Streaming Ranked-Tree-to-String Transducers

Yuta Takahashi[1], Kazuyuki Asada[2] and Keisuke Nakano[2]

[1] The University of Electro-Communications takahashi@ipl.cs.uec.ac.jp
[2] Tohoku University {asada,ksk}@riec.tohoku.ac.jp

Abstract. *Streaming tree transducers with single-use restriction* ($\mathrm{STT}_{\mathrm{sur}}$s) were introduced by Alur and D'Antoni as an analyzable, executable, and expressive model for transforming unranked ordered trees in a single pass. The equivalence problem of $\mathrm{STT}_{\mathrm{sur}}$s is decidable because their class is as expressive as the class of MSO-definable tree transformations. In this paper, we present *streaming ranked-tree-to-string transducers* (SRTSTs), based on $\mathrm{STT}_{\mathrm{sur}}$s: SRTSTs are released from the single-use restriction while their input and output are restricted to ranked trees and strings, respectively. We show that the expressiveness of SRTSTs coincides with that of *deterministic top-down tree transducers with regular look-ahead* ($\mathrm{yDT}^{\mathrm{R}}$s), whose equivalence problem is known to be decidable. Our proof is done by constructing equivalent transducers in both directions.

Keywords: ranked trees · streaming transducers · expressiveness · equivalence.

1 Introduction

Streaming tree transducers with single-use restriction ($\mathrm{STT}_{\mathrm{sur}}$s) were introduced in [1] which can characterize MSO-definable tree transformations in a single path. An $\mathrm{STT}_{\mathrm{sur}}$ defines a function over unranked ordered trees (and forests), which are encoded as *nested words*. A nested word is a string over symbols tagged with *open/close* brackets. An $\mathrm{STT}_{\mathrm{sur}}$ reads an input nested word from left to right in a single path. Here each state is equipped with *a visibly pushdown stack* and a finite number of *variables* that store output chunks. An $\mathrm{STT}_{\mathrm{sur}}$ updates variables and a stack that stores *stack symbols* along with updated values of variables.

Intuitively, the single-use restriction is a restriction that variables are updated in a manner that ensures that each value of a variable contributes *at most once* to the eventual output without duplication (see [1] for the definition). Due to this restriction, the class of $\mathrm{STT}_{\mathrm{sur}}$s equals that of MSO-definable tree transducers.

The equivalence problem of transducers is an important topic in formal language theory. One of the remarkable results is that the equivalence of MSO-definable tree transducers is decidable [8]. Furthermore, in [1] the decidability of the equivalence problem for $\mathrm{STT}_{\mathrm{sur}}$s was proved without using the result for MSO-definable transducers, with better complexity.

In this paper, we propose *streaming ranked-tree-to-string transducers* (SRTSTs) as a model of transformations from ranked trees to strings. The definition of

SRTSTs is based on that of $\mathrm{STT}_{\mathrm{sur}}$s but released from the single-use restriction. In addition, the input and output of SRTSTs are restricted to ranked trees and strings, respectively, while those of $\mathrm{STT}_{\mathrm{sur}}$s are both unranked trees. SRTSTs with single-use restriction ($\mathrm{SRTST}_{\mathrm{sur}}$s) (which are nothing but $\mathrm{STST}_{\mathrm{sur}}$s [1] whose input is restricted to ranked trees) have the same expressive power as MSO-definable ranked-tree-to-string transducers (see Appendix A); and SRTSTs are more expressive than $\mathrm{SRTST}_{\mathrm{sur}}$s. Streaming transducers without single-use restriction were not studied in [1].[3]

Deterministic top-down tree-to-string transducers with regular look-ahead ($\mathrm{yDT}^{\mathrm{R}}$s) [5] are a classical model for structural-recursive tree-to-string transformation. A $\mathrm{yDT}^{\mathrm{R}}$ defines a transformation from ranked trees to strings by mutually recursive functions with regular look-ahead. The equivalence of $\mathrm{yDT}^{\mathrm{R}}$s had been a long-standing open problem [6], that was recently solved by Seidl et al. [12,13] and their decision procedure was implemented and experimented in [15]. For a subclass of $\mathrm{yDT}^{\mathrm{R}}$, the notion of *finite copying* for top-down tree transducers was introduced in [9]. Intuitively, a $\mathrm{yDT}^{\mathrm{R}}$ is finite copying ($\mathrm{yDT}^{\mathrm{R}}_{\mathrm{fc}}$), if each subtree of an input is copied at most a bounded number of times. The expressiveness of $\mathrm{yDT}^{\mathrm{R}}_{\mathrm{fc}}$s coincides with that of MSO-definable tree-to-string transducers [7], thereby $\mathrm{yDT}^{\mathrm{R}}_{\mathrm{fc}}$ and $\mathrm{SRTST}_{\mathrm{sur}}$ are equi-expressive.

In this paper, we characterize the class of SRTSTs in terms of $\mathrm{yDT}^{\mathrm{R}}$s. Our contributions are:

(i) SRTSTs and $\mathrm{yDT}^{\mathrm{R}}$s are equi-expressive,
(ii) bottom-up SRTSTs and SRTSTs are equi-expressive, and
(iii) the equivalence problem for SRTSTs is decidable.

Our main contribution is (i). In Section 4, we show how to construct equivalent transducers for each direction. In the direction from $\mathrm{yDT}^{\mathrm{R}}$s to SRTSTs,used we construct the equivalent SRTST to be bottom-up one for every given $\mathrm{yDT}^{\mathrm{R}}$, which shows (ii). As an immediate corollary, (iii) is derived from (i) and the decidability of the equivalence problem for $\mathrm{yDT}^{\mathrm{R}}$s [13].

Related work There is a subclass of $\mathrm{yDT}^{\mathrm{R}}$s besides $\mathrm{yDT}^{\mathrm{R}}_{\mathrm{fc}}$s, called $\mathrm{yDT}_{\mathrm{seq}}$s, which are restricted to non-copying and order-preserving on input variables occurring in right-hand sides of rules. $\mathrm{yDT}_{\mathrm{seq}}$s are equi-expressive to *deterministic nested word-to-word transducers* [14], which is a subclass of $\mathrm{SRTST}_{\mathrm{sur}}$s hence their equivalence problem is known to be decidable.

As for streaming string-to-string transducers rather than tree-to-string ones, *copyless streaming string transducers* ($\mathrm{SST}_{\mathrm{cl}}$s) have been introduced in [3]. The expressiveness of $\mathrm{SST}_{\mathrm{cl}}$s coincides with that of MSO-definable string transducers. *Copyful streaming string transducers* ($\mathrm{SST}_{\mathrm{cf}}$s) that are released from the copyless restriction are studied in [10]. It is shown that $\mathrm{SST}_{\mathrm{cf}}$s and HDT0L systems are equi-expressive.

[3] In the current paper, notions with (resp. without) single-use restriction are written by words with (resp. without) the subscript $_{\mathrm{sur}}$ such as $\mathrm{STT}_{\mathrm{sur}}$ (resp. STT). In [1], $\mathrm{STT}_{\mathrm{sur}}$ and $\mathrm{STST}_{\mathrm{sur}}$ are written just as STT and STST, respectively.

Macro forest transducers that are richer than yDT$^{\mathrm{R}}$s have been translated into streaming transducers to obtain efficient XML stream processors [11]. The streaming model is rather informal and its expressiveness has not been studied.

2 Preliminaries

For $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, \ldots, n\}$; in particular, $[0] = \emptyset$. We use boldface letters such as $\boldsymbol{t}$ to denote tuples. For a k-tuple $\boldsymbol{t}$, we write $|\boldsymbol{t}|$ for the length k of $\boldsymbol{t}$. For a set A and $k \in \mathbb{N}$, A^k denotes the set of all k-tuples of elements of A, and $A^{(\leq k)} \triangleq \bigcup_{i=0}^{k} A^i$. For $\boldsymbol{t} = (a_1, \ldots, a_k) \in A^k$ and $a \in A$, we denote by $\boldsymbol{t} \,||\, a$ the $(k+1)$-tuple $(a_1, \ldots, a_k, a)$. We write ε for the empty string, and $\mathrm{Dom}(f)$ for the domain of definition of a partial function f.

A *ranked alphabet* is a pair of a finite set Σ and a function $\mathrm{rank}_\Sigma : \Sigma \to \mathbb{N}$; the value $\mathrm{rank}_\Sigma(\sigma)$ of a symbol σ is called the *rank* of σ. We define $\Sigma^{(n)} \triangleq \{\sigma \in \Sigma \mid \mathrm{rank}_\Sigma(\sigma) = n\}$, and write σ also as $\sigma^{(n)}$ when $\sigma \in \Sigma^{(n)}$. The set of (*ranked*) *trees over* Σ, denoted by $\mathcal{T}_\Sigma$, is the smallest set T such that if $\sigma \in \Sigma^{(n)}$ and $t_1, \ldots, t_n \in T$ then $\sigma(t_1, \ldots, t_n) \in T$. We use an English letter (mainly e) as a metavariable ranging over rank-0 letters, i.e., leaves; a tree of the form $e()$ is written as e.

An *alphabet* is just a finite set. Let Σ be an alphabet. A *tagged alphabet* $\hat{\Sigma}$ consists of the *call symbols* $\langle\sigma$, and the *return symbols* $\sigma\rangle$, for all $\sigma \in \Sigma$. A *nested word* over Σ is a finite sequence over $\hat{\Sigma}$. We simply write $\langle e \rangle$ for a nested word of the form $\langle e\ e\rangle$. A nested word w is called *well-matched* if all the left and right angle brackets occurring in w are well-bracketed; and w is called *well-labeled* if each matched pair of call and return symbols is labeled with the same symbol in Σ. Any well-labeled nested word is well-matched. For example, $\langle \mathtt{a}\ \mathtt{b}\rangle\ \langle \mathtt{a}\ \mathtt{b}\rangle$ is well-matched but not well-labeled; $\langle \mathtt{a}\ \mathtt{a}\rangle\ \langle \mathtt{b}\ \langle \mathtt{a}\ \mathtt{a}\rangle\ \mathtt{b}\rangle$ is well-labeled; and $\langle \mathtt{a}\ \mathtt{a}\rangle\ \langle \mathtt{b}\ \langle \mathtt{b}$ is not well-matched.

