) \mid t \in L(A),\ s \in T_M(t)\}$ is the *translation* defined by M; and $\text{val}(M) = \sup\{\text{val}_M(t) \mid t \in T_\Sigma\}$ is the *valuedness*. M is called

- *single-valued*, if $\text{val}(M) \leq 1$;
- *k-valued*, if $\text{val}(M) \leq k$;
- *finite-valued*, if $\text{val}(M) < \infty$; and
- *infinite-valued*, if $\text{val}(M) = \infty$.

As an example of an FST consider $M = (A, T)$, where $A = (Q, \Sigma, \delta, Q_F)$, with $Q = \{0, 1, 2, 3\}$, where $Q_F = \{0\}$; $\Sigma = \{a, b\}$, where a has rank 2 and b rank 0, and δ consists of the transitions:

$$\tau_1 = (0, a, 0\,2), \quad \text{where } T(\tau_1) = x_1,$$

$$\tau_2 = (0, a, 1\,3), \quad \text{where } T(\tau_2) = d(c_1, x_2),$$

$$\tau_3 = (0, a, 1\,2), \quad \text{where } T(\tau_3) = d(x_1, c_2),$$

$$\tau_4 = (1, a, 1\,2), \quad \text{where } T(\tau_4) = x_1,$$

$$\tau_5 = (1, a, 3\,2), \quad \text{where } T(\tau_5) = x_2,$$

$$\tau_6 = (2, b, \varepsilon), \quad \text{where } T(\tau_6) = c_1,$$

and

$$\tau_7=(3,b,\varepsilon), \qquad \text{where } T(\tau_7)=c_2.$$

An accepting computation ϕ of A for $t=a(a(a,b),b),b)$ is, e.g., $\phi=\tau_1(\tau_2(\tau_5(\tau_7,\tau_6),\tau_7),\tau_6)$, where $T(\phi)=d(c_1,c_2)$. In general, $T(M)$ consists of all pairs $(a(x_1,b)^k b, d(c_1,c_2))$, where $k>1$. Hence, M is single-valued. If we add a transition $\tau_8=(1,a,2\,3)$, where $T(\tau_8)=x_2$, then the resulting transducer is no longer single-valued.

To measure the computational complexity of our algorithms we need some notion of size of our transducers. For this we refer to the following internal representation of the output function T. For every transition τ we introduce a distinct set of variables $x_{\tau,j}$ as pattern variables. Assume t_τ is obtained from $T(\tau)$ by renaming variable leaves x_j with $x_{\tau,j}$. Then T is represented internally by the subtree graph

$$\boldsymbol{m}(T)=\boldsymbol{m}(\langle t_\tau\rangle_{\tau\in\delta})$$

of the output-pattern forest. Therefore, the nodes of $\boldsymbol{m}(T)$ correspond to the (isomorphy classes of) subtrees of output patterns. Since we made the variable sets for distinct output patterns disjoint, $\boldsymbol{m}(T)$ contains for every transition τ a "disjoint copy" of all paths in $\boldsymbol{m}(T(\tau))$ from the root to the variable occurrences of this pattern. Note that this internal representation can be computed in linear time from a representation where the outputs $T(\tau)$ are given by trees and not by graphs.

Since our algorithms always operate on this internal representation, we define the *size* $|M|$ of M by

$$|M|=\sum_{(q,a,q_1\dots q_m)\in\delta}(m+1)+|\boldsymbol{m}(T)|.$$

A state $q\in Q$ is called *useful* iff an accepting computation ϕ and a node o in ϕ exists such that $\phi(o)$ is 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 further and more sophisticated normalizations of transducers. Especially, 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 $T_\Delta(X)\cup\{\perp\}$. 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 the following hold:

(i) A is reduced;

(ii) there is some subset $U(M)$ of states such that for every $\tau=(q,a,q_1\dots q_m)\in\delta$ the following holds:

if $q\notin U(M)$ then $T(\tau)\neq\perp$ and ($q_j\in U(M)$ iff x_j does not occur in $T(\tau)$),

if $q\in U(M)$ then $T(\tau)=\perp$ and $q_j\in U(M)$ for all j;

(iii) if $f \in Q_F$ then $f \neq q_j$ for every transition $(q, a, q_1 \ldots q_m) \in \delta$ and every j.

The states in $U(M)$ are exactly those which are used by subcomputations ϕ which produce $\perp$. If a computation 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.

It should be noted that in [13] reducedness does not include property (iii) above. However, property (iii) can be achieved with little extra computational effort; moreover, it simplifies the constructions in the present context.

Considering the example of a single-valued transducer above we find that we need a stronger normal form of FSTs. This normal form should allow transitions with output patterns $\neq \perp$ for nonfinal states q only if the outputs produced by q-computations *depend* on the input trees. Assume $M = (A, T)$ is a reduced FST with $A = (Q, \Sigma, \delta, Q_F)$. A state $q \in Q$ is called *constant* iff for all q-computations ϕ, ϕ': $T(\phi) = T(\phi')$. Const(M) denotes the set of constant states of M, whereas $C_M : \text{Const}(M) \to T_\Delta \cup \{\perp\}$ the map defined by $C_M(q) = T(\phi)$ for some q-computation ϕ. Observe that always $U(M) \subseteq \text{Const}(M)$. In general, this inclusion is proper; also, $C_M(q)$ possibly has exponential size. However, the number of *different* subtrees of all the trees $C_M(q)$, $q \in \text{Const}(M)$, is only linear in the size of M. In fact, this is the reason why we are forced to employ graph representations of trees in order to make our algorithms run in polynomial time.

The FST $M = (A, T)$ is called *strongly reduced* iff M is reduced and $U(M) = \text{Const}(M) \backslash Q_F$. We prove:

Theorem 3.1. (i) *For every FST M a reduced FST M_r exists such that $T(M_r) = T(M)$. M_r can be constructed from M in linear time.*

(ii) *For every reduced FST $M = (A, T)$ a strongly reduced FST $M_s = (A, T_s)$ exists such that $T(M) = T(M_s)$.*

M_s can be computed in linear time.

Statement (i) of Theorem 3.1 is essentially taken from [13]. Consider, e.g., the FST $M = (A, T)$, where $A = (Q, \Sigma, \delta, Q_F)$ and T are defined as follows:

$Q = \{q, p\}$ with $Q_F = \{q\}$;

$\Sigma_2 = \{a\}$ and $\Sigma_0 = \{b\}$, whereas

$\delta = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ and T are given by

$\tau_1 = (q, a, q\,p)$, where $T(\tau_1) = d(x_1, c)$,

$\tau_2 = (q, a, q\,q)$, where $T(\tau_2) = d(c, x_2)$,

$\tau_3 = (q, b, \varepsilon)$, where $T(\tau_3) = c$,

$\tau_4 = (p, b, \varepsilon)$, where $T(\tau_4) = c$.

Then, the corresponding reduced FST is obtained in two stages. First, we add a tag from $\{0, 1\}$ as a new component to Q which indicates whether the present output is part of the final output or not. Hence, we define

$$Q_r = Q \times \{0, 1\} \quad \text{with} \quad Q_F = \{(q, 1)\};$$

$\delta_r = \{\tau'_1, \dots, \tau'_8\}$ and T are constructed as follows. The first four transitions which are $\langle \_, 1 \rangle$-transitions generate the same outputs as the corresponding transitions in δ, but the jth successor state is tagged with 0 iff variable x_j does not occur in the output pattern. The second half of transitions all produce output $\perp$, all states involved have tag 0. These transitions mimic computations of A for subtrees whose output is abandoned.

First group:

$$\tau'_1 = (\langle q, 1\rangle, a, \langle q, 1\rangle \langle p, 0\rangle), \quad \text{where } T(\tau'_1) = d(x_1, c),$$

$$\tau'_2 = (\langle q, 1\rangle, a, \langle q, 0\rangle \langle q, 1\rangle), \quad \text{where } T(\tau'_2) = d(c, x_2),$$

$$\tau'_3 = (\langle q, 1\rangle, b, \varepsilon), \quad \text{where } T(\tau'_3) = c,$$

$$\tau'_4 = (\langle p, 1\rangle, b, \varepsilon), \quad \text{where } T(\tau'_4) = c.$$

Second group:

$$\tau'_5 = (\langle q, 0\rangle, a, \langle q, 0\rangle \langle p, 0\rangle), \quad \text{where } T(\tau'_5) = \perp,$$

$$\tau'_6 = (\langle q, 0\rangle, a, \langle q, 0\rangle \langle q, 0\rangle), \quad \text{where } T(\tau'_6) = \perp,$$

$$\tau'_7 = (\langle q, 0\rangle, b, \varepsilon), \quad \text{where } T(\tau'_7) = \perp, \quad \text{and}$$

$$\tau'_8 = (\langle p, 0\rangle, b, \varepsilon), \quad \text{where } T(\tau'_8) = \perp.$$

Now, the reduced FST is obtained by adding a new final state and removing useless states (like $\langle p, 1\rangle$).

In order to prove statement (ii) we first show that for every reduced FST $M = (A, T)$ the set $\mathrm{Const}(M)$ together with (a representation of) the map C_M can be computed in linear time.

Proposition 3.2. *Let $M = (A, T)$ be a reduced FST. Then*

(i) $\mathrm{Const}(M)$ *can be computed in linear time*;

(ii) *the graph $\boldsymbol{m}(C_M) = \boldsymbol{m}(\langle C_M(q)\rangle_{q \in \mathrm{Const}(M)})$ can be computed in linear time.*

Proof. Define the *reachability* and *connectivity* relations $\rightarrow_A, \leftrightarrow_A \subseteq Q \times Q$. q is reachable from p w.r.t. A ($p \rightarrow_A q$) iff there is a proper (p, q)-computation of A. q is connected with p w.r.t. A ($p \leftrightarrow_A q$) iff $p \rightarrow_A q$ and $q \rightarrow_A p$. $\leftrightarrow_A$ is an equivalence relation on Q. The equivalence classes $Q_1, \dots, Q_k$ w.r.t. $\leftrightarrow_A$ are also called the *strong components* of A. A strong component $Q_{j'}$ is reachable from a strong component Q_j iff $p \rightarrow_A q$ for some $p \in Q_j$ and $q \in Q_{j'}$. The following facts can easily be shown:

- $p \in \mathrm{Const}(M)$ and $p \rightarrow_A q$ implies $q \in \mathrm{Const}(M)$;
- $p \in \mathrm{Const}(M)$ and $p \leftrightarrow_A q$ implies $T(\phi) \in \{\perp, x_1\}$ for every proper (p, q)-computation (and, hence, $C_M(p) = C_M(q)$);
- $U(A, T) \subseteq \mathrm{Const}(M)$.

We construct $\mathrm{Const}(M)$ and C_M according to reachability. Assume $Q_1, \ldots, Q_k$ are the strong components Q_j of A such that for every strong component Q' reachable from Q_j, either $Q' \subseteq U(M)$ or $T(\phi) = x_1$ for every proper (p,q)-computation with $p,q \in Q'$. Provided $Q_j \subseteq \mathrm{Const}(M)$, define $C_M(Q_j) = C_M(q)$ for some $q \in Q_j$. Computing the set $\mathrm{Const}(M)$ and C_M is done in two steps.

Step 1: For every $j \in \{1, \ldots, k\}$, select a transition $\tau_j = (q^{(j)}, a^{(j)}, q_1^{(j)} \ldots q_{m_j}^{(j)}) \in \delta$, where $q^{(j)} \in Q_j$ but either $q_\mu^{(j)} \notin Q_j \cup U(M)$ for at least one μ or $q_\mu^{(j)} \in U(M)$ for all μ. Define $t = \langle t_1, \ldots, t_k \rangle$, where t_j is the tree obtained from $T(\tau_j)$ by replacing x_μ with $x_{j'}$ provided $q_\mu^{(j)} \in Q_{j'}$. By definition of t, the digraph G_t in the assumption of Proposition 2.3 is acyclic. Define $\langle C_M^*(1), \ldots, C_M^*(k) \rangle = t^k$. Then the following holds for every j:

$$(*) \qquad C_M^*(j) = C_M(Q_j) \quad \text{provided } Q_j \subseteq \mathrm{Const}(M).$$

By Proposition 2.3, $g^{(1)} = \boldsymbol{m}(\langle C_M^*(1), \ldots, C_M^*(k) \rangle)$ can be computed in linear time.

Step 2: We compute the set of those j such that $Q_j \subseteq \mathrm{Const}(M)$. Let δ' be the set of all q-transitions where $q \in Q_1 \cup \cdots \cup Q_k$. For $\tau = (q, a, q_1 \ldots q_m) \in \delta'$ define t_τ as the tree obtained from $T(\tau)$ by replacing x_μ with $x_{j'}$ provided $q_\mu \in Q_{j'}$. Then $Q_j \subseteq \mathrm{Const}(M)$ iff the following hold:

(1) $\forall \tau = (q, a, q_1 \ldots q_m) \in \delta'$ with $q \in Q_j$: $C_M^*(j) = t_\tau[C_M^*(1), \ldots, C_M^*(k)]$;

(2) whenever $Q_{j'}$ is reachable from Q_j then $Q_{j'} \subseteq \mathrm{Const}(M)$.

By Proposition 2.2 the graph $g^{(2)} = \boldsymbol{m}(\langle t_\tau[C_M^*(1), \ldots, C_M^*(k)] \rangle_{\tau \in \delta'})$ can be computed in linear time. For $g^{(2)}$, Properties (1) and (2) can be tested in linear time.

Finally, knowing $\mathrm{Const}(M)$, the graph $\boldsymbol{m}(C_M)$ can be extracted from $g^{(1)}$ in linear time as well. This finishes the proof. $\square$

Having computed the set $\mathrm{Const}(M)$ together with the graph $\boldsymbol{m}(C_M)$ it is no longer difficult to compute (a representation of) the output function T_s such that $M_s = (A, T_s)$ is strongly reduced.

Proof of Theorem 3.1(ii). For $\tau = (q, a, q_1 \ldots q_m) \in \delta$ define

$$T_s(\tau) = \begin{cases} T(\tau)[s_1, \ldots, s_m] & \text{if } q \notin \mathrm{Const}(M), \\ C_M(q) & \text{if } q \in \mathrm{Const}(M) \cap Q_F, \\ \perp & \text{if } q \in \mathrm{Const}(M) \setminus Q_F, \end{cases}$$

where

$$s_j = \begin{cases} C_M(q_j) & \text{if } q_j \in \mathrm{Const}(M), \\ x_j & \text{otherwise.} \end{cases}$$

T_s produces the output "as late as possible". Especially, $T_s(\phi) = T(\phi)$ for every accepting computation ϕ. By Proposition 2.2 we find that, given $\mathrm{Const}(M)$ and $\boldsymbol{m}(C_M)$ the graph $\boldsymbol{m}(T_s)$ can be constructed in linear time. $\square$

4. Pairings

Instead of comparing *two* computations w.r.t. *one* output function T, it is technically more convenient to consider only one computation but two output functions. Therefore, we introduce the notion of a pairing which consists of one FTA and two output functions. For every FST M, we construct the canonical pairing M^2. Then, M is single-valued iff the two output functions of M^2 are equivalent (Proposition 4.1). We give necessary conditions (U1) and (U2) for a strongly reduced pairing to have two equivalent output functions (Proposition 4.3). For the proof of necessity we employ Proposition 4.2, which describes the relation between the outputs produced by equivalent output functions for a proper computation. Proposition 4.2 will be used a second time in Section 5 to prove necessity of a third Property (U3), which, together with Properties (U1) and (U2), precisely characterizes equivalence of output functions.

Assume $A=(Q,\Sigma,\delta,Q_F)$ is an FTA and $T_1,T_2:\delta\to T_\Delta(X)\cup\{\perp\}$ are maps such that (A,T_i) is an FST for $i=1,2$. Then $\Pi=(A,T_1,T_2)$ is called *pairing*. As the size of Π, $|\Pi|$, we simply define $|\Pi|=|(A,T_1)|+|(A,T_2)|$.

If (A,T_1) and (A,T_2) are reduced (strongly reduced), then Π is also called reduced (strongly reduced).

Assume $v_1,v_2\in\tilde{T}_\Delta(*)$. We say T_1 is (v_1,v_2)-*equivalent* to T_2 w.r.t. A ($v_1T_1\equiv v_2T_2$ for short) iff $v_1T_1(\phi)=v_2T_2(\phi)$ for every accepting computation ϕ of A.

We defined the notion of equivalence somewhat more generally as necessary in our algorithm deciding single-valuedness. Our procedure deciding single-valuedness only refers to $(*,*)$-equivalence (see the next proposition). This important case of equivalence is denoted by $T_1\equiv T_2$. However, our polynomial algorithm deciding finite-valuedness also makes use of the more general notion (see Section 6).

