PAPER Special Section on Foundations of Computer Science—Developments of the Theory of Algorithms and Computation—

Determinacy and Subsumption of Single-Valued Bottom-Up Tree Transducers*

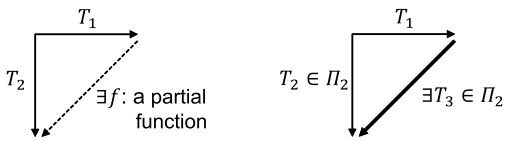
Kenji HASHIMOTO[†a)], ***Member***, **Ryuta SAWADA**[††], ***Nonmember***, **Yasunori ISHIHARA**[††], ***Member***, **Hiroyuki SEKI**[†], ***and*** **Toru FUJIWARA**[††], ***Fellows***

SUMMARY This paper discusses the decidability of determinacy and subsumption of tree transducers. For two tree transducers T_1 and T_2, T_1 determines T_2 if the output of T_2 can be identified by the output of T_1, that is, there is a partial function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$ where $[\![T_1]\!]$ and $[\![T_2]\!]$ are tree transformation relations induced by T_1 and T_2, respectively. Also, T_1 subsumes T_2 if T_1 determines T_2 and the partial function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$ can be defined by a transducer in a designated class that T_2 belongs to. In this paper, we show that determinacy is in coNEXPTIME for single-valued linear extended bottom-up tree transducers as the determiner class and single-valued bottom-up tree transducers as the determinee class. We also show that subsumption is in coNEXPITME for these classes, and a bottom-up tree transducer T_3 such that $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$ can be constructed if T_1 subsumes T_2.

key words: *determinacy, subsumption, tree transducer*

1. Introduction

The importance of data transformation has been emerging due to the recent extensive studies on data integration, data explosion, etc. Especially, it is desirable to guarantee that information is preserved in some sense through data transformation. As a formalization for information preservation in data transformation, the notions of *determinacy* and *subsumption* (or query rewriting) are known [2]–[4]. Let Q be a query to a database and V be a data transformation (or a view definition) of the database. Determinacy of Q by V means that the answer to Q can be identified by the answer to V. When information to be preserved is specified by a query Q, determinacy guarantees that for any database instance D, $V(D)$ gives enough information to uniquely determine the specified information $Q(D)$ for D. Subsumption means that the answer to Q can always be computed from the answer to V by some query in a designated class that Q belongs to. Compared with determinacy, subsumption guarantees that the necessary information $Q(D)$ can be extracted from the transformed data $V(D)$ by the same query language expressing Q.



(a) T_1 determines T_2 (b) T_1 subsumes T_2 w.r.t. Π_2

Fig. 1 Determinacy and subsumption

We study the decidability and complexity of determinacy and subsumption when both a query and a data transformation are given by tree transducers. Tree transducers are abstract machines that model binary relations on labeled ordered trees. A tree transducer is said to be *single-valued* if the tree transformation induced by the transducer is a partial function. Since an XML document has a tree structure, tree transducers are often used as a model of XML document transformations. Formally, for two single-valued tree transducers T_1 and T_2 in classes Π_1 and Π_2 of transducers, respectively, we say T_1 *determines* T_2 if there is a partial function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$ (see Fig. 1 (a)), where $[\![T_1]\!]$ and $[\![T_2]\!]$ are the tree transformation relations induced by T_1 and T_2, respectively. Π_1 and Π_2 are called the determiner class and the determinee class, respectively. We also say T_1 *subsumes* T_2 with respect to Π_2, if T_1 determines T_2 and the partial function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$ can be defined by a transducer in the class Π_2 (see Fig. 1 (b)). Determinacy and subsumption are undecidable in general. Our goal is to find practical subclasses of tree transducers for which determinacy and subsumption can be decided with lower complexity than known results, and to consider the problem of constructing a tree transducer T_3 in the determinee class such that $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$ if T_1 subsumes T_2.

In this paper, we first show that determinacy is in coNEXPTIME for single-valued linear extended bottom-up tree transducers (*sl*-xbots) as the determiner class and single-valued bottom-up tree transducers (*s*-bots) as the determinee class running over a ranked-tree encoding of a given XML document. Transformations induced by transducers in the classes include simple filterings, relabelings, insertions, and deletions of elements. Especially, *sl*-xbots do not allow duplications of elements. Given an *sl*-xbot T_1 and an *s*-bot T_2, the decision procedure works as follows: (1) construct a transducer T_1^{inv} that induces the inverse of T_1, (2) construct a transducer T_3 that induces the composition of T_1^{inv} followed

Manuscript received March 27, 2015.
Manuscript revised August 5, 2015.
Manuscript publicized December 16, 2015.

[†]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8601 Japan.
[††]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.

*A preliminary version of this paper was presented at the 7th International Conference on Language and Automata Theory and Applications [1].

a) E-mail: k-hasimt@is.nagoya-u.ac.jp

DOI: 10.1587/transinf.2015FCP0015

by T_2, then (3) decide whether T_3 is single-valued. We introduce a class of transducers with *grafting*, which allows to insert any tree in a specified tree language, in order to capture the inverses of transformations induced by *sl*-xbots as well as the composition of the inverses and *s*-bots. Next, we prove that single-valuedness is in coNP for the class. Lastly, we show that subsumption is in coNEXPTIME for *sl*-xbots as the determiner class and *s*-bots as the determinee class. The proof gives a construction method of an *s*-bot T_3 satisfying $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$ if T_1 subsumes T_2.

Due to space limitations, we omit some of the proofs including details of some construction methods. They are presented in the full version of this paper [5].

Related Work

Determinacy and subsumption (or query rewriting) have been well studied mainly for relational queries such as first-order logic and conjunctive queries [2]–[4]. In XML context, the problems has been studied in a setting where transformations and queries are modeled by tree transducers [6]. It was shown that determinacy is undecidable for a subclass of deterministic top-down tree transducers which can copy at most once as the determiner class and the identities as the determinee class. Also, determinacy and subsumption were shown to be decidable for compositions of single-valued linear extended top-down tree transducers with regular look-ahead (*sl*-xtop$^{\mathrm{R}}$s) as the determiner class, and for either deterministic top-down tree transducers with regular look-ahead (*d*-top$^{\mathrm{R}}$s) or deterministic MSO definable tree transducers as the determinee class. The decision relies on the decidability of equivalence and the closure property under composition for the classes of tree transducers.

In this paper, we adopt another approach to deciding determinacy and subsumption for *sl*-xbots and *s*-bots, which leads us a lower complexity result. In our preliminary version [1], we showed only the decidability results of determinacy (and subsumption) for *sl*-xbots and *s*-bots, and did not give their complexity results. Actually, the decidability result itself can be concluded from [6] because the class of transformations induced by *sl*-xbots is equivalent with *sl*-xtop$^{\mathrm{R}}$s, and the class of *s*-bots is a subclass of *d*-top$^{\mathrm{R}}$s. However, the complexity is not discussed in [6]. The size of composition of an *sl*-xbot v, a uniformizer u of v, and an *s*-bot q are at most doubly exponential in their sizes in general. Although we do not know a tight upper bound of the equivalence test for the composition and an *s*-bot, it is at most in doubly exponential time in their sizes from the known complexity result for *d*-top$^{\mathrm{R}}$s. In summary, the rough estimation shows that the complexity result by [6] is 4EXPTIME, while the complexity result in this paper is coNEXPTIME.

2. Preliminaries

2.1 Trees and Tree Automata

We treat only ranked labeled ordered trees and tree transducers which work on such trees. Though an XML document is often modeled by an unranked labeled ordered tree, we assume that an unranked tree is encoded to a ranked tree by some encoding such as First-Child-Next-Sibling encoding [7] and DTD-based encoding [8].

We denote the set of nonnegative integers by $\mathbb{N}$. Let $[i, j] = \{d \in \mathbb{N} \mid i \leq d \leq j\}$. In particular, we denote $[1, k]$ by $[k]$. A (ranked) alphabet is a finite set Σ of symbols with a mapping rk from Σ to $\mathbb{N}$. We denote the set of k-ary symbols of Σ by $\Sigma^{(k)} = \{\sigma \in \Sigma \mid \mathsf{rk}(\sigma) = k\}$. The set $\mathcal{T}_\Sigma$ of *ranked trees* over an alphabet Σ is the smallest set T such that $\sigma(t_1, \ldots, t_k) \in T$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T$. If $\sigma \in \Sigma^{(0)}$, we write σ instead of $\sigma()$. Let $\Gamma \subseteq \Sigma$ and $H \subseteq \mathcal{T}_\Sigma$. We define $\Gamma(H) = \{\gamma(t_1, \ldots, t_k) \mid \gamma \in \Gamma^{(k)}, t_1, \ldots, t_k \in H\}$. Let Δ be an alphabet. We denote by $\mathcal{T}_\Delta(H)$ the smallest set $T \subseteq \mathcal{T}_{\Sigma\cup\Delta}$ such that $H \subseteq T$ and $\Delta(T) \subseteq T$. The set of *positions* of $t = \sigma(t_1, \ldots, t_k) \in \mathcal{T}_\Sigma$ where $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$, denoted by $\mathrm{pos}(t)$, is defined by $\mathrm{pos}(t) = \{\epsilon\} \cup \{ip \mid i \in [k], p \in \mathrm{pos}(t_i)\}$. Note that the empty string ϵ is the position of the root of t and that $\mathrm{pos}(t) \subseteq \mathbb{N}^*$. The size $|t|$ of a tree t is $|\mathrm{pos}(t)|$. We write $p \preceq p'$ when p is a prefix of p', that is, p is an ancestor position of p', and $p \prec p'$ when p is a proper prefix of p'. For $p, p' \in \mathrm{pos}(t)$, let $\mathrm{nca}(p, p')$ be the nearest common ancestor position of p and p', that is, the longest common prefix of p and p'. For a tree $t = \sigma(t_1, \ldots, t_k)$ and $p \in \mathrm{pos}(t)$, the subtree $t|_p$ of t at p is defined as $t|_\epsilon = t$ and $t|_{ip} = t_i|_p$ for $i \in [k]$ and $p \in \mathrm{pos}(t_i)$. For trees t, t' and $p \in \mathrm{pos}(t)$, $t[t']_p$ denotes the tree obtained from t by replacing $t|_p$ with t' at position p. Let $\lambda_t(p)$ be the symbol of tree t at p, defined as $\lambda_t(p) = \sigma'$ where $t|_p = \sigma'(t'_1, \ldots, t'_k)$.

Let $X = \{x_*\} \cup \{x_i \mid i \geq 1\}$ be a set of variables of rank 0, and for every $k \geq 1$, $X_k = \{x_i \mid i \in [k]\}$. For $V \subseteq X$, we often write $\mathcal{T}_\Sigma(V)$ to mean $\mathcal{T}_{\Sigma\cup V}$. A tree $t \in \mathcal{T}_\Sigma(V)$ is *linear* in V if each variable in V occurs at most once in t. Let $C_\Sigma(V)$ denote the set of linear trees in $\mathcal{T}_\Sigma(V)$. A tree $t \in \mathcal{T}_\Sigma(V)$ is *complete* (or *non-deleting*) in V if each variable in V occurs at least once in t. Let $\bar{\mathcal{T}}_\Sigma(V)$ (resp. $\bar{C}_\Sigma(V)$) be the set of complete trees in $\mathcal{T}_\Sigma(V)$ (resp. $C_\Sigma(V)$). Note that, for a set V' of variables disjoint with V, $\bar{\mathcal{T}}_{\Sigma\cup V}(V')$ denotes the set of trees in $\mathcal{T}_\Sigma(V \cup V')$ such that every variable in V' occurs at least once. For $t \in \mathcal{T}_\Sigma(X)$ and $\sigma \in \Sigma \cup X$, let $\mathrm{pos}_\sigma(t) = \{p \in \mathrm{pos}(t) \mid \lambda_t(p) = \sigma\}$ and $\mathrm{pos}_Y(t) = \bigcup_{\sigma\in Y} \mathrm{pos}_\sigma(t)$ for $Y \subseteq \Sigma \cup X$. Let $\mathrm{var}(t)$ be the set of variables of t, and $\mathrm{yield}_X : \mathcal{T}_\Sigma(X) \to X^*$ be the function such that $\mathrm{yield}_X(x) = x$ for every $x \in X$ and $\mathrm{yield}_X(\sigma(t_1, \ldots, t_k)) = \mathrm{yield}_X(t_1) \cdots \mathrm{yield}_X(t_k)$ for every $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma(X)$. A tree $t \in \mathcal{T}_\Sigma(X)$ is *normalized* if $\mathrm{yield}_X(t) = x_1 \cdots x_k$ for some $k \in \mathbb{N}$. Every mapping $\theta : V \to \mathcal{T}_\Sigma(X)$ with $V \subseteq X$ is called a substitution. It can be naturally extended to the mapping from $\mathcal{T}_\Sigma(V)$ to $\mathcal{T}_\Sigma(X)$. If $V = X_k$ and $x_i\theta = t_i$ for each $i \in [k]$, we also denote $t\theta$ by $t[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$ or $t[t_1, \ldots, t_k]$ if the total order among variables is understood. If $V = \{x_*\}$ and $\theta(x_*) = t'$, we denote $t\theta$ by $t[t']$ or often tt' without brackets.