Let Σ be a ranked alphabet. A mapping from ranked trees to nested words $\lfloor - \rfloor : \mathcal{T}_\Sigma \to \hat{\Sigma}^*$ is defined by $\lfloor t \rfloor = \langle\sigma\ \lfloor t_1 \rfloor \cdots \lfloor t_n \rfloor\ \sigma\rangle$ for $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$. The set of *ranked nested words*, denoted by $\lfloor \mathcal{T}_\Sigma \rfloor$, is defined by $\left\{ \lfloor t \rfloor \,\middle|\, t \in \mathcal{T}_\Sigma \right\}$. Ranked nested words respect ranks but well-labeled nested words do not necessarily (furthermore, well-labeled nested words can express not just unranked trees but even forests). For example, for $\sigma^{(2)} \in \Sigma$, $\langle\sigma\ \sigma\rangle$ is well-labeled, but is not a ranked nested word as $\sigma()$ is not a ranked tree. Note that $\lfloor - \rfloor$ is injective, and $\lfloor \mathcal{T}_\Sigma \rfloor$ is isomorphic to $\mathcal{T}_\Sigma$; we write the converse function from ranked nested words to ranked trees as $\lceil - \rceil : \lfloor \mathcal{T}_\Sigma \rfloor \to \mathcal{T}_\Sigma$. In this paper, we are interested in ranked nested words rather than well-labeled nested words.

Definition 1. A *non-deterministic finite state bottom-up tree automaton* (*NBTA*) is a tuple (Π, Σ, θ) where:

- Π is a finite set of *states*,
- Σ is a ranked alphabet of *input symbols*, and
- $\theta : \bigcup_{n \in \mathbb{N}} (\Sigma^{(n)} \times \Pi^n) \to 2^\Pi$ is a function called a *transition function*.

Let $A = (\Pi, \Sigma, \theta)$ be an NBTA. We extend θ to $\hat{\theta} : \mathcal{T}_\Sigma \to 2^\Pi$ by induction: $\hat{\theta}(\sigma(t_1, \ldots, t_n)) \triangleq \bigcup_{\pi_1 \in \hat{\theta}(t_1), \ldots, \pi_n \in \hat{\theta}(t_n)} \theta(\sigma, (\pi_1, \ldots, \pi_n))$. We say that a state $\pi \in \Pi$ *accepts* a tree $t \in \mathcal{T}_\Sigma$ if $\pi \in \hat{\theta}(t)$. We denote by $\mathcal{L}_A(\pi)$ the set of trees accepted by π; we simply write $\mathcal{L}(\pi)$ for $\mathcal{L}_A(\pi)$ when A is clear from the context. We assume that $\mathcal{L}_A(\pi) \neq \emptyset$ for all $\pi \in \Pi$. An NBTA A is called a *deterministic finite state bottom-up tree automaton* (DBTA) if $\theta(\sigma, (\pi_1, \ldots, \pi_n))$ has exactly one element for every $\sigma \in \Sigma^{(n)}$ and $\pi_1, \ldots, \pi_n \in \Pi$. For a DBTA, we write $\theta(\sigma, (\pi_1, \ldots, \pi_n))$ for its unique element. DBTAs and NBTAs recognize the same class of tree languages known as *regular tree languages* [4]. We denote by *REG* the set of all regular tree languages. *REG* is closed under union, intersection, and complementation.

3 Transducers

We here define the two main notions to be compared: $\mathrm{yDT}^{\mathrm{R}}$ and SRTST.

3.1 Deterministic Top-Down Tree-to-String Transducers with Regular Look-ahead

A deterministic top-down tree-to-string transducer with regular look-ahead ($\mathrm{yDT}^{\mathrm{R}}$ for short) works on ranked trees.

First we prepare the range of the right hand sides of transition rules of $\mathrm{yDT}^{\mathrm{R}}$s. We fix a set of input variables $X = \{x_1, x_2, \ldots\}$ and define $X_n \triangleq \{x_1, \ldots, x_n\}$ for $n \in \mathbb{N}$. Also, for a set Q and $n \in \mathbb{N}$, we define $Q(X_n) \triangleq \{q(x_i) \mid q \in Q, i \leq n\}$ where $q(x_i) \triangleq (q, x_i)$. For a finite set Q (of *states*), an alphabet Δ, and $n \in \mathbb{N}$, we define the set $Rhs_{Q,\Delta}(X_n)$ of *expressions* by $\tau ::= \varepsilon \mid a\tau \mid q(x_i)\tau$ where $a \in \Delta$ and $q(x_i) \in Q(X_n)$. The following definition of $\mathrm{yDT}^{\mathrm{R}}$ is taken from [13], except that we adopt the style in [5] for regular look-ahead (i.e., we consider directly regular tree languages rather than states of an NBTA).

Definition 2 ($\mathrm{yDT}^{\mathrm{R}}$). A *deterministic top-down tree-to-string transducer with regular look-ahead* ($\mathrm{yDT}^{\mathrm{R}}$) is a tuple $(Q, \Sigma, \Delta, Init, R)$ satisfying the following conditions:

- Q is a finite set of *states*.
- Σ is a ranked alphabet of *input symbols*.
- Δ is an alphabet of *output symbols*.
- $Init \subseteq Rhs_{Q,\Delta}(X_1) \times (REG \setminus \{\emptyset\})$ is a finite set of *initial sequences*, and must satisfy that, for any two distinct initial sequences (τ, L) and (τ', L') in $Init$, $L \cap L' = \emptyset$. For $L \in REG \setminus \{\emptyset\}$, τ such that $(\tau, L) \in Init$ is unique (if it exists), and is denoted by $Init(L)$.
- $R \subseteq \bigcup_{n \in \mathbb{N}} (Q \times \Sigma^{(n)} \times Rhs_{Q,\Delta}(X_n) \times (REG \setminus \{\emptyset\})^n)$ is a finite set of *rules*, and must satisfy that, for any two distinct rules $(q, \sigma, \tau, (L_1, \ldots, L_n))$ and $(q, \sigma, \tau', (L'_1, \ldots, L'_n))$ in R, $L_i \cap L'_i = \emptyset$ for some $i \in [n]$. A rule $(q, \sigma, \tau, (L_1, \ldots, L_n))$ is written as

$$q(\sigma(x_1, \ldots, x_n)) \to \tau \qquad \langle L_1, \ldots, L_n \rangle \ .$$

Here τ is uniquely determined (if it exists) from $(q, \sigma, (L_1, \ldots, L_n))$; hence we call this rule a $(q, \sigma, (L_1, \ldots, L_n))$-*rule*, and denote τ by $\mathrm{rhs}(q, \sigma, (L_1, \ldots, L_n))$. We write $q(e) \to \tau$ for $q(e()) \to \tau\ \langle\rangle$ when $\mathrm{rank}_\Sigma(e) = 0$.

We define the semantics of a yDT$^{\mathrm{R}}$, which transforms a tree in a top-down manner. Let $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$ be a yDT$^{\mathrm{R}}$, and we define a partial function $[\![M]\!] : \mathcal{T}_\Sigma \rightharpoonup \Delta^*$.

First, we define auxiliary partial functions $[\![q]\!]_M : \mathcal{T}_\Sigma \rightharpoonup \Delta^*$ (for $q \in Q$) and $[\![\tau]\!]_M : \mathcal{T}_\Sigma^n \rightharpoonup \Delta^*$ (for $n \in \mathbb{N}$ and $\tau \in \mathit{Rhs}_{Q,\Delta}(X_n)$), by simultaneous induction on input trees. Let $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ and $q \in Q$. If there exist $L_1, \ldots, L_n$ and a $(q, \sigma, (L_1, \ldots, L_n))$-rule in R such that $t_i \in L_i$ for all $i \in [n]$, then $[\![q]\!]_M(t)$ is defined as

$$[\![q]\!]_M(\sigma(t_1, \ldots, t_n)) \triangleq [\![\mathrm{rhs}(q, \sigma, (L_1, \ldots, L_n))]\!]_M(t_1, \ldots, t_n)$$

(if the r.h.s. is defined), and $[\![q]\!]_M(t)$ is not defined otherwise. Here note that, due to the condition on R, the tuple $(L_1, \ldots, L_n)$ above is unique if exists. For $n \in \mathbb{N}$ and $\tau \in \mathit{Rhs}_{Q,\Delta}(X_n)$, $[\![\tau]\!]_M(t_1, \ldots, t_n)$ is defined as follows:

$$\begin{aligned}
[\![\varepsilon]\!]_M(t_1, \ldots, t_n) &\triangleq \varepsilon \\
[\![a\tau']\!]_M(t_1, \ldots, t_n) &\triangleq a\,[\![\tau']\!]_M(t_1, \ldots, t_n) \\
[\![q'(x_i)\tau']\!]_M(t_1, \ldots, t_n) &\triangleq [\![q']\!]_M(t_i)\,[\![\tau']\!]_M(t_1, \ldots, t_n)\ .
\end{aligned}$$

Now let us define $[\![M]\!] : \mathcal{T}_\Sigma \rightharpoonup \Delta^*$. For $t \in \mathcal{T}_\Sigma$, if there exists a pair $(\tau, L) \in \mathit{Init}$ such that $t \in L$ and if $[\![\tau]\!]_M(t)$ is defined, then we define $[\![M]\!](t) \triangleq [\![\tau]\!]_M(t)$; and $[\![M]\!](t)$ is not defined otherwise. Again, note that such a pair (τ, L) is unique. We denote by $\mathrm{Dom}(M)$ the domain of $[\![M]\!]$.

In the above definition, regular look-ahead is realized by regular tree languages directly. As a finite representation for expressing regular tree languages, we use an NBTA or a DBTA, switching the two styles conveniently. (Recall that the two notions recognize the same class of languages, *REG*.) We call an automaton used for regular look-ahead a *regular look-ahead automaton*. Given a regular look-ahead automaton, we use states of the automaton instead of regular tree languages when we refer to initial sequences or rules (e.g., we write $\mathrm{rhs}(q, \sigma, (\pi_1, \ldots, \pi_n))$ for $\mathrm{rhs}(q, \sigma, (\mathcal{L}(\pi_1), \ldots, \mathcal{L}(\pi_n)))$).

The equivalence problem for yDT$^{\mathrm{R}}$ is known to be decidable.

Theorem 3 (Corollary 8.1 in [13]). *Given two* yDT$^{\mathrm{R}}$s M_1, M_2, *it is decidable whether* $[\![M_1]\!] = [\![M_2]\!]$.

See Appendix D for an example of a yDT$^{\mathrm{R}}$ (Example 28).

3.2 Streaming Ranked-Tree-to-String Transducers

A streaming ranked-tree-to-string transducer (SRTST for short) works on ranked nested words. We first prepare some auxiliary definitions.