Assume $A=(Q,\Sigma,\delta,Q_F)$ is a reduced FTA. Consider the FTA $A^{(k)}=(Q^k,\Sigma,\delta^k,Q_F^k)$, where $\tau=(\langle q^{(1)},\dots,q^{(k)}\rangle,a,\langle q_1^{(1)},\dots,q_1^{(k)}\rangle\cdots\langle q_m^{(1)},\dots,q_m^{(k)}\rangle)\in\delta^k$ iff $\tau^{(i)}=(q^{(i)},a,q_1^{(i)}\dots q_m^{(i)})\in\delta$ for all i. $\tau\in\delta^k$ can be viewed as the k-tuple $\tau=\langle\tau^{(1)},\dots,\tau^{(k)}\rangle$ and $\tau^{(j)}$ as its jth component. Accordingly, every computation ϕ of $A^{(k)}$ for some tree t represents a k-tuple $\langle\phi^{(1)},\dots,\phi^{(k)}\rangle$ of computations $\phi^{(j)}$ of A for t; and vice versa. The kth *power* of A, A^k, is the reduced FTA obtained from $A^{(k)}$ by removing all useless states and transitions. Assume $M=(A,T)$ is a reduced FST. The kth *power* of M, M^k, is the $(k+1)$-tuple $(A^k,T_1,\dots,T_k)$, where T_i produces the output pattern according to the ith component of transitions. Since (A,T) is reduced, (A^k,T_j) is reduced for $j=1,\dots,k$. In the sequel, we use powers M^k only for $k=2$ or $k=3$. M^2 is also called *canonical pairing* of M. We find:

Proposition 4.1. *Assume* $M=(A,T)$ *is a reduced FST and* $M^2=(A^2,T_1,T_2)$ *is the canonical pairing of* M. *Then* $\text{val}(M)=1$ *iff* $T_1\equiv T_2$ *w.r.t.* A^2.

Since M^2 can be computed from M in polynomial (quadratic) time it suffices, by Theorem 3.1, to deal with (strongly) reduced pairings. The proof of necessity of our characterization of (v_1,v_2)-equivalence is based on the following (technical) proposition.

Proposition 4.2. *Assume $\Pi=(A, T_1, T_2)$ is a reduced pairing with $v_1 T_1 \equiv v_2 T_2$.*

(i) *If ϕ and ϕ' are q-computations for some state q of A then $T_1(\phi)=T_1(\phi')$ iff $T_2(\phi)=T_2(\phi')$. Hence, especially,* $\mathrm{Const}(A, T_1)=\mathrm{Const}(A, T_2)$.

(ii) *Assume ϕ is an X_k-proper $(f, q_1 \ldots q_k)$-computation of A with $f \in Q_F$ and $q_1, \ldots, q_k \notin \mathrm{Const}(A, T_i)$. Then $v_1 T_1(\phi) \approx v_2 T_2(\phi)$.*

Proof. First consider statement (i). Since Π is reduced, there is a proper (f, q)-computation ψ of A with $f \in Q_F$. W.l.o.g. assume $T_1(\phi)=T_1(\phi')$ but $T_2(\phi) \neq T_2(\phi')$. By assumption, $v_1 T_1(\psi\phi)=v_2 T_2(\psi\phi)$. However,

$$\begin{aligned} v_1 T_1(\psi\phi) &= v_1 T_1(\psi\phi')=v_2 T_2(\psi\phi')=v_2 T_2(\psi) T_2(\phi') \\ &\neq v_2 T_2(\psi) T_2(\phi)=v_2 T_2(\psi\phi): \quad \text{contradiction.} \end{aligned}$$

Next, we prove (ii). Since $q_j \notin \mathrm{Const}(A, T_i)$, we also have $q_j \notin U(A, T_i)$. Hence x_j occurs both in $T_1(\phi)$ and $T_2(\phi)$. By (i), q_j-computations ϕ_j, ϕ_j' exist with $T_1(\phi_j) \neq T_1(\phi_j')$ and $T_2(\phi_j) \neq T_2(\phi_j')$. Define $s_i=v_i T_i(\phi)$, $s_{i,j}=T_i(\phi_j)$ and $s_{i,j}'=T_i(\phi_j')$ for $i=1,2$ and $j=1, \ldots, k$. Then

$$s_1[\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,k}]=s_2[\tilde{s}_{2,1}, \ldots, \tilde{s}_{2,k}],$$

whenever $\tilde{s}_{i,j} \in \{s_{i,j}, s_{i,j}'\}$ and $\tilde{s}_{1,j}=s_{1,j}$ iff $\tilde{s}_{2,j}=s_{2,j}$.

Consider the equations

$$s_1[s_{1,1}, \ldots, s_{1,j}, \ldots, s_{1,k}]=s_2[s_{2,1}, \ldots, s_{2,j}, \ldots, s_{2,k}],$$

$$s_1[s_{1,1}, \ldots, s_{1,j}', \ldots, s_{1,k}]=s_2[s_{2,1}, \ldots, s_{2,j}', \ldots, s_{2,k}].$$

Factorization w.r.t. x_j yields some tree $u_j \in \tilde{T}_\Delta(x_j)$ such that either (1) or (2) holds:

(1) $u_j s_{1,j}=s_{2,j}$ and $u_j s_{1,j}'=s_{2,j}'$,

(2) $s_{1,j}=u_j s_{2,j}$ and $s_{1,j}'=u_j s_{2,j}'$.

Let J_1 denote the set of j where (1) holds and J_2 the set of the remaining ones. Consider the equations

$$s_1[\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,k-1}, s_{1,k}]=s_2[\tilde{s}_{2,1}, \ldots, \tilde{s}_{2,k-1}, s_{2,k}],$$

$$s_1[\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,k-1}, s_{1,k}']=s_2[\tilde{s}_{2,1}, \ldots, \tilde{s}_{2,k-1}, s_{2,k}'].$$

If $k \in J_1$ then bottom cancellation yields

$$s_1[\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,k-1}, x_k]=s_2[\tilde{s}_{2,1}, \ldots, \tilde{s}_{2,k-1}, u_k]$$

and, otherwise,

$$s_1[\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,k-1}, u_k]=s_2[\tilde{s}_{2,1}, \ldots, \tilde{s}_{2,k-1}, x_k].$$

Continuing bottom cancellation for $x_{k-1}, \ldots, x_1$ we obtain

$$s_1[v_{1,1}, \ldots, v_{1,k}]=s_2[v_{2,1}, \ldots, v_{2,k}] \text{ with } v_{i,j}=\begin{cases} u_j & \text{if } j \in J_i, \\ x_j & \text{otherwise.} \end{cases}$$

This proves (ii). □

As an application of Proposition 4.2 we find:

Proposition 4.3. *Assume $\Pi=(A,T_1,T_2)$ is a strongly reduced pairing. If $v_1T_1\equiv v_2T_2$ then Π has Properties* (U1) *and* (U2):

(U1) $\mathrm{Const}(A,T_1)=\mathrm{Const}(A,T_2)$.

(U2) *Assume $\tau=(q,a,q_1\ldots q_m)\in\delta$, $T_i(\tau)$ contains variables $x_j,x_{j'}$ for $j\neq j'$ and $T_i(\tau)=u_iy_i$, where u_i is the maximal prefix in $\tilde{T}_\Delta(*)$. Then $y_1\approx y_2$.*

Since Π is strongly reduced, Property (U1) implies $U(A,T_1)=U(A,T_2)$. Especially, for every transition τ, the sets of variables occurring in the output patterns $T_1(\tau)$ and $T_2(\tau)$ coincide.

Proof. Property (U1) immediately follows from Proposition 4.2(i). Therefore, it remains to prove that Π also has Property (U2). Since A is reduced, there is a proper (f,q)-computation ϕ of A for some $f\in Q_F$. By Proposition 4.2(ii),

$$v_1T_1(\phi)u_1y_1=v_1T_1(\phi\tau(x_1,\ldots,x_m))\approx v_2T_2(\phi\tau(x_1,\ldots,x_m))=v_2T_2(\phi)u_2y_2.$$

Therefore,

$$\begin{aligned}v_1T_1(\phi)u_1(y_1\theta_1)&=(v_1T_1(\phi)u_1y_1)\theta_1=(v_2T_2(\phi)u_2y_2)\theta_2\\&=v_2T_2(\phi)u_2(y_2\theta_2)=s\end{aligned}$$

for substitutions θ_i with $x_j\theta_i\in\tilde{T}_\Delta(x_j)$ for all i and j. Since both $v_1T_1(\phi)u_1$ and $v_2T_2(\phi)u_2$ are maximal prefixes of s in $\tilde{T}_\Delta(*)$ they are, by Proposition 1.5, equal. Therefore, we can apply top cancellation and deduce that $y_1\theta_1=y_2\theta_2$. Hence, $y_1\approx y_2$, which we wanted to prove. □

5. Translations of paths

In Section 4 we found Properties (U1) and (U2) of a strongly reduced pairing $\Pi=(A,T_1,T_2)$, where $v_1T_1\equiv v_2T_2$ w.r.t. A. In this section we exhibit a third Property (U3′). Property (U3′) for (v_1,v_2) is concerned with computations along paths of input trees. Provided Π has Properties (U1) and (U2), we give a reformulation of Property (U3′) by a suitable graph property (U3) (Proposition 5.1). We show that Properties (U1), (U2) and (U3) for (v_1,v_2) are not only necessary but also sufficient for (v_1,v_2)-equivalence of output functions (Theorem 5.2). Since (U1), (U2) and (U3) for (v_1,v_2) can be decided in deterministic polynomial time, we obtain a deterministic polynomial-time algorithm deciding (v_1,v_2)-equivalence (Theorem 5.3).

Assume $w=(a_1,j_1)\cdots(a_k,j_k)\in\Sigma_B^*$. If $\langle\varepsilon,w,j_1\ldots j_k\rangle$ is a path in $t\in T_\Sigma$, then t can be factored according to w, $t=u_1\ldots u_kt'$, where $t'=t/j_1\ldots j_k$ and $u_\kappa=a_\kappa(t_{\kappa,1},\ldots,t_{\kappa,m_\kappa})$ with

$$\rho(a_\kappa)=m_\kappa\quad\text{and}\quad t_{\kappa,i}=\begin{cases}t/j_1\ldots j_{\kappa-1}i & \text{if } i\neq j_\kappa,\\ * & \text{if } i=j_\kappa.\end{cases}$$

By definition, the u_κ are $*$-irreducible. Assume $\Pi=(A, T_1, T_2)$ is a strongly reduced pairing and $\phi=\phi_1...\phi_k\phi'$ is the corresponding decomposition of an accepting computation of A for t, where $\phi(j_1...j_k)$ is a q-transition with $q\notin U(A,T_i)$. Assume $v_1T_1\equiv v_2T_2$ w.r.t. A. Applying Proposition 4.3 to $\phi_1...\phi_k$, we deduce that either $v_1T_1(\phi_1)...T_1(\phi_k)$ is a $*$-prefix of $v_2T_2(\phi_1)...T_2(\phi_k)$ or vice versa. Note that this may be wrong if Π is not strongly reduced. In general there is no finite subalphabet of $I_\Delta(*)$ by which these outputs can be spelled. Therefore, we "strip off" the trees $t_{\kappa,j}$, i.e. we replace them in the factorization of t according to w by variables $x_{\kappa,j}$. We make use of doubly indexed variables merely for convenience in order to distinguish between the variables at different levels of the tree. We may as well have used variables from some variable set X_k.

Formally, let now X denote the set of variables $\{x_{i,j}\mid i,j\in\mathbb{N}\}$. X is supposed to be disjoint from any set $X_k=\{x_1,...,x_k\}$. For $w=(a_1,j_1)...(a_k,j_k)$, the tree $\mathrm{tr}(w)\in T_{\Sigma\cup X}(*)$ is defined by $\mathrm{tr}(w)=*$ provided $k=0$ (i.e. $w=\varepsilon$), and $\mathrm{tr}(w)=y_1...y_k$ otherwise, where

$$y_\kappa=a_\kappa(u_{\kappa,1},...,u_{\kappa,m_\kappa}) \quad \text{with} \quad a_\kappa\in\Sigma_{m_\kappa} \quad \text{and}$$

$$u_{\kappa,i}=\begin{cases} x_{\kappa,i} & \text{if } i\neq j_\kappa \\ * & \text{if } i=j_\kappa. \end{cases}$$

So, e.g., for $w=(d,1)(a,1)(a,2)(b,1)$, with $a\in\Sigma_2$, $b\in\Sigma_1$ and $d\in\Sigma_3$, $\mathrm{tr}(w)$ is the tree $\mathrm{tr}(w)=d(a(a(x_{3,1},b*),x_{2,2}),x_{1,2},x_{1,3})$.

For $s\in T_\Delta(X_m)$ and $1\leqslant j\leqslant m$ we introduce the abbreviation $s[\kappa,j]=s[x_{\kappa,1},..., x_{\kappa,j-1},*,x_{\kappa,j+1},...,x_{\kappa,m}]$, i.e. the variable x_j is replaced by $*$, whereas the others get the additional index κ. Observe that $s[\kappa,j]\in\tilde{T}_{\Delta\cup X}(*)$ iff s contains x_j.

Let $M=(A,T)$ be an FST with $A=(Q,\Sigma,\delta,Q_F)$. We want M to consume w via tr(_) and produce output in the free monoid $\tilde{T}_{\Delta\cup X}(*)$. Since we are not interested in the states of computations corresponding to variables $x_{\kappa,j}$, we only specify those at the root of $\mathrm{tr}(w)$ and at the variable leaf $*$. Therefore, for states $p,q\in Q$, a *partial* (p,q)-computation π of A for w is a computation of A for $\mathrm{tr}(w)$, where $p=q$ and $\pi(\varepsilon)=*$ provided $w=\varepsilon$; otherwise, $\pi(\varepsilon)$ is a p-transition and $\pi(j_1...j_{k-1})=\tau$ for some transition $\tau=(q_0,a_{k-1},q_1...q_m)$, with $q_{j_k}=q$. Because of the special structure of the tree $\mathrm{tr}(w)$, we can represent π as the sequence $(\tau_1,j_1)...(\tau_k,j_k)\in(\delta\times\mathbb{N})^k$, where τ_κ is the transition chosen in π at node $j_1...j_{\kappa-1}$. We have $T(\pi)=T(\tau_1)[1,j_1]...T(\tau_k)[k,j_k]$. If some variable $x_{\kappa,j}$ occurs in $T(\pi)$ then the state q' *corresponding* to $x_{\kappa,j}$ is given by $q'=q_j$ provided $\pi(j_1...j_{\kappa-1})=(q_0,a,q_1...q_m)$. If $\pi=\varepsilon$ then $T(\pi)=*$. If $\pi\neq\varepsilon$ then

$$T(\pi)\in\tilde{T}_{\Delta\cup X}(*) \text{ iff } T(\tau_k) \text{ contains } x_{j_k}.$$

A transition τ is called *T-initial* iff $T(\tau)\in T_\Delta$.

Assume the pairing $\Pi=(A,T_1,T_2)$ has Property (U1). Then $T_1(\tau)$ contains variable x_j iff $T_2(\tau)$ contains x_j for every transition τ of A. Especially, τ is T_1-initial iff τ is T_2-initial. We say Π has Property (U3′) for (v_1,v_2) iff

(U3′) For every partial (f, q)-computation π with $f \in Q_F$ and $q \in [Q \setminus U(A, T_i)] \cup Q_F$, the following holds: Let u_i denote the maximal $*$-suffix of $v_i T_i(\pi)$ in $\tilde{T}_\Delta(*)$. Then for every transition $\tau = (q, a, q_1 \ldots q_m) \in \delta$,
(1) If $T_i(\tau)$ contains variables x_j and $x_{j'}$ with $j \neq j'$, and u_i' is the maximal prefix of $T_i(\tau)$ in $\tilde{T}_\Delta(*)$, then $u_1 u_1' = u_2 u_2'$.
(2) If τ is T_i-initial then $u_1 T_1(\tau) = u_2 T_2(\tau)$.

Observe that the trees u_i, u_i' occurring in (U3′) for (v_1, v_2) can be spelled over a *finite* subalphabet $I \subseteq I_\Delta(*)$. However, we still have to consider possibly infinitely many partial computations. Therefore, we represent the set of all partial computations by a graph called the *trace graph* $G(A)$ of A, and transform Property (U3′) for (v_1, v_2) into a property of $G(A)$.

The *trace graph* of A is the (directed unordered edge labeled) graph $G(A) = (V, E)$ where $V = [Q \setminus U(A, T_i)] \cup Q_F$, and E consists of all edges $\langle q, (\tau, j), q' \rangle$ with $\tau = (q, a, q_1 \ldots q_m) \in \delta$ and $q_j = q'$. The paths in $G(A)$ describe the partial computations of A.

To every node q of $G(A)$ we attach the "difference" between outputs when reaching this state q during a partial computation. Formally, the *difference* $\text{diff}(t_1, t_2)$ of trees $t_1, t_2 \in \tilde{T}_\Delta(*)$ is defined as follows. Assume r is the maximal common $*$-prefix of t_1 and t_2. Then $\text{diff}(t_1, t_2) = (s_1, s_2)$ iff $t_1 = r s_1$ and $t_2 = r s_2$. Clearly, $\text{diff}(t_1, t_2) = (*, *)$ iff $t_1 = t_2$.

Property (U3) for (v_1, v_2) is divided into four assertions. Assertions (1) and (2) give the initial and the final conditions for the outputs produced along such a path. Assertion (4) describes the situation at "branching points" i.e. at nodes o where the output depends on at least two subtrees at o. It states that such nodes are "synchronizing", i.e. both the outputs produced above and below agree. To complete, Assertion (3) describes what happens on paths without branching points.