Let A, B, C be sets. A relation R from A to B is a subset of $A \times B$. The domain of R is $\mathrm{dom}(R) = \{t \mid (t, t') \in R\}$ and the range of R is $\mathrm{rng}(R) = \{t' \mid (t, t') \in R\}$. The inverse of R

is $R^{-1} = \{(t', t) \mid (t, t') \in R\}$. The composition of $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ is $R_2 \circ R_1 = \{(t, t'') \mid (t, t') \in R_1, (t', t'') \in R_2\} \subseteq A \times C$.

A *finite tree automaton* (TA for short) is a 4-tuple $A = (Q, \Sigma, Q_f, \gamma)$, where Q is a finite set of states, Σ is an alphabet, $Q_f \subseteq Q$ is the set of accepting states, and γ is a finite set of transition rules, each of which is of the form $C[q_1, \ldots, q_k] \to q$ where $q, q_1, \ldots, q_k \in Q$ and $C \in \bar{C}_\Sigma(X_k)$. Note that $Q \cap \Sigma = \emptyset$. In an ordinary definition of TAs the left-hand side of a rule is a tree with height one, like $a(x_1, \ldots, x_k)$ where $a \in \Sigma^{(k)}$ and $x_i \in X_k$. In this paper, we allow the left-hand side tree with height two or more. This extension does not change the class of tree languages recognized by TAs. The move relation $\Rightarrow_A \subseteq \mathcal{T}_\Sigma(Q) \times \mathcal{T}_\Sigma(Q)$ of the TA A is defined as follows: if $C[q_1, \ldots, q_k] \to q \in \gamma$ and $t|_p = C[q_1, \ldots, q_k]$ for some $p \in \mathrm{pos}(t)$, then $t \Rightarrow_A t[q]_p$. The tree language *recognized* by A is $L(A) = \{t \in \mathcal{T}_\Sigma \mid t \Rightarrow_A^* q \text{ for some } q \in Q_f\}$ where $\Rightarrow_A^*$ is the reflexive transitive closure of $\Rightarrow_A$. For a state q of A, let $A(q)$ be the TA obtained from A by replacing the set Q_f of accepting states with the singleton $\{q\}$. A set L of trees is *regular* if it is recognized by some TA. The size $|\rho|$ of a transition rule ρ is the size of the tree in its left-hand side, and the size of the TA A is $|Q| + |\Sigma| + \sum_{\rho \in \gamma} |\rho|$.

2.2 Tree Transducers

An *extended bottom-up tree transducer* (xbot) [9] is a 5-tuple $T = (Q, \Sigma, \Delta, Q_f, \delta)$, where Q is a finite set of states, Σ is an input alphabet, Δ is an output alphabet, $Q_f \subseteq Q$ is a set of final states, and δ is a set of transduction rules of the form $C[q_1(x_1), \ldots, q_k(x_k)] \to q(t)$ where $k \in \mathbb{N}$, $C \in \bar{C}_\Sigma(X_k)$, $t \in \mathcal{T}_\Delta(X_k)$, $q, q_1, \ldots, q_k \in Q$. A rule is normalized if its left-hand side is normalized. Without loss of generality, we can assume that every rule is normalized. A rule $\rho \in \delta$ is an *ϵ-rule* if the left-hand side of ρ has the form $q(x)$ for some $q \in Q$, and it is *input-consuming* otherwise. The xbot T is a *bottom-up tree transducer* (bot) if the left-hand side of every rule in δ contains exactly one symbol in Σ. Also, we define an xbot$^{-\epsilon}$ as an xbot without ϵ-rules. T is a *linear extended bottom-up tree transducer* (l-xbot) if the tree t in the right-hand side of each rule in δ is linear. The size $|\rho|$ of a transduction rule ρ is the sum of the sizes of the trees in both sides of ρ, and the size $|T|$ of the xbot T is $|Q| + |\Sigma| + |\Delta| + \sum_{\rho \in \delta} |\rho|$.

The move relation $\Rightarrow_T \subseteq \mathcal{T}_\Sigma(Q(\mathcal{T}_\Delta(X))) \times \mathcal{T}_\Sigma(Q(\mathcal{T}_\Delta(X)))$ of the xbot T is defined as follows: $t \Rightarrow_T^\rho t'$ for a rule $\rho = (l \to r) \in \delta$ if there exists a position $p \in \mathrm{pos}(t)$ and a substitution $\theta : X \to \mathcal{T}_\Delta(X)$ such that $t|_p = l\theta$ and $t' = t[r\theta]_p$, and $t \Rightarrow_T t'$ if there exists $\rho \in \delta$ such that $t \Rightarrow_T^\rho t'$. The *transformation induced by T*, denoted as $[\![T]\!]$, is the relation defined as $\{(t, t') \in \mathcal{T}_\Sigma \times \mathcal{T}_\Delta \mid t \Rightarrow_T^* q(t') \text{ for some } q \in Q_f\}$ where $\Rightarrow_T^*$ is the reflexive transitive closure of $\Rightarrow_T$. We abbreviate $\mathrm{dom}([\![T]\!])$ by $\mathrm{dom}(T)$ and $\mathrm{rng}([\![T]\!])$ as $\mathrm{rng}(T)$. For a tree t, $[\![T]\!](t) = \{t' \mid (t, t') \in [\![T]\!]\}$. For a TA A, the image $T(A)$ of $L(A)$ by T is $\{t' \mid (t, t') \in [\![T]\!], t \in L(A)\}$. For a state q of T, let $T(q)$ be the xbot obtained from T by replacing the set Q_f of final states with the singleton $\{q\}$.

Tree transducers T and T' are *equivalent* if $[\![T]\!] = [\![T']\!]$. A transducer T is said to be *single-valued* (or *functional*) if any two pairs of (t, t') and (t, t'') in $[\![T]\!]$ satisfy $t' = t''$. If T is single-valued and $(t, t') \in [\![T]\!]$, then we also write $T(t)$ for t'. It is known that the single-valuedness of bots is decidable in polynomial time [10]. We use the prefix 's' to represent that a transducer is single-valued, e.g., we write for short an s-xbot to denote a single-valued xbot.

We recall the notion of reducedness [10], which is defined for bots but can be naturally applied to xbots. Let $\perp$ be the special symbol not in Δ. An xbot $T = (Q, \Sigma, \Delta \cup \{\perp\}, Q_f, \delta)$ is *reduced* if and only if the following three conditions hold:

1. T has no useless states, that is, for every state $q \in Q$, there exists a tree $t = Ct_s \in \mathrm{dom}(T)$ such that $C \in \bar{C}_\Sigma(\{x_*\})$, $t_s \in \mathcal{T}_\Sigma$, and $t \Rightarrow_T^* C[q(t'_s)] \Rightarrow_T^* q_f(t')$ for some $q_f \in Q_f$ and $t'_s, t' \in \mathcal{T}_\Delta$.
2. There exists a subset $U(T)$ of Q such that for every rule $C[q_1(x_1), \ldots, q_k(x_k)] \to q(t) \in \delta$,
   - if $q \in U(T)$ then $t = \perp$ and $q_i \in U(T)$ for each $i \in [k]$, and
   - if $q \notin U(T)$ then (1) $t \neq \perp$ and (2) for each $i \in [k]$, $q_i \in U(T)$ if and only if $x_i \notin \mathrm{var}(t)$.
3. No $q \in Q_f$ occurs in the left-hand side of any rule in δ.

Note that for any $q \in U(T)$ and $t = Ct_s \in \mathrm{dom}(T)$ where $C \in \bar{C}_\Sigma(\{x_*\})$, if $t \Rightarrow_T^* C[q(t')]$ then $t' = \perp$ and the final output for t does not contain $\perp$. That is, the intermediate output at q is always $\perp$ and it is eventually abandoned. Conversely, for $q \in Q - U(T)$, the intermediate output at q is in $\mathcal{T}_\Delta$ and it is contained in the final output. For every xbot T, a reduced xbot equivalent with T can be constructed in linear time in the same way as the construction for bots [10].

A *multi bottom-up tree transducer* (mbot) [9], whose states might have ranks different from one, is a 5-tuple $T = (Q, \Sigma, \Delta, Q_f, \delta)$ where Q is a ranked alphabet, $Q_f \subseteq Q - Q^{(0)}$ and δ is a finite set of rules of the form $l \to r$ where $l \in \Sigma(Q(X))$ is linear in X and $r \in Q(\mathcal{T}_\Delta(\mathrm{var}(l)))$. The size $|T|$ and the move relation $\Rightarrow_T$ are defined in the same way as xbots. The induced transformation $[\![T]\!]$ is defined as $\{(t, t'_1) \in \mathcal{T}_\Sigma \times \mathcal{T}_\Delta \mid t \Rightarrow_T^* q(t'_1, \ldots, t'_n) \text{ for some } q \in Q_f\}$. For $q \in Q^{(k)}$ and a tuple $\mathbf{x} = (x_{i_1}, \ldots, x_{i_k})$ in $(X - \{x_*\})^k$, let $q(\mathbf{x})$ denote $q(x_{i_1}, \ldots, x_{i_k})$.

In addition, we define *computations* of the mbot T. A tree φ in $C_\delta(X_k)$ is a $(q, q_1 \cdots q_k)$-computation of T for $t \in C_\Sigma(X_k)$ if the following conditions hold.

1. If $t = x_i$, then $q = q_i$ and $\varphi = x_i$, and
2. If $t = a(t_1, \ldots, t_m)$, then $\varphi = \rho(\varphi_1, \ldots, \varphi_m)$ for some rule $\rho = a(q'_1(\mathbf{x}_1), \ldots, q'_m(\mathbf{x}_m)) \to q(t^{(1)}, \ldots, t^{(\mathrm{rk}(q))}) \in \delta$ and $(q'_j, q_1 \cdots q_k)$-computations φ_j for t_j ($j \in [m]$).

We say that φ is a $(q, q_1 \cdots q_k)$-computation of T if it

is a $(q, q_1 \cdots q_k)$-computation for some $t \in C_\Sigma(X_k)$. A (q, ϵ)-computation is also called a q-computation. Any q-computation with $q \in Q_f$ is *accepting*.

The output $T(\varphi)$ produced by a $(q, q_1 \cdots q_k)$-computation φ is defined inductively as follows. Let $Y = \{y_i^j \mid i \geq 1, j \in [N]\}$ be a set of variables disjoint with X where $N = \max\{\mathsf{rk}(q) \mid q \in Q\}$. If $\varphi = x_i$, then $T(\varphi) = (y_i^1, \ldots, y_i^{\mathsf{rk}(q_i)})$. Assume $\varphi = \rho(\varphi_1, \ldots, \varphi_m)$ where $\rho = (a(q'_1(x_{11}, \ldots, x_{1\mathsf{rk}(q'_1)}), \ldots, q'_m(x_{m1}, \ldots, x_{m\mathsf{rk}(q'_m)})) \to q(t^{(1)}, \ldots, t^{(\mathsf{rk}(q))})) \in \delta$ by renaming variables without loss of generality. Also assume inductively that $T(\varphi_j) = (\tau_{j1}, \ldots, \tau_{j\mathsf{rk}(q'_j)})$ for $j \in [m]$. Define $T(\varphi) = (t^{(1)}\theta, \ldots, t^{(\mathsf{rk}(q))}\theta)$ where $\theta(x_{jl}) = \tau_{jl}$ for $j \in [m]$ and $l \in [\mathsf{rk}(q'_j)]$. We can see that $(t, t') \in [\![T]\!]$ if and only if there is an accepting computation φ of T for $t \in \mathcal{T}_\Sigma$ such that $T(\varphi) = t'$.