For a finite set Γ (of *variables*) and an alphabet Δ, the set $E(\Gamma, \Delta)$ of *expressions over Γ and Δ* is defined by the following grammar: $E ::= \varepsilon \mid aE \mid \gamma E$ where $a \in \Delta$ and $\gamma \in \Gamma$. Note that $E(\emptyset, \Delta) = \Delta^*$.

Let Γ and Γ' be finite sets and Δ be an alphabet. We call a mapping $\rho : \Gamma \to E(\Gamma', \Delta)$ an *assignment*, and denote ρ by $[\gamma_1 := e_1, \ldots, \gamma_n := e_n]$ where $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ and $e_i = \rho(\gamma_i)$; in this notation, $\gamma_i := e_i$ may be omitted if $e_i = \gamma_i$. An assignment ρ is naturally extended to $\rho : E(\Gamma, \Delta) \to E(\Gamma', \Delta)$: for $e \in E(\Gamma, \Delta)$, $\rho(e)$ is the expression over Γ' and Δ obtained by replacing all occurrences of every variable γ in e with $\rho(\gamma)$. We denote $\rho(e)$ by $e\rho$. The set of all assignments over Γ, Γ' and Δ is denoted by $\mathcal{A}(\Gamma, \Gamma', \Delta)$. An element of $\mathcal{A}(\Gamma, \emptyset, \Delta)$ (i.e., a function $\Gamma \to \Delta^*$) is called an *evaluation function*, and a returned value of an evaluation function is called a *variable value*.

Given two assignments $\rho_1 : \Gamma_1 \to E(\Gamma_1', \Delta)$ and $\rho_2 : \Gamma_2 \to E(\Gamma_2', \Delta)$, the assignment $\rho_1\rho_2 : \Gamma_1 \to E((\Gamma_1' \setminus \Gamma_2) \cup \Gamma_2', \Delta)$ is defined (as usual) as follows. For $\gamma \in \Gamma_1$, $(\rho_1\rho_2)(\gamma)$ is an expression in $E((\Gamma_1' \setminus \Gamma_2) \cup \Gamma_2', \Delta)$ obtained by replacing all occurrences of variable $\gamma_1' \in \Gamma_1' \cap \Gamma_2$ in $\rho_1(\gamma) \in E(\Gamma_1', \Delta)$ with $\rho_2(\gamma_1') \in E(\Gamma_2', \Delta)$, and keeping all occurrences of other variable $\gamma_1' \in \Gamma_1' \setminus \Gamma_2$ in $\rho_1(\gamma) \in E(\Gamma_1', \Delta)$. Given two assignments $\rho_1 : \Gamma_1 \to E(\Gamma, \Delta)$ and $\rho_2 : \Gamma_2 \to E(\Gamma, \Delta)$ where Γ_1 and Γ_2 are disjoint, the assignment $\rho_1 \uplus \rho_2 : \Gamma_1 \uplus \Gamma_2 \to E(\Gamma, \Delta)$ is defined by

$$\rho_1 \uplus \rho_2 \triangleq [\gamma_1 := \rho_1(\gamma_1), \ldots, \gamma_n := \rho_1(\gamma_n), \gamma_1' := \rho_2(\gamma_1'), \ldots, \gamma_m' := \rho_2(\gamma_m')]$$

where $\Gamma_1 = \{\gamma_1, \ldots, \gamma_n\}$ and $\Gamma_2 = \{\gamma_1', \ldots, \gamma_m'\}$.

We introduce the notion of an SRTST based on the definition of $\mathrm{STT}_{\mathrm{sur}}$ in [1]. The difference is as follows: an $\mathrm{STT}_{\mathrm{sur}}$ is restricted by *single-use restriction* (sur for short), while an SRTST is not restricted; the input and output of an $\mathrm{STT}_{\mathrm{sur}}$ are well-matched nested words, while the input and output of an SRTST are ranked nested words and strings, respectively.

Definition 4. A *streaming ranked-tree-to-string transducer* (SRTST for short) is a tuple $(S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$ where:

- S is a finite set of *states*,
- Σ is a ranked alphabet of *input symbols*,
- Δ is an alphabet of *output symbols*,
- P is a finite set of *stack symbols*,
- $s_0 \in S$ is an *initial state*,
- Γ is a finite set of *variables*,
- $F : S \rightharpoonup E(\Gamma, \Delta)$ is a partial function called an *output function*,
- $\delta_c : S \times \Sigma \to S \times P$ is a *call state-transition function*,
- $\delta_r : S \times P \times \Sigma \to S$ is a *return state-transition function*,
- $\rho_c : S \times \Sigma \to \mathcal{A}(\Gamma, \Gamma, \Delta)$ is a *call variable-update function*,
- $\rho_r : S \times P \times \Sigma \to \mathcal{A}(\Gamma, \Gamma \uplus \overline{\Gamma}, \Delta)$ is a *return variable-update function* where $\overline{\Gamma}$ is a "copy" of Γ, i.e., $\overline{\Gamma} \triangleq \{\overline{\gamma} \mid \gamma \in \Gamma\}$ and each $\overline{\gamma}$ is a fresh symbol.

We define the semantics of an SRTST, which transforms a tree (a ranked nested word) in a top-down and bottom-up way. Let $T = (S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$

be an SRTST. We define the set of *configurations* of T, denoted by Ψ, as $\Psi := S \times (P \times \mathcal{A}(\Gamma, \emptyset, \Delta))^* \times \mathcal{A}(\Gamma, \emptyset, \Delta)$. For a configuration $(s, \Lambda, \alpha) \in \Psi$, Λ is called the *stack*, and α is called the *current evaluation function*. Let $\alpha_\varepsilon^\Gamma \triangleq [\gamma := \varepsilon]_{\gamma \in \Gamma}$, which is called the *emptyword evaluation function*; we often omit the superscript Γ of $\alpha_\varepsilon^\Gamma$. We call $(s_0, \varepsilon, \alpha_\varepsilon^\Gamma)$ the *initial configuration*. We define the *transition function* $\delta : \Psi \times \hat{\Sigma} \rightharpoonup \Psi$ *over configurations* as follows:

Call transitions For $\sigma \in \Sigma$, $\delta\big((s, \Lambda, \alpha), \langle\sigma\big) \triangleq \big(s',\ (p, \alpha')\Lambda,\ \alpha_\varepsilon^\Gamma\big)$ where:
- $(s', p) \triangleq \delta_c(s, \sigma)$: we invoke the state-transition function δ_c, which reads $\langle\sigma$ in the state s,
- $\alpha' \triangleq \rho_c(s, \sigma)\alpha$: we push (p, α') on the stack Λ (rather than setting α' as the current evaluation function); α' is almost $\rho_c(s, \sigma)$ but each variable γ in $\rho_c(s, \sigma)$ is substituted for $\alpha(\gamma)$ (and $\alpha(\gamma)$ is discarded if γ does not occur in $\rho_c(s, \sigma)$),
- we reset the current evaluation function α to the emptyword evaluation function $\alpha_\varepsilon^\Gamma$.

Return transitions For $\sigma \in \Sigma$, $\delta\big((s, (p, \beta)\Lambda, \alpha),\ \sigma\rangle\big) \triangleq \big(s', \Lambda, \alpha'\big)$ where:
- $s' \triangleq \delta_r(s, p, \sigma)$: we invoke the state-transition function δ_r, which read $\sigma\rangle$ in the state s with the stack symbol p on the stack,
- we pop (p, β) from the stack,
- $\alpha' \triangleq \rho_r(s, p, \sigma)(\alpha \uplus \overline{\beta})$ where $\overline{\beta}$ is the "copy" of β: i.e., $\overline{\beta} \triangleq [\overline{\gamma} := \beta(\gamma)]_{\gamma \in \Gamma}$: we replace α with α', which is almost $\rho_r(s, p, \sigma)$ but each variable γ in $\rho_r(s, p, \sigma)$ is substituted for $\alpha(\gamma)$ and each variable $\overline{\gamma}$ in $\rho_r(s, p, \sigma)$ is substituted for $\beta(\gamma)$ (and $\alpha(\gamma)$ / $\beta(\gamma)$ is discarded if γ / $\overline{\gamma}$ does not occur in $\rho_c(s, \sigma)$).

Now we define the meaning $[\![T]\!] : \lfloor \mathcal{T}_\Sigma \rfloor \rightharpoonup \Delta^*$. First, the transition function $\delta : \Psi \times \hat{\Sigma} \rightharpoonup \Psi$ naturally extends to $\delta^* : \Psi \times \hat{\Sigma}^* \rightharpoonup \Psi$ by iterating δ. For a nested word $w \in \hat{\Sigma}^*$, we denote by $c \overset{w}{\Rightarrow}_T c'$ if $\delta^*(c, w) = c'$; we omit the subscript T if it is clear from the context. Note that for any configuration c and any well-matched nested word w, $\delta^*(c, w)$ is always defined. For a ranked nested word $w \in \lfloor \mathcal{T}_\Sigma \rfloor$, if $(s_0, \varepsilon, \alpha_\varepsilon^\Gamma) \overset{w}{\Rightarrow} (s, \varepsilon, \alpha)$ and if $F(s)$ is defined then $[\![T]\!](w) \triangleq F(s)\alpha$; otherwise $[\![T]\!](w)$ is undefined. We denote by $\mathrm{Dom}(T)$ the domain of $[\![T]\!]$.