Assume Π has Property (U1). Then Π has Property (U3) for (v_1, v_2) iff

(U3) There is a map diff: $V \to \tilde{T}_\Delta(*)^2$, with:
(1) If $q \in Q_F$ then $\text{diff}(q) = \text{diff}(v_1, v_2)$.
(2) If $\text{diff}(q) = (s_1, s_2)$ and there is some T_i-initial q-transition then $s_1 T_1(\tau) = s_2 T_2(\tau)$.
Assume $\langle q, (\tau, j), q' \rangle \in E$ and $\text{diff}(q) = (s_1, s_2)$.
(3) If $T_i(\tau) \in \tilde{T}_\Delta(x_j)$ and $u_i = T_i(\tau)[*]$ then $\text{diff}(q') = \text{diff}(s_1 u_1, s_2 u_2)$.
(4) Assume $T_i(\tau) \notin \tilde{T}_\Delta(x_j)$ and $T_i(\tau) = u_i y_i u_i'$, where $u_i \in \tilde{T}_\Delta(*)$, $u_i' \in \tilde{T}_\Delta(x_j)$ and y_i is the minimal subword in $I^*_{\Delta \cup X}(*)$ containing all variable occurrences $x_{j'}$ with $j' \neq j$. Then $\text{diff}(q') = \text{diff}(u_1' *, u_2' *)$; and $\text{diff}(s_1 u_1, s_2 u_2) = (*, *)$.

Property (U3) is an appropriate generalization of a corresponding property for GSMs characterizing equivalence of two output functions. However, for words (viewed as monadic trees with one special leaf) the situation of (4) never occurs. This observation was exploited by Karhumaki et al. [8] to construct linear sized test sets for regular word languages.

All trees in the image of diff(_) contain at least one occurrence of $*$. They can be spelled by a *finite* set I of irreducible trees which can be computed from Π in polynomial time. Observe again that these trees may have exponential size, although they can be represented as a word over I of polynomial length.

The next (technical) proposition relates Properties (U3′) and (U3) for (v_1, v_2) to compatibility of outputs of computations. It, therefore, will be used to prove sufficiency of our characterization of (v_1, v_2)-equivalence.

Proposition 5.1. *Assume* $\Pi = (A, T_1, T_2)$ *is strongly reduced. If* Π *has Properties* (U1) *and* (U2) *then the following three statements are equivalent*:

(1) *For every partial* (f, q)*-computation* π *with* $f \in Q_F$ *and* $q \in [Q \setminus U(A, T_i)] \cup Q_F$, $v_1 T_1(\pi) \approx v_2 T_2(\pi)$ *and* $v_1 T_1(\pi) T_1(\tau) \approx v_2 T_2(\pi) T_2(\tau)$ *for every* T_i*-initial* q*-transition* τ;

(2) Π *has Property* (U3′) *for* (v_1, v_2);

(3) Π *has Property* (U3) *for* (v_1, v_2).

Observe that by Proposition 4.3, statement (1) holds true whenever $v_1 T_1 \equiv v_2 T_2$ w.r.t. A.

Proof of Proposition 5.1. (1) *implies* (2): Assume π is a partial (f, q)-computation of length k with $f \in Q_F$, $q \in [Q \setminus U(A, T_i)] \cup Q_F$, and $v_i T_i(\pi) = s_i u_i$, where u_i is the maximal $*$-suffix in $\tilde{T}_\Delta(*)$, $i = 1, 2$. Assume $\tau = (q, a, q_1 \ldots q_m) \in \delta$. By (1), trees $r_1, r_2 \in \tilde{T}_\Delta(*)$ and X-substitutions θ_1, θ_2 exist with $(s_1\theta_1)u_1 r_1 = (s_2\theta_2)u_2 r_2 = s$, where for every variable $x \neq *$ in s_i, $x\theta_i \in \tilde{T}_\Delta(x)$ and either $x\theta_1 = x$ or $x\theta_2 = x$. Since both $u_1 r_1$ and $u_2 r_2$ are maximal $*$-suffixes of s in $\tilde{T}_\Delta(*)$, we deduce that $u_1 r_1 = u_2 r_2$ and $s_1\theta_1 = s_2\theta_2$. First, consider the case where τ is T_i-initial. By (1),

$$(s_1 u_1 T_1(\tau))\theta'_1 = (s_2 u_2 T_2(\tau))\theta'_2$$

for X-substitutions θ'_i with $x\theta'_i \in \tilde{T}_\Delta(x)$ for every $x \neq *$ in s_i. By Fact 1.8 we may choose $\theta'_i = \theta_i$. Thus, we find

$$(s_1\theta_1)u_1 T_1(\tau) = (s_1 u_1 T_1(\tau))\theta_1 = (s_2 u_2 T_2(\tau))\theta_2 = (s_2\theta_2)u_2 T_2(\tau)$$

and, hence, by top cancellation, $u_1 T_1(\tau) = u_2 T_2(\tau)$.

Now assume $T_i(\tau)$ contains occurrences of variables x_j, $x_{j'}$ with $j \neq j'$. Let $T_i(\tau)[k+1, j] = u'_i y_i$, where u'_i is the maximal prefix in $\tilde{T}_\Delta(*)$, $i = 1, 2$. By (1), there are trees $\bar{r}_1, \bar{r}_2 \in \tilde{T}_\Delta(*)$ and X-substitutions $\bar{\theta}_1, \bar{\theta}_2$ such that

$$(s_1 u_1 u'_1 y_1)\bar{\theta}_1 \bar{r}_1 = (s_2 u_2 u'_2 y_2)\bar{\theta}_2 \bar{r}_2,$$

where for every variable $x \neq *$ in s_i, $x\bar{\theta}_i \in \tilde{T}_\Delta(x)$ and either $x\bar{\theta}_1 = x$ or $x\bar{\theta}_2 = x$. Moreover, for every variable x occurring in s_i, $x\bar{\theta}_i = x\theta_i$. Hence,

$$(s_1\theta_1)u_1 u'_1 (y_1\bar{\theta}_1)\bar{r}_1 = (s_2\theta_2)u_2 u'_2 (y_2\bar{\theta}_2)\bar{r}_2.$$

Therefore, again by top cancellation,

$$u_1 u'_1 (y_1\bar{\theta}_1)\bar{r}_1 = u_2 u'_2 (y_2\bar{\theta}_2)\bar{r}_2,$$

where both $u_1u'_1$ and $u_2u'_2$ are maximal $*$-prefixes in $\tilde{T}_\Delta(*)$. Hence, by Proposition 1.5, $u_1u'_1=u_2u'_2$, in accordance with Property (U3′) for (v_1,v_2).

(2) *implies* (1): Assume π is a partial (f,q)-computation of length k with $f\in Q_F$ and $q\in[Q\backslash U(A,T_i)]\cup Q_F$ and $v_iT_i(\pi)=s_iu_i$, where u_i is the maximal $*$-suffix in $\tilde{T}_\Delta(*)$. We proceed by induction on the length of π.

Assume $\pi=\varepsilon$. Then $T_i(\pi)=*$, and v_i itself is the maximal $*$-suffix of $v_iT_i(\pi)$. Therefore, (2) implies

$$v_1T_1(\pi)=v_1\approx v_2=v_2T_2(\pi),$$

in accordance with (1). Also, if τ is a T_i-initial q-transition then by (2),

$$v_1T_1(\tau)=u_1T_1(\tau)=u_2T_2(\tau)=v_2T_2(\tau)$$

and therefore, trivially,

$$v_1T_1(\pi)T_1(\tau)=v_1T_1(\tau)\ \approx\ v_2T_2(\tau)=v_2T_2(\pi)T_2(\tau).$$

Now, assume $\pi=\pi'(\tau_k,j_k)$, where π' has length $k-1\geqslant 0$. Then $T_i(\pi)=T_i(\pi')T_i(\tau_k)[k,j_k]$. Assume $v_iT_i(\pi')=s_iu_i$, where u_i is the maximal suffix in $\tilde{T}_\Delta(*)$. By the inductive assumption $(s_1\theta_1)u_1r_1=(s_2\theta_2)u_2r_2$ for substitutions θ_i of the variables $x_{\kappa,j}$, $\kappa=1,\ldots,k-1$, where $x_{\kappa,j}\theta_i\in\tilde{T}_\Delta(x_{\kappa,j})$, and trees $r_1,r_2\in\tilde{T}_\Delta(*)$. We distinguish two cases.

Case 1: $T_i(\tau_k)\notin\tilde{T}_\Delta(x_{j_k})$. Then $T_i(\tau_k)$ contains some variable x_j with $j\neq j_k$. Assume $T_i(\tau_k)[k,j_k]=u'_iy_i$, where u'_i is the maximal $*$-prefix in $\tilde{T}_\Delta(*)$. By assertion (1) of Property (U3′), we have $u_1u'_1=u_2u'_2$. By Property (U2), $y_1\theta'_1=y_2\theta'_2$ for substitutions θ'_i, where $x\theta'_i\in\tilde{T}_\Delta(x)$ for every variable x occurring in y_i. Since the set of variables occurring in y_i are disjoint from the set of variables $\neq *$ occurring in s_i, we can combine θ_i and θ'_i to obtain substitutions $\bar{\theta}_i$ of the variables $x_{\kappa,j}$ with $\kappa=1,\ldots,k$ and $*$ such that

$$\begin{aligned}v_1T_1(\pi)\bar{\theta}_1&=v_1(T_1(\pi')u_1u'_1y_1)\bar{\theta}_1=v_1(T_1(\pi')\theta_1)u_1u'_1(y_1\theta'_1)\\&=v_2(T_2(\pi')\theta_2)u_2u'_2(y_2\theta'_2)=v_2(T_2(\pi')u_2u'_2y_2)\bar{\theta}_2=v_2T_2(\pi)\bar{\theta}_2.\end{aligned}$$

Moreover, assume τ is a T_i-initial q-transition. We factorize $y_i=y'_iw_i$, where w_i is the maximal $*$-suffix of y_i in $\tilde{T}_\Delta(*)$. Since the maximal $*$-suffix of $T_i(\pi)$ in $\tilde{T}_\Delta(*)$ is w_i, we obtain by assertion (2) of Property (U3′) that $w_1T_1(\tau)=w_2T_2(\tau)$. Hence, also $v_1(T_1(\pi)T_1(\tau))\bar{\theta}_2=v_2(T_2(\pi)T_2(\tau))\bar{\theta}_1$, and statement (1) holds true.

Case 2: $T_i(\tau_k)\in\tilde{T}_\Delta(x_{j_k})$. This case causes some (technical) trouble since Property (U3′) for (v_1,v_2) does not speak (explicitly) about patterns containing a single variable at all! First, we show

$$(+)\qquad u_1T_1(\tau_k)[k,j_k]\approx u_2T_2(\tau_k)[k,j_k].$$

For a proof of $(+)$, we again distinguish two cases.

Case 2.1: A partial (q,p)-computation π_1 and a transition $\tau=(p,a,q_1\ldots q_m)\in\delta$ exist such that $T_i(\pi_1)\in T_\Delta(*)$, and $T_i(\tau)$ contains occurrences of variables $x_j,x_{j'}$, with $j\neq j'$. Assume $T_i(\tau)=u'_iy_i$, where u'_i is the maximal $*$-prefix in $\tilde{T}_\Delta(*)$. Define $r'_i=T_i(\pi_1)u'_i$.

Then by Property (U3′) for (v_1, v_2), $u_1 T_1(\tau_k)[k,j_k]r_1' = u_2 T_2(\tau_k)[k,j_k]r_2'$, which we wanted to prove.

Case 2.2: A partial (q,p_1)-computation π_1, a partial (p_1,p_2)-computation π_2, and T_i-initial p_j-transitions τ_j', $j=1,2$, exist such that $T_i(\pi_j)\in\tilde{T}_\Delta(*)$ and $T_1(\tau_1')\neq T_1(\pi_2)T_1(\tau_2')$.

We claim that also $T_2(\tau_1')\neq T_2(\pi_2)T_2(\tau_2')$.

To prove this assume, for a contradiction, $T_2(\tau_1')=T_2(\pi_2)T_2(\tau_2')$.

Then by assertion (2) of Property (U3′), substitutions θ_1,θ_2 exist with $x_{\kappa,j}\theta_i\in\tilde{T}_\Delta(x_{\kappa,j})$ such that both

$$v_1 T_1(\pi(\tau,j))\theta_1 T_1(\pi_1)T_1(\tau_1') = v_2 T_2(\pi(\tau,j))\theta_2 T_2(\pi_1)T_2(\tau_1'), \text{ and}$$

$$v_1 T_1(\pi(\tau,j))\theta_1 T_1(\pi_1\pi_2)T_1(\tau_2') = v_2 T_2(\pi(\tau,j))\theta_2 T_2(\pi_1\pi_2)T_2(\tau_2').$$

By assumption, the two left-hand sides are equal, whereas the two right-hand sides are not: contradiction.

Therefore, $T_i(\tau_1')\neq T_i(\pi_2)T_i(\tau_2')$ for $i=1,2$. Define $r_i'=T_i(\pi_1)\in\tilde{T}_\Delta(*)$, $s_i=T_i(\tau_1')$, and $s_i'=T_i(\pi_2)T_i(\tau_2')$ for $i=1,2$. By assertion (2) of Property (U3′), $u_1 T_1(\tau_k)[k,j_k]r_1's_1 = u_2 T_2(\tau_k)[k,j_k]r_2's_2$ and $u_1 T_1(\tau_k)[k,j_k]r_1's_1' = u_2 T_2(\tau_k)[k,j_k]r_2's_2'$. Therefore, we can apply factorization and obtain that either $u_1 T_1(\tau_k)[k,j_k]$ is a $*$-prefix of $u_2 T_2(\tau_k)[k,j_k]$ or vice versa. From this follows assertion $(+)$.

Applying $(+)$ we conclude that $(s_1\theta_1)u_1 T_1(\tau_k)[k,j_k]r_1' = (s_2\theta_2)u_2 T_2(\tau_k)[k,j_k]r_2'$ for suitable $r_i'\in\tilde{T}_\Delta(*)$. Hence, $T_1(\pi)\approx T_2(\pi)$. Moreover, if there is a T_i-initial q-transition τ then Property (U3′) for (v_1,v_2) yields $u_1 T_1(\tau_k)[k,j_k]T_1(\tau) = u_2 T_2(\tau_k)[k,j_k]T_2(\tau)$. Hence,

$$\begin{aligned} v_1(T_1(\pi)T_1(\tau))\theta_1 &= (s_1\theta_1)u_1 T_1(\tau_k)[k,j_k]T_1(\tau) \\ &= (s_2\theta_1)u_2 T_2(\tau_k)[k,j_k]T_2(\tau) = v_2(T_2(\pi)T_2(\tau))\theta_2. \end{aligned}$$

Therefore, $T_1(\pi)T_1(\tau)\approx T_2(\pi)T_2(\tau)$, in accordance with assertion (1).

For a proof that (2) and (3) are equivalent observe that whenever $f=q\in Q_F$ or $q\notin U(A,T_i)$, the set of partial (f,q)-computations $\pi=(\tau_1,j_1)\cdots(\tau_k,j_k)$ of A equals the set of sequences of labels of paths in $G(A)$ from node f to q. Therefore, Property (U3) for (v_1,v_2) implies Property (U3′) for (v_1,v_2). For the opposite implication assume Π has Property (U3′) for (v_1,v_2). Then, a map diff(_) with Property (U3) exists and is uniquely defined iff for every $f\in Q_F$ and $q\in[Q\setminus U(A,T_i)]\cup Q_F$, every partial (f,q)-computation π has the following property:

$(++)$ If u_i is the maximal suffix of $v_iT_i(\pi)$ in $\tilde{T}_\Delta(*)$ then $\text{diff}(u_1,u_2)=\text{diff}(q)$.

So, assume π and π' are partial (f,q)-computations where, for $i=1,2$, u_i is the maximal suffix of $v_iT_i(\pi)$ in $\tilde{T}_\Delta(*)$ and u_i' is the maximal suffix of $v_iT_i(\pi')$ in $\tilde{T}_\Delta(*)$.

Assume $\text{diff}(u_1,u_2)=(s_1,s_2)$ and $\text{diff}(u_1',u_2')=(s_1',s_2')$. As above, we distinguish two cases.

Case 1: A partial (q,p)-computation π_1 and a transition $\tau=(p,a,q_1\ldots q_m)\in\delta$ exist such that $T_i(\pi_1)\in T_\Delta(*)$, and $T_i(\tau)$ contains occurrences of variables $x_j, x_{j'}$ with $j\neq j'$.

Define $r_i = T_i(\pi_1)u_i'$, where u_i' is the maximal prefix of $T_i(\tau)$ in $\tilde{T}_\Delta(*)$. By Property (U3′) for (v_1, v_2), $u_1 r_1 = u_2 r_2$ and $u_1' r_1 = u_2' r_2$. Hence, by the definition of diff(_,_), $s_1 r_1 = s_2 r_2$ and $s_1' r_1 = s_2' r_2$. W.l.o.g. assume $s_1 = *$. Then $r_1 = s_2 r_2$, which implies $s_1' s_2 r_2 = s_2' r_2$. By bottom cancellation we obtain $s_1' s_2 = s_2'$. By the definition of diff(_,_), either $s_1' = *$ or $s_2' = *$. Therefore, $s_1' = *$ and $s_2' = s_2$. Consequently, $(s_1, s_2) = (s_1', s_2')$, which we wanted to prove.