2.3 Determinacy and Subsumption of Tree Transducers

Let Π_1 and Π_2 be arbitrary classes of tree transducers.

Definition 1 (Determinacy): Let T_1 and T_2 be tree transducers in Π_1 and Π_2, respectively, such that $\mathrm{dom}(T_2) \subseteq \mathrm{dom}(T_1)$. T_1 *determines* T_2 if and only if there exists a partial function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$. Π_1 is called the *determiner class* and Π_2 is called the *determinee class*.

Definition 2 (Subsumption): Let T_1 and T_2 be tree transducers in Π_1 and Π_2, respectively, such that $\mathrm{dom}(T_2) \subseteq \mathrm{dom}(T_1)$. T_1 *subsumes* T_2 with respect to Π_2 if and only if there exists a single-valued transducer $T_3 \in \Pi_2$ such that $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$.

From the definition, if T_1 subsumes T_2 then T_1 determines T_2. Conversely, even if there exists some function f such that $[\![T_2]\!] = f \circ [\![T_1]\!]$, f cannot always be induced by some transducer in Π_2 in general.

If determinacy is decidable for a determiner class Π_1 and a determinee class Π_2, we simply say determinacy is decidable for (Π_1, Π_2). We will use a similar notation for subsumption.

3. Decidability Results on Determinacy

3.1 Overview

We consider the problem of deciding whether, given single-valued linear xbot (*sl*-xbot) T_1 and single-valued bot (*s*-bot) T_2 such that $\mathrm{dom}(T_2) \subseteq \mathrm{dom}(T_1)$, T_1 determines T_2 or not. Our approach is based on the following proposition.

Proposition 1: For any single-valued transducers T_1 and T_2 such that $\mathrm{dom}(T_2) \subseteq \mathrm{dom}(T_1)$, T_1 determines T_2 if and only if $[\![T_2]\!] \circ [\![T_1]\!]^{-1}$ is a partial function.

According to Proposition 1, given *sl*-xbot T_1 and *s*-bot T_2, our decision algorithm works as follows:

Step 1: Construct a transducer T_1^{inv} such that $[\![T_1^{inv}]\!] = [\![T_1]\!]^{-1}$;

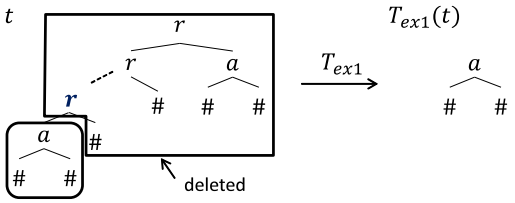


Fig. 2 A transducer T_{ex1}

Step 2: Construct a transducer T_3 such that $[\![T_3]\!] = [\![T_2]\!] \circ [\![T_1^{inv}]\!]$;

Step 3: Decide whether T_3 is single-valued.

In Step 1, the inverse transducer T_1^{inv} of T_1 is computed. T_1^{inv} is not necessarily an *l*-xbot. Due to this, we introduce a slightly larger class, linear extended bottom-up tree transducers with *grafting* (*l*-xbot^{+g} for short), that can represent not only inverses of *l*-xbots but also the composition of the inverses with *s*-bots. In Step 2, an xbot^{+g} T_3 which represents the composition of T_1^{inv} followed by T_2 is constructed. Lastly, it is determined whether the composition transducer T_3 is single-valued.

Before we explain the detail of each step, we give an example of an *sl*-bot whose inverse cannot be computed by any *l*-xbot.

Example 1: Let $\Sigma = \{r, a, \#\}$ and $\Delta = \{a, \#\}$. Consider an *sl*-bot $T_{ex1} = (\{q_r, q\}, \Sigma, \Delta, \{q_r\}, \delta)$ where δ consists of the following four rules:

$$\# \to q(\#), \quad a(q(x_1), q(x_2)) \to q(a(x_1, x_2)),$$
$$r(q(x_1), q(x_2)) \to q_r(x_1), \quad r(q_r(x_1), q(x_2)) \to q_r(x_1).$$

T_{ex1} is defined on the trees t where an r-node appears only as the root of t or the left child of another r-node. T_{ex1} leaves only the subtree of t at the left child of the bottom-most r-node (see Fig. 2). There is an infinite number of trees t' such that $T_{ex1}(t') = T_{ex1}(t)$ because the inverse of T_{ex1} allows to insert any number of r-labeled ancestor nodes having arbitrary trees in $\mathcal{T}_{\Sigma-\{r\}}$ as their right subtrees. Even if ϵ-rules are allowed, no *l*-xbot allows to insert a node having an arbitrary tree in $\mathcal{T}_{\Sigma-\{r\}}$ as its right subtree. Therefore, there is no *l*-xbot T such that $[\![T]\!] = [\![T_{ex1}]\!]^{-1}$.

To express the inverse of T_{ex1} in Example 1, a transducer has to, for an input tree, insert any number of internal nodes and subtrees non-deterministically. To capture the inverse of *sl*-xbots, we extend xbots by grafting. Let Z be a set of symbols of rank 0, called g-variables, disjoint with Σ, Δ, and X. A grafting is a mapping $G : Z \to 2^{\mathcal{T}_\Delta}$. For $t \in \mathcal{T}_\Delta(X \cup Z)$, let $G(t)$ be the set of trees in $\mathcal{T}_\Delta(X)$ obtained from t by replacing each g-variable z by a tree in $G(z)$ in all possible ways.

An xbot^{+g} is a system $T = (Q, \Sigma, \Delta, Q_f, G, \delta)$ where Q, Σ, Δ, and Q_f are the same as for an xbot, G is a grafting, and δ is a finite set of rules of the form $C[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)$ where $k \in \mathbb{N}$, $C \in \bar{C}_\Sigma(X_k)$, $t_r \in \mathcal{T}_\Delta(X_k \cup Z)$, and $q, q_1, \ldots, q_k \in Q$. The move relation by a rule $C[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)$ is as follows:

if $t|_p = C[q_1(t_1), \ldots, q_k(t_k)]$ where $t_1, \ldots, t_k \in \mathcal{T}_\Delta$, then $t \Rightarrow t[q(t'[t_1, \ldots, t_k])]_p$ for some $t' \in G(t_r)$.

An xbot^{+g} is called an xbot$^{+g(R)}$ when $G(z)$ is regular for each g-variable z. Also, an xbot^{+g} is called an xbot$^{+g(B(R))}$ when for each g-variable z, $G(z)$ can be written as $T(A)$ for some bot T and TA A.

Example 2: Consider an l-xbot$^{+g(R)}$ $T_{ex2} = (\{q, q_r\}, \Delta, \Sigma, \{q_r\}, G, \delta')$ where δ' consists of the following four rules

$$\#() \to q(\#), \quad a(q(x_1), q(x_2)) \to q(a(x_1, x_2)),$$
$$q(x_1) \to q_r(r(x_1, z)), \quad q_r(x_1) \to q_r(r(x_1, z)),$$

and $G(z) = \mathcal{T}_{\Sigma-\{r\}}$. Then, T_{ex2} induces the inverse of T_{ex1}.

Steps 1 to 3 of the decision algorithm can be refined as follows.

3.2 Step 1: Inversion of *sl*-xbots

We provide a construction procedure of an l-xbot^{+g} representing the inverse of a given *sl*-xbot. Intuitively, the l-xbot^{+g} is obtained by swapping the left-hand and right-hand sides of each rule of the *sl*-xbot. However, we must take care of variables occurring only in the left-hand side, which mean deletions of subtrees. In the construction procedure, such variables are replaced with g-variables.

Let $T = (Q, \Sigma, \Delta, Q_f, \delta)$ be an l-xbot. The construction procedure is as follows.

1. Construct a TA $A_T = (Q, \Sigma, Q_f, \gamma)$ where $\gamma = \{C_l[q_1, \ldots, q_k] \to q \mid C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(C_r) \in \delta\}$. Note that A_T recognizes dom(T).
2. Construct an l-xbot$^{+g(R)}$ $T' = (Q, \Delta, \Sigma, Q_f, G, \delta')$ such that $G(z_i^q) = L(A_T(q))$ for each g-variable z_i^q and δ' is the smallest set satisfying the following condition: Let $C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(C_r)$ be an arbitrary rule in δ. Let θ_l be the substitution such that $\theta_l(x_i) = q_i(x_i)$ for each $i \in [k]$, and let θ_r be the substitution such that $\theta_r(x_i) = x_i$ if $x_i \in \mathrm{var}(C_r)$ and $\theta_r(x_i) = z_i^{q_i}$ otherwise. Moreover, let θ_n be the substitution for normalization, which is the bijective function from var(C_r) to $X_{k'}$ ($k' = |\mathrm{var}(C_r)|$) making $(C_r\theta_l)\theta_n$ normalized. Then, $(C_r\theta_l)\theta_n \to q((C_l\theta_r)\theta_n) \in \delta'$.

Here, we can represent the grafting G of T' as the mapping from Z to the set of states of A_T: $G(z) = q$ instead of $G(z) = L(A_T(q))$ for each g-variable z. The construction procedure can be done in linear time, and the size of the inverse transducer T' is $O(|T|)$.

Lemma 1: For any l-xbot T, an l-xbot$^{+g(R)}$ T^{inv} such that $[\![T^{inv}]\!] = [\![T]\!]^{-1}$ can be constructed.

Proof. It can be shown by induction on move relations of the transducers that the inverse transducer T^{inv} of T is correctly constructed by the procedure above. □

3.3 Step 2: Composition of l-xbot$^{+g(R)}$ and s-bot

This step constructs an xbot^{+g} equivalent with the composition of the l-xbot$^{+g(R)}$ T_1^{inv} followed by an s-bot T_2.

Lemma 2: For any l-xbot$^{+g(R)}$ T and bot T', an xbot$^{+g(B(R))}$ T'' such that $[\![T'']\!] = [\![T']\!] \circ [\![T]\!]$ can be constructed.

Proof. It is known that for an l-bot T and a bot T', a bot T'' such that $[\![T'']\!] = [\![T']\!] \circ [\![T]\!]$ can be constructed [11], [12]. We extend straightforwardly the construction for an l-xbot$^{+g(R)}$ T and a bot T'. Let $T = (Q, \Sigma, \Delta, Q_f, G, \delta)$ and $T' = (Q', \Delta, \Gamma, Q'_f, \delta')$. Construct $T'' = (Q \times Q', \Sigma, \Gamma, Q_f \times Q'_f, G'', \delta'')$, where

- δ'' is the set satisfying the following condition: for each $C[q_1(x_1), \ldots, q_n(x_n)] \to q(t) \in \delta$,

  $$C[(q_1, q'_1)(x_1), \ldots, (q_n, q'_n)(x_n)] \to (q, q')(t') \in \delta''$$

  if and only if $t\theta \Rightarrow^*_{T'} q'(t')^\dagger$ for some substitution $\theta : X_n \cup Z \to Q'(X_n \cup Z \times Q')$ such that
  - if $x_i \in X_n$ then $\theta(x_i) = q'_i(x_i)$, and
  - if $z \in Z$ then $\theta(z) = q'((z, q'))$ for some $q' \in Q'$,

  and
- $G'' : Z \times Q' \to 2^{\mathcal{T}_\Gamma}$ such that $G''(z, q') = T'(q')(G(z))$ for each $(z, q') \in Z \times Q'$.

The correctness of the above construction can be proved by showing that for $u \in \mathcal{T}_\Sigma$ and $u' \in \mathcal{T}_\Gamma$, $u \Rightarrow^*_{T''} (q, q')(u')$ if and only if there exists $u'' \in \mathcal{T}_\Delta$ such that $u \Rightarrow^*_T q(u'')$ and $u'' \Rightarrow^*_{T'} q'(u')$. □

The grafting G'' of the composition transducer T'' of T_1^{inv} and T_2 can be expressed by a mapping from g-variables to pairs of the states in A_{T_1} and T_2, i.e., $G''(z, q') = (q, q')$ instead of $G''(z, q') = T_2(q')(G(z))$ where $G(z) = L(A_T(q))$ for the grafting G of T_1^{inv}. Thus, it can be expressed by using $O(|A_{T_1}| + |T_2|)$ space. Let N_1 and N_2 be the number of rules of T_1^{inv} and T_2, respectively. Let S_{l1} (resp. S_{r1}) be the maximum size of the left-hand size (resp. the right-hand side) of rules of T_1^{inv}, and S_{r2} be the maximum size of the right-hand side of rules of T_2. Let M_1 be the maximum number of symbols in neither Q nor X appearing in the right-hand side of rules of T_1^{inv}. The size of each rule of T'' is at most $S_{l1} + S_{r2}{}^{M_1}$, and the number of rules of T'' is at most $N_1 N_2{}^{M_1}$. Thus, the size of T'' is exponential in the sizes of T_1^{inv} and T_2. If M_1 is bounded by some constant, the size of T'' is polynomial in the sizes of T_1^{inv} and T_2.