Example 5. We give an SRTST T: let $S \triangleq \{s_?, s_\mathtt{a}, s_\mathtt{b}\}$; $\Sigma \triangleq \{\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}\}$; $\Delta \triangleq \{\mathtt{a}, \mathtt{b}\}$; $P \triangleq \{p_?, p_\mathtt{a}, p_\mathtt{b}\}$; $s_0 \triangleq s_?$; $\Gamma \triangleq \{\gamma\}$; $F(s) \triangleq \gamma$ for every $s \in S$; $\delta_c(s_d, \sigma) \triangleq (s_?, p_d)$ for every $\sigma \in \Sigma$ and $d \in \{\mathtt{a}, \mathtt{b}, ?\}$; δ_r is defined by

$$\delta_r(s, p_d, \sigma) \triangleq s_d, \quad \delta_r(s_d, p_?, \sigma) \triangleq s_d, \quad \delta_r(s_?, p_?, \mathtt{f}) \triangleq s_?, \quad \delta_r(s_?, p_?, d) \triangleq s_d,$$

for every $s \in S$, $\sigma \in \Sigma$, $d \in \{\mathtt{a}, \mathtt{b}\}$; ρ_c is defined by

$$\rho_c(s, \mathtt{f}) \triangleq [\gamma := \gamma],\ \rho_c(s_?, d) \triangleq [\gamma := d],\ \rho_c(s_\mathtt{a}, d) \triangleq [\gamma := \gamma d],\ \rho_c(s_\mathtt{b}, d) \triangleq [\gamma := d\gamma],$$

for every $s \in S$ and $d \in \{\mathtt{a}, \mathtt{b}\}$; and ρ_r is defined by

$$\rho_r(s, p_?, \mathtt{f}) \triangleq [\gamma := \gamma], \quad \rho_r(s, p_\mathtt{a}, \mathtt{f}) \triangleq [\gamma := \overline{\gamma}\gamma], \quad \rho_r(s, p, \mathtt{a}) \triangleq [\gamma := \overline{\gamma}],$$
$$\rho_r(s, p_\mathtt{b}, \mathtt{f}) \triangleq [\gamma := \gamma\overline{\gamma}], \quad \rho_r(s, p, \mathtt{b}) \triangleq [\gamma := \overline{\gamma}],$$

for every $s \in S$ and $p \in P$. Given $t \in \mathcal{T}_\Sigma$, $[\![T]\!](\lfloor t \rfloor)$ recursively swaps the two subtrees of the root if the leftmost leaf is $\mathtt{b}$, and skip the swapping otherwise; and then produces the leaves as the output. For instance, $[\![T]\!]$ transforms $\lfloor \mathtt{f}(\mathtt{f}(\mathtt{b}, \mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{b})) \rfloor$ as in Fig. 1, where for clarity we denote stacks with $[\![-]\!]$.

$$
\begin{array}{l|l}
(s_?, [\![]\!], \alpha_\varepsilon) & \overset{\langle \mathtt{f}}{\Longrightarrow} (s_?, [\![(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) \\
\overset{\langle \mathtt{f}}{\Longrightarrow} (s_?, [\![(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) & \overset{\langle \mathtt{a}}{\Longrightarrow} (s_?, [\![(p_?, [\gamma{:=}\mathtt{a}])(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) \\
\overset{\langle \mathtt{f}}{\Longrightarrow} (s_?, [\![(p_?, \alpha_\varepsilon)(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) & \overset{\mathtt{a}\rangle}{\Longrightarrow} (s_\mathtt{a}, [\![(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{a}]) \\
\overset{\langle \mathtt{b}}{\Longrightarrow} (s_?, [\![(p_?, [\gamma{:=}\mathtt{b}])(p_?, \alpha_\varepsilon)(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) & \overset{\langle \mathtt{b}}{\Longrightarrow} (s_?, [\![(p_\mathtt{a}, [\gamma{:=}\mathtt{ab}])(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) \\
\overset{\mathtt{b}\rangle}{\Longrightarrow} (s_\mathtt{b}, [\![(p_?, \alpha_\varepsilon)(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{b}]) & \overset{\mathtt{b}\rangle}{\Longrightarrow} (s_\mathtt{a}, [\![(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{ab}]) \\
\overset{\langle \mathtt{a}}{\Longrightarrow} (s_?, [\![(p_\mathtt{b}, [\gamma{:=}\mathtt{ab}])(p_?, \alpha_\varepsilon)(p_?, \alpha_\varepsilon)]\!], \alpha_\varepsilon) & \overset{\mathtt{f}\rangle}{\Longrightarrow} (s_\mathtt{b}, [\![(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{abab}]) \\
\overset{\mathtt{a}\rangle}{\Longrightarrow} (s_\mathtt{b}, [\![(p_?, \alpha_\varepsilon)(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{ab}]) & \overset{\mathtt{f}\rangle}{\Longrightarrow} (s_\mathtt{b}, [\![]\!], [\gamma{:=}\mathtt{abab}]) \\
\overset{\mathtt{f}\rangle}{\Longrightarrow} (s_\mathtt{b}, [\![(p_?, \alpha_\varepsilon)]\!], [\gamma{:=}\mathtt{ab}]) &
\end{array}
$$

Fig. 1. Transitions for $\lfloor \mathtt{f}(\mathtt{f}(\mathtt{b}, \mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{b})) \rfloor$

4 SRTST and yDT$^{\mathrm{R}}$ are equi-expressive

We show the equi-expressiveness between SRTSTs and yDT$^{\mathrm{R}}$s, by giving effective constructions in the both directions. Then (only) the construction of a yDT$^{\mathrm{R}}$ from an SRTST is used to show the decidability of the equivalence of SRTST. The both constructions are basically *component-wise*: the regular look-ahead automaton of a yDT$^{\mathrm{R}}$ M corresponds to the state-transition functions δ_c, δ_r of an SRTST T (also see Remark 20); rules of M correspond to the variable-update functions ρ_c, ρ_r of T; and *Init* of M corresponds to F of T.

4.1 The construction of SRTST from yDT$^{\mathrm{R}}$

We construct a *bottom-up* SRTST T from a yDT$^{\mathrm{R}}$ M; an SRTST T is *bottom-up* [1] if $\delta_c(s, \sigma) = (s_0, p)$ for some p and $\rho_c(s, p, \sigma) = [\gamma := \gamma]_{\gamma \in \Gamma}$ for every s, p and σ, (i.e., if in any call transition T always resets the state s to the initial state and never change the evaluation function α being pushed on the stack). The construction in the next lemma is inspired by the proof of the decidability of the equivalence of yDT$^{\mathrm{R}}$s given in [12, Sections 3 and 4].

Lemma 6. *For any* yDT$^{\mathrm{R}}$ M*, there exists a bottom-up* SRTST T *such that* $\mathrm{Dom}(T) = \lfloor \mathrm{Dom}(M) \rfloor$ *and* $[\![T]\!](\lfloor t \rfloor) = [\![M]\!](t)$ *for all* $t \in \mathrm{Dom}(M)$.

Proof. Let $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$ be a yDT$^{\mathrm{R}}$ and we assume that its regular look-ahead is given by DBTA $A = (\Sigma, \Pi, \theta)$. Further, w.l.o.g. (see Lemma 12), we can assume that M satisfies the following conditions: (i) for any tree $\sigma(t_1, \ldots, t_n)$ and $q \in Q$, $\sigma(t_1, \ldots, t_n) \in \mathrm{Dom}([\![q]\!]_M)$ iff there exists a $(q, \sigma, (L_1, \ldots, L_n))$-rule in R such that $t_i \in L_i$ for every $i \in [n]$; and (ii) $\mathrm{Dom}(M) = \bigcup_{(\tau,\pi) \in \mathit{Init}} \mathcal{L}(\pi)$.

We define an equivalent SRTST $T = (S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$ as follows. (For its behavior, see also the explanation after the lemma.)

Let $m = \max(\{1\} \cup \{\mathrm{rank}_\Sigma(\sigma) \mid \sigma \in \Sigma\})$, $S = P = \Pi^{(\leq m)}$, and $s_0 = ()$. We define $\Gamma = Q(X_m)$, so that $E(\Gamma, \Delta) = \mathit{Rhs}_{Q,\Delta}(X_m)$. We define the output (partial) function $F : S \rightharpoonup E(\Gamma, \Delta)$ as $F((\pi)) = \mathit{Init}(\mathcal{L}(\pi)) \in \mathit{Rhs}_{Q,\Delta}(X_1)$.

The call state-transition function $\delta_c : S \times \Sigma \to S \times P$ is defined by $\delta_c(\boldsymbol{\pi}, \sigma) = (s_0, \boldsymbol{\pi})$. The return state-transition function $\delta_r : S \times P \times \Sigma \to S$ is defined by $\delta_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma) = \boldsymbol{\pi}' || \theta(\sigma, \boldsymbol{\pi})$ for $\boldsymbol{\pi}, \boldsymbol{\pi}' \in \Pi^{(\leq m)}$ and $\sigma \in \Sigma$ such that $|\boldsymbol{\pi}| = \mathrm{rank}_\Sigma(\sigma)$ and $|\boldsymbol{\pi}'| \neq m$, and by $\delta_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma) = s_0$ for the other case of arguments $\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma$.

The call variable-update function $\rho_c : S \times \Sigma \to \mathcal{A}(\Gamma, \Gamma, \Delta)$ is defined by $\rho_c(\boldsymbol{\pi}, \sigma) = [\gamma := \gamma]_{\gamma \in \Gamma}$. The return variable-update function $\rho_r : S \times P \times \Sigma \to \mathcal{A}(\Gamma, \Gamma \uplus \overline{\Gamma}, \Delta)$ is defined by

$$\begin{aligned}\rho_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma) = \quad & \left[q(x_k) := \overline{q(x_k)}\right]_{q \in Q,\, k \leq |\boldsymbol{\pi}'|} \\ \uplus\ & \left[q(x_{|\boldsymbol{\pi}'|+1}) := \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi})\right]_{q \in Q} \\ \uplus\ & \left[q(x_k) := \varepsilon\right]_{q \in Q,\, |\boldsymbol{\pi}'|+1 < k \leq m}\end{aligned}$$

for $\boldsymbol{\pi}, \boldsymbol{\pi}' \in \Pi^{(\leq m)}$ and $\sigma \in \Sigma$ such that $|\boldsymbol{\pi}| = \mathrm{rank}_\Sigma(\sigma)$ and $|\boldsymbol{\pi}'| \neq m$, and by $\rho_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma) = \alpha_\varepsilon$ for the other case of arguments $\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma$. Above, the function $\underline{\mathrm{rhs}} : \bigcup_{n \in \mathbb{N}} (Q \times \Sigma^{(n)} \times \Pi^n) \to E(\Gamma, \Delta)$ is defined by

$$\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}) = \begin{cases} \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}) & ((q, \sigma, \boldsymbol{\pi})\text{-rule is defined}) \\ \varepsilon & (\text{otherwise}) \end{cases}$$

where $\mathrm{rhs}(q, \sigma, \boldsymbol{\pi}) \in \mathit{Rhs}_{Q,\Delta}(X_n) \subseteq E(\Gamma, \Delta)$. Above, ε in the definitions of ρ_r and $\underline{\mathrm{rhs}}$ are not used in the actual computation. We control "definedness" by the determinism of the DBTA A and the above assumptions (i) and (ii) for M.

The equivalence of M and T follows from Lemmas 15 and 16. □

The construction of T is designed to behave for each input symbol as follows. Let $\sigma'(t'_1, \ldots, t'_{n'})$ be a subtree of an input tree, and $t'_i = \sigma(t_1, \ldots, t_n)$.