Case 2: A partial (q, p_1)-computation π_1, a partial (p_1, p_2)-computation π_2, and two T_i-initial p_j-transitions τ_j', $j = 1, 2$, exist such that $T_i(\pi_j) \in \tilde{T}_\Delta(*)$ and $T_1(\tau_1') \neq T_1(\pi_2) T_1(\tau_2')$. As above, we conclude that also $T_2(\tau_1') \neq T_2(\pi_2) T_2(\tau_2')$.

Define $r_i = T_i(\pi_1) T_i(\tau_1)$ and $r_i' = T_i(\pi_1 \pi_2) T_i(\tau_2)$. Then, by Property (U3′) for (v_1, v_2),

$$u_1 r_1 = u_2 r_2, \qquad u_1 r_1' = u_2 r_2', \qquad u_1' r_1 = u_2' r_2 \text{ and } u_1' r_1' = u_2' r_2'.$$

Hence, by the definition of diff(_,_) and top cancellation,

$$s_1 r_1 = s_2 r_2, \qquad s_1 r_1' = s_2 r_2', \qquad s_1' r_1 = s_2' r_2 \text{ and } s_1' r_1' = s_2' r_2'.$$

By factorization we, w.l.o.g., assume that s_1 is a $*$-prefix of s_2. Then $s_1 = *$ and, therefore, $r_1 = s_2 r_2$ and $r_1' = s_2 r_2'$. Again, substituting this result into the equations for s_i' and using bottom cancellation we obtain $s_1' s_2 = s_2'$. Consequently, $s_1' = *$ and $s_2' = s_2$. Hence, $(s_1, s_2) = (s_1', s_2')$, which we wanted to prove. □

We are now ready to prove the main theorem of this section:

Theorem 5.2. *For a strongly reduced pairing* $\Pi = (A, T_1, T_2)$ *and* $v_1, v_2 \in \tilde{T}_\Delta(*)$ *the following two statements are equivalent*:

(1) $v_1 T_1 \equiv v_2 T_2$ *w.r.t.* A;

(2) Π *has Properties* (U1) *and* (U2) *and Property* (U3) *for* (v_1, v_2).

It can be decided in polynomial time whether or not $v_1 T_1 \equiv v_2 T_2$ w.r.t. A.

Proof. (1) *implies* (2): Assume statement (1) is true. Then by Proposition 4.3, Π has Properties (U1) and (U2). From Proposition 4.2, we deduce that for every partial (f, q)-computation π with $f \in Q_F$ and $q \in [Q \setminus U(A, T_i)] \cup Q_F$, $T_1(\pi) \approx T_2(\pi)$ and $T_1(\pi) T_1(\tau) \approx T_2(\pi) T_2(\tau)$ for all T_i-initial q-transitions τ. Hence by Proposition 5.1, Π has Property (U3) for (v_1, v_2). Thus, statement (1) implies statement (2).

(2) *implies* (1): For a proof of the converse implication assume Π has Properties (U1), (U2) and (U3) for (v_1, v_2) but $v_1 T_1(\phi) \neq v_2 T_2(\phi)$, where ϕ is an accepting computation of A for some tree t. Since $v_1 T_1(\phi) \neq v_2 T_2(\phi)$, there is some node o in $O(v_1 T_1(\phi)) \cap O(v_2 T_2(\phi))$ with $v_1 T_1(\phi)(o) \neq v_2 T_2(\phi)(o)$.

Let $\langle \varepsilon, w_i, \bar{o}_i \rangle$, $i = 1, 2$, be paths in t of minimal length with the following property. Assume τ_i is the transition chosen in ϕ at node $\bar{o}_i$ and π_i the subcomputation of ϕ on $\mathrm{tr}(w_i)$ with $u_i = v_i T_i(\pi_i) T_i(\tau_i)$. Then for $i = 1, 2$, o is a node in u_i with label in Δ. Especially,

$$u_1(o) = v_1 T_1(\phi)(o) \neq v_2 T_2(\phi)(o) = u_2(o).$$

Let $w_0=(a_1,j_1)\cdots(a_{k-1},j_{k-1})$ be the maximal common prefix of w_1 and w_2. Let π_0 be the subcomputation of ϕ on $\mathrm{tr}(w_0)$ with $u_i'=v_iT_i(\pi_0)$, and $\tau=\phi(j_1\ldots j_{k-1})$.

First assume $w_1\neq w_0\neq w_2$. Then, $w_i=w_0(a,j^{(i)})w_i'$ for some $j^{(1)}\neq j^{(2)}$. Since Π has Properties (U1), (U2) and (U3) for (v_1,v_2) we deduce from Proposition 5.1 that

$$(+)\qquad u_1'T_1(\tau)[k,j^{(1)}]=v_1T_1(\pi_0(\tau,j^{(1)}))\approx v_2T_2(\pi_0(\tau,j^{(1)}))=u_2'T_2(\tau)[k,j^{(1)}].$$

Moreover, $u_1=u_1'T_1(\tau)[k,j^{(1)}]r_1$ and $u_2=u_2'T_2(\tau)[k,j^{(2)}]r_2$ for some trees r_1,r_2. Hence, node o can be factored $o=o_1o_1'=o_2o_2'$, where o_1 is a node in $u_1'T_1(\tau)[k,j^{(1)}]$ labeled with $*$, and o_2 is a node in $u_2'T_2(\tau)[k,j^{(2)}]$ labeled with $*$. Either o_1 is a prefix of o_2 or vice versa. In $u_2'T_2(\tau)[k,j^{(1)}]$ however, o_2 is labeled with $x_{k,j^{(2)}}$. Applying Fact 1.7 to (+), we conclude that neither o_1 is a prefix of o_2 nor o_2 a prefix of o_1: contradiction.

It remains to consider the case where $w_0=w_1$ or $w_0=w_2$. W.l.o.g. assume $w_0=w_1$, and consider $\tau_2=\phi(o_2)$.

By Property (U1), τ_2 is either both T_1-initial and T_2-initial or both $T_1(\tau_2)$ and $T_2(\tau_2)$ contain occurrences of variables. Assume τ_2 is both T_1-initial and T_2-initial. Since o is a node in $v_iT_i(\pi_2)T_i(\tau_2)$ for $i=1,2$, $v_1T_1(\pi_2)T_1(\tau_2)$ and $v_2T_2(\pi_2)T_2(\tau_2)$ are not comparable. Hence by Proposition 5.1, Π cannot have Property (U3) for (v_1,v_2): contradiction.

If τ_2 is not T_i-initial for $i=1,2$, then $T_i(\tau_2)$ contains an occurrence of a variable x_j. Again, since o is a node in $v_iT_i(\pi_0)T_i(\tau_2)$ for $i=1,2$, $v_1T_1(\pi_2(\tau_2,j))$ and $v_2T_2(\pi_2(\tau_2,j))$ cannot be comparable. Now Proposition 5.1 applied to $\pi(\tau_2,j)$ implies that Π cannot have Property (U3) for (v_1,v_2). This finishes the proof.

The algorithm deciding (v_1,v_2)-equivalence is as follows:

(0) Input: strongly reduced pairing $\Pi=(A,T_1,T_2)$.
(1) Decide whether or not Π has Property (U1).
If Π does not have Property (U1) then return: "$v_1T_1\not\equiv v_2T_2$ w.r.t. A".
(time: $O(|\Pi|)$).
(2) Decide whether or not Π has Property (U2).
If Π does not have Property (U2) then return: "$v_1T_1\not\equiv v_2T_2$ w.r.t. A".
(time: $O(|\Pi|)$).
(3) Decide whether or not Π has Property (U3) for (v_1,v_2).
If Π does not have Property (U3) for (v_1,v_2) then return: "$v_1T_1\not\equiv v_2T_2$ w.r.t. A"; otherwise return: "$v_1T_1\equiv v_2T_2$ w.r.t. A"
(time: $O(|\Pi|\cdot\log(|v_1|+|v_2|+|\Pi|))$).

It remains to prove that all three steps can be executed within the given time bounds. By Proposition 3.2, Property (U1) can be decided in linear time. By Proposition 2.5, we can compute the kernel decompositions of all output patterns containing at least one variable. This gives a linear-time procedure deciding (U2). Moreover, using the kernel decompositions we can, by Proposition 2.4, compute the subset $I\subseteq\tilde{T}_\Delta(*)$ needed to spell the trees occurring in the description of Property (U3) for (v_1,v_2). It remains to show that Property (U3) for (v_1,v_2) can be decided within the given time bounds. Clearly, the graph $G(A)$ can be constructed in linear time. By a depth-first traversal through $G(A)$ the values diff(q) can be computed where every edge is considered only once. The lengths of words from I^* occurring are bounded by $|v_1|+|v_2|+|\Pi|$. Therefore, the map diff(_) with the given properties can be shown to exist or not to exist in time $O(|\Pi|\cdot(|v_1|+|v_2|+|\Pi|))$. However, using the same algorithmic idea as in [8] for implementing the necessary comparisons of occurring strings one can improve this upper time bound to $O(|\Pi|\cdot\log(|v_1|+|v_2|+|\Pi|))$. □

The complexity bounds for the given algorithm are remarkable since it meets the best-known upper bound for the corresponding problem for words [8]. Hence, an improvement of the given result is only possible if one finds a more efficient algorithm for the word case as well.

Applying the algorithm of the proof of Theorem 5.2 to the problem of single-valuedness we obtain:

Theorem 5.3. *For every FST $M=(A,T)$ it can be decided in polynomial time whether or not* $\text{val}(M)\leqslant 1$.

Proof. The algorithm is as follows:

(0) Input: FST $M=(A,T)$ with n states.
(1) If $L(A)=\emptyset$ then return: "val(M)=0";
otherwise construct an equivalent reduced FST (time: $O(|M|)$).
W.l.o.g. M is reduced.
(2) Compute the canonical reduced pairing $M^2=(A^2,T_1,T_2)$ (time: $O(|M|^2)$).
(3) Compute the sets Const(A^2,T_i) and the functions G_{T_i} (time: $O(|M|^2)$).
Compute the equivalent strongly reduced pairing $\Pi=(A^2,T_1',T_2')$ (time: $O(|M|^2)$).
(4) Decide whether or not $T_1'\equiv T_2'$ w.r.t. A^2.
If $T_1'\equiv T_2'$ w.r.t. A^2 then return: "val(M)=1";
otherwise return: "val(M)>1" (time: $O(|M|^2\cdot\log|M|)$). □

Theorem 5.3 should be seen in contrast to the result of Engelfriet in [4]. Engelfriet proves a polynomial upper bound on the depth of a witness for nonsingle-valuedness. From this upper bound, it is not difficult to derive a nondeterministic polynomial-time algorithm deciding nonsingle-valuedness. This algorithm can roughly be described as follows:

(1) Guess a node in the output where the two output values differ from each other;
(2) Guess two nodes of the witness which produce this output node;
(3) Verify that these guesses have been reasonable.

This method gives no hint how a deterministic polynomial-time algorithm may look like. However, as pointed out in [13], it allows for a generalization to construct a nondeterministic polynomial-time algorithm which decides whether an FST is *not*

k-valued. So far, it is open whether a *deterministic* polynomial algorithm exists for this problem for any $k > 1$.

As further applications of Theorem 5.2 we obtain:

Corollary 5.4. *Assume $M_i = (A_i, T_i)$ are single-valued FSTs such that $L(A_1) = L(A_2)$. Then it can be decided in deterministic polynomial time whether or not $T(M_1) = T(M_2)$.*

Observe, however, that by [12], deciding the equivalence of the underlying FTAs A_i is deterministic exponential-time-complete in general. Nevertheless, there are important subclasses which admit faster algorithms. If, e.g., the underlying FTAs are deterministic then equivalence of A_1 and A_2 is decidable in polynomial time. In fact, by the results in [12] it suffices that A_1 and A_2 are m-ambiguous for some constant m, where m-ambiguous means that for every input there are at most m different accepting computations (e.g., for deterministic FTAs there is at most one accepting computation for every input). Thus we have:

Corollary 5.5. *Assume $m \geqslant 1$ is a constant, A_i are m-ambiguous FTAs, and $M_i = (A_i, T_i)$, $i = 1, 2$, are single-valued FSTs. Then it can be decided in polynomial time whether or not $T(M_1) = T(M_2)$.*

6. Finite-valuedness

In this section we show how the ideas and algorithms of the last two sections can be used to construct an algorithm deciding finite-valuedness in deterministic polynomial time. We start with the two Properties (F1) and (F2) of [13] that characterize finite-valuedness and derive an equivalent set of properties, each of which can be decided in deterministic polynomial time. The derivation is done in three steps. First, we subdivide the Properties (Fi) into pairs of Properties (Fi.0) and (Fi.1) (Proposition 6.2) from which (Fi.0) are easy to decide. Secondly, we reduce Properties (Fi.1) to properties of partial computations on paths of input trees. We start with a property (F0.1) which is implied by Property (F1.1). We give three Properties (G1), (G2) and (G3) which, together with a length property (L0), characterize (F0.1) (Theorem 6.6). Properties (Gi) correspond to the Properties (U1), (U2) and (U3) characterizing single-valuedness. In the third step we consider FSTs M having Property (F0.1). We find that then M already has Property (F2.1) (Theorem 6.4). Moreover, Property (F1.1) is equivalent to a simple length property (L1) (Theorem 6.7), which generalizes (L0). Since (F1.0), (F2.0), (G1), (G2), (G3) and (L1) are decidable in deterministic polynomial time we conclude that finite-valuedness can be decided in deterministic polynomial time (Theorem 6.9).

For this section, assume $M = (A, T)$ is a reduced FST with $A = (Q, \Sigma, \delta, Q_F)$. All properties are formulated for pairs of states $(q, p) \in Q^2$. First, we reformulate criteria from [13] by means of $M^{(3)} = (A^{(3)}, T_1, T_2, T_3)$ and proper $(\langle q, q, p\rangle, \langle q, p, p\rangle)$-computations ϕ of $A^{(3)}$.

M has Property (Fi) for (q,p), $i=0,1,2$, iff the corresponding statement (Fi) holds:

(F0) For every proper $(\langle q,q,p\rangle,\langle q,p,p\rangle)$-computation ϕ of $A^{(3)}$,

$$T_1(\phi)T_2(\phi)=T_2(\phi)T_3(\phi). \tag{0}$$

(F1) For every proper $(\langle q,q,p\rangle,z)$-computation ϕ_1, (z,z)-computation ϕ_2 and $(z,\langle q,p,p\rangle)$-computation ϕ_3 of $A^{(3)}$,

$$T_1(\phi_1\phi_2\phi_3)T_2(\phi_1\phi_3)=T_2(\phi_1\phi_2\phi_3)T_3(\phi_1\phi_3). \tag{1}$$

(F2) For every proper $(\langle q,q,p\rangle,\langle q,p,p\rangle z)$-computation ϕ_1, (z,z)-computation ϕ_2 and (z,ε)-computation ϕ_3 of $A^{(3)}$,

$$T_1(\phi_1[x_1,\phi_2\phi_3])T_2(\phi_1[x_1,\phi_3])=T_2(\phi_1[x_1,\phi_2\phi_3])T_3(\phi_1[x_1,\phi_3]). \tag{2}$$

Observe that Property (F0) for (q,p) is a special case of (F1) for (q,p) (choose $\phi_2=x_1$).

Theorem 6.1 (Seidl [13]). *For a reduced FST $M=(A,T)$, the following two statements are equivalent*:

(1) $\mathrm{val}(M)<\infty$;

(2) *M has Properties* (F1) *and* (F2) *for all* (p,q).

For $i=0,1,2$, Property (Fi.0) for (q,p) is obtained from Property (Fi) for (q,p) by replacing the conclusion (i) with $(i.0)$:

$$T_1(\phi)\in\{\bot,x_1\} \text{ iff } T_3(\phi)\in\{\bot,x_1\}, \tag{0.0}$$

$$T_1(\phi_2)\in\{\bot,x_1\} \text{ iff } T_2(\phi_2)\in\{\bot,x_1\}, \tag{1.0}$$

$$T_1(\phi_2)\in\{\bot,x_1\} \text{ iff } T_2(\phi_2)\in\{\bot,x_1\}. \tag{2.0}$$

Observe that Property (F0.0) for (q,p) is implied by Property (F1.0) for (q,p). Using size arguments as in [13], it can be shown that for $i=0,1$, (Fi.0) for (q,p) is implied by (Fi) for (q,p), whereas (F2.0) for (q,p) is implied by (F0) and (F2) for (q,p). If $p\notin U(A,T)$, then the corresponding output trees $T_i(\phi)$ in the formulation of Property (F0) and $T_i(\phi_j)$ in the formulation of (F1) are in $\tilde{T}_\Delta(x_1)$. The versions of Properties (Fi) for (p,q) dealing only with $T_1(\phi), T_3(\phi)\notin\{\bot,x_1\}$ and $T_1(\phi_2), T_2(\phi_2)\notin\{\bot,x_1\}$, respectively, are called (Fi.1) for (q,p). We have:

Proposition 6.2. *Assume M is a reduced FST.*

(i) *M has Property* (F0) *for (q,p) iff M has Properties* (F0.0) *and* (F0.1) *for (q,p).*

(ii) *Assume M has Property* (F0) *for (q,p). Then for $i=1,2$, M has Property* (Fi) *for (q,p) iff M has Properties* (Fi.0) *and* (Fi.1) *for (q,p).*

Assume $q,p\in Q$. The reduced FST $M=(A,T)$ has Property (G0) for (q,p) iff

(G0) A proper $(\langle q,q,p\rangle,\langle q,p,p\rangle)$-computation of $A^{(3)}$ exists; and a proper $(\langle q,p\rangle,\langle q,p\rangle)$-computation $\phi=\langle\phi_1,\phi_2\rangle$ of $A^{(2)}$ exists with $T(\phi_1)\neq x_1$ and $T(\phi_2)\neq x_1$.