3.4 Step 3: Deciding Single-Valuedness of xbot$^{+g(B(R))}$

This step decides whether the xbot$^{+g(B(R))}$ obtained in Step 2 is single-valued. It is known that single-valuedness of bots is decidable in polynomial time [10]. However, the class of transformations induced by xbot^{+g}s is a proper superclass of the class induced by bots.

Let T_3 be the xbot$^{+g(B(R))}$ obtained in Step 2. Let G be the grafting of T_3. The overview of Step 3 is as follows:

†The move relation $\Rightarrow_{T'}$ is naturally extended to be the relation on $\mathcal{T}_\Delta(Q'(\mathcal{T}_\Gamma(X \cup Z \times Q'))) \times \mathcal{T}_\Delta(Q'(\mathcal{T}_\Gamma(X \cup Z \times Q')))$: $t \Rightarrow_{T'} t'$ if there exists $p \in \mathrm{pos}(t)$ and a substitution $\theta : X \to \mathcal{T}_\Gamma(X \cup Z \times Q')$ such that $t|_p = l\theta$ and $t' = t[r\theta]_p$.

Step 3-1 Check if there exists an xbot equivalent with T_3. If there exists no such xbot, answer that T_3 is not single-valued. Otherwise, construct a reduced xbot $T_{3.1}$ equivalent with T_3 and go to 3-2.

Step 3-2 Check if there exists an xbot$^{-\epsilon}$ equivalent with $T_{3.1}$. If there exists no such xbot$^{-\epsilon}$, answer that T_3 is not single-valued. Otherwise, construct a reduced xbot$^{-\epsilon}$ $T_{3.2}$ equivalent with $T_{3.1}$ and go to 3-3.

Step 3-3 Decide whether $T_{3.2}$ is single-valued or not.

We further refine the above sub-steps as follows.

(1) Step 3-1: Constructing an equivalent xbot without g-variables

Step 3-1 tries to eliminate g-variables from T_3 keeping the induced transformation. For this, we check if the language (expressed by) $G(z)$ contains two or more trees for each g-variable z. If there exists a g-variable mapped to a language of size more than one, T_3 is not single-valued. Otherwise, we can easily obtain an equivalent xbot without g-variables. Step 3-1 consists of the following substeps.

(i) For each rule containing a g-variable z of T_3,

- if $G(z) = \emptyset$ then delete the rule, and
- if $G(z) = \{t\}$ for some tree t then replace z with t.

(ii) Construct an equivalent reduced xbot$^{+\mathrm{g(B(R))}}$ $T_{3.1}$.

(iii) If $T_{3.1}$ has a rule containing a g-variable z (which satisfies $|G(z)| \geq 2$ by (i)), answer that T_3 is not single-valued and halt. Otherwise, $T_{3.1}$ is an xbot.

If the procedure outputs a transducer, it is an xbot equivalent with T_3 because it contains no g-variables and the deletion and replacement in (i) certainly do not ruin the transformation induced by T_3. In what follows, we show in Lemma 3 that if the procedure fails to output an equivalent transducer, T_3 is not single-valued.

Lemma 3: Let $T = (Q, \Sigma, \Delta, Q_f, G, \delta)$ be a reduced xbot$^{+\mathrm{g}}$. If T has a rule whose right-hand side has a state $q \in Q - U(T)$ and a g-variable z such that $|G(z)| \geq 2$, then T is not single-valued.

Proof. Assume that T has a rule $C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)$ where $q \in Q - U(T)$ and t_r has a g-variable z such that $|G(z)| \geq 2$. Since T is reduced, there exist $t = CC_l[t_1, \ldots, t_k] \in \mathrm{dom}(T)$ where $C \in \bar{C}_\Sigma(\{x_*\})$, $t' \in \bar{\mathcal{T}}_\Delta(\{x_*\})$, $t'_1, \ldots, t'_k \in \mathcal{T}_\Delta$, and $q_f \in Q_f$ such that $t \Rightarrow^*_T CC_l[q_1(t'_1), \ldots, q_k(t'_k)] \Rightarrow_T C[q(t_r[t'_1, \ldots, t'_k])] \Rightarrow^*_T q_f(t't'_r[t'_1, \ldots, t'_k])$ for any $t'_r \in G(t_r)$. Since $|G(z)| \geq 2$, $G(t_r)$ has at least two distinct trees t^1_r and t^2_r. Also, the positions of each variable in t^1_r and t^2_r are identical. Hence, $t^1_r[t'_1, \ldots, t'_k] \neq t^2_r[t'_1, \ldots, t'_k]$ and thus the final outputs $t't^1_r[t'_1, \ldots, t'_k]$ and $t't^2_r[t'_1, \ldots, t'_k]$ are different. Therefore, $|[\![T]\!](t)| \geq 2$. □

We need to check in the procedure which one holds, $|T(A)| = 0$, $|T(A)| = 1$, or $|T(A)| \geq 2$ for an s-bot T and a TA A because $G(z) = T_2(q')(A_{T_1}(q))$ for some states q and q' of A_{T_1} and T_2, respectively. It can be done in polynomial time by using alternating tree automata. An alternating tree automaton (ATA) [7] is a tuple $A = (Q, \Sigma, Q_f, \Phi)$ where Q is a finite set of states, Σ is an alphabet, $Q_f \subseteq Q$ is a set of accepting states, and Φ is a mapping from $Q \times \Sigma$ to $B^+(Q \times \mathbb{N})$ such that $\Phi(q, a) \in B^+(Q \times [\mathsf{rk}(a)])$ where $B^+(P)$ is the set of positive propositional formulas over the set P of propositional variables. The acceptance of a tree in a state by an ATA is inductively defined as follows: the ATA A accepts a tree $a(t_1, \ldots, t_n)$ in a state q if $\Phi(q, a)$ is satisfied with the truth assignment ν that assigns true to (q, i) if and only if A accepts t_i in q for each $(q, i) \in Q \times [\mathsf{rk}(a)]$.

Lemma 4: Given an s-bot $T = (Q^T, \Sigma, \Delta, Q_f, \delta)$ and a TA $A = (Q^A, \Sigma, Q^A_f, \gamma)$, it can be decided in polynomial time which of the three cases holds: $|T(A)| = 0$, $|T(A)| = 1$, or $|T(A)| \geq 2$.

Proof. The decision procedure is shown below.

1. Decide whether $L(A_T) \cap L(A) = \emptyset$, where A_T recognizes $\mathrm{dom}(T)$. If this is the case, answer that $|T(A)| = 0$. Otherwise, go to 2.
2. Construct a TA A_t that accepts only a single tree $t \in \mathrm{dom}(T) \cap L(A)$.
3. Construct a TA A_o such that $L(A_o) = T(A_t) = \{T(t)\}$.
4. Construct an ATA $A_{\bar{o}}$ recognizing the complement of $L(A_o)$, i.e., $\mathcal{T}_\Delta - L(A_o)$.
5. Construct an ATA A' such that $L(A') = \{t \in \mathcal{T}_\Sigma \mid T(t) \in L(A_{\bar{o}})\}$.
6. Decide whether $L(A') \cap L(A) = \emptyset$. If so, answer that $|T(A)| = 1$. Otherwise, answer that $|T(A)| \geq 2$.

It is clear that $\mathrm{dom}(T) \cap L(A) = \emptyset$ if and only if $|T(A)| = 0$. Assume that $t \in \mathrm{dom}(T) \cap L(A)$. If there exists a tree t' in $L(A') \cap L(A)$, then $T(t')$ is defined and $T(t') \neq T(t)$, that is, $|T(A)| \geq 2$, because $T(t') \in L(A_{\bar{o}})$, $T(t) \in L(A_o)$, and $L(A_{\bar{o}}) \cap L(A_o) = \emptyset$. Otherwise, for all $t' \in \mathrm{dom}(T) \cap L(A)$, $t' \notin L(A')$ and thus $T(t') \in \mathcal{T}_\Delta - L(A_{\bar{o}}) = L(A_o) = \{T(t)\}$. Therefore, $|T(A)| = 1$.

Constructing the intersection TA A_I of A_T and A, the size of which is $O(|T||A|)$, and the emptiness test of A_I can be done in $O(|T||A|)$ time. We can construct A_t in $O(|A_I|)$ time by choosing for each accessible state an appropriate rule of A_I with the state in the right-hand side, along with the emptiness test of A_I. TA A_o can be constructed in $O(|T||A_t|)$ time as follows. We can assume without loss of generality that T is reduced and A_t is ordinary and reduced. For rules $a(p_1(x_1), \ldots, p_n(x_n)) \to p(t)$ of T and $a(q_1, \ldots, q_n) \to q$ of A_t, respectively, where t is not $\perp$, construct a rule $t' \to (p, q)$ where $t' = t[x_1 \leftarrow (p_1, q_1), \ldots, x_n \leftarrow (p_n, q_n)]$. The set of accepting states of A_o is the set of state pairs (p_f, q_f) such that p_f is a final state of T and q_f is an accepting state of A_t. And, since A_o accepts only a single tree, we

can construct ATA $A_{\bar{o}}$ in $O(|A_o|)$ time such that each transition formula of $A_{\bar{o}}$ is a disjunction of atomic propositions. Let $A_{\bar{o}} = (Q, \Delta, Q_f, \Phi)$. Then, we construct ATA $A' = (Q^T \times Q, \Sigma, Q'_f, \Phi')$ from $A_{\bar{o}}$ and T such that for $(q^T, q) \in Q^T \times Q$ and $a \in \Sigma$,

$$\Phi'((q^T, q), a) = \bigvee_{a(q_1^T(x_1),\cdots,q_n^T(x_n))\to q^T(t_r)\in\delta} \mathrm{Inf}(t_r[q_1^T(x_1), \ldots, q_n^T(x_n)], q).$$

The function Inf is defined inductively as follows:

$$\mathrm{Inf}(b(t_1, \ldots, t_n), q) = \begin{cases} \mathrm{Inf}(t_1, q_1) \vee \cdots \vee \mathrm{Inf}(t_n, q_n) \\ \qquad \text{if } \Phi(q, b) = (q_1, 1) \vee \cdots \vee (q_n, n), \\ true \qquad \text{if } \Phi(q, b) = true, \\ false \qquad \text{if } \Phi(q, b) = false. \end{cases}$$

$$\mathrm{Inf}(q_i^T(x_i), q_j) = ((q_i^T, q_j), i).$$

Observe that each transition formula of A' is a disjunction of atomic propositions. From the observation, the emptiness of the intersection TA of A' and A can be decided in $O(|A'||A|)$. □

(2) Step 3-2: Constructing an equivalent xbot$^{-\epsilon}$ without ϵ-rules

We say that a nonempty subset δ_ϵ of ϵ-rules is *repeatedly-producing at state* q if $q(x_*) \Rightarrow^*_{\delta_\epsilon} q(t)$ for some tree $t \in \bar{\mathcal{T}}_\Delta(\{x_*\}) - \{x_*\}$, where $\Rightarrow^*_{\delta_\epsilon}$ means zero or more applications of rules in δ_ϵ.

Step 3-2 first finds a subset of ϵ-rules of $T_{3.1}$ repeatedly-producing at some state, which produces many distinct output trees for an input tree. If such subset exists, it answers that T is not single-valued. Otherwise, it eliminates ϵ-rules keeping the transformation induced by $T_{3.1}$. We give two lemmas to show the correctness of the above procedure. We will use an idea similar to the proof of Proposition 10 of [9].

Lemma 5: Let $T = (Q, \Sigma, \Delta, Q_f, \delta)$ be a reduced xbot. If there is a subset δ_ϵ of ϵ-rules in δ repeatedly-producing at some $q \in Q - U(T)$, then T is not single-valued.