Call symbol $\langle\sigma$: By the definitions of δ_c and ρ_c above, in general a call transition is $(\boldsymbol{\pi}', \Lambda, \beta) \overset{\langle\sigma}{\Longrightarrow} ((), (\boldsymbol{\pi}', \beta)\Lambda, \alpha_\varepsilon)$. Suppose that T starts the computation of $t'_i = \sigma(t_1, \ldots, t_n)$. The computation so far for $t'_1, \ldots, t'_{i-1}$ is recorded in $\boldsymbol{\pi}'$ and β and the further earlier computation is recorded in Λ. Then the call transition by $\langle\sigma$ saves the record $(\boldsymbol{\pi}', \beta)$ to the stack.

Return symbol $\sigma\rangle$: A return transition is basically of the following form: $(\boldsymbol{\pi}, (\boldsymbol{\pi}', \beta)\Lambda, \alpha) \overset{\sigma\rangle}{\Longrightarrow} (\boldsymbol{\pi}' \,||\, \theta(\sigma, \boldsymbol{\pi}), \Lambda, \alpha')$. Suppose that T is finishing the computation of some child tree $t'_i = \sigma(t_1, \ldots, t_n)$.

The state-transition function δ_r simulates the DBTA A as follows. In the transition $\delta_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma) = \boldsymbol{\pi}' \,||\, \theta(\sigma, \boldsymbol{\pi})$, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ is a tuple of A-states and $\pi_1, \ldots, \pi_n$ accept $t_1, \ldots, t_n$, respectively. Hence $\theta(\sigma, \boldsymbol{\pi})$ is an A-state which accepts $t'_i = \sigma(t_1, \ldots, t_n)$. Likewise, $\boldsymbol{\pi}'$ is a tuple of A-states which accept $t'_1, \ldots, t'_{i-1}$, respectively. In this way, δ_r plays the role of regular look-ahead, and each component of a state $\boldsymbol{\pi}$ of T (and of a stack symbol $\boldsymbol{\pi}'$ on a stack) represents a regular language of the look-ahead.

According to the definitions of the semantics of an SRTST and of ρ_r,

$$\begin{aligned}\alpha' = \rho_r(\boldsymbol{\pi}, \boldsymbol{\pi}', \sigma)(\alpha \uplus \overline{\beta}) = \quad & [q(x_k) := \beta(q(x_k))]_{q \in Q,\, k \leq |\boldsymbol{\pi}'|} \\ \uplus\ & [q(x_{|\boldsymbol{\pi}'|+1}) := \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi})\alpha]_{q \in Q} \\ \uplus\ & [q(x_k) := \varepsilon]_{q \in Q,\, |\boldsymbol{\pi}'|+1 < k \leq m}\ .\end{aligned}$$

Here, variable values of α are the result of computation of $t_1, \ldots, t_n$ by M (i.e., by every $q \in Q$), and hence $\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi})\alpha$ is the result of computation of $t'_i = \sigma(t_1, \ldots, t_n)$ by q; note that $q(x_{|\boldsymbol{\pi}'|+1}) = q(x_i)$. Likewise, β is the result of computation of $t'_1, \ldots, t'_{i-1}$ by M. In this way, T computes as M does, and the result is recorded in the current evaluation function α of a configuration (and evaluation function β on a stack).

Note that in our construction the finiteness of the width of trees is used for the finiteness of sets S, P, and Γ, in both δ_r and ρ_r.

See Appendix D for an example of the above construction (Example 29).

4.2 The construction of yDT$^{\mathrm{R}}$ from SRTST

For an SRTST T, $s_0, \ldots, s_n \in S$, and nested words $w_1 \ldots, w_n$, we write $s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n$ if $(s_0, \Lambda_0, \alpha_0) \overset{w_1}{\Longrightarrow} (s_1, \Lambda_1, \alpha_1) \cdots \overset{w_n}{\Longrightarrow} (s_n, \Lambda_n, \alpha_n)$ for some Λ_i and α_i $(i = 0, \ldots, n)$. We write $s \overset{\langle\sigma}{\Longrightarrow}_p s'$ if $\delta_c(s, \sigma) = (s', p)$, and $s \overset{\sigma\rangle}{\Longrightarrow}_p s'$ if $\delta_r(s, p, \sigma) = s'$.

Lemma 7. *For any* SRTST T*, there exists a* yDT$^{\mathrm{R}}$ M *such that* $\mathrm{Dom}(M) = \lceil \mathrm{Dom}(T) \rceil$ *and* $[\![M]\!](\lceil w \rceil) = [\![T]\!](w)$ *for all* $w \in \mathrm{Dom}(T)$.

Proof. Let $T = (S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$ be an SRTST. We define a yDT$^{\mathrm{R}}$ $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$ with a regular look-ahead NBTA $A = (\Sigma, \Pi, \theta)$.

First we define a predicate vst (stands for *valid state transition*): for given $\big((s^c_{n+1}, \sigma, s_{n+1}), (s_1, \sigma_1, s^r_1), \ldots, (s_n, \sigma_n, s^r_n)\big) \in \bigcup_{n\in\mathbb{N}} (S \times \Sigma^{(n)} \times S) \times (S \times \Sigma \times S)^n$,

$$\begin{aligned} &\mathrm{vst}((s^c_{n+1}, \sigma^{(n)}, s_{n+1}), (s_1, \sigma_1, s^r_1), \ldots, (s_n, \sigma_n, s^r_n)) \ \text{ if} \\ &s^c_{n+1} \overset{\langle\sigma}{\Longrightarrow}_p s_1 \overset{\langle\sigma_1}{\Longrightarrow}_{p_1} s'_1, \quad s^r_1 \overset{\sigma_1\rangle}{\Longrightarrow}_{p_1} s_2 \overset{\langle\sigma_2}{\Longrightarrow}_{p_2} s'_2, \quad s^r_2 \overset{\sigma_2\rangle}{\Longrightarrow}_{p_2} s_3 \cdots \\ &\qquad\qquad \cdots s_n \overset{\langle\sigma_n}{\Longrightarrow}_{p_n} s'_n, \quad s^r_n \overset{\sigma_n\rangle}{\Longrightarrow}_{p_n} s_{n+1} \end{aligned} \tag{1}$$

where $s'_i \in S$ and $p, p_i \in P$ are given by the call-transitions in (1). Note that $\mathrm{vst}((s^c_1, \sigma^{(0)}, s_1))$ iff $s^c_1 \overset{\langle\sigma}{\Longrightarrow}_p s_1$, when $n = 0$. Eq. (1) holds if we have

$$\begin{aligned} s^c_{n+1} \overset{\langle\sigma}{\Longrightarrow} s_1 \overset{\langle\sigma_1}{\Longrightarrow} s'_1 \overset{w_1}{\Longrightarrow} s^r_1 \overset{\sigma_1\rangle}{\Longrightarrow} s_2 \overset{\langle\sigma_2}{\Longrightarrow} s'_2 \overset{w_2}{\Longrightarrow} s^r_2 \overset{\sigma_2\rangle}{\Longrightarrow} s_3 \cdots \\ \cdots s_n \overset{\langle\sigma_n}{\Longrightarrow} s'_n \overset{w_n}{\Longrightarrow} s^r_n \overset{\sigma_n\rangle}{\Longrightarrow} s_{n+1} \end{aligned} \tag{2}$$

for some well-matched w_i. Conversely, to obtain (2) we need the condition vst recursively for the parts $s_i \overset{\langle\sigma_i}{\Longrightarrow} s'_i \overset{w_i}{\Longrightarrow} s^r_i$, which leads to the next definition. Note that (1) and (2) extend with $\overset{\sigma\rangle}{\Longrightarrow}_p s'$ and $\overset{\sigma\rangle}{\Longrightarrow} s'$ for some unique s', respectively.

States of NBTA Let $\Pi_0 \triangleq \emptyset$, and for $i > 0$,

$$\begin{aligned} \Pi_i \ \triangleq \ \Pi_{i-1} \cup \{(s^c, \sigma^{(n)}, s^r) \in S \times \Sigma \times S \mid n \in \mathbb{N} \wedge \\ \exists (\pi_1, \ldots, \pi_n) \in (\Pi_{i-1})^n.\ \mathrm{vst}((s^c, \sigma, s^r), \pi_1, \ldots, \pi_n)\}. \end{aligned}$$

Thus we have defined a chain: $\Pi_0 \subseteq \Pi_1 \subseteq \Pi_2 \subseteq \cdots \subseteq S \times \Sigma \times S$, and we define the set of states as $\Pi \triangleq \bigcup_i \Pi_i \subseteq S \times \Sigma \times S$, which is finite. Then, $(s^c_{n+1}, \sigma^{(n)}, s_{n+1}) \in \Pi$ iff the T-transition (2) exists.

Transition function of NBTA The transition function $\theta : \bigcup_{n\in\mathbb{N}}(\Sigma^{(n)} \times \Pi^n) \to 2^{\Pi}$ is defined by

$$\theta(\sigma^{(n)}, (\pi_1, \ldots, \pi_n)) \triangleq \big\{(s^c, \sigma, s^r) \in S\times\Sigma\times S \;\big|\; \mathrm{vst}((s^c, \sigma, s^r), \pi_1, \ldots, \pi_n)\big\}.$$

This ensures that a tree $\sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ is accepted by an A-state (s^c, σ, s^r) iff a transition $s^c \xrightarrow{\langle\sigma\ \lfloor t_1\rfloor\cdots\lfloor t_n\rfloor} s^r \xrightarrow{\sigma\rangle} s'$ exists; any $t \in \mathcal{T}_\Sigma$ is accepted by at least one state of A (though $F(s')$ is not necessarily defined).

Rules We define $Q = S \times S \times \Gamma$. For every $(s^c, \sigma, s^r) \in \Pi$, $\gamma \in \Gamma$, and $\pi_1, \ldots, \pi_n \in \Pi$ such that $(s^c, \sigma, s^r) \in \theta(\sigma, (\pi_1, \ldots, \pi_n))$, the following rule is added in R:

$$(s^c, s^r, \gamma)(\sigma(x_1, \ldots, x_n)) \to \mathcal{W}(\pi_1, \ldots, \pi_n)(\gamma) \qquad \langle\pi_1, \ldots, \pi_n\rangle$$

where $\mathcal{W} : \Pi^n \to \mathcal{A}(\Gamma, Q(X_n), \Delta)$ is defined from ρ_c and ρ_r as follows, and note that $\mathcal{W}(\pi_1, \ldots, \pi_n)(\gamma) \in E(Q(X_n), \Delta) = Rhs_{Q,\Delta}(X_n)$.