If (G0) does not hold for (q,p) then Properties (F*i*.1) for (q,p) trivially hold. Therefore, for testing Properties (F*i*.1) for (q,p), we always can, w.l.o.g., assume that M has Property (G0) for (q,p). Clearly, (G0) for (q,p) can be decided in deterministic polynomial time.

If M has Property (G0) for (q,p) then especially $p,q\notin\text{Const}(A,T)$.

Note that neither $\langle q,q,p\rangle$ nor $\langle q,p,p\rangle$ are necessarily useful for $A^{(3)}$. Therefore, we modify $A^{(3)}$ as follows. For $M^{(3)}=(A^{(3)},T_1,T_2,T_3)$ with $A^{(3)}=(Q,\Sigma,\delta,Q_F)$, we define $M^{(3)}_{q,p}=(A^{(3)}_{q,p},T'_1,T'_2,T'_3)$. We first add a tag 0 or 1 to the states of $A^{(3)}$. The states tagged 1 are those occurring on partial computations from $\langle q,q,p\rangle$ to $\langle q,p,p\rangle$; the remaining ones are tagged 0. The output functions are modified accordingly. To enforce that $\langle\langle q,q,p\rangle,1\rangle=\langle qqp,1\rangle$ and $\langle\langle q,p,p\rangle,1\rangle=\langle qpp,1\rangle$ are useful we introduce a new transition $(\langle qpp,1\rangle,\#,\varepsilon)$ for some new symbol $\#$ of rank 0 and include $\langle qqp,1\rangle$ into the set of final states. Thus, $A^{(3)}_{q,p}=(Q\times\{0,1\},\Sigma\cup\{\#\},\delta',\{\langle qqp,1\rangle\})$, where

- $(\langle qpp,1\rangle,\#,\varepsilon)\in\delta'$ with $T_i((\langle qpp,1\rangle,\#,\varepsilon))=\#$; and
- if $\tau=(z,a,z_1\ldots z_m)\in\delta$ then
  $\tau_0=(\langle z,0\rangle,a,\langle z_1,0\rangle\cdots\langle z_m,0\rangle)\in\delta'$; and whenever $m>0$,
  $\tau_j=(\langle z,1\rangle,a,\langle z_1,0\rangle\cdots\langle z_{j-1},0\rangle\langle z_j,1\rangle\langle z_{j+1},0\rangle\cdots\langle z_m,0\rangle)\in\delta'$, $j=1,\ldots,m$, with $T'_i(\tau_j)=T_i(\tau)$ for $j=0,\ldots,m$.

Let $M^3_{q,p}=(A^3_{q,p},T'_1,T'_2,T'_3)$, where $A^3_{q,p}$ is obtained from $A^{(3)}_{q,p}$ by removing all useless states, and the output functions of $M^3_{q,p}$ are the output functions of $M^{(3)}_{q,p}$ restricted to the set of remaining transitions.

Then, $\langle z,1\rangle$ is a state of $A^3_{q,p}$ iff a proper $(\langle q,q,p\rangle,z)$-computation and a proper $(z,\langle q,p,p\rangle)$-computation of $A^{(3)}$ exist. Moreover, the proper $(\langle qqp,1\rangle,\langle z,1\rangle)$-computations ϕ of $A^3_{q,p}$ are in one-to-one correspondence to the proper $(\langle q,q,p\rangle,z)$-computations of $A^{(3)}$. Also, if ϕ corresponds to $\langle\phi_1,\phi_2,\phi_3\rangle$ then $T'_i(\phi)=T(\phi_i)$. Therefore, we also write $\phi=\langle\phi_1,\phi_2,\phi_3\rangle$. Accordingly, $\langle q_1q_2q_3,0\rangle$-computations ψ of $\bar{A}$ are in one-to-one correspondence to $\langle q_1,q_2,q_3\rangle$-computations of $A^{(3)}$ whenever $\langle q_1q_2q_3,0\rangle$ is a state of $A^3_{q,p}$. We allow ourselves to decompose ψ into a triple of q_i-computations as well.

For the following, let $M^3_{q,p}=(\bar{A},T_1,T_2,T_3)$. Since we do not know how to handle the infinite alphabet $I_\Delta(x_1)$ necessary to spell the outputs $T_i(\phi)$ of a proper $(\langle qqp,1\rangle,\langle qpp,1\rangle)$-computation ϕ of $\bar{A}$ we try to find equivalent formulations of the given properties by means of *partial* $(\langle qqp,1\rangle,\langle qpp,1\rangle)$-computations π of $\bar{A}$. π corresponds to a partial $(\langle q,q,p\rangle,\langle q,p,p\rangle)$-computation $\langle\pi_1,\pi_2,\pi_3\rangle$ of $A^{(3)}$. For convenience, we write: $\pi=\langle\pi_1,\pi_2,\pi_3\rangle$.

Assume $v_1,v_2,v_3\in\tilde{T}_\Delta(*)$ and $\langle z,0\rangle$ is a state of $\bar{A}$. For $i=1,2$, we write $v_iT_i\equiv_z v_{i+1}T_{i+1}$ iff $v_iT_i(\phi)=v_{i+1}T_{i+1}(\phi)$ for every $\langle z,0\rangle$-computation ϕ of $\bar{A}$. By Theorem 5.2, it can be decided in polynomial time whether or not $v_iT_i\equiv_z v_{i+1}T_{i+1}$.

The following fundamental proposition must be seen in analogy to Proposition 4.2 for single-valuedness.

Proposition 6.3. *Assume $M=(A,T)$ is a reduced FST. If M has Property (F0.1) for (q,p) then for every proper $(\langle qqp,1\rangle,\langle z_1,1\rangle\langle z_2,0\rangle\cdots\langle z_m,0\rangle)$-computation ϕ of $\bar{A}$ the following holds:*

(i) *For all $j=2,\ldots,m$, and $\langle z_j,0\rangle$-computations ψ_1,ψ_2 of $\bar{A}$,*

$$T_1(\psi_1)=T_1(\psi_2)\text{ iff }T_2(\psi_1)=T_2(\psi_2)\text{ iff }T_3(\psi_1)=T_3(\psi_2).$$

Especially,

$$\langle z_j,0\rangle\in\text{Const}(\bar{A},T_1)\text{ iff }\langle z_j,0\rangle\in\text{Const}(\bar{A},T_2)\text{ iff }\langle z_j,0\rangle\in\text{Const}(\bar{A},T_3).$$

(ii) *Assume $m>1$, $z_j\notin\text{Const}(\bar{A},T_i)$ for all $j>1$, and $T_i(\phi)=v_iy_i[u_{i1},\ldots,u_{im}]$ is the kernel decomposition of $T_i(\phi)$. Then*

- $y_1=y_2=y_3$.
- $v_1=v_2$; $u_{21}=u_{31}$; $u_{11}\approx u_{21}$;
- *if* $z_1=qpp$ *then* $u_{11}v_2=u_{21}v_3$.
- *For every* $j=2,\ldots,m$, $u_{1j}T_1\equiv_{z_j}u_{2j}T_2$ *and* $u_{2j}T_2\equiv_{z_j}u_{3j}T_3$.

Proof. Let $\phi=\langle\phi_1,\phi_2,\phi_3\rangle$. Let $m>1$. W.l.o.g. we assume $z_1=qpp$ and $m=2$. We start with a proof of statement (i).

Let $\psi_j=\langle\psi_{j1},\psi_{j2},\psi_{j3}\rangle$. First assume $T_1(\psi_1)=T_1(\psi_2)$ but $T_2(\psi_1)\neq T_2(\psi_2)$. Assume the maximal depth of a node in $T_2(\phi)$ with label x_1 is n. Consider

$$\phi^{(n+1)}=\langle\phi_1^{n+1},\phi_2\phi_3^n,\phi_3^{n+1}\rangle,$$

where the x_1-substitution is written as concatenation. From Property (F0.1) for (q,p) we deduce that

$$(1)\qquad T_1\phi^{(n+1)}[x_1,\psi_i])T_2(\phi^{(n+1)}[x_1,\psi_i])=T_2(\phi^{(n+1)}[x_1,\psi_i])T_3(\phi^{(n+1)}[x_1,\psi_i]).$$

By assumption, $T_1(\phi[x_1,\psi_1])=T_1(\phi[x_1,\psi_2])$; therefore,

$$(2)\qquad T_1(\phi^{(n+1)}[x_1,\psi_1])=T_1(\phi[x_1,\psi_1])^{n+1}=T_1(\phi[x_1,\psi_2])^{n+1}$$
$$=T_1(\phi^{(n+1)}[x_1,\psi_2]).$$

Since $T_1(\phi)\neq x_1$, equation (1) and the definition of n imply that some node o exists such that $T_1(\phi^{(n+1)}[x_1,\psi_i])/o=T_2(\psi_i)$ for both $i=1$ and $i=2$. Therefore, (2) implies that $T_2(\psi_1)=T_2(\psi_2)$: contradiction.

Now assume $T(\psi_{11})\neq T(\psi_{21})$ but $T(\psi_{12})=T(\psi_{22})$. Define $\phi^{(1)}=\langle\phi_1,\phi_1,\phi_3\rangle$ and $\phi^{(2)}=\langle\phi_1[x_1,x_3],\phi_2[x_1,x_3],\phi_3[x_1,x_3]\rangle$.

By Property (F0.1) for (q,p),

$$T_1((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])T_2((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])$$
$$=T_2((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])T_3((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])$$

or equivalently,

$$T(\phi_1[x_1,\psi_{i1}])T(\phi_1[x_1,\psi_{j1}])T(\phi_1[x_1,\psi_{i1}])T(\phi_2[x_1,\psi_{j2}])$$
$$=T(\phi_1[x_1,\psi_{i1}])T(\phi_2[x_1,\psi_{j2}])T(\phi_3[x_1,\psi_{i3}])T(\phi_3[x_1,\psi_{j3}])$$

for every $i,j\in\{1,2\}$. Thus, top cancellation yields:

$$(1)\qquad T(\phi_1[x_1,\psi_{j1}])T(\phi_1[x_1,\psi_{i1}])T(\phi_2[x_1,\psi_{j2}])$$
$$=T(\phi_2[x_1,\psi_{j2}])T(\phi_3[x_1,\psi_{i3}])T(\phi_3[x_1,\psi_{j3}])\quad\text{for every } i,j\in\{1,2\}.$$

If $T(\psi_{13})=T(\psi_{23})$ then the right-hand side of (1) is independent of any choice of i and j, which immediately gives a contradiction. Therefore, $T(\psi_{13})\neq T(\psi_{23})$. It follows that we can apply factorization and deduce that

$$(2)\qquad T(\phi_1[x_1,x_3])T(\phi_1[x_1,x_2])T(\phi_2[x_1,\psi_{j2}])$$
$$\approx T(\phi_2[x_1,\psi_{j2}])T(\phi_3[x_1,x_2])T(\phi_3[x_1,x_3]).$$

Especially, the first x_1-irreducible factor of the left-hand side of (2) containing an occurrence of x_3 is to the left of the first x_1-irreducible factor containing an occurrence of x_2, whereas the first x_1-irreducible factor of the right-hand side containing an occurrence of x_3 is to the right of the first x_1-irreducible factor containing an occurrence of x_2. By Fact 1.9, this is impossible. We conclude that $T_1(\psi_1)=T_1(\psi_2)$ iff $T_2(\psi_1)=T_2(\psi_2)$.

To prove the last equivalence of (i) first assume $T(\psi_{1i})=T(\psi_{2i})$ for $i=1,2$ but $T(\psi_{13})\neq T(\psi_{23})$. Hence, also

$$T_i(\phi[x_1,\psi_1])=T(\phi_i[x_1,\psi_{1i}])=T(\phi_i[x_1,\psi_{2i}])=T_i(\phi[x_1,\psi_2])\quad\text{for } i=1,2$$

but

$$T_3(\phi[x_1,\psi_1])=T(\phi_3[x_1,\psi_{13}])\neq T(\phi_3[x_1,\psi_{23}])=T_3(\phi[x_1,\psi_2]),$$

since M is reduced. It follows that

$$T_1(\phi[x_1,\psi_1])T_2(\phi[x_1,\psi_1])=T_1(\phi[x_1,\psi_2])T_2(\phi[x_1,\psi_2])$$

but

$$T_2(\phi[x_1,\psi_1])T_3(\phi[x_1,\psi_1])\neq T_2(\phi[x_1,\psi_2])T_3(\phi[x_1,\psi_2])$$

Therefore, either $\phi[x_1,\psi_1]$ or $\phi[x_1,\psi_2]$ gives a contradiction to (F0.1) for (q,p).

So, finally assume $T(\psi_{1i})\neq T(\psi_{2i})$ for $i=1,2$ but $T(\psi_{13})=T(\psi_{23})$. Define

$$\phi^{(1)}=\langle\phi_1,\phi_2,\phi_3\rangle\quad\text{and}\quad\phi^{(2)}=\langle\phi_1[x_1,x_3],\phi_3[x_1,x_3],\phi_3[x_1,x_3]\rangle.$$

By Property (F0.1) for (q,p),

$$T_1((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])T_2((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])$$
$$=T_2((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])T_3((\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j])$$

or equivalently,

$$T(\phi_1[x_1,\psi_{i1}])T(\phi_1[x_1,\psi_{j1}])T(\phi_2[x_1,\psi_{i2}])T(\phi_3[x_1,\psi_{j3}])$$
$$=T(\phi_2[x_1,\psi_{i2}])T(\phi_3[x_1,\psi_{j3}])T(\phi_3[x_1,\psi_{i3}])T(\phi_3[x_1,\psi_{j3}])$$

for every $i,j\in\{1,2\}$. By assumption, the right-hand side is independent of j, whereas the left-hand side is not. This gives a contradiction and finishes the proof of (i).

For a proof of (ii) let $T(\phi_i)=v_i y_i[u_{1i},u_{2i}]$ be the kernel decompositions of $T(\phi_i)$, $i=1,2,3$. Consider $\langle z_2,0\rangle$-computations $\psi_i=\langle\psi_{i1},\psi_{i2},\psi_{i3}\rangle$, $i=1,2$, of $\bar{A}$ with $T(\psi_{1i})\neq T(\psi_{2i})$ for $i=1,2$.

Define $\phi^{(1)}=\langle\phi_1,\phi_2,\phi_3\rangle$ and $\phi^{(2)}=\langle\phi_1[x_1,x_3],\phi_3[x_1,x_3],\phi_3[x_1,x_3]\rangle$. Applying Property (F0.1) for (q,p) to $(\phi^{(1)}\phi^{(2)})[x_1,\psi_i,\psi_j]$ we deduce that $T(\phi_2[x_1,\psi_{i2}])$ is a x_1-prefix both of $T(\phi_1[x_1,\psi_{i1}])T(\phi_1[x_1,\psi_{11}])$ and $T(\phi_1[x_1,\psi_{i1}])T(\phi_1[x_1,\psi_{21}])$. It follows that $T(\phi_2)[x_1,T(\psi_{i2})]$ is a x_1-prefix of $T(\phi_1)[v_1,T(\psi_{i1})]$. Accordingly, $T(\phi_1)[x_1,T(\psi_{i1})]$ is a x_1-prefix of $T(\phi_2)[v_3,T(\psi_{i2})]$. We conclude that $r_1,r_2\in\tilde{T}_\Delta(x_1)$ exist with

(1) $\quad T(\phi_1)[r_1,T(\psi_{i1})]=T(\phi_2)[r_2,T(\psi_{i2})]\quad$ for $i=1,2$.

Therefore, we can apply factorization and deduce that

(2) $\quad T(\phi_1)[r_1,s_1]=T(\phi_2)[r_2,s_2]\quad$ for some $s_1,s_2\in\tilde{T}_\Delta(x_1)$.

By Fact 1.8, $v_1=v_2$ and $y_1=y_2$. Since also

$$y_1[x_1,u_{12}T(\psi_{i1})]=y_2[x_1,u_{22}T(\psi_{i2})],$$

we can apply top cancellation to obtain:

(3) $\quad u_{11}T(\phi_2[x_1,\psi_{i2}])=u_{21}T(\phi_3[x_1,\psi_{i3}])\quad$ for $i=1,2$.

Applying again factorization we derive from (3),

(4) $\quad u_{11}T(\phi_2)[x_1,t_1]=u_{21}T(\phi_3)[x_1,t_2]\quad$ for some $t_1,t_2\in\tilde{T}_\Delta(x_1)$.

Hence, again by Fact 1.8, $u_{11}v_2=u_{21}v_3$. It remains to show that for every $\langle z_2,0\rangle$-computation ψ, $u_{12}T_1(\psi)=u_{22}T_2(\psi)$ and $u_{22}T_2(\psi)=u_{32}T_3(\psi)$.