Proof. Assume that there is a subset δ_ϵ of ϵ-rules in δ repeatedly-producing at some $q \in Q - U(T)$. Then, there are trees $t \in \mathcal{T}_\Sigma$, $t' \in \mathcal{T}_\Delta$, $D \in \bar{C}_\Sigma(\{x_*\})$, $t_c, t_D \in \bar{\mathcal{T}}_\Delta(\{x_*\})$, and $q_f \in Q_f$ such that t_c has at least one symbol in Δ and $D[t] \Rightarrow^*_T D[q(t')] \Rightarrow^*_T D[q(t_c t')] \Rightarrow^*_T D[q(t_c^n t')] \Rightarrow^*_T q_f(t_D t_c^n t')$ for any positive integer n. □

Following [9], we call a state $q_e \in Q$ an *end state* if there exists an input-consuming rule whose left-hand side contains q_e. The set of all end states of Q is denoted by $E(T)$. For each input-consuming rule $\rho = (C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)) \in \delta$, let $\mathrm{rhs}^+(\rho) = \{q'(t) \mid q(t_r) \Rightarrow^*_T q'(t), q' \in E(T) \cup Q_f\}$. Note that only ϵ-rules can be used in the derivation $q(t_r) \Rightarrow^*_T q'(t)$.

Lemma 6: Let $T = (Q, \Sigma, \Delta, Q_f, \delta)$ be a reduced xbot. If there is no subset δ_ϵ of ϵ-rules in δ repeatedly-producing at any $q \in Q - U(T)$, then $\mathrm{rhs}^+(\rho)$ is finite for every input-consuming rule ρ of T and an xbot$^{-\epsilon}$ equivalent with T can be constructed.

Proof. Assume that there is no subset δ_ϵ of ϵ-rules in δ repeatedly-producing at any $q \in Q - U(T)$. By the assumption, for any tree t, $q(t) \Rightarrow^+_T q(t')$ implies $t = t'$. Thus, suppose that n_r is the number of rules of T, and then we can say that $q(t) \Rightarrow^*_T q(t')$ if and only if $q(t) \Rightarrow^i_T q'(t')$ for some $i \in [n_r]$ where $\Rightarrow^i_T$ denotes the move relation by i times applications of rules. We just show the only if part by induction on the number n of rules used twice or more in the rewriting sequence of $q(t) \Rightarrow^*_T q(t')$. If $n = 0$, it is clear that $q(t) \Rightarrow^i_T q'(t')$ for some $i \in [n_r]$. If $n > 0$, let r be a rule used twice or more, and then $q''(t''_1)$ and $q''(t''_2)$ be the first and last trees obtained by applying r in the rewriting sequence, that is, $q(t) \Rightarrow^*_T q''(t''_1) \Rightarrow^+_T q''(t''_2) \Rightarrow^*_T q'(t')$. Since $q''(t''_1) \Rightarrow^+_T q''(t''_2)$ implies $t''_1 = t''_2$, we have $q(t) \Rightarrow^*_T q''(t''_1) \Rightarrow^*_T q'(t')$, in which the number of rules used twice or more is at most $n - 1$. From the inductive hypothesis, we can conclude $q_(t) \Rightarrow^i_T q'(t')$ for some $i \in [n_r]$.

For each rule $\rho = (C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)) \in \delta$, $\mathrm{rhs}^+(\rho) = \{q'(t) \mid q(t_r) \Rightarrow^i_T q'(t), q' \in E(T) \cup Q_f, i \in [n_r]\}$. Since $\mathrm{rhs}^+(\rho)$ is finite, we can construct an equivalent xbot$^{-\epsilon}$ $T_e = (Q, \Sigma, \Delta, Q_f, \delta')$ where $\delta' = \bigcup_{l\to r\in\delta^\Sigma}\{l \to r' \mid r' \in \mathrm{rhs}^+(l \to r)\}$ and δ^Σ is the subset of input-consuming rules of δ. □

It can be decided in polynomial time by the following algorithm whether there is a subset of ϵ-rules of $T_{3.1}$ repeatedly-producing at some state.

(i) Construct the weighted graph $G_{rp} = (Q - U(T_{3.1}), E_e)$ from $T_{3.1} = (Q, \Sigma, \Delta, Q_f, \delta)$ where $E_e = \{(q, q') \mid q(x_1) \to q'(t) \in \delta, t \in \bar{\mathcal{T}}_\Delta(\{x_1\})\}$, and the weight of each (q, q') is 1 if there is a rule $q(x_1) \to q'(t)$ such that t includes at least one output symbol, and otherwise 0.

(ii) Find a cycle whose weight is at least one. If such a cycle exists, answer that T_3 is not single-valued and halt.

If there is no such subset of ϵ-rules, according to the proof of Lemma 6, we can construct a reduced xbot$^{-\epsilon}$ $T_{3.2}$ equivalent with $T_{3.1}$ in polynomial time.

(3) Step 3-3: Deciding single-valuedness of xbot$^{-\epsilon}$

In this substep, we prove the decidability of single-valuedness of xbot$^{-\epsilon}$s. The idea of the proof is the same as that of the proof of the decidability of k-valuedness of bottom-up tree transducers (bots) [13]. The proof in [13] uses Engelfriet's property T1(i) to show that for a bot T, there exists an input tree t of height polynomial in $|T|$ such that $|[\![T]\!](t)| \geq 2$ if and only if T is not single-valued. We use a variant of the property (Lemma 7) to prove the decidability of single-valuedness of xbot$^{-\epsilon}$s.

We give some notations for the property. Let $\mathcal{T}_\Sigma[X_n] = \bar{\mathcal{T}}_\Sigma(X_n) \cup \mathcal{T}_\Sigma$, that is, every $t \in \mathcal{T}_\Sigma[X_n]$ has all the variables in X_n or has no variable. For $t, s \in \mathcal{T}_\Sigma[X_n]$, ts is the tree

obtained from t by replacing each variable with s. Note that $ts = t$ if t has no variable. For $m \in [n]$, let $\mathcal{T}_\Sigma^{n,m}[X_n] = \mathcal{T}_\Sigma^{m-1} \times \mathcal{T}_\Sigma[X_n] \times \mathcal{T}_\Sigma^{n-m}$. For $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, we denote by $t^{(i)}$ the ith element of $\mathbf{t}$, i.e., $\mathbf{t} = (t^{(1)}, \ldots, t^{(n)})$. For $s \in \mathcal{T}_\Sigma[X_n]$ and $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $s\mathbf{t}$ is the tree obtained from s by replacing x_i with $t^{(i)}$ for all $i \in [n]$. Let $\mathbf{tu} = (t^{(1)}\mathbf{u}, \ldots, t^{(n)}\mathbf{u})$ for $\mathbf{u} \in \mathcal{T}_\Sigma^{n,m}[X_n]$. Notice that since $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, so is $\mathbf{tu}$. For $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5 \in \mathcal{T}_\Sigma^{n,m}[X_n]$ and $S = \{i_1, \ldots, i_{|S|}\} \subseteq [1,5]$, let $\mathbf{t}_S = \mathbf{t}_{i_1} \cdots \mathbf{t}_{i_{|S|}}$ where $i_j < i_{j+1}$ for $j \in [|S|-1]$.

Now, we give an extension of Engelfriet's property T1(i) (see Appendix for the proof).

Lemma 7: Let n, n' be arbitrary positive integers, and $m \in [n], m' \in [n']$. Suppose that $t_0 \in \mathcal{T}_\Sigma[X_n]$, $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5 \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $t'_0 \in \mathcal{T}_\Sigma[X_{n'}]$, $\mathbf{t}'_1, \mathbf{t}'_2, \mathbf{t}'_3, \mathbf{t}'_4, \mathbf{t}'_5 \in \mathcal{T}_\Sigma^{n',m'}[X_{n'}]$. If $t_0\mathbf{t}_S = t'_0\mathbf{t}'_S$ for every S such that $\{5\} \subseteq S \subset [1,5]$, then $t_0\mathbf{t}_{[1,5]} = t'_0\mathbf{t}'_{[1,5]}$.

Before we use Lemma 7, we decompose the left-hand side of each rule of $T_{3.2}$ into several rules each of which has only one input symbol. Actually, we construct a multi bottom-up tree transducer (mbot) equivalent with a given xbot$^{-\epsilon}$. We follow the construction in [9, Lemma 14] . We first give an example of the decomposition of xbot$^{-\epsilon}$ rules. The decomposition of a rule is done by inserting new states for each symbol appearing in the left-hand side of the rule.

Example 3: Assume that an xbot$^{-\epsilon}$ T has the following transduction rule

$$\rho = a(b(q_1(x_1), q_2(x_2), q_3(x_3)), q_4(x_4)) \to q(c(x_1, x_2, x_4)).$$

Then, the mbot T_a obtained by decomposing T contains the following rules

$$b(q_1(x_1), q_2(x_2), q_3(x_3)) \to q_1^\rho(x_1, x_2) \text{ and}$$
$$a(q_1^\rho(x_1, x_2), q_4(x_4)) \to q(c(x_1, x_2, x_4))$$

(see Fig. 3). Note that q_1^ρ maintains x_1 and x_2 but not x_3 because x_3 does not occur in the right-hand side of ρ.

Formally, the translation of an xbot$^{-\epsilon}$ into an mbot is as follows: Given an xbot$^{-\epsilon}$ $T = (Q, \Sigma, \Delta, Q_f, \delta)$, construct an mbot $T_a = (Q \cup Q_m, \Sigma, \Delta, Q_f, \delta_a)$ equivalent with T where

- $Q_m = \{q_p^\rho \mid \rho = (C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)) \in \delta,\ p \in \mathrm{pos}_\Sigma(C_l) - \{\epsilon\}\}$,
- δ_a is the smallest set satisfying the following condition. For each $\rho = (C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r)) \in \delta$, let
  - $V_p = \{x_i\}$ and $q_p^\rho = q_i$ if $p \in \mathrm{pos}_{x_i}(C_l)$ for some x_i and
  - $V_p = \mathrm{var}(C_l|_p) \cap \mathrm{var}(t_r)$ and q_p^ρ be a new state if $p \in \mathrm{pos}_\Sigma(C_l) - \{\epsilon\}$,

  and $\mathbf{x}_p$ be the sequence of all the variables in V_p in index order, and
  - $\sigma_\epsilon(q_1^\rho(\mathbf{x}_1), \ldots, q_{k_\epsilon}^\rho(\mathbf{x}_{k_\epsilon})) \to q(t_r) \in \delta_a$ where $\sigma_\epsilon = \lambda_{C_l}(\epsilon)$ and $k_\epsilon = \mathsf{rk}(\sigma_\epsilon)$, which is called the *top rule* of ρ,
  - for each position $p \in \mathrm{pos}_\Sigma(C_l) - \{\epsilon\}$, let $\theta_n : V_p \to X_{|V_p|}$ such that $C_l|_p\theta_n$ is normalized, then the rule obtained by applying θ_n to the both sides of $\sigma_p(q_{p1}^\rho(\mathbf{x}_{p1}), \ldots, q_{pk_p}^\rho(\mathbf{x}_{pk_p})) \to q_p^\rho(\mathbf{x}_p)$ belongs to δ_a, where $\sigma_p = \lambda_{C_l}(p)$ and $k_p = \mathsf{rk}(\sigma_p)$.

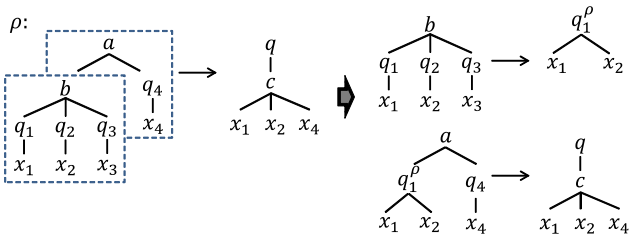


Fig. 3 An example of translating an xbot$^{-\epsilon}$ into an mbot

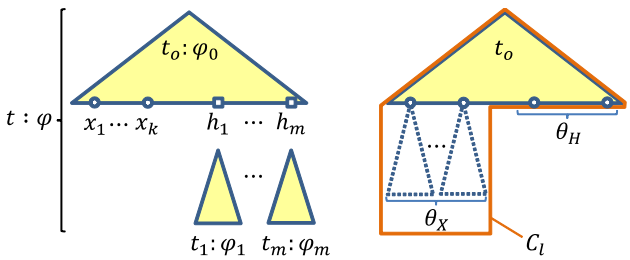


Fig. 4 Decomposition of an x-partial computation φ for t into φ_0 and $\varphi_1, \ldots, \varphi_m$

We call the above translation the *rule-decomposition*. The rule-decomposition can be done in linear time. Let $\delta^T = \{\rho \in \delta_a \mid \rho \text{ is the top rule of a rule in } \delta\}$. A $(q, q_1 \cdots q_k)$-computation φ of T_a is *x-partial* if $\lambda_\varphi(\epsilon) \in \delta^T$. Note that every accepting computation of T_a is x-partial. If φ is x-partial, $T_a(\varphi)$ is a tree by construction of T_a.