W.l.o.g., we assume that $Q(X_n) \cap (\Gamma \uplus \overline{\Gamma}) = \emptyset$. For $\pi_i = (s^c_i, \sigma_i, s^r_i)$ $(i \in [n])$, let $\delta_c(s^c_i, \sigma_i) = (s_i, p_i)$. Then

$$\mathcal{W}(\pi_1, \ldots, \pi_n) \triangleq \beta^r_n\big(\eta_n \uplus (\beta^c_n\Big(\cdots\Big(\beta^r_1\big(\eta_1 \uplus (\beta^c_1\alpha^\Gamma_\varepsilon)\big)\Big)\cdots\Big))\big) \tag{3}$$

$$\begin{aligned}
\text{where: } \alpha^\Gamma_\varepsilon &= [\gamma := \varepsilon]_{\gamma\in\Gamma} && \in \mathcal{A}(\Gamma, Q(X_n), \Delta)\\
\beta^c_i &\triangleq \overline{\rho_c(s^c_i, \sigma_i)} && \in \mathcal{A}(\overline{\Gamma}, \Gamma, \Delta)\\
\eta_i &\triangleq [\gamma := (s^c_i, s^r_i, \gamma)(x_i)]_{\gamma\in\Gamma} && \in \mathcal{A}(\Gamma, Q(X_n), \Delta)\\
\beta^r_i &\triangleq \rho_r(s^r_i, p_i, \sigma_i) && \in \mathcal{A}(\Gamma, \Gamma \uplus \overline{\Gamma}, \Delta).
\end{aligned}$$

The intuition for the above rule is as follows. Recall that the computation of T is top-down and bottom up, by call and return transitions, respectively. Also recall that state-transition functions δ_c and δ_r are already simulated by regular look-ahead, resulting $\pi_i = (s^c_i, \sigma_i, s^r_i)$ $(i \in [n])$. In (3), β^c_1 (with $\alpha^\Gamma_\varepsilon$) corresponds to the call transition by the root σ_1 of the leftmost subtree x_1:

$$\big(s^c_1,\ \Lambda,\ \alpha^\Gamma_\varepsilon\big) \xrightarrow{\langle\sigma_1} \big(s'_1,\ \big(p_1, \rho_c(s^c_1, \sigma_1)\alpha^\Gamma_\varepsilon\big)\Lambda,\ \alpha^\Gamma_\varepsilon\big)$$

where s'_1 and Λ are some state and stack, and the current evaluation function before this transition is $\alpha^\Gamma_\varepsilon$ because the configuration is just after the call transition for the parent σ. Then, η_1 is the recursive computation by T of the subtree x_1. And then $\beta^r_1\big(\eta_1 \uplus (\beta^c_1\alpha^\Gamma_\varepsilon)\big)$ corresponds to the return transition

$$\big(s^r_1,\ \big(p_1, \rho_c(s^c_1, \sigma_1)\alpha^\Gamma_\varepsilon\big)\Lambda,\ \eta_1\big) \xrightarrow{\sigma_1\rangle} \big(s^c_2,\ \Lambda,\ \beta^r_1\big(\eta_1 \uplus (\beta^c_1\alpha^\Gamma_\varepsilon)\big)\big)$$

where note that $\overline{\rho_c(s^c_1, \sigma_1)\alpha^\Gamma_\varepsilon} = \overline{\rho_c(s^c_1, \sigma_1)}\alpha^\Gamma_\varepsilon = \beta^c_1\alpha^\Gamma_\varepsilon$. Repeating this also for the remaining sibling subtrees $x_2, \ldots, x_n$, we obtain (3).

Initial sequences Finally, we define *Init* in a similar way to R:

$$Init = \bigcup_{\sigma\in\Sigma} \Big\{ \big(F(s')\beta^r_{s^r,p,\sigma}\big(\eta_{s^r} \uplus (\beta^c_\sigma \alpha^\Gamma_\varepsilon)\big),\ (s_0,\sigma,s^r)\big) \ \Big|\ (s_0,\sigma,s^r)\in \Pi,\quad s_0 \overset{\langle\sigma}{\Longrightarrow}_p s_1,\quad s^r \overset{\sigma\rangle}{\Longrightarrow}_p s' \in \mathrm{Dom}(F) \Big\}$$

where: (i) s_0 is the initial state of T, and s_1, s', and p are given by the transitions, (ii) $\beta^c_\sigma \triangleq \overline{\rho_c(s_0,\sigma)}$, (iii) $\eta_{s^r} \triangleq [\gamma := (s_0, s^r, \gamma)(x_1)]_{\gamma\in\Gamma}$, and (iv) $\beta^r_{s^r,p,\sigma} \triangleq \rho_r(s^r,p,\sigma)$.

The intuition is as follows. Let $t = \sigma(t_1,\ldots,t_n)$ be an input tree, and

$$(s_0,\varepsilon,\alpha^\Gamma_\varepsilon) \overset{\langle\sigma}{\Longrightarrow} (s_1,(p,\rho_c(s_0,\sigma)\alpha^\Gamma_\varepsilon),\alpha^\Gamma_\varepsilon) \overset{\lfloor t_1\rfloor}{\Longrightarrow} \cdots \cdots \overset{\lfloor t_n\rfloor}{\Longrightarrow} (\tilde{s}^r,(p,\rho_c(s_0,\sigma)\alpha^\Gamma_\varepsilon),\tilde{\eta}) \overset{\sigma\rangle}{\Longrightarrow} (\tilde{s}',\varepsilon,\rho_r(\tilde{s}^r,p,\sigma)(\tilde{\eta}\uplus(\beta^c_\sigma\alpha^\Gamma_\varepsilon)))$$

be the transition for $\lfloor t\rfloor$. Now t, $\tilde{s}^r$, $\tilde{\eta}$, $\tilde{s}'$, and $\rho_r(\tilde{s}^r,p,\sigma)$ respectively correspond to x_1, s^r, η_{s^r}, s', and $\beta^r_{s^r,p,\sigma} = \rho_r(s^r,p,\sigma)$ in the above definition. Here η_{s^r} involves $[\![(s_0,s^r,\gamma)]\!]_M(t)$, which is computed as in **Rules** above.

The equivalence of M and T follows from Lemmas 23 and 27. □

See Appendix D for an example of the above construction (Example 30).

4.3 Expressiveness and Decidability of Equivalence

Combining Lemmas 6 and 7, we can conclude:

Theorem 8. SRTSTs *and* yDT$^{\mathrm{R}}$s *are equi-expressive.*

Theorem 9. *For all* SRTST T*, there exists a bottom-up* SRTST T' *such that* $[\![T]\!] = [\![T']\!]$*. Thus,* SRTSTs *and bottom-up* SRTSTs *are equi-expressive.*

The STT$_{\mathrm{sur}}$-version of Theorem 9 is given in [1, Theorem 3.7], where, though, it was shown directly. As an immediate consequence of Lemma 7 and Theorem 3, we have:

Theorem 10. *Given* SRTSTs T_1 *and* T_2*, it is decidable whether* $[\![T_1]\!] = [\![T_2]\!]$.

Concluding remark By dropping the condition of sur from STT$_{\mathrm{sur}}$, we obtain the notion of an STT. We hope that our method can be applied to a similar result to Theorem 8, modified from SRTST to a similar model to STT whose input and output are ranked trees. If we further consider STT, whose input and output are *unranked* trees, it is not clear what model we should compare STT with, to solve the open problem of the decidability of the equivalence for STTs.

Acknowledgments We thank anonymous referees for useful comments. This work was supported by JSPS KAKENHI Grant Numbers JP17K00007, JP17H06099, JP18H04093, and JP18K11156.

References

1. Alur, R., D'Antoni, L.: Streaming tree transducers. J. ACM **64**(5), 31:1–31:55 (Aug 2017)
2. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing. pp. 202–211. STOC '04, ACM (2004)
3. Alur, R., Černý, P.: Streaming transducers for algorithmic verification of single-pass list-processing programs. In: Proc. of POPL. pp. 599–610. ACM (2011)
4. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2007)
5. Engelfriet, J.: Top-down tree transducers with regular look-ahead. Mathematical systems theory **10**(1), 289–303 (Dec 1976)
6. Engelfriet, J.: Some open questions and recent results on tree transducers and tree languages. In: Formal Language Theory, pp. 241 – 286. Academic Press (1980)
7. Engelfriet, J., Maneth, S.: Macro tree transducers, attribute grammars, and MSO definable tree translations. Information and Computation **154**(1), 34 – 91 (1999)
8. Engelfriet, J., Maneth, S.: The equivalence problem for deterministic MSO tree transducers is decidable. Information Processing Letters **100**(5), 206 – 212 (2006)
9. Engelfriet, J., Rozenberg, G., Slutzki, G.: Tree transducers, l systems, and two-way machines. Journal of Computer and System Sciences **20**(2), 150 – 202 (1980)
10. Filiot, E., Reynier, P.A.: Copyful streaming string transducers. In: Reachability Problems. pp. 75–86. Springer (2017)
11. Nakano, K., Mu, S-C.: A pushdown machine for recursive XML processing. In: Proc. of APLAS. pp. 340–356. Springer (2006)
12. Seidl, H., Maneth, S., Kemper, G.: Equivalence of deterministic top-down tree-to-string transducers is decidable. In: Proc. of FOCS. pp. 943–962 (2015)
13. Seidl, H., Maneth, S., Kemper, G.: Equivalence of deterministic top-down tree-to-string transducers is decidable. J. ACM **65**(4), 21:1–21:30 (Apr 2018)
14. Staworko, S., Laurence, G., Lemay, A., Niehren, J.: Equivalence of deterministic nested word to word transducers. In: Proc. of Fundamentals of Computation Theory. pp. 310–322. Springer (2009)
15. Takahashi, Y., Nakano, K.: Evaluating an algorithm deciding equivalence of deterministic top-down tree-to-string transducers (in Japanese). Computer Sofware **35**(4), 52–71 (2018)

A $\mathbf{SRTST_{sur}}$, $\mathbf{STST_{sur}}$, and MSO-definability

Here, as an immediate corollary of results in [1], we explain that MSO-definable ranked-tree-to-string transducers and $\mathrm{SRTST}_{\mathrm{sur}}$s are equi-expressive.

For the notions in this section, see [1], but note that, in [1]:

(i) the word "tree" means a well-matched nested word, rather than well-labeled nested word,
(ii) ranked trees are considered just as binary trees, just for presentation,
(iii) the notion of a nested word (in [1]) consists also of *internal symbols*, and ranked trees are embedded in nested words that contain no internal symbol (this is the reason why we omit the notion of an internal symbol from our notion of a nested word),

(iv) strings are embedded in nested words as strings of internal symbols.