For a contradiction assume a $\langle z_2,0\rangle$-computation ψ exists with $u_{12}T_1(\psi)\neq u_{22}T_2(\psi)$. (The proof for a $\langle z_1,0\rangle$-computation ψ with $u_{22}T_1(\psi)\neq u_{32}T_2(\psi)$ is analogous and therefore omitted.) By assumption, there is a node o both in $T_1(\phi)$ and $T_2(\phi)$ such that $T_1(\phi)/o=u_{12}$ and $T_2(\phi)/o=u_{22}$. It follows that

$$T_1(\phi[x_1,\psi])T_2(\phi[x_1,\psi])/o=u_{12}T_1(\psi)\neq u_{22}T_2(\psi)$$
$$=T_2(\phi[x_1,\psi])T_3(\phi[x_1,\psi])/o$$

Hence, especially, $T_1(\phi[x_1,\psi])T_2(\phi[x_1,\psi])\neq T_2(\phi[x_1,\psi])T_3(\phi[x_1,\psi])$ in contradiction to (F0.1) for (q,p). $\square$

A somewhat surprising consequence of Proposition 6.3 is that Property (F2.1) for (q,p) is already implied by Property (F0.1) for (q,p).

Theorem 6.4. *If M has Property* (F0.1) *for (q,p) then also Property* (F2.1) *for (q,p).*

Proof. Let ϕ_1 be a proper $(\langle qqp,1\rangle,\langle qpp,1\rangle\langle z,0\rangle)$-computation, ϕ_2 a proper $(\langle z,0\rangle,\langle z,0\rangle)$-computation and ϕ_3 a $\langle z,0\rangle$-computation of $\bar{A}$, where $T_1(\phi_2)\notin\{\perp,x_1\}$. Hence, especially, $\langle z,0\rangle\notin\text{Const}(\bar{A},T_1)$ and, therefore, by Proposition 6.3(i), also $\langle z,0\rangle\notin\text{Const}(\bar{A},T_2)$ and $\langle z,0\rangle\notin\text{Const}(\bar{A},T_3)$. For $i=1,2,3$, let $T_i(\phi_1)=v_iy_i[u_{i1},u_{i2}]$ be the kernel decomposition of $T_i(\phi_1)$. From Proposition 6.3(ii) we deduce that

(1) $\quad v_1=v_2;\quad y_1=y_2\quad\text{and}\quad u_{11}v_2=u_{21}v_3.$

Moreover,

(2) $\quad u_{12}T_1\equiv_z u_{22}T_2,\quad\text{and}\quad u_{22}T_2\equiv_z u_{32}T_3.$

It follows that

$$u_{12}T_1(\phi_2\phi_3)=u_{22}T_2(\phi_2\phi_3),$$

and

$$u_{22}T_2(\phi_3)=u_{32}T_3(\phi_3).$$

Hence, we conclude

$$\begin{aligned}
&T_1(\phi_1[x_1,\phi_2\phi_3])T_2(\phi_1[x_1,\phi_3])\\
&=v_1y_1[u_{11},u_{12}T_1(\phi_2\phi_3)]v_2y_2[u_{21},u_{22}T_2(\phi_3)]\\
&=v_1y_1[x_1,u_{12}T_1(\phi_2\phi_3)](u_{11}v_2)y_2[u_{21},u_{22}T_2(\phi_3)]\\
&=v_2y_2[x_1,u_{22}T_2(\phi_2\phi_3)](u_{21}v_3)y_3[u_{31},u_{32}T_3(\phi_3)]\\
&=T_2(\phi_1[x_1,\phi_2\phi_3])T_3(\phi_1[x_1,\phi_3]),
\end{aligned}$$

which we wanted to prove. □

M has Property (G1) for (q,p) iff

(G1) $\quad \text{Const}(\bar{A},T_1)=\text{Const}(\bar{A},T_2)=\text{Const}(\bar{A},T_3).$

By Proposition 3.2, the sets $\text{Const}(\bar{A},T_i)$, together with mappings $C_i=C_{\bar{A},T_i}$, $i=1,2,3$, can be computed in polynomial time. For a transition $\tau=(\langle z,1\rangle,a,\langle z_1,0\rangle\ldots\langle z_j,1\rangle\ldots\langle z_m,0\rangle)$ of $\bar{A}$ define $\bar{T}_i(\tau)=T_i(\tau)[s_1,\ldots,s_m]$, where

$$s_j=\begin{cases}C_i(z_\mu) & \text{if } z_\mu\in\text{Const}(\bar{A},T_i),\\ x_\mu & \text{otherwise.}\end{cases}$$

We extend $\bar{T}_i$ to partial $(\langle qqp, 1\rangle, \langle z, 1\rangle)$-computations π of $\bar{A}$ in the natural way. Recall that by our assumption (G0), $\langle qpp, 1\rangle \notin \text{Const}(\bar{A}, T_i)$. Therefore, the same holds for every state $\langle z, 1\rangle$ of $\bar{A}$.

The modification of T_i to $\bar{T}_i$ contains a peculiarity. Assume $\pi = \langle \pi_1, \pi_2, \pi_3\rangle$ is a $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computation of $\bar{A}$. Then clearly, $\pi^{(1)} = \langle \pi_1, \pi_1, \pi_3\rangle$ is a $(\langle qqp, 1\rangle, \langle qqp, 1\rangle)$-computation of $\bar{A}$. However, although $T_1(\pi^{(1)}) = T_1(\pi)$, neither $\bar{T}_1(\pi^{(1)})$ and $\bar{T}_1(\pi)$, nor $\bar{T}_3(\pi^{(1)})$ and $\bar{T}_3(\pi)$ are necessarily equal, since a state $\langle q_1 q_1 q_3, 0\rangle$ may not be constant in $(\bar{A}, T_i)$ even if $\langle q_1 q_2 q_3, 0\rangle$ is. Therefore, $\bar{T}_1(\pi^{(1)})$ may contain some variables which do no longer occur in $\bar{T}_1(\pi)$.

Assume the reduced FST M has Property (G1). If (τ, j) occurs on a partial $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computation of $\bar{A}$ then $\bar{T}_1(\tau)$ contains variable $x_{j'}$ with $j' \neq j$ iff $\bar{T}_2(\tau)$ contains $x_{j'}$. Especially, $\bar{T}_1(\tau) \in \tilde{T}_\Delta(x_j)$ iff $\bar{T}_2(\tau) \in \tilde{T}_\Delta(x_j)$. M has Property (G2), (G3′) or (L0) for (q, p) iff the corresponding statement holds:

(G2) Assume $\tau = (\langle z, 1\rangle, a, \langle z_1, 0\rangle \ldots \langle z_j, 1\rangle \ldots \langle z_m, 0\rangle)$ is a transition of $\bar{A}$ and $\bar{T}_i(\tau) = u_i y_i \theta_i$ is the kernel decomposition of $\bar{T}_i(\tau)$. Then
(1) $y_1 = y_2$;
(2) If $x_{j'}$ occurs in $\bar{T}_i(\tau)$ for $j' \neq j$ and $v_i = (x_{j'} \theta_i)[*]$ then $v_1 T_1 \equiv_{z_{j'}} v_2 T_2$.

(G3′) For every partial $(\langle qqp, 1\rangle, \langle z, 1\rangle)$-computation π of $\bar{A}$, where u_i is the maximal $*$-suffix of $T_i(\pi)$ in $\tilde{T}_\Delta(*)$ the following holds:

(1) Assume τ is a $\langle z, 1\rangle$-transition of $\bar{A}$ such that $T_i(\tau)$ contains occurrences of variables x_j and $x_{j'}$ with $j \neq j'$, let u_i' be the maximal $*$-prefix of $T_i(\tau)$ in $\tilde{T}_\Delta(*)$. Then

$$u_1 u_1' = u_2 u_2'.$$

(2) $u_1 \approx u_2$.

(L0) For every partial $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation π of $\bar{A}$ with $\bar{T}_i(\pi) \in \tilde{T}_\Delta(*)$,

$$|T_1(\pi)|_* = |T_2(\pi)|_*.$$

Statement (1) of Property (G3′) is rather similar to statement (1) of Property (U3′). It is simpler in that it does not speak about T_i-initial transitions. Also note that the trees u_i, u_i', v_i occurring in (G3′) can be spelled over a *finite* subalphabet $I \subseteq I_\Delta(*)$. Again, we would like to reformulate (G3′) as a graph property (G3) which can be tested in polynomial time.

Let $G_{q,p} = (V, E)$ denote the subgraph of the trace graph $G(\bar{A})$ that contains all paths from $\langle qqp, 1\rangle$ to $\langle qpp, 1\rangle$. Hence $G_{q,p}$ is the maximal subgraph of $G(\bar{A})$ containing only nodes $\langle z, 1\rangle$. Property (G3) is formulated in analogy with Property (U3). However, now the map diff no longer consists of a single pair of trees but of a *compatible set* of pairs. A set S of pairs $(s_1, s_2) \in \tilde{T}_\Delta(*)$ is called *compatible* iff $v_1, v_2 \in \tilde{T}_\Delta(*)$ exist such that for every $(s_1, s_2) \in S$,

- either $s_1 = *$ or $s_2 = *$,
- s_1 is a prefix of v_1 and s_2 is a prefix of v_2.

Observe that the cardinality of S is bounded by $|v_1|_* + |v_2|_* + 1$.

Property (G3) for (q, p) is divided into three assertions. Assertion (1) gives an initial condition for the outputs produced along paths in $G_{q,p}$. Assertion (2) describes what happens on paths with branching points, whereas Assertion (3) deals with paths without branching points.

M has Property (G3) for (q, p) iff

(G3) There is a map diff: $V \to 2^{\tilde{T}_\Delta(*)^2}$ such that
(0) Every set diff(q) is compatible;
(1) diff$(\langle qqp, 1\rangle) = \{(*, *)\}$;
(2) Assume $(s_1, s_2) \in$ diff$(\langle z, 1\rangle)$ and $G_{q,p}$ contains an edge $\langle\langle z, 1\rangle, (\tau, j), \langle z', 1\rangle\rangle$, where $\bar{T}_i(\tau)$ contains an occurrence of a variable $x_{j'}$ with $j' \neq j$. Let u_i be the maximal $*$-prefix of $\bar{T}_i(\tau)$ in $\tilde{T}_\Delta(*)$, and u_i' the maximal x_j-suffix of $\bar{T}_i(\tau)$. Then

$$s_1 u_1 = s_2 u_2 \text{ and diff}(u_1'*, u_2'*) \in \text{diff}(z');$$

(3) Assume $(s_1, s_2) \in$ diff$(\langle z, 1\rangle)$, $\langle\langle z, 1\rangle, (\tau, j), \langle z', 1\rangle\rangle \in E$, $\bar{T}_i(\tau) \in \tilde{T}_\Delta(x_j)$ and $u_i = \bar{T}_i(\tau)[*]$. Then

$$\text{diff}(s_1 u_1, s_2 u_2) \in \text{diff}(\langle z', 1\rangle).$$

The following example shows that the sets diff(v) in fact can contain more than one element. Let $M = (A, T)$ be the (reduced) FST, where $A = (\{p, q\}, \Sigma, \delta, q)$, with $\Sigma_0 = \Delta_0 = \{\#\}$; $\Sigma_1 = \{c, d\}$ and $\Delta_1 = \{a, b\}$; δ consists of the transitions:

$$\tau_0 = (p, \#, \varepsilon) \text{ with } T(\tau_0) = \#;$$

$$\tau_1 = (p, c, p) \text{ with } T(\tau_1) = ababx_1;$$

$$\tau_2 = (p, d, p) \text{ with } T(\tau_2) = ababx_1;$$

$$\tau_3 = (q, c, p) \text{ with } T(\tau_3) = babx_1;$$

$$\tau_4 = (q, d, p) \text{ with } T(\tau_4) = bababx_1;$$

$$\tau_5 = (q, c, q) \text{ with } T(\tau_5) = babax_1;$$

$$\tau_6 = (q, d, q) \text{ with } T(\tau_6) = babax_1.$$

Then M has Property (G3′) but diff$(\langle qqp, 1\rangle) = \{(*, *)\}$ and diff$(\langle qpp, 1\rangle) = \{(a, *), (*, b)\}$.

The above Property (G3) can be tested for (q, p) in polynomial time only if we succeed to give polynomial upper bounds to the maximal $*$-lengths of trees occurring in diff(z). It turns out that such bounds can be derived from Property (L1) for (q, p) below.

The next proposition relates Properties (G3′) and (G3) for (q, p) to comparability of outputs of partial computations.

Proposition 6.5. *Assume the reduced FST M has Properties* (G1) *and* (G2) *for* (q,p). *Then the following three statements are equivalent*:

(1) *For every partial* $(\langle qqp,1\rangle,\langle z,1\rangle)$*-computation of* $\bar{A}$, *the following holds: if* $\bar{T}_i(\pi)\in\tilde{T}_\Delta(*)$ *then* $\bar{T}_1(\pi)\approx\bar{T}_2(\pi)$; *if* $\bar{T}_i(\pi)\notin\tilde{T}_\Delta(*)$ *and* $\bar{T}_i(\pi)=v_iy_i\theta_i$ *is the kernel decomposition of* $\bar{T}_i(\pi)$ *then*

- $v_1=v_2, y_1=y_2, *\theta_1\approx*\theta_2$, *and*
- *for every variable* $x'\neq*$ *occurring in* $\bar{T}_i(\pi)$ *with corresponding state* z', $(x'\theta_1)T_1\equiv_{z'}(x'\theta_2)T_2$.

(2) *M has Property* (G3′) *for* (q,p).

(3) *M has* (G3) *for* (q,p).

Observe that by Proposition 6.3, statement (1) holds true whenever M has Property (F0.1) for (q,p).

Proof of Proposition 6.5. (1) *implies* (2): Assume statement (1) holds. Let π be a partial $(\langle qqp,1\rangle,\langle z,1\rangle)$-computation of length k, and $\bar{T}_i(\pi)=s_iu_i$, where u_i is the maximal $*$-suffix in $\tilde{T}_\Delta(*)$, $i=1,2$. Then especially, $u_1\approx u_2$ in accordance with assertion (2) of (G3′). To deduce also assertion (1) of Property (G3′), assume $\tau=(\langle z,1\rangle,a,\langle z_1,0\rangle\ldots\langle z_j,1\rangle\ldots\langle z_m,0\rangle)$ is a transition of $\bar{A}$, where $\bar{T}_i(\tau)$ contains occurrences of some variable $x_{j'}$ for $j'\neq j$. Let $\bar{T}_i(\tau)[k+1,j]=v_i's_i'$, where v_i' is the maximal prefix in $\tilde{T}_\Delta(*)$. First, assume $\bar{T}_i(\pi)=u_i$. Then applying (1) to the partial computation $\pi(\tau,j)$ yields $u_1v_1'=u_2v_2'$ in accordance with Property (G3′). So, assume $s_i\neq*$. Let $s_i=v_iy_i\theta_i$ be the kernel decomposition of s_i, and $v_i'y_i'\theta_i'$ be the kernel decomposition of $\bar{T}_i(\tau)[k+1,j]$. By Property (G2) for (q,p), $y_1'=y_2'$. Moreover, by (1),

$$y_1=y_2\quad\text{and}\quad y_1u_1v_1'y_1'=y_2u_2v_2'y_2'.$$

Therefore, by top and bottom cancellation, $u_1v_1'=u_2v_2'$, which we wanted to prove.

(2) *implies* (1): Let π be a partial $(\langle qqp,1\rangle,\langle z,1\rangle)$-computation of length k. We proceed by induction on the length of π.

If $\pi=\varepsilon$, statement (1) trivially holds. Therefore, assume $\pi=\pi'(\tau,j)$, where π' has length $k-1\geqslant0$. Then $\bar{T}_i(\pi)=\bar{T}_i(\pi')\bar{T}_i(\tau)[k,j]$. Assume $\bar{T}_i(\pi')=s_iu_i$, where u_i is the maximal suffix in $\tilde{T}_\Delta(*)$.

First assume $s_i=*$. If $\bar{T}_i(\tau)\in\tilde{T}_\Delta(x_j)$ then statement (1) for π is immediately implied by assertion (2) of Property (G3′). If $\bar{T}_i(\tau)$ contains at least one occurrence of some variable $x_{j'}$ with $j'\neq j$ then let $\bar{T}_i(\tau)[k,j]=v_iy_i\theta_i$ be the kernel decomposition of $\bar{T}_i(\tau)[k,j]$. Hence, the kernel decomposition of $\bar{T}_i(\pi)$ is $\bar{T}_i(\pi)=(u_iv_i)y_i\theta_i$. By Property (G3′) for (q,p), $u_1v_1=u_2v_2$, whereas the remaining identities are implied by (G2) instantiated with τ.

Now assume $s_i\neq*$. Let $s_i=v_iy_i\theta_i$ be the kernel decomposition of s_i. We distinguish two cases.

If $\bar{T}_i(\tau)\in\tilde{T}_\Delta(x_j)$ then $\bar{T}_i(\pi)=s_i\bar{u}_i$, where $\bar{u}_i=u_i\bar{T}_i(\tau)[k,j]$. By Property (G3′) for (q,p), $\bar{u}_1\approx\bar{u}_2$. The remaining assertions of statement (1) follow from the inductive assumption.