We give two facts on the mbot obtained from an xbot$^{-\epsilon}$ by the rule-decomposition, and prove that the single-valuedness of the mbot is in coNP by using the facts and Lemma 7. Let H_m be a set of m variables $h_1, \ldots, h_m$ disjoint with X and Y. Recall that Y is the set of special variables appearing in the output produced by a computation of mbots (see the definition of the output of mbots in Sect. 2.2). We denote $t[h_1 \leftarrow t_1, \ldots, h_m \leftarrow t_m]$ by $t[t_1, \ldots, t_m]_H$.

Proposition 2: Let $T_a = (Q_a, \Sigma, \Delta, Q_f, \delta_a)$ be the mbot obtained from an xbot$^{-\epsilon}$ $T = (Q, \Sigma, \Delta, Q_f, \delta)$ by the rule-decomposition, and let $t \in \mathcal{T}_\Sigma(X_k)$. If there is an x-partial $(q, q_1 \cdots q_k)$-computation φ of T_a for t, then there exist $t_0 \in \mathcal{T}_\Sigma(X_k \cup H_m)$ and subtrees $t_1, \ldots, t_m$ of t satisfying the following conditions (see Fig. 4).

- $t = t_0[t_1, \ldots, t_m]_H$.
- There exist $\varphi_0 \in \mathcal{T}_{\delta_a}(X_k \cup H_m)$ and x-partial computations $\varphi_1, \ldots, \varphi_m$ for $t_1, \ldots, t_m$ such that $\varphi = \varphi_0[\varphi_1, \ldots, \varphi_m]_H$.
- There exist mappings $\theta_X : X_k \to \mathcal{T}_\Sigma(X_m)$ and $\theta_H : H_m \to X_m$ satisfying $C_l = (t_0\theta_X)\theta_H$ where $\lambda_{\varphi_0}(\epsilon)$ is the top rule of $\rho = (C_l[q'_1(x_1), \ldots, q'_m(x_m)] \to q(t_r)) \in \delta$.

We call φ_0 the *x-initial* computation of φ, and θ_H the *x-fitting mapping* of φ_0. By letting $T_a(h_i) = h_i$ for each $h_i \in H_m$, we

have that $T_a(\varphi) = T_a(\varphi_0)[T_a(\varphi_1), \ldots, T_a(\varphi_m)]_H$. Note that $T_a(\varphi_0)$ and t_r are identical except for variable positions, i.e., $T_a(\varphi_0)\theta_V = t_r\theta_V$ where θ_V maps every $z \in X \cup Y \cup H$ to some unique variable[†].

Lemma 8: Let $T_a = (Q_a, \Sigma, \Delta, Q_f, \delta_a)$ be the mbot obtained from an xbot$^{-\epsilon}$ $T = (Q, \Sigma, \Delta, Q_f, \delta)$ by the rule-decomposition. For every $q \in Q_a$ and $C \in \bar{C}_\Sigma(\{x_*\})$, if $C[q(x_1, \ldots, x_n)] \Rightarrow^+_{T_a} q(t_1, \ldots, t_n)$, then $(t_1, \ldots, t_n) \in \mathcal{T}^{n,m}_\Delta[X_n]$ for some $m \in [n]$.

Proof. Assume that $q \in Q$. Since every state in Q has rank one, for any $C \in \bar{C}_\Sigma(\{x_*\})$, if $C[q(x_*)] \Rightarrow^*_{T_a} q(t)$, then $t \in \mathcal{T}_\Delta(\{x_*\}) = \mathcal{T}^{1,1}_\Delta$.

Assume that $q \in Q_{\mathrm{m}}$ and C is an arbitrary tree in $\bar{C}_\Sigma(\{x_*\})$ such that $C[q(x_1, \ldots, x_n)] \Rightarrow^+_{T_a} q(t_1, \ldots, t_n)$. From the decomposition procedure, there exists a rule $\rho = (C_l[q_1(x_1), \ldots, q_n(x_n)] \to q_\rho(t_r)) \in \delta$ of T such that $q = q^\rho_p$ for some position p in $\mathrm{pos}_\Sigma(C_l) - \{\epsilon\}$. Thus, $C[q(x_1, \ldots, x_n)] \Rightarrow^*_\rho C'[q_\rho(t'')] \Rightarrow^*_{T_a} q(t_1, \ldots, t_n)$ for some $C' \in \bar{C}_\Sigma(\{x_*\})$ and $t'' \in \bar{\mathcal{T}}_\Delta(X_n)$, where $\Rightarrow^*_\rho$ means zero or more applications of only the rules obtained by decomposing ρ. Note that t'' must contain all the variables $x_1, \ldots, x_n$. In the above derivation $C'[q_\rho(t'')] \Rightarrow^*_{T_a} q(t_1, \ldots, t_n)$, t'' is either abandoned or contained as a subtree in t_m for some $m \in [n]$. Thus, $(t_1, \ldots, t_n) \in \mathcal{T}^{n,m}_\Delta$ for some $m \in [n]$. □

Henceforth, we denote $q(t_1, \ldots, t_n)$ by $q(\mathbf{t})$ where $\mathbf{t} = (t_1, \ldots, t_n)$. The following lemma is proved in a similar way to the proof of Theorem 2.2(i) in the Ref. [13].

Lemma 9: Let $T_a = (Q_a, \Sigma, \Delta, Q_f, \delta_a)$ be the mbot obtained from an xbot$^{-\epsilon}$ T by the rule-decomposition. Assume that the maximum rank of states is N.

(1) T_a is not single-valued if and only if there is a tree t of depth less than $5(|Q_a|N)^2$ such that $|[\![T_a]\!](t)| > 1$.

(2) It can be decided in non-deterministic polynomial time whether T_a is not single-valued.

Proof. (1) The if part is trivial and so we prove the only if part. Assume that $t \in \mathcal{T}_\Sigma$ is a tree of minimal size such that there are two distinct derivations $t \Rightarrow^*_{T_a} q_{f1}(t_{o1})$ and $t \Rightarrow^*_{T_a} q_{f2}(t_{o2})$ where $q_{f1}, q_{f2} \in Q_f$, and $t_{o1} \neq t_{o2}$. For a contradiction, assume that the depth of t is greater than or equal to $5(|Q_a|N)^2$. Consider a path of length greater than or equal to $5(|Q_a|N)^2$. From Lemma 8 and the pigeonhole principle, we can choose

- five different positions $p_1, \ldots, p_5$ of t over the path, where $p_1 \prec \cdots \prec p_5$ and
- two pairs (q_1, m_1) and (q_2, m_2) in $Q_a \times [N]$ where $m_1 \in [\mathsf{rk}(q_1)]$ and $m_2 \in [\mathsf{rk}(q_2)]$

satisfying the following condition: Let $C_i = t[x_*]_{p_i}$ for $i \in [0, 4]$ and $C_5 = t|_{p_5}$, and for each $j \in \{1, 2\}$, there exist $\mathbf{t}^j_0, \ldots, \mathbf{t}^j_5 \in \mathcal{T}^{\mathsf{rk}(q_j),m_j}_\Delta[X_{\mathsf{rk}(q_j)}]$ such that

$$\begin{aligned} t = C_0C_1C_2C_3C_4C_5 &\Rightarrow^*_{T_a} C_0C_{[1,4]}[q_j(\mathbf{t}^j_5)] \\ &\Rightarrow^*_{T_a} C_0C_{[1,3]}[q_j(\mathbf{t}^j_{[4,5]})] \\ &\Rightarrow^*_{T_a} \cdots \\ &\Rightarrow^*_{T_a} C_0[q_j(\mathbf{t}^j_{[1,5]})] \\ &\Rightarrow^*_{T_a} q_{fj}(t^j_0\mathbf{t}^j_{[1,5]}) = q_{fj}(t_{oj}). \end{aligned}$$

By the minimality of t, we have $t^1_0\mathbf{t}^1_S = t^2_0\mathbf{t}^2_S \in [\![T_a]\!](C_0C_S)$ for every S such that $\{5\} \subseteq S \subset [1, 5]$. From Lemma 7, $t_{o1} = t^1_0\mathbf{t}^1_{[1,5]} = t^2_0\mathbf{t}^2_{[1,5]} = t_{o2}$. This is a contradiction.
(2) As the proof for ordinary bots in [13], if T_a is not single-valued, we can find paths of two distinct computations of T_a for some input tree that produce distinct paths of output trees leading to the same position. Let $B(\Gamma) := \{(a, j) \mid a \in \Gamma, 0 \leq j \leq \mathsf{rk}(a)\}$. Let $z' \in X \cup Y \cup H$. Let $b_0(z') = \emptyset$. For $z \in X \cup Y \cup H$, let $b_z(z') = \{\varepsilon\}$ if $z = z'$ and $b_z(z') = \emptyset$ otherwise. If $s = a(s_1, \ldots, s_m)$ then let

$$b_0(s) = \{(a, 0)\} \cup \bigcup_{i=1}^{m} (a, i) \cdot b_0(s_i)$$

and for $z \in X \cup Y \cup H$, let

$$b_z(s) = \bigcup_{i=1}^{m} (a, i) \cdot b_z(s_i).$$

For example, for $s = a(b(x_1, c), x_2)$, $b_0(s) = \{(a, 0), (a, 1)(b, 0), (a, 1)(b, 2)(c, 0)\}$ and $b_{x_1}(s) = \{(a, 1)(b, 1)\}$. For $w = (a_1, j_1) \cdots (a_r, j_r)(a, 0) \in B(\Gamma)^*$, we say that path w of s leads to position $p_r = j_1 \cdots j_r$. We denote $j_1 \cdots j_r$ by $o(w)$. And, for $w = (a_1, j_1) \cdots (a_r, j_r)$ where $j_r \neq 0$, we denote $j_1 \cdots j_r$ by $os(w)$.

Next, we define the set of paths of the output tree produced by paths of an x-partial computation φ of T_a. For $\rho \in \delta_a$, let $\mathsf{rk}(\rho)$ be the number of states appearing in the left-hand side of ρ, and let $BS(\delta_a) = B(\delta^T) \cdot B(\delta_a - \delta^T)^*$. Remember that δ^T is the set of top rules. For $\pi \in BS(\delta_a)^+$, let $hdr(\pi) = \rho$ when the head of π is (ρ, j_1). Note that any $\pi \in b_0(\varphi)$ is in $BS(\delta_a)^+$. For $\pi \in BS(\delta_a)^+$, we define $\Pi_{T_a}(\pi)$ as follows.

- If $\pi \in BS(\delta_a)$, $hdr(\pi)$ is the top rule of $C_l[q'_1(x_1), \ldots, q'_k(x_m)] \to q(t_r)$, and $o(\pi) \in \mathrm{pos}_\Sigma(C_l)$, then let
$$\Pi_{T_a}(\pi) = b_0(t_r).$$
- If $\pi = \pi' \cdot \pi''$ where $\pi' \in BS(\delta_a)$, $\pi'' \in BS(\delta_a)^+$, $hdr(\pi)$ is the top rule of $C_l[q'_1(x_1), \ldots, q'_k(x_m)] \to q(t_r)$, and $C_l|_{os(\pi')} = x_i$, then let
$$\Pi_{T_a}(\pi) = b_{x_i}(t_r) \cdot \Pi_{T_a}(\pi'').$$

Then, for any x-partial computation φ of T_a, we have

$$b_0(T_a(\varphi)) = \bigcup_{\pi \in b_0(\varphi)} \Pi_{T_a}(\pi)$$

[†]The counterparts of $T_a(\varphi_0)$ and t_r before transformation T_a are t_0 and C_l, which are not the same in general, because t_0 lacks $\theta_X(x_i)$ below x_i ($i \in [k]$). However, $\lambda_{\varphi_0}(\epsilon)$ is the top rule of ρ. By construction of T_a, the right-hand side of $\lambda_{\varphi_0}(\epsilon)$ contains t_r. Hence $T_a(\varphi_0)\theta_V = t_r\theta_V$ holds for some θ_V.