Here we denote by $\langle\Sigma\rangle^{\mathrm{i}}$ the set of well-nested words (with internal symbols) in [1] and denote by $\langle\Sigma\rangle$ the set of well-nested words defined in the current paper. There is an obvious embedding $e^{\mathrm{i}} : \langle\Sigma\rangle \to \langle\Sigma\rangle^{\mathrm{i}}$. Also we write $e^{\mathrm{str}} : \Delta^* \to \langle\Delta\rangle^{\mathrm{i}}$ for the embedding of (iv) above. Given $f : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \Delta^*$, for $\mathrm{X} = \mathrm{STT}_{\mathrm{sur}}/\ \mathrm{MSO}$, we say f is X-*definable* if $e^{\mathrm{str}} \circ f : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \Delta^* \hookrightarrow \langle\Delta\rangle^{\mathrm{i}}$ is X-definable. Also, given $f : \lfloor\mathcal{T}_\Sigma\rfloor \rightharpoonup \Delta^*$, f is called MSO-*definable* if there exists MSO-definable $g : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \langle\Delta\rangle^{\mathrm{i}}$ such that $e^{\mathrm{str}} \circ f = g|_{\lfloor\mathcal{T}_\Sigma\rfloor}$.

In [1], the authors considered a variation of the notion of an $\mathrm{STT}_{\mathrm{sur}}$, i.e., that of an $\mathrm{STST}_{\mathrm{sur}}$, whose input and output are well-matched nested words and strings, respectively. (Recall that both the input and output of an $\mathrm{STT}_{\mathrm{sur}}$ are well-matched nested words.) We can restrict the notion of an SRTST with single-use restriction, which we call an $\mathrm{SRTST}_{\mathrm{sur}}$. The syntax for $\mathrm{SRTST}_{\mathrm{sur}}$s is the same as that for $\mathrm{STST}_{\mathrm{sur}}$s except that: (i) the input in the semantics is restricted to ranked nested words, and (ii) since the input does not involve internal symbols, we eliminate redundant notions (an internal state-transition function and an internal variable-update function) from the definition.

Corollary 11 ([1]). *Given $g : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \Delta^*$, g is* MSO-*definable iff g is* $\mathrm{STST}_{\mathrm{sur}}$-*definable. Therefore, given $f : \lfloor\mathcal{T}_\Sigma\rfloor \rightharpoonup \Delta^*$, f is* MSO-*definable iff f is* $\mathrm{SRTST}_{\mathrm{sur}}$-*definable.*

Proof. The latter part trivially follows from the former part. The former part is nothing but a combination of the following two results: Given $g : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \Delta^*$, g is $\mathrm{STT}_{\mathrm{sur}}$-definable iff g is $\mathrm{STST}_{\mathrm{sur}}$-definable [1, Theorem 3.18]. Given $g : \langle\Sigma\rangle^{\mathrm{i}} \rightharpoonup \langle\Delta\rangle^{\mathrm{i}}$, g is MSO-definable iff g is $\mathrm{STT}_{\mathrm{sur}}$-definable [1, Theorem 4.6]. □

B Lemmas for Section 4.1

Lemma 12. *Given a* $\mathrm{yDT}^{\mathrm{R}}$ $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$*, there exists a* $\mathrm{yDT}^{\mathrm{R}}$ $M' = (Q, \Sigma, \Delta, \mathit{Init}', R')$ *such that* $[\![M]\!] = [\![M']\!]$ *and M' satisfies the following conditions:*

(i) For any tree $\sigma(t_1, \ldots, t_n)$ and $q \in Q$,

$\sigma(t_1, \ldots, t_n) \in \mathrm{Dom}([\![q]\!]_{M'}) \iff$
there exists a $(q, \sigma, (L_1, \ldots, L_n))$-rule in R' such that $t_i \in L_i$ for every $i \in [n]$.

(ii) $\mathrm{Dom}(M') = \biguplus_{(\tau, L) \in \mathit{Init}'} L$.

Proof. Let $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$. We define $M' = (Q, \Sigma, \Delta, \mathit{Init}', R')$ as follows.

$$\mathit{Init}' \triangleq \Big\{ \ (\tau, L') \ \Big| \ (\tau, L) \in \mathit{Init}, \quad L' = L \cap \bigcap_{q' \in Q, q'(x_1) \text{ occurs in } \tau} \mathrm{Dom}([\![q']\!]_M) \neq \emptyset \Big\}$$

$$R' \triangleq \Big\{ \quad q(\sigma(x_1,\dots,x_n)) \to \tau \qquad \langle L'_1,\dots,L'_n\rangle \quad \Big| \\ \big(\ q(\sigma(x_1,\dots,x_n)) \to \tau \qquad \langle L_1,\dots,L_n\rangle\ \big) \in R, \\ L'_i \quad = \quad L_i \cap \bigcap_{q'\in Q, q'(x_i) \text{ occurs in } \tau} \mathrm{Dom}([\![q']\!]_M) \quad \neq \quad \emptyset \Big\}$$

Above, $\mathrm{Dom}([\![q']\!]_M)$ is (effectively) a regular tree language (Corollary 2.7 in [5]) and hence so are L' and L'_i.

First let us check that: (i) for $(\tau_1, L'_1) \neq (\tau_2, L'_2)$ in Init', we have $L'_1 \cap L'_2 = \emptyset$, and (ii) for $(q,\sigma,\tau_1,(L'_{1,1},\dots,L'_{1,n})) \neq (q,\sigma,\tau_2,(L'_{2,1},\dots,L'_{2,n}))$ in R', we have $L'_{1,i} \cap L'_{2,i} = \emptyset$ for some i. For (i), suppose $(\tau_1, L'_1) \neq (\tau_2, L'_2)$, and for each i let L_i be a witness for $L'_i \in \mathit{Init}'$. In the case where $\tau_1 = \tau_2$, we must have $L'_1 \neq L'_2$, which implies $L_1 \neq L_2$ (by contraposition) and hence $(\tau_1, L_1) \neq (\tau_2, L_2)$. In the case where $\tau_1 \neq \tau_2$, trivially $(\tau_1, L_1) \neq (\tau_2, L_2)$. Thus in any case we have $(\tau_1, L_1) \neq (\tau_2, L_2)$, and hence $L'_1 \cap L'_2 \subseteq L_1 \cap L_2 = \emptyset$. We can check (ii) in the same way.

Now, for any tree $\sigma(t_1,\dots,t_n)$ and $q \in Q$,

$\sigma(t_1,\dots,t_n) \in \mathrm{Dom}([\![q]\!]_M) \iff$
there exists a $(q,\sigma,(L'_1,\dots,L'_n))$-rule in R' such that $t_i \in L'_i$ for every $i \in [n]$.

Thus the item (i) is equivalent to the following:

$$\sigma(t_1,\dots,t_n) \in \mathrm{Dom}([\![q]\!]_{M'}) \iff \sigma(t_1,\dots,t_n) \in \mathrm{Dom}([\![q]\!]_M) \qquad \text{for all } q \in Q.$$

This and the following

$$[\![q]\!]_{M'}(\sigma(t_1,\dots,t_n)) \quad = \quad [\![q]\!]_M(\sigma(t_1,\dots,t_n)) \qquad \text{for all } q \in Q$$

can be shown easily by induction on $\sigma(t_1,\dots,t_n)$. From these, the item (ii) (which is equivalent to $\mathrm{Dom}(M') = \mathrm{Dom}(M)$) and $[\![M']\!] = [\![M]\!]$ immediately follow. □

Lemma 13. *Let $M = (Q,\Sigma,\Delta,\mathit{Init},R)$ be a* yDT$^{\mathrm{R}}$*, $n \in \mathbb{N}$, $t_1,\dots,t_n \in \mathcal{T}_\Sigma$, $\Gamma' \subseteq Q(X_n)$, and $\tau \in E(\Gamma',\Delta) \subseteq \mathit{Rhs}_{Q,\Delta}(X_n)$. Then if $t_i \in \mathrm{Dom}([\![q]\!]_M)$ for every $q(x_i) \in \Gamma'$, we have*

$$[\![\tau]\!]_M(t_1,\dots,t_n) = \tau\big[q(x_i) := [\![q]\!]_M(t_i)\big]_{q(x_i)\in\Gamma'}\ .$$

Proof. Trivial (induction on the length of τ). □

Let $M = (Q,\Sigma,\Delta,\mathit{Init},R)$ be a yDT$^{\mathrm{R}}$ that satisfies the two properties of M' in Lemma 12, and $T = (S,\Sigma,\Delta,P,s_0,\Gamma,F,\delta_c,\delta_r,\rho_c,\rho_r)$ be the SRTST constructed by the proof of Lemma 6 from M.

The next lemma describes how the semantics of M and that of T are related. We define

$$[\![q]\!]^{\varepsilon}_M(t) \triangleq \begin{cases} [\![q]\!]_M(t) & (t \in \mathrm{Dom}([\![q]\!]_M)) \\ \varepsilon & (\text{otherwise}). \end{cases}$$

Lemma 14. *For any $t \in \mathcal{T}_\Sigma$ and configuration $(\boldsymbol{\pi}, \Lambda, \alpha)$ where $|\boldsymbol{\pi}| \neq m$, we have $(\boldsymbol{\pi}, \Lambda, \alpha) \stackrel{\lfloor t \rfloor}{\Longrightarrow} (\boldsymbol{\pi} \,||\, \hat{\theta}(t),\ \Lambda,\ \alpha')$ with*

$$\begin{aligned}\alpha' = {} & \big[q(x_k) := \alpha(q(x_k))\big]_{q\in Q,\, 1\le k\le |\boldsymbol{\pi}|} \uplus \big[q(x_{|\boldsymbol{\pi}|+1}) := [\![q]\!]^{\varepsilon}_M(t)\big]_{q\in Q} \\ & \uplus \big[q(x_k) := \varepsilon\big]_{q\in Q,\, |\boldsymbol{\pi}|+1<k\le m}\ .\end{aligned}$$