If $\bar{T}_i(\tau)\notin\tilde{T}_\Delta(x_j)$ then $\bar{T}_i(\tau)$ contains some variable $x_{j''}$ with $j\neq j''$. Assume $\bar{T}_i(\tau)[k,j]=v_i'y_i'\theta_i'$ is the kernel decomposition of $\bar{T}_i(\tau)[k,j]$. Then, the kernel decomposition of $\bar{T}_i(\pi)$ is $\bar{T}_i(\pi)=v_i\bar{y}_i\bar{\theta}_i$, where $\bar{y}_i=y_iu_iv_i'y_i'$ and $\bar{\theta}_i$ is defined by

$$x\bar{\theta}_i=\begin{cases} x\theta_i' & \text{if } x=* \quad\text{or}\quad x=x_{k,j'}, \\ x\theta_i & x=x_{\kappa,j'} \quad\text{with}\quad \kappa<k. \end{cases}$$

By induction hypothesis we have $v_1=v_2$, $y_1=y_2$ and $(x_{\kappa,j'}\bar{\theta}_1)T_1\equiv_{z'}(x_{\kappa,j'}\bar{\theta}_2)T_2$ whenever $x_{\kappa,j'}$ occurs in $\bar{y}_i$ with $\kappa<k$ and $\langle z',0\rangle$ is the state of $\bar{A}$ corresponding to $x_{\kappa,j'}$. By Property (G3′) for (q,p) applied to π' and τ we have $u_1v_1'=u_2v_2'$. Moreover, by Property (G2) for (q,p), $y_1'=y_2'$. Hence, we deduce that also $\bar{y}_1=\bar{y}_2$. Finally, Property (G2) for (q,p) also implies that $(x_{k,j'}\bar{\theta}_1)T_1\equiv_{z'}(x_{k,j'}\bar{\theta}_2)T_2$ whenever $x_{k,j'}$ occurs in $\bar{y}_i$, $\langle z',0\rangle$ corresponds to $x_{k,j'}$ in $\pi(\tau,j)$, and also $*\bar{\theta}_1\approx*\bar{\theta}_2$. Hence, all assertions of statement (1) hold. Therefore, (2) also implies (1). Since there is a one-to-one correspondence between the paths in $G_{q,p}$ from $\langle qqp,1\rangle$ to $\langle qpp,1\rangle$ with the partial $(\langle qqp,1\rangle,\langle qpp,1\rangle)$-computations of $\bar{A}$, Property (G3) for (q,p) implies Property (G3′) for (q,p). The harder part is the reverse direction.

(2) *implies* (3): Assume z is a node of $G_{q,p}$. From the definition we deduce that for every partial $(\langle qqp,1\rangle,\langle z,1\rangle)$-computation π, diff(z) contains diff(u_1,u_2), where u_i is the maximal $*$-suffix of $\bar{T}_i(\pi)$ in $\tilde{T}_\Delta(*)$. Thus, it remains to show:

(+) Assume π and π' are partial $(\langle qqp,1\rangle,\langle z,1\rangle)$-computations of $\bar{A}$ and u_i and u_i' are the maximal $*$-suffixes of $\bar{T}_i(\pi)$ and $\bar{T}_i(\pi')$, respectively. Then the set $S=\{(s_1,s_2),(s_1',s_2')\}$ is compatible, where $(s_1,s_2)=\text{diff}(u_1,u_2)$ and $(s_1',s_2')=\text{diff}(u_1',u_2')$.

To prove this, we distinguish two cases.

Case I: A partial $(\langle z,1\rangle,\langle z',1\rangle)$-computation π'' and a $\langle z',1\rangle$-transition $\tau=(\langle z',1\rangle,a,\langle z_1,0\rangle\ldots\langle z_j,1\rangle\ldots\langle z_m,0\rangle)$ of $\bar{A}$ exist such that $\bar{T}_i(\pi'')\in\tilde{T}_\Delta(*)$, and $\bar{T}_i(\tau)$ contains an occurrence of some variable $x_{j'}$ for $j\neq j'$.

Let v_i denote the maximal $*$-prefix of $\bar{T}_i(\tau)$ in $\tilde{T}_\Delta(*)$, and define $r_i=\bar{T}_i(\pi'')v_i$. Applying assertion (1) of Property (G3′) to $\pi\pi''$ and $\pi'\pi''$ we find:

$$u_1r_1=u_2r_2 \quad\text{and}\quad u_1'r_1=u_2'r_2.$$

Hence, by top cancellation also

$$s_1r_1=s_2r_2 \quad\text{and}\quad s_1'r_1=s_2'r_2.$$

Therefore, either $s_1=*$ or $s_2=*$. Moreover, similar to the proof of Proposition 5.1 we find that $(s_1',s_2')=(s_1,s_2)$. Hence, set S is compatible.

Case II: The condition of Case I does not hold. Then, a partial $(\langle z,1\rangle,\langle qpp,1\rangle)$-computation π'' exists with $\bar{T}_i(\pi'')\in\tilde{T}_\Delta(*)$. By assertion (2) of Property (G3′),

$$u_1\bar{T}_1(\pi'')\approx u_2\bar{T}_2(\pi'') \quad\text{and}\quad u_1'\bar{T}_1(\pi'')\approx u_2'\bar{T}_2(\pi'')$$

and, hence, by top cancellation,

$$s_1 \bar{T}_1(\pi'') \approx s_2 \bar{T}_2(\pi'') \quad \text{and} \quad s_1' \bar{T}_1(\pi'') \approx s_2' \bar{T}_2(\pi'').$$

Again, either $s_1 = *$ or $s_2 = *$ and, likewise, $s_1' = *$ or $s_2' = *$. If $s_1 = *$ and $s_2' = *$, S trivially is compatible. The same holds when $s_2 = *$ and $s_1' = *$.

By assumption (G0) for (q, p), some $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation $\bar{\pi} = \langle \pi_1, \pi_3, \pi_3 \rangle$ exists such that $\bar{T}_1(\bar{\pi}) \neq * \neq \bar{T}_2(\bar{\pi})$. Since the condition of Case I does not hold, $\bar{T}_i(\bar{\pi}) \in \bar{T}_\Delta(*)$. For $k > 0$, consider $\pi^{(k)} = \pi'' \bar{\pi}^{k-1}$. First, assume $s_1 = *$ and $s_1' = *$. Then some k exists such that both s_2 and s_2' are prefixes of $\bar{T}_1(\pi^{(k)})$. Hence, S is compatible. Analogously, if $s_2 = *$ and $s_2' = *$ then some k exists such that both s_1 and s_1' are prefixes of $\bar{T}_2(\pi^{(k)})$, which proves S compatible. □

The key observation is that Properties (G1)–(G3) together with (L0) for (q, p) give an equivalent characterization of (F0.1) for (q, p). We have:

Theorem 6.6. *Assume $M = (A, T)$ is a reduced FST and condition* (G0) *holds for (q, p). Then, the following two statements are equivalent*:

(1) *M has Property* (F0.1) *for (q, p)*;

(2) *M has Properties* (G1)–(G3) *and* (L0) *for (q, p)*.

Proof. (1) *implies* (2): Assume M has Property (F0.1) for (q, p). Then by Proposition 6.3, M also has Properties (G1), (G2) and (G3') for (q, p). To prove that M also has Property (L0) consider a partial $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation $\pi = \langle \pi_1, \pi_2, \pi_3 \rangle$ of $\bar{A}$ such that $\bar{T}_i(\pi) \in \bar{T}_\Delta(*)$. Define $\tilde{\pi} = \langle \pi_1, \pi_3, \pi_2 \rangle$ which is a partial $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation of $\bar{A}$ as well. Since $\bar{A}$ is reduced, some $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation ϕ exists such that $\phi = \tilde{\pi}[\ldots, \psi_{\kappa,j}, \ldots]$ for some $\psi_{\kappa,j}$ and $T_i(\phi) = \bar{T}_i(\tilde{\pi})$. Moreover, some $(\langle qqp, 1\rangle, \langle qpp\rangle)$-computation ϕ' of $\bar{A}$ exists such that

$$T_1(\phi') T_2(\phi') = T_2(\phi') T_3(\phi') \quad \text{and} \quad T_1(\phi'\phi) T_2(\phi'\phi) = T_2(\phi'\phi) T_3(\phi'\phi).$$

The first equation yields $|T_1(\phi')|_{x_1} = |T_3(\phi')|_{x_1}$. From this and the second equation above we obtain

$$|\bar{T}_1(\pi)|_* = |T_1(\phi)|_{x_1} = |T_3(\phi)|_{x_1} = |\bar{T}_2(\pi)|_*,$$

which we wanted to prove.

(2) *implies* (1): Assume M has Properties (G1)–(G3) and (L0) for (q, p). Consider a $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computation ϕ of $\bar{A}$. Then, $\phi = \pi[\ldots, \psi_{\kappa,j}, \ldots]$ for some partial $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computation $\pi = \langle \pi_1, \pi_2, \pi_3 \rangle$ and $\langle z_{\kappa,j}, 0\rangle$-computations $\psi_{\kappa,j} = \langle \psi_{\kappa,j,1}, \psi_{\kappa,j,2}, \psi_{\kappa,j,3} \rangle$. Define

$$\pi^{(1)} = \langle \pi_1, \pi_1, \pi_3 \rangle; \qquad \pi^{(2)} = \langle \pi_1, \pi_3, \pi_3 \rangle, \quad \text{and}$$

$$\psi_{\kappa,j}^{(1)} = \langle \psi_{\kappa,j,1}, \psi_{\kappa,j,1}, \psi_{\kappa,j,3} \rangle; \qquad \psi_{\kappa,j}^{(2)} = \langle \psi_{\kappa,j,1}, \psi_{\kappa,j,3}, \psi_{\kappa,j,3} \rangle.$$

First, assume $\bar{T}_i(\pi^{(2)}) \in \tilde{T}_\Delta(*)$. Then also $\bar{T}_i(\pi^{(1)}) \in \tilde{T}_\Delta(*)$ and $\bar{T}_i(\pi) \in \tilde{T}_\Delta(*)$. Hence, by Property (L0) applied to $\pi^{(2)}$,

$$|T_1(\phi)|_{x_1} = |\bar{T}_1(\pi^{(2)})|_* = |\bar{T}_2(\pi^{(2)})|_* = |T_3(\phi)|_{x_1}.$$

Therefore, especially,

$$(+) \qquad |T_1(\phi)T_2(\phi)|_{x_1} = |T_2(\phi)T_3(\phi)|_{x_1}.$$

By Property (G3′) for (q,p), some $k>1$ exists such that $T_1(\phi)T_2(\phi) = T_2(\pi^{(1)}\pi)$ is a prefix of $T_1(\phi)^k = T_1(\pi^{(1)}\pi(\pi^{(2)})^{k-2})$ and, likewise, $T_2(\phi)T_3(\phi) = T_2(\pi\pi^{(2)})$ is a prefix of $T_1(\phi)^k = T_1(\pi(\pi^{(2)})^{k-1})$. Therefore, by $(+)$, $T_1(\phi)T_2(\phi) = T_2(\phi)T_3(\phi)$, in accordance with Property (F0.1) for (q,p).

Now, assume $\bar{T}_i(\pi^{(2)}) \notin \tilde{T}_\Delta(*)$. Let $\bar{T}_i(\pi^{(2)}) = \bar{s}_i\bar{u}_i$, where $\bar{u}_i$ is the maximal $*$-suffix of $\bar{T}_i(\pi^{(2)})$ in $\tilde{T}_\Delta(*)$, and $\bar{s}_i = \bar{v}_i\bar{y}_i\bar{\theta}_i$ be the kernel decomposition of $\bar{s}_i$.

Case I: $\bar{T}_i(\pi) \in \tilde{T}_\Delta(*)$. Consider the partial $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computations $\pi\pi^{(2)}$ and $\pi\pi^{(2)}\pi^{(2)}$. Proposition 6.5 for the first one gives

$$\bar{T}_1(\pi)\bar{v}_1 = \bar{T}_2(\pi)\bar{v}_2 \quad \text{and} \quad \bar{y}_1 = \bar{y}_2.$$

Therefore, Proposition 6.5 for the second one, together with top cancellation, gives:

$$\bar{u}_1\bar{v}_1 = \bar{u}_2\bar{v}_2.$$

It follows that some $t \in \tilde{T}_\Delta(*)$ exists with

$$\bar{T}_1(\pi)t = \bar{T}_2(\pi);\ \bar{v}_1 = t\bar{v}_2 \quad \text{and hence} \quad \bar{u}_1 t = \bar{u}_2,$$

or

$$\bar{T}_1(\pi) = \bar{T}_2(\pi)t;\ t\bar{v}_1 = \bar{v}_2 \quad \text{and hence} \quad \bar{u}_1 = \bar{u}_2 t.$$

For convenience, assume the former to be the case. The proof for the second possibility is analogous. Since $\bar{T}_i(\pi) = T_i(\phi)*$, we have

$$T_1(\phi)T_1(\phi)t = T_1(\phi)T_2(\phi)*.$$

Moreover, Proposition 6.5 applied to $\pi\pi^{(2)}$ yields

$$T_1(\phi)T_1(\phi)t = T_1(\pi)\bar{v}_1(\bar{y}_1\theta_{\psi,1})(\bar{u}_1 t) = T_2(\pi)\bar{v}_2(\bar{y}_2\theta_{\psi,2})\bar{u}_2 = T_2(\phi)T_3(\phi)*,$$

where $\theta_{\psi,i}$ is the substitution which inserts $T_i(\psi_{\kappa,j'})$ into variable $x_{\kappa,j'}$. Hence, we conclude

$$T_1(\phi)T_2(\phi)* = T_1(\phi)T_1(\phi)t = T_2(\phi)T_3(\phi)*,$$

which we wanted to prove.

Case II: $\bar{T}_i(\pi) \notin \tilde{T}_\Delta(*)$. Let $\bar{T}_i(\pi) = v_i y_i \theta_i$ be the kernel decomposition of $\bar{T}_i(\pi)$. Consider the partial $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computations π, $\pi\pi^{(2)}$ and $\pi\pi^{(2)}\pi^{(2)}$. Proposition 6.5 for the first, the second and the third partial computation, respectively, yields

- $v_1 = v_2$ and $y_1 = y_2$,
- $u_1\bar{v}_1 = u_2\bar{v}_2$ and $\bar{y}_1 = \bar{y}_2$,
- $\bar{u}_1\bar{v}_1 = \bar{u}_2\bar{v}_2$.

Again, some $t \in \tilde{T}_\Delta(*)$ exists such that

$$u_1 t = u_2 \quad \text{and} \quad \bar{u}_1 t = \bar{u}_2;$$

or

$$u_2 t = u_1 \quad \text{and} \quad \bar{u}_2 t = \bar{u}_1.$$

For example, assume the latter to be the case. The proof for the first possibility is analogous. Let $\theta_{\psi,i}$ denote the substitution which inserts $T_i(\psi_{\kappa,j'})$ into variable $x_{\kappa,j'}$. Applying Proposition 6.5 to π we find:

$$T_1(\phi)* = v_1(y_1\theta_{\psi,1})u_1 = v_2(y_2\theta_{\psi,2})(u_2 t) = T_2(\phi)t.$$

Hence,

(1) $\quad T_1(\phi)T_1(\phi)* = T_1(\phi)T_2(\phi)t.$

Moreover, Proposition 6.5 applied to $\pi\pi^{(2)}$ yields

(2) $$T_1(\phi)T_1(\phi)* = v_1(y_1\theta_{\psi,1})(u_1\bar{v}_1)(\bar{y}_1\theta_{\psi,1})\bar{u}_1 = v_2(y_2\theta_{\psi,2})(u_2\bar{v}_2)(\bar{y}_2\theta_{\psi,2})(\bar{u}_2 t)$$
$$= T_2(\phi)T_3(\phi)t$$

Hence, (1) and (2) together yield

$$T_1(\phi)T_2(\phi)t = T_1(\phi)T_1(\phi)* = T_2(\phi)T_3(\phi)t,$$

from which we derive the conclusion according to (F0.1) for (q,p) by bottom cancellation. □

Assume M has Properties (G1)–(G3) and (L0) for (q,p). We give two equivalent characterizations (G4) and (L1) for Property (F1.1) by means of paths. As (L0), Property (L1) is a length property. M has Property (G4) or Property (L1) for (q,p) iff

(G4) For every partial $(\langle qqp, 1\rangle, \langle z, 1\rangle)$-computation π_1, every partial $(\langle z, 1\rangle, \langle z, 1\rangle)$-computation π_2 and every partial $(\langle z, 1\rangle, \langle qpp, 1\rangle)$-computation p_3 of $\bar{A}$,

(1) If $\bar{T}_i(\pi_1\pi_2\pi_3) \in \tilde{T}_\Delta(*)$ then

$$\bar{T}_1(\pi_1\pi_2\pi_3)\bar{T}_2(\pi_1\pi_2) = \bar{T}_2(\pi_1\pi_2\pi_3)\bar{T}_3(\pi_1\pi_2).$$

(2) If $\bar{T}_i(\pi_1)$, $\bar{T}_i(\pi_3) \in \tilde{T}_\Delta(*)$ but $\bar{T}_i(\pi_2) \notin \tilde{\tilde{T}}_\Delta(*)$ then

$$u_1\bar{T}_1(\pi_3)\bar{T}_2(\pi_1\pi_3) = u_2\bar{T}_2(\pi_3)\bar{T}_3(\pi_1\pi_3),$$

where u_i is the maximal $*$-suffix of $\bar{T}_i(\pi_2)$.