Now, we give a proof sketch for the above equation. Assume that $\varphi = x_i$. Then we have $b_0(T_a(\varphi)) = b_0(\varphi) = \emptyset$. Next, assume that φ is an x-partial computation which is not a variable. By Proposition 2, we can write φ as $\varphi = \varphi_0[\varphi_1, \ldots, \varphi_m]_H$ where φ_0 is the x-initial computation of φ with its fitting mapping θ_H, and $\lambda_\varphi(\epsilon)$ is the top rule of $C_l[q'_1(x_1), \ldots, q'_m(x_m)] \to q(t_r) \in \delta$. We have

$$\begin{aligned} b_0(T_a(\varphi)) &= b_0(T_a(\varphi_0)[T_a(\varphi_1), \ldots, T_a(\varphi_m)]_H) \\ &= b_0(t_r) \cup \bigcup_{i=1}^{m} b_{\theta_H(h_i)}(t_r) \cdot b_0(T_a(\varphi_i)) \\ &= \bigcup_{\pi \in b_0(\varphi_0)} \Pi_{T_a}(\pi) \cup \bigcup_{\pi \in b_0(\varphi) - b_0(\varphi_0)} \Pi_{T_a}(\pi) \\ &= \bigcup_{\pi \in b_0(\varphi)} \Pi_{T_a}(\pi). \end{aligned}$$

We give a necessary and sufficient condition for T_a to be not single-valued, which can be decided in nondeterministic polynomial time:

$T_a = (Q_a, \Sigma, \Delta, Q_f, \delta_a)$ is not single-valued if and only if T_a satisfies the following property: Let L and N be the maximum ranks of symbols in Σ and states in Q_a, respectively. Let $U = 5(|Q_a|N)^2$. Then, there exist $k \leq 2LU$, $t \in \bar{\mathcal{T}}(X_k)$, and $(q^i, q^i_1 \cdots q^i_k)$-computations φ_i $(i = 1, 2)$ of T_a for t such that the followings hold:

1. $|t| \leq 2LU$.
2. $q^1, q^2 \in Q_f$.
3. $\mathrm{dom}(T(q^1_j)) \cap \mathrm{dom}(T(q^2_j)) \neq \emptyset$ for $j \in [k]$.
4. There exist paths $\pi_i \in b_0(\varphi_i)$ and $w_i \in \Pi_{T_a}(\pi_i)$ $(i = 1, 2)$ such that w_1 and w_2 lead to the same position but $w_1 \neq w_2$. □

Theorem 1: Deciding single-valuedness of xbot$^{+\mathrm{g(B(R))}}$s is in coNP.

It is still open whether single-valuedness is decidable in polynomial time for xbots.

Theorem 2: Determinacy for (*sl*-xbots, *s*-bots) is in coNEXPTIME.

Proof. Our decision algorithm takes polynomial time at Step 1, and exponential time at Step 2. Since the size of the transducer T_3 constructed at Step 2 is at most exponential in the input size, from Theorem 1, single-valuedness of T_3 at Step 3 can be decided in co-nondeterministic exponential time. Thus, determinacy for (*sl*-xbots, *s*-bots) is in coNEXPTIME. □

If the left-hand side of every rule of a given *sl*-xbot T_1 contains at most M_1 symbols which are neither states nor variables for some constant M_1, the right-hand side of every rule of the inverse transducer T_1^{inv} also contains at most M_1. Then, the size of the transducer T_3 is polynomial as stated in Sect. 3.3, and thus determinacy for such class is in coNP.

4. Decidability Result on Subsumption

We show that subsumption is decidable for (*sl*-xbots, *s*-bots). As shown in Sect. 3, given an *sl*-xbot T_1 and an *s*-bot T_2, if T_1 determines T_2, we can construct a reduced *s*-xbot$^{-\epsilon}$ T_3 such that $[\![T_3]\!] = [\![T_2]\!] \circ [\![T_1]\!]^{-1}$. So, in order to decide subsumption, it suffices to check whether each rule of the reduced *s*-xbot$^{-\epsilon}$ T_3 can be decomposed into non-extended rules.

Roughly speaking, this check is accomplished by examining the 'relative positions' among variables. For example, consider the following extended rule of a transducer T:

$$a(b(q_1(x_1), q_2(x_2)), q_3(x_3)) \to q(c(x_1, d(x_2, x_3))).$$

such that $\mathrm{rng}(T(q_j))$ is infinite for all $j \in [3]$. This rule cannot be decomposed into non-extended ones because the relative positions among variables are different in the both side of the rule. More precisely, the left-hand side has a subtree with x_1 and x_2 but without x_3, while the right-hand side does not have such subtrees. For an extended rule to be decomposed, the right-hand side must preserve the relative positions among variables in the left-hand side. The next lemma precisely states the notion of 'relative positions' and shows that it is a necessary and sufficient condition for an *s*-xbot$^{-\epsilon}$ to have an equivalent *s*-bot.

Lemma 10: Let $T = (Q, \Sigma, \Delta, Q_f, \delta)$ be a reduced *s*-xbot$^{-\epsilon}$. An *s*-bot equivalent with T can be constructed if and only if T satisfies the following condition (X): for every rule $C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r) \in \delta$ and any three variables $x_{i_1}, x_{i_2}, x_{i_3} \in \mathrm{var}(t_r)$, if

(X1) $\mathrm{rng}(T(q_{i_j}))$ is infinite for all $j \in [3]$, and

(X2) $\mathrm{nca}(p_1, p_2) \succ \mathrm{nca}(p_1, p_3)$ where $\{p_j\} = \mathrm{pos}_{x_{i_j}}(C_l)$ for $j \in [3]$, then

(X3) the minimal suffix $t_s \in \mathcal{T}_\Sigma(X_k)$ such that $t_r = t_p t_s$ for some $t_p \in \bar{\mathcal{T}}_{\Sigma \cup X_k - \{x_{i_1}, x_{i_2}\}}(\{x_*\})$ does not contain x_{i_3}.

Proof Sketch. Assume (X) does not hold and we can construct an *s*-bot T' equivalent with a given *s*-xbot$^{-\epsilon}$ T. Since (X) does not hold, there is a rule $C_l[q_1(x_1), \ldots, q_k(x_k)] \to q(t_r) \in \delta$ and $x_{i_1}, x_{i_2}, x_{i_3} \in \mathrm{var}(t_r)$ such that (X1) and (X2) hold but (X3) does not. Let $p_{12} = \mathrm{nca}(p_1, p_2)$ in (X2), and t_s be the minimal suffix of t_r in (X3). Since T' is an *s*-bot equivalent with T, T' must have rules of which left-hand sides 'cover' the subtree $C_l|_{p_{12}}$, which contains x_{i_1} and x_{i_2} and does not contain x_{i_3}. Also, since $C_l[x_*]_{p_{12}}$ does not contain x_{i_1} and x_{i_2}, some suffix t'_s of t_r in the right-hand side such that $t_r = t'_p t'_s$ for some $t'_p \in \bar{\mathcal{T}}_{\Sigma \cup X_k - \{x_{i_1}, x_{i_2}\}}(\{x_*\})$ should be generated by T' corresponding to $C_l|_{p_{12}}$. However, the minimal suffix t_s contains x_{i_3}, and thus so does t'_s. That is, t'_s including x_{i_3} will inevitably be generated from $C_l|_{p_{12}}$ without x_{i_3}, which leads to a contradiction. Conversely, if (X) holds, we can divide

each rule of T into non-extended rules, each of which has exactly one symbol in the left-hand side.

□

For any xbot T, it can be decided in polynomial time whether $\mathrm{rng}(T)$ is infinite. Thus, it is decidable in polynomial time whether there is an s-bot equivalent with a given s-xbot$^{-\epsilon}$. If such an s-bot exists, it can be constructed in doubly-exponential time in the worst case.

Theorem 3: Subsumption for (sl-xbots, s-bots) is in coNEXPTIME. If an sl-xbot T_1 subsumes an s-bot T_2, an s-bot T_3 such that $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$ can be constructed.

5. Conclusion

We have shown that determinacy and subsumption are decidable in coNEXPTIME for single-valued linear extended bottom-up tree transducers as the determiner class and single-valued bottom-up tree transducers as the determinee class.

As future work, we will investigate whether subsumption for more powerful determiner classes, such as deterministic top-down tree transducers (with regular look-ahead) and deterministic MSO definable tree transducers, is decidable or not. Though determinacy is undecidable for such classes, decidability of subsumption is still open. We also consider whether, given two transducers T_1 and T_2 in the classes such that T_1 subsumes T_2, a transducer T_3 satisfying $[\![T_2]\!] = [\![T_3]\!] \circ [\![T_1]\!]$ can be effectively constructed or not.

References

[1] K. Hashimoto, R. Sawada, Y. Ishihara, H. Seki, and T. Fujiwara, "Determinacy and subsumption for single-valued bottom-up tree transducers," Proc. 7th International Conference on Language and Automata Theory and Applications, Lecture Notes in Computer Science, vol.7810, pp.335–346, 2013.

[2] L. Zheng and H. Chen, "Determinacy and rewriting of conjunctive queries over unary database schemas," Proc. 2011 ACM Symposium on Applied Computing, pp.1039–1044, 2011.

[3] F.N. Afrati, "Determinacy and query rewriting for conjunctive queries and views," Theoretical Computer Science, vol.412, no.11, pp.1005–1021, 2011.

[4] W. Fan, F. Geerts, and L. Zheng, "View determinacy for preserving selected information in data transformations," Information Systems, vol.37, no.1, pp.1–12, 2012.

[5] K. Hashimoto, R. Sawada, Y. Ishihara, H. Seki, and T. Fujiwara, "Determinacy and subsumption for single-valued bottom-up tree transducers," Available at http://www.trs.cm.is.nagoya-u.ac.jp/~k-hasimt/paper/201503.pdf, 2015.

[6] M. Benedikt, J. Engelfriet, and S. Maneth, "Determinacy and rewriting of top-down and MSO tree transformations," Proceedings of 38th International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, vol.8087, pp.146–158, Springer Berlin Heidelberg, 2013.

[7] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi, "Tree automata techniques and applications," http://tata.gforge.inria.fr/, 2007.

[8] A. Lemay, S. Maneth, and J. Niehren, "A learning algorithm for top-down XML transformations," Proc. 29th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp.285–296, 2010.

[9] J. Engelfriet, E. Lilin, and A. Maletti, "Extended multi bottom-up tree transducers," Acta Informatica, vol.46, no.8, pp.561–590, 2009.

[10] H. Seidl, "Single-valuedness of tree transducers is decidable in polynomial time," Theoretical Computer Science, vol.106, no.1, pp.135–181, 1992.

[11] B.S. Baker, "Composition of top-down and bottom-up tree transductions," Information and Control, vol.41, no.2, pp.186–213, 1979.

[12] J. Engelfriet, "Bottom-up and top-down tree transformations – a comparison," Mathematical Systems Theory, vol.9, no.3, pp.198–231, 1975.

[13] H. Seidl, "Equivalence of finite-valued tree transducers is decidable," Theory of Computing Systems, vol.27, no.4, pp.285–346, 1994.