Proof. The proof proceeds by induction on t. Let $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$. The transition from $(\boldsymbol{\pi}, \Lambda, \alpha)$ by $\lfloor t \rfloor$ is as follows, where we use induction hypothesis:

$$\begin{aligned}
& (\boldsymbol{\pi}, \Lambda, \alpha) \\
\stackrel{\langle\sigma}{\Longrightarrow}\ & \big((), (\boldsymbol{\pi}, \alpha)\Lambda, \alpha_\varepsilon\big) \\
\stackrel{\lfloor t_1 \rfloor}{\Longrightarrow}\ & \Big(\big(\hat{\theta}(t_1)\big),\ (\boldsymbol{\pi}, \alpha)\Lambda,\ [q(x_1) := [\![q]\!]^{\varepsilon}_M(t_1)]_{q\in Q}\ \uplus\ [q(x_k) := \varepsilon]_{q\in Q,\, 1<k\le m}\Big) \\
& \cdots \\
\stackrel{\lfloor t_n \rfloor}{\Longrightarrow}\ & \Big(\big(\hat{\theta}(t_1), \ldots, \hat{\theta}(t_n)\big),\ (\boldsymbol{\pi}, \alpha)\Lambda, \\
& \qquad [q(x_k) := [\![q]\!]^{\varepsilon}_M(t_k)]_{q\in Q, 1\le k\le n}\ \uplus\ [q(x_k) := \varepsilon]_{q\in Q,\, n<k\le m}\Big) \\
\stackrel{\sigma\rangle}{\Longrightarrow}\ & \big(\boldsymbol{\pi} \,||\, \theta(\sigma, \boldsymbol{\pi}''),\ \Lambda,\ \rho_r\big(\boldsymbol{\pi}'', \boldsymbol{\pi}, \sigma\big)\alpha''\big)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{\pi}'' &\triangleq \big(\hat{\theta}(t_1), \ldots, \hat{\theta}(t_n)\big) \\
\alpha'' &\triangleq \big[q(x_k) := [\![q]\!]^{\varepsilon}_M(t_k)\big]_{q\in Q, 1\le k\le n}\ \uplus\ \big[q(x_k) := \varepsilon\big]_{q\in Q,\, n<k\le m}\ \uplus\ \overline{\alpha} \\
\overline{\alpha} &= \big[\overline{q(x_k)} := \alpha(q(x_k))\big]_{q\in Q, 1\le k\le m}\ .
\end{aligned}$$

Here, as required, we have $\theta\big(\sigma, \boldsymbol{\pi}''\big) = \hat{\theta}(t)$ by the definition of $\hat{\theta}$ (for DBTA).

By the definition of ρ_r given in Section 4.1, we have

$$\begin{aligned}
& \rho_r\big(\boldsymbol{\pi}'', \boldsymbol{\pi}, \sigma\big)\alpha'' \\
= {} & \begin{pmatrix} \big[q(x_k) := \overline{q(x_k)}\big]_{q\in Q,\, 1\le k\le |\boldsymbol{\pi}|} \\ \uplus\ \big[q(x_{|\boldsymbol{\pi}|+1}) := \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\big]_{q\in Q} \\ \uplus\ \big[q(x_k) := \varepsilon\big]_{q\in Q,\, |\boldsymbol{\pi}|+1<k\le m} \end{pmatrix} \alpha'' \\
= {} & \big[q(x_k) := \overline{q(x_k)}\alpha''\big]_{q\in Q,\, 1\le k\le |\boldsymbol{\pi}|} \\
& \uplus\ \big[q(x_{|\boldsymbol{\pi}|+1}) := \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha''\big]_{q\in Q} \\
& \uplus\ \big[q(x_k) := \varepsilon\alpha''\big]_{q\in Q,\, |\boldsymbol{\pi}|+1<k\le m} \\
= {} & \big[q(x_k) := \alpha(q(x_k))\big]_{q\in Q,\, 1\le k\le |\boldsymbol{\pi}|} \\
& \uplus\ \big[q(x_{|\boldsymbol{\pi}|+1}) := \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha''\big]_{q\in Q} \\
& \uplus\ \big[q(x_k) := \varepsilon\big]_{q\in Q,\, |\boldsymbol{\pi}|+1<k\le m}\ .
\end{aligned}$$

Then the remaining to show is that, for any $q \in Q$, $\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha'' = [\![q]\!]_M(t)$ if $t \in \mathrm{Dom}([\![q]\!]_M)$; and $\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha'' = \varepsilon$ if $t \notin \mathrm{Dom}([\![q]\!]_M)$.

Since M satisfies the property (i) in Lemma 12 and since A is a DBTA, $t \in \mathrm{Dom}([\![q]\!]_M)$ iff $(q, \sigma, \boldsymbol{\pi}'')$-rule is defined, and furthermore if this is the case, then, for any $q'(x_k)$ in

$$\Gamma' \quad \triangleq \quad \{q'(x_k) \in Q(X_n) \mid q'(x_k) \text{ occurs in } \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'')\},$$

$[\![q']\!]_M(t_k)$ is defined and hence $[\![q']\!]^{\varepsilon}_M(t_k) = [\![q']\!]_M(t_k)$. Now, recall the definition of $\underline{\mathrm{rhs}}$:

$$\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'') \triangleq \begin{cases} \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'') & ((q, \sigma, \boldsymbol{\pi}'')\text{-rule is defined}) \\ \varepsilon & (\text{otherwise}). \end{cases}$$

Then, if $t \in \mathrm{Dom}([\![q]\!]_M)$,

$$\begin{aligned} \underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha'' &= \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'')\alpha'' \\ &= \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'')\big[q'(x_k) := [\![q']\!]^{\varepsilon}_M(t_k)\big]_{q'(x_k) \in \Gamma'} \\ &= \mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'')\big[q'(x_k) := [\![q']\!]_M(t_k)\big]_{q'(x_k) \in \Gamma'} \\ &= [\![\mathrm{rhs}(q, \sigma, \boldsymbol{\pi}'')]\!]_M(t_1, \ldots, t_n) \qquad \text{(by Lemma 13)} \\ &= [\![q]\!]_M(t) \end{aligned}$$

and if $t \notin \mathrm{Dom}([\![q]\!]_M)$,

$$\underline{\mathrm{rhs}}(q, \sigma, \boldsymbol{\pi}'')\alpha'' = \varepsilon\alpha'' = \varepsilon.$$

□

Lemma 15. *The domain of $[\![T]\!]$ is isomorphic to the domain of $[\![M]\!]$ by $\lfloor - \rfloor$.*

Proof. We have:

$$\begin{aligned} t \in \mathrm{Dom}(M) &\iff t \in \bigcup_{(\tau,\pi) \in Init} \mathcal{L}(\pi) && \text{(by Lemma 12 - (ii))} \\ &\iff \hat{\theta}(t) \in \mathrm{Dom}(Init) \quad (t \in \mathcal{L}(\pi) \iff \hat{\theta}(t) = \pi \text{ by definition}) \\ &\iff (\hat{\theta}(t)) \in \mathrm{Dom}(F) && \text{(by the definition of } F) \\ &\iff \lfloor t \rfloor \in \mathrm{Dom}(T). && \text{(by Lemma 14)} \end{aligned}$$

□

Lemma 16. $[\![T]\!](\lfloor t \rfloor) = [\![M]\!](t)$ *for every* $t \in \mathrm{Dom}(M)$.

Proof. By Lemma 14, we have $\big((), \varepsilon, \alpha_\varepsilon\big) \overset{\lfloor t \rfloor}{\Longrightarrow} \big((\hat{\theta}(t)), \varepsilon, \alpha'\big)$ with

$$\alpha' \quad = \quad \big[q(x_1) := [\![q]\!]^{\varepsilon}_M(t)\big]_{q \in Q} \ \uplus \ \big[q(x_k) := \varepsilon\big]_{q \in Q, 1 < k \leq m} .$$

Let $t = \sigma(t_1, \ldots, t_n)$. Since $t \in \mathrm{Dom}(M)$ and M satisfies the property (ii) in Lemma 12, $\mathit{Init}(\hat{\theta}(t))$ is defined, and furthermore for any $q(x_1)$ in

$$\Gamma' \quad \triangleq \quad \{q(x_1) \in Q(X_1) \mid q(x_1) \text{ occurs in } \mathit{Init}(\hat{\theta}(t))\},$$

$[\![q]\!]_M(t)$ is defined, so that $[\![q]\!]^{\varepsilon}_M(t) = [\![q]\!]_M(t)$. Then

$$\begin{aligned}
[\![T]\!](\lfloor t \rfloor) &= F((\hat{\theta}(t)))\alpha' \\
&= \mathit{Init}(\hat{\theta}(t))\big[q(x_1) := [\![q]\!]^{\varepsilon}_M(t)\big]_{q(x_1)\in\Gamma'} \\
&\text{(where recall that } F((\hat{\theta}(t))) = \mathit{Init}(\hat{\theta}(t)) \in \mathit{Rhs}_{Q,\Delta}(X_1) \big(\subseteq \mathit{Rhs}_{Q,\Delta}(X_m)\big)\text{)} \\
&= \mathit{Init}(\hat{\theta}(t))\big[q(x_1) := [\![q]\!]_M(t)\big]_{q(x_1)\in\Gamma'} \\
&= [\![\mathit{Init}(\hat{\theta}(t))]\!]_M(t) \qquad \text{(by Lemma 13)} \\
&= [\![M]\!](t)
\end{aligned}$$

□

C Lemmas for Section 4.2

Let $T = (S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$ be an SRTST and $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$ be the yDT$^{\mathrm{R}}$ constructed in the proof of Lemma 7 from T.

First recall that, for an SRTST T, $s_0, \ldots, s_n \in S$, and nested words $w_1 \ldots, w_n$,

$$s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \tag{4}$$

iff

$$(s_0, \Lambda_0, \alpha_0) \overset{w_1}{\Longrightarrow} (s_1, \Lambda_1, \alpha_1) \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} (s_n, \Lambda_n, \alpha_n) \text{ for some } \Lambda_i \text{ and } \alpha_i \tag{5}$$

by definition. Note that this notation is *essentially ambiguous* if we divide this as a composition of relations; i.e., given

$$s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_i}{\Longrightarrow} s_i \quad \text{and} \quad s_i \overset{w_{i+1}}{\Longrightarrow} s_{i+1} \overset{w_{i+2}}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \tag{6}$$

(4) (namely (5)) does not necessarily hold, (although the converse always holds). Throughout this paper, when we write (4), this always means (5), and does not mean (6) for any i, except the case where the ambiguity can be avoided by Lemma 17 - (iv), (v) below.

Lemma 17. *(i)* $s \overset{\langle\sigma}{\Longrightarrow} s'$ *iff* $s \overset{\langle\sigma}{\Longrightarrow}_p s'$ *for some (unique)* $p \in P$.

(ii) If $