(L1) $\quad |\bar{T}_1(\pi_2)|_* = |\bar{T}_2(\pi_2)|_*.$

(L1) is the generalization of a corresponding criterion for finite-valuedness of [16]. Property (L1) for (q,p) implies Property (L0) for (q,p). To see this, consider a partial $(\langle qpp, 1\rangle, \langle qpp, 1\rangle)$-computation π. By assumption (G0) for (q,p), a partial $(\langle qqp, 1\rangle, \langle qpp, 1\rangle)$-computation π' of $\bar{A}$ exists. Then, apply the conclusion of (L1) for (q,p) to $\pi_1 = \pi'$, $\pi_2 = \pi$ and $\pi_3 = *$.

Theorem 6.7. *Assume $M=(A,T)$ is a reduced FST, and condition* (G0) *holds for* (q,p). *Then, the following three statements are equivalent:*

(1) *M has Property* (F1.1) for (q,p);

(2) *M has Properties* (G1)–(G4) *for* (q,p);

(3) *M has Properties* (G1)–(G3) *and* (L1) *for* (q,p).

Proof. (1) *implies* (2): Assume M has Property (F1.1) for (q,p). Then M also has Property (F0.1) for (q,p). Hence by Theorem 6.6, M has Properties (G1)–(G3) and (L0). Let π_1,π_2,π_3 as in the assumption of Property (G4). If for $j=1,2,3$, $\bar{T}_i(\pi_j)\in\tilde{T}_\Delta(*)$ then for (q,p), the conclusion of (G4) trivially follows from the conclusion of (F1.1). So, let $\bar{T}_i(\pi_1)$, $\bar{T}_i(\pi_3)\in\tilde{T}_\Delta(*)$ but $\bar{T}_i(\pi_2)\notin\tilde{T}_\Delta(*)$. Let $\bar{T}_i(\pi_2)=v_iy_iu_i$, where v_i is the maximal $*$-prefix and u_i is the maximal $*$-suffix in $\tilde{T}_\Delta(*)$. Assume ϕ_i are x_1-proper computations with $\phi_i=\pi_i[\ldots,\psi_{\kappa,j}^{(i)},\ldots]$. Let θ_i denote the substitution with $x_{\kappa,j}\theta_i=T_i(\psi_{\kappa,j}^{(2)})$. From Proposition 6.5 we deduce that $T_i(\phi_2)=v_i(y_i\theta_i)u_i$, where

$$T_1(\phi_1)v_1(y\theta_1)=T_2(\phi_1)v_2(y\theta_2).$$

Therefore, applying top cancellation to the conclusion of (F1.1) for ϕ_i yields

$$u_1\bar{T}_1(\pi_3)\bar{T}_2(\pi_1\pi_3)=u_2\bar{T}_2(\pi_3)\bar{T}_3(\pi_1\pi_3),$$

which we wanted to prove.

(2) *implies* (3): The proof is a case distinction on whether or not $\bar{T}_i(\pi_j)$ are in $\tilde{T}_\Delta(*)$. In most of these cases equality of the $*$-lengths of the outputs for the partial computations in question already follow from (G1)–(G3) for (q,p) by means of Proposition 6.5. Only for two remaining cases Property (G4) is needed explicitly. Again, the case where $\bar{T}_i(\pi_j)\in\tilde{T}_\Delta(*)$ for $j=1,2,3$ is trivial. So, assuming $\bar{T}_i(\pi_1)$, $\bar{T}_i(\pi_3)\in\tilde{T}_\Delta(*)$ but $\bar{T}_i(\pi_2)\notin\tilde{T}_\Delta(*)$, we argue as follows. Assume $\bar{T}_i(\pi_2)=v_iy_iu_i$, where v_i is the maximal $*$-prefix and u_i is the maximal $*$-suffix in $\tilde{T}_\Delta(*)$. Then by Proposition 6.5,

$$|\bar{T}_1(\pi_1)v_1|_*=|\bar{T}_2(\pi_1)v_2|_* \quad\text{and}\quad |y_1|_*=|y_2|_*.$$

From Property (G4) we know

$$|u_1\bar{T}_1(\pi_3)\bar{T}_2(\pi_1\pi_3)|_*=|u_2\bar{T}_2(\pi_3)\bar{T}_3(\pi_1\pi_3)|_*.$$

Adding up all three equations and subtracting

$$|\bar{T}_1(\pi_1\pi_3)\bar{T}_2(\pi_1\pi_3)|_*=|\bar{T}_2(\pi_1\pi_3)\bar{T}_3(\pi_1\pi_3)|_*$$

we obtain

$$|v_1y_1u_1|_*=|v_2y_2u_2|_*,$$

which we wanted to prove.

(3) *implies* (1): Assume M has Properties (G1)–(G3) and (L1) for (q,p). Then M also has Property (L0) for (q,p) and, hence, by Theorem 6.6, also Property (F0.1) for (q,p). Let ϕ_1,ϕ_2,ϕ_3 be three proper computations as in the assumption of Property (F1.1)

for (q,p). For $\phi_\nu=\langle\phi_{\nu 1},\phi_{\nu 2},\phi_{\nu 3}\rangle$, $\nu=1,2,3$, define $\phi_\nu^{(1)}=\langle\phi_{\nu 1},\phi_{\nu 1},\phi_{\nu 3}\rangle$ and $\phi_\nu^{(2)}=\langle\phi_{\nu 1},\phi_{\nu 3},\phi_{\nu 3}\rangle$. By assumption (G0) for (q,p), some $(\langle q,p\rangle,\langle q,p\rangle)$-computation $\langle\phi_1'\phi_2'\rangle$ of $A^{(2)}$ exists with $T(\phi_i')\neq *$ for $i=1,2$. Define $\phi'=\langle\phi_1',\phi_2',\phi_2'\rangle$. Then for every k, $\phi^{(k)}=\phi_1\phi_2\phi_3\phi_1^{(2)}\phi_3^{(2)}\phi'^k$ and $\bar\phi^{(k)}=\phi_1^{(1)}\phi_2^{(1)}\phi_3^{(1)}\phi_1\phi_3\phi'^k$ are proper $(\langle qqp,1\rangle,\langle qpp,1\rangle)$-computations of $\bar A$ with

$$T_1(\phi^{(k)})=T_1(\bar\phi^{(k)})=T_1(\phi_1\phi_2\phi_3)T_1(\phi_1\phi_3)T(\phi_1')^k,$$

$$T_2(\phi^{(k)})=T_2(\phi_1\phi_2\phi_2)T_3(\phi_1\phi_3)T(\phi_2')^k,$$

$$T_2(\bar\phi^{(k)})=T_1(\phi_1\phi_2\phi_2)T_2(\phi_1\phi_3)T(\phi_2')^k.$$

Since M has Property (F0.1) for (q,p), some k exists such that both $s_1=T_2(\phi_1\phi_2\phi_3)T_3(\phi_1\phi_2)$ and $s_2=T_1(\phi_1\phi_2\phi_3)T_2(\phi_1\phi_2)$ are prefixes of $T_1(\phi_1\phi_2\phi_3)T_1(\phi_1\phi_3)T(\phi_1')^k$. Therefore, $s_1=s_2$ provided the x_1-lengths of s_1 and s_2 agree. By Property (F0.1) for (q,p),

$$|T_1(\phi_1\phi_3)T_2(\phi_1\phi_3)|x_1=|T_2(\phi_1\phi_3)T_3(\phi_1\phi_3)|x_1.$$

Hence, it remains to prove that

$$(+)\qquad |T_1(\phi_2)|x_1=|T_2(\phi_2)|x_1.$$

Let $\phi_2=\pi_2[\ldots,\psi_{\kappa,j},\ldots]$ for a partial computation π_2 and define substitutions θ_i which substitute variables $x_{\kappa,j}$ with $T_i(\psi_{\kappa,j})$. By Property (L1) for (q,p), $|\bar T_1(\pi_2)|_*=|\bar T_2(\pi_2)|_*$. By Proposition 6.5, $\bar T_1(\pi_2)\in\tilde T_\Delta(*)$ iff $\bar T_2(\pi_2)\in\tilde T_\Delta(*)$.

If $\bar T_1(\pi_2)\in\tilde T_\Delta(*)$ then statement (+) trivially follows since $T_i(\phi_2)=\bar T_i(\pi_2)x_1$ for $i=1,2$. If $\bar T_1(\pi_2)\notin\tilde T_\Delta(*)$ then $\bar T_i(\pi_2)=v_iy_iu_i$, where v_i is the maximal $*$-prefix and u_i is the maximal $*$-suffix in $\tilde T_\Delta(*)$. We have: $T_i(\phi_2)=v_i(y_i\theta_i)u_ix_1$. By Proposition 6.5, $|y_1|_*=|y_2|_*$ and $y_1\theta_1=y_2\theta_2$. The first equation, together with (L1) applied to π_2, gives

$$|v_1|_*+|u_1|_*=|v_2|_*+|u_2|_*.$$

Therefore, we can conclude that

$$|T_1(\phi_2)|x_1=|v_1|_*+|(y_1\theta_1)|_*+|u_2|_*=|v_2|_*+|(y_2\theta_2)|_*+|u_2|_*=|T_2(\phi_2)|x_1,$$

which we wanted to prove. □

Before we state the main theorem of this section we present an estimation of the sets diff(v) in Property (G3) for (q,p), provided the given FST M has Property (L1).

Proposition 6.8. *Assume the reduced FST $M=(A,T)$ has Properties* (G1), (G2) *and* (L1) *for (q,p), and let $G_{q,p}=(V,E)$.*

(i) *Then M has Property* (G3) *for (q,p) iff a map* diff: $V\to 2^{\tilde T_\Delta(*)}$ *satisfying conditions* (0), (1), (2) *and* (3) *of Property* (G3) *exists such that for every $v\in V$, $\#\mathrm{diff}(v)\leqslant 2\cdot|M|^3+1$.*

(ii) *It can be decided in polynomial time whether or not M has Property* (G3) *for (q,p).*

Proof. First, we prove statement (i). Assume M has Property (G3) for (q,p) and, hence, by Proposition 6.5, also Property (G3′). Consider $\langle z,1\rangle \in V$.

Case I: A node $\langle z',1\rangle$ is reachable from $\langle z,1\rangle$ for which a $\langle z',1\rangle$-transition in $\bar{A}$ exists such that $\bar{T}_i(\tau)$ contains occurrences of at least two distinct variables. Then assertion (2) of Property (G3) implies that $\#\mathrm{diff}(\langle z,1\rangle)=1$.

Case II: The condition of Case I does not hold. Let $(s_1,s_2)\in\mathrm{diff}(\langle z,1\rangle)$, and assume π is a partial $(\langle z,1\rangle,\langle z,1\rangle)$-computation of $\bar{A}$. Then, $u_i=\bar{T}_i(\pi)\in\tilde{T}_\Delta(*)$. We prove

$$(+)\qquad \mathrm{diff}(s_1u_1,s_2u_2)=(s_1,s_2).$$

The observation of Case I and statement $(+)$ together imply the upper bound given in (i). To show $(+)$, let $\mathrm{diff}(s_1u_1,s_2u_2)=(s_1',s_2')$. Then,

$$s_1u_1s_2'=s_2u_2s_1'.$$

Since $(s_1,s_2)\in\mathrm{diff}(\langle z,1\rangle)$, either $s_1=*$ or $s_2=*$. By Property (L1) for (q,p), $|u_1|_*=|u_2|_*$. Therefore, $s_1=*$ implies $s_1'=*$ and $|s_2|_*=|s_2'|_*$. Accordingly, $s_2=*$ implies $s_2'=*$ and $|s_1|_*=|s_1'|_*$. Since $\mathrm{diff}(\langle z,1\rangle)$ is compatible, we conclude that $(s_1,s_2)=(s_1',s_2')$.

(ii) Now, we can compute a map diff as follows:

(1) compute a maximal acyclic subgraph $G'=(V,E')$ of $G_{q,p}$;

(2) compute a map diff with Properties (0), (1), (2) and (3) of (G3);

(3) verify that for the given map diff(_), Properties (2) and (3) also hold for the edges in $E\setminus E'$.

The map diff constructed in steps (1) and (2) can be computed in polynomial time. Considering the two cases in the proof of (i) we find that if M has Property (G3) then this map also passes the test of (3). This finishes the proof of statement (ii). □

The following theorem collects the properties which together give an alternative characterization of finite-valuedness.

Theorem 6.9. *Assume M is a reduced FST. Then the following two statements are equivalent*:

(1) *M is finite-valued*;

(2) *M has Properties* (F1.0), (F2.0) *for all pairs of states and, whenever condition* (G0) *holds for (q,p), also Properties* (G1)–(G3) *and* (L1) *for (q,p).*

It is decidable in deterministic polynomial time whether or not M is finite-valued.

Proof. (1) *implies* (2): Assume M is finite-valued. Then by Theorem 6.1, M has Properties (F0), (F1) and (F2) for all (q,p). Therefore, by Proposition 6.2, M has Properties (F*i*.0) for $i=1,2$ and Properties (F0.1) and (F1.1) for all pairs (q,p). Assume (G0) holds for (q,p). If M has Property (F0.1) for (q,p) then by Proposition 6.3, also Properties (G1)–(G3) for (q,p). If M has Property (F1.1) for (q,p) then, by Theorem 6.7, also Property (L1) for (q,p). Hence, statement (2) follows.

(2) *implies* (1): Assume M has Properties (F1.0), (F2.0) for all pairs of states and Properties (G1)–(G3) and (L1) for every pair of states (q,p) where (G0) holds. By Theorem 6.7, M also has Property (F1.1) and hence also (F0.1) for (q,p). Then by Theorem 6.4, M has Property (F2.1) for (q,p). Therefore, M has Properties (F1.0), (F1.1), (F2.0) and (F2.1) for all (q,p). Hence statement (1) follows from Theorem 6.1. By Proposition 6.8, Property (G3) for (q,p) can be decided in deterministic polynomial time. Since the remaining Properties in statement (2) can be decided in deterministic polynomial time as well we conclude that finite-valuedness is decidable in deterministic polynomial time. □

References

[1] B. Courcelle and P. Franchi-Zannettacci, Attribute grammars and recursive program schemes, part I, *Theoret. Comput. Sci.* **17** (1982) 163–191.

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

[3] P.J. Downey, R. Sethi and E.R. Tarjan, Variations on the common subexpression problem. *J. ACM* **27** (1980) 758–771.

[4] J. Engelfriet, Some open questions and recent results on tree transducers and tree languages, in: R.V. Book, ed., *Formal Language Theory* (Academic Press, New York, 1980) 241–286.

[5] Z. Esik, Decidability results concerning tree transducers I. *Acta Cybernet.* **5** (1980) 1–20.

[6] F. Gecseg and M. Steinby, *Tree Automata* (Akademiai Kiado, Budapest, 1984).

[7] R. Giegerich and K. Schmal, Code selection techniques: pattern matching, tree parsing and inversion of derivors, in: *Proc. ESOP'1988*, Lecture Notes in Computer Science, Vol. 300 (Springer, Berlin, 1988) 245–268.

[8] J. Karhumaki, W. Rytter and S. Jarominek, Efficient constructions of test sets for regular and context-free languages, in: *Proc. MFCS*, Lecture Notes in Computer Science, Vol. 520 (Springer, Berlin, 1991) 249–258.

[9] W. Paul, *Komplexitätstheorie* (Teubner, Stuttgart, 1978).

[10] M. Schützenberger, Sur les relations rationelles entre monoides libres, *Theoret. Comput. Sci.* **3** (1976) 243–259.

[11] H. Seidl, On the finite degree of ambiguity of finite tree automata, *Acta Inform.* **26** (1989) 527–542.

[12] H. Seidl, Deciding equivalence of finite tree automata, *SIAM J. Comput.* **19** (1990) 424–437.

[13] H. Seidl, Equivalence of finite-valued bottom-up finite state tree transducers is decidable. in: *Proc. CAAP '90*, Lecture Notes in Computer Science, Vol. 431 (Springer, Berlin, 1990) 269–284; a full version will appear in *Math. Systems Theory.*

[14] H. Seidl, Single-valuedness of bottom-up finite state tree transducers is decidable in polynomial time, in: *Proc. of the Toyohashi Symposium on Theoretical Computer Science* (1990) 69–73.

[15] A. Weber, Ueber die Mehrdeutigkeit und Wertigkeit von endlichen Automaten und Transducern, Doct. Thesis, Frankfurt/Main, 1987.

[16] A. Weber, On the valuedness of finite transducers, *Acta Inform.* **27** (1990) 749–780.

[17] A. Weber, A decomposition theorem for finite-valued transducers and an application to the equivalence problem, in: *Proc. MFCS'1988*, Lecture Notes in Computer Science, Vol. 324 (Springer, Berlin, 1988) 552–562; *SIAM J. Comput.*, to appear.

[18] A. Weber, On the lengths of values in a finite transducer, in: *Proc. MFCS'1989*, Lecture Notes in Computer Science, Vol. 379 (Springer, Berlin, 1989) 523–533.

[19] A. Weber and H. Seidl, On the degree of ambiguity of finite automata, in: *MFCS'1986*, Lecture Notes in Computer Science, Vol. 233 (Springer, Berlin, 1986) 620–629; also *Theoret. Comput. Sci.* **88** (1991) 325–349.

[20] Z. Zachar, The solvability of the equivalence problem for deterministic frontier-to-root tree transducers, *Acta Cybernet.* **4** (1978) 167–177.