Appendix: Proof of Lemma 7

We first recall the notations. Let $\mathcal{T}_\Sigma[X_n] = \bar{\mathcal{T}}_\Sigma(X_n) \cup \mathcal{T}_\Sigma$, that is, every $t \in \mathcal{T}_\Sigma[X_n]$ has all the variables in X_n or has no variable. For $t, s \in \mathcal{T}_\Sigma[X_n]$, ts is the tree obtained from t by replacing each variable with s. Note that $ts = t$ if t has no variable. For $m \in [n]$, let $\mathcal{T}_\Sigma^{n,m}[X_n] = \mathcal{T}_\Sigma^{m-1} \times \mathcal{T}_\Sigma[X_n] \times \mathcal{T}_\Sigma^{n-m}$. For $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, we denote by $t^{(i)}$ the ith element of $\mathbf{t}$, i.e., $\mathbf{t} = (t^{(1)}, \ldots, t^{(n)})$. For $s \in \mathcal{T}_\Sigma[X_n]$ and $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $s\mathbf{t}$ is the tree obtained from s by replacing x_i with $t^{(i)}$ for all $i \in [n]$. Let $\mathbf{t}s = (t^{(1)}s, \ldots, t^{(n)}s)$, and $\mathbf{tu} = (t^{(1)}\mathbf{u}, \ldots, t^{(n)}\mathbf{u})$ for $\mathbf{u} \in \mathcal{T}_\Sigma^{n,m}[X_n]$. Notice that since $\mathbf{t} \in \mathcal{T}_\Sigma^{n,m}[X_n]$, so are $\mathbf{t}s$ and $\mathbf{tu}$. Lastly, the above substitution operation is associative. That is, for $s \in \mathcal{T}_\Sigma[X_n]$ and $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3 \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $(s\mathbf{t}_1)\mathbf{t}_2 = s(\mathbf{t}_1\mathbf{t}_2)$ and $(\mathbf{t}_1\mathbf{t}_2)\mathbf{t}_3 = \mathbf{t}_1(\mathbf{t}_2\mathbf{t}_3)$ hold. For $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5 \in \mathcal{T}_\Sigma^{n,m}[X_n]$ and $S = \{i_1, \ldots, i_{|S|}\} \subseteq [1, 5]$, let $\mathbf{t}_S = \mathbf{t}_{i_1} \cdots \mathbf{t}_{i_{|S|}}$ where $i_j < i_{j+1}$ for $j \in [|S| - 1]$.

We use some propositions to prove Lemma 7. The following is the same as the top cancellation in [13] except that we use tuples of trees in $\mathcal{T}_\Sigma^{n,m}[X_n]$.

Proposition 3 (Top Cancellation): Let $s \in \mathcal{T}_\Sigma[X_n]$ and $\mathbf{t}_1, \mathbf{t}_2 \in \mathcal{T}_\Sigma^{n,m}[X_n]$. If $s\mathbf{t}_1 = s\mathbf{t}_2$, then $s \in \mathcal{T}_\Sigma$ or $\mathbf{t}_1 = \mathbf{t}_2$.

Proof. Assume that $s\mathbf{t}_1 = s\mathbf{t}_2$, $s \notin \mathcal{T}_\Sigma$ and $\mathbf{t}_1 \neq \mathbf{t}_2$. Then, s contains all the variables in X_n, and $t_1^{(i)} \neq t_2^{(i)}$ for some $i \in [n]$. Since distinct trees are substituted to x_i, $s\mathbf{t}_1 \neq s\mathbf{t}_2$. This is a contradiction. □

The decidability of single-valuedness (or more generally, k-valuedness) of bots was proved in [13] by using Engelfriet's Property T1(i), which was proved by Engelfriet's Property T2(i).

Proposition 4 (Engelfriet's Property T1(i) [13]): Assume $t_i, t_i' \in \mathcal{T}_\Sigma(\{x_*\})$, $i = 0, 1, 2, 3, 4$. Then,

$$t_0 \cdots t_{i-1} t_j \cdots t_4 = t_0' \cdots t_{i-1}' t_j' \cdots t_4' \text{ for all } 0 < i < j \leq 4$$

implies $t_0 t_1 t_2 t_3 t_4 = t_0' t_1' t_2' t_3' t_4'$.

Proposition 5 (Engelfriet's Property T2(i) [13]): Assume $s_i, u_i, v_i, y_i, z_i \in \mathcal{T}_\Sigma(\{x_*\})$ $(i = 1, 2)$, s_1 or s_2 contains x_*, $y_1 \neq z_1$, and $y_2 \neq z_2$. Then,

$$\begin{array}{l} s_1y_1 = s_2y_2 \\ s_1z_1 = s_2z_2 \\ s_1v_1 = s_2v_2 \\ u_1y_1 = u_2y_2 \\ u_1z_1 = u_2z_2 \end{array} \quad \text{implies} \quad u_1v_1 = u_2v_2.$$

Now, we prove Lemma 7, a variant of Engelfriet's Property T1(i). For $\mathbf{t} = (t^{(1)}, \ldots, t^{(n)}) \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $\mathbf{t}^c$ denotes the n-tuple of trees obtained from $\mathbf{t}$ by replacing the mth element with x_*, that is, $\mathbf{t}^c = (t^{(1)}, \ldots, t^{(m-1)}, x_*, t^{(m+1)}, \ldots, t^{(n)})$. Hence, we have $\mathbf{t} = \mathbf{t}^c t^{(m)}$.

Lemma 7: Let n, n' be arbitrary positive integers, and $m \in [n], m' \in [n']$. Suppose that $t_0 \in \mathcal{T}_\Sigma[X_n]$, $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5 \in \mathcal{T}_\Sigma^{n,m}[X_n]$, $t'_0 \in \mathcal{T}_\Sigma[X_{n'}]$, $\mathbf{t}'_1, \mathbf{t}'_2, \mathbf{t}'_3, \mathbf{t}'_4, \mathbf{t}'_5 \in \mathcal{T}_\Sigma^{n',m'}[X_{n'}]$. Then, if $t_0\mathbf{t}_S = t'_0\mathbf{t}'_S$ for every S such that $\{5\} \subseteq S \subset [1,5]$, then $t_0\mathbf{t}_{[1,5]} = t'_0\mathbf{t}'_{[1,5]}$.

Proof. Assume that $t_0\mathbf{t}_S = t'_0\mathbf{t}'_S$ for every S such that $\{5\} \subseteq S \subset [1,5]$. Let

$$\begin{array}{ll} s_1 = t_0\mathbf{t}_2^c, & s_2 = t'_0\mathbf{t}_2'^c, \\ u_1 = t_0\mathbf{t}_1\mathbf{t}_2^c, & u_2 = t'_0\mathbf{t}'_1\mathbf{t}_2'^c, \\ y_1 = t_2^{(m)}\mathbf{t}_5, & y_2 = t_2'^{(m')}\mathbf{t}'_5, \\ z_1 = t_2^{(m)}\mathbf{t}_{[4,5]}, & z_2 = t_2'^{(m')}\mathbf{t}'_{[4,5]}, \\ v_1 = t_2^{(m)}\mathbf{t}_{[3,5]}, & v_2 = t_2'^{(m')}\mathbf{t}'_{[3,5]}. \end{array}$$

By the assumption, we have

$$\begin{aligned} s_1y_1 &= (t_0\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_5) = t_0\mathbf{t}_2\mathbf{t}_5 = t'_0\mathbf{t}'_2\mathbf{t}'_5 \\ &= (t'_0\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_5) = s_2y_2, \\ s_1z_1 &= (t_0\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_{[4,5]}) = t_0\mathbf{t}_2\mathbf{t}_{[4,5]} = t'_0\mathbf{t}'_2\mathbf{t}'_{[4,5]} \\ &= (t'_0\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_{[4,5]}) = s_2z_2, \\ s_1v_1 &= (t_0\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_{[3,5]}) = t_0\mathbf{t}_{[2,5]} = t'_0\mathbf{t}'_{[2,5]} \\ &= (t'_0\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_{[3,5]}) = s_2v_2, \\ u_1y_1 &= (t_0\mathbf{t}_1\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_5) = t_0\mathbf{t}_1\mathbf{t}_2\mathbf{t}_5 = t'_0\mathbf{t}'_1\mathbf{t}'_2\mathbf{t}'_5 \\ &= (t'_0\mathbf{t}'_1\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_5) = u_2y_2, \\ u_1z_1 &= (t_0\mathbf{t}_1\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_{[4,5]}) = t_0\mathbf{t}_{[1,2]}\mathbf{t}_{[4,5]} = t'_0\mathbf{t}'_{[1,2]}\mathbf{t}'_{[4,5]} \\ &= (t'_0\mathbf{t}'_1\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_{[4,5]}) = u_2z_2. \\ u_1v_1 &= (t_0\mathbf{t}_1\mathbf{t}_2^c)(t_2^{(m)}\mathbf{t}_{[3,5]}) = t_0\mathbf{t}_{[1,5]} \\ u_2v_2 &= (t'_0\mathbf{t}'_1\mathbf{t}_2'^c)(t_2'^{(m')}\mathbf{t}'_{[3,5]}) = t'_0\mathbf{t}'_{[1,5]} \end{aligned}$$

There are four cases.
Case (1) Both of s_1 and s_2 contain no variable.
Then, t_0 and t'_0 do not contain x_m and $x_{m'}$, respectively. Moreover, by definition of $\mathcal{T}_\Sigma[X]$, t_0 and t'_0 do not contain any variable. Thus, $t_0\mathbf{t}_{[1,5]} = t_0 = t_0\mathbf{t}_5$ and $t'_0\mathbf{t}'_{[1,5]} = t'_0 = t'_0\mathbf{t}'_5$. Since $t_0\mathbf{t}_5 = t'_0\mathbf{t}'_5$, we have $t_0\mathbf{t}_{[1,5]} = t'_0\mathbf{t}_{[1,5]}$.
Case (2) s_1 or s_2 contains variable x_*, $y_1 \neq z_1$ and $y_2 \neq z_2$.
Proposition 5 implies $t_0\mathbf{t}_{[1,5]} = u_1v_1 = u_2v_2 = t'_0\mathbf{t}_{[1,5]}$.
Case (3) s_1 or s_2 contains variable x_* and $y_1 = z_1$.
Because $y_1 = z_1$, by top cancellation (Proposition 3), $\mathbf{t}_5 = \mathbf{t}_{[4,5]}$ or $t_2^{(m)}$ has no variable. If $\mathbf{t}_5 = \mathbf{t}_{[4,5]}$ then $t_0\mathbf{t}_{[1,3]}\mathbf{t}_5 = t_0\mathbf{t}_{[1,5]}$. If $t_2^{(m)}$ has no variable then we also have $t_0\mathbf{t}_{[1,3]}\mathbf{t}_5 = t_0\mathbf{t}_{[1,2]} = t_0\mathbf{t}_{[1,5]}$. Moreover, we have $s_1y_1 = s_1z_1$ and thus $s_2y_2 = s_2z_2$. By top cancellation, $y_2 = z_2$ or s_2 has no variable. Assume that $y_2 = z_2$. By the same argument as the above, we have $t'_0\mathbf{t}'_{[1,3]}\mathbf{t}'_5 = t'_0\mathbf{t}'_{[1,5]}$. On the other hand, assume that s_2 has no variable. Then, t'_0 has no variable and thus we also have $t'_0\mathbf{t}'_{[1,3]}\mathbf{t}'_5 = t'_0 = t'_0\mathbf{t}'_{[1,5]}$. Therefore, $t_0\mathbf{t}_{[1,5]} = t'_0\mathbf{t}'_{[1,5]}$ because $t_0\mathbf{t}_{[1,3]}\mathbf{t}_5 = t'_0\mathbf{t}'_{[1,3]}\mathbf{t}'_5$.
Case (4) s_1 or s_2 contains variable x_* and $y_2 = z_2$.
This is analogous to Case (3). □

Kenji Hashimoto received the Ph.D. degree in Information and Computer Sciences from Osaka University in 2009. From 2009 to 2013, he was an Assistant Professor of Nara Institute of Science and Technology. Since Oct. 2013, he has been an Assistant Professor of Graduate School of Information Science, Nagoya University. His research interests include formal language, database theory, and information security.

Ryuta Sawada received the B.E. and M.E. degrees in information and computer sciences from Osaka University in 2010 and 2012.

Yasunori Ishihara received the Ph.D. degree in information and computer sciences from Osaka University in 1995. In 1994, he joined the faculty of Nara Institute of Science and Technology. In 1999, he joined the faculty of Osaka University, and since 2002, he has been an Associate Professor of Graduate School of Information Science and Technology, Osaka University. His research interests include database theory and information security.

Hiroyuki Seki received his Ph.D. degree from Osaka University in 1987. He was an Assistant Professor, and later, an Associate Professor from 1987 to 1994. In 1994, he joined Nara Institute of Science and Technology, where he was a Professor during 1996 to 2013. Currently, he is a Professor in Nagoya University. His current research interests include formal language theory and formal approach to software development.

Toru Fujiwara received the B.E., M.E., and Ph.D. degrees in information and computer sciences form Osaka University in 1981, 1983, and 1986, respectively. In 1986, he joined the faculty of Osaka University. Since 1997, he has been a Professor at Osaka University. He is currently with Graduate School of Information Science and Technology. In 1998-2000, he was a Professor at Graduate School of Information Science, Nara Institute of Science and Technology, simultaneously. His current research interests include coding theory and information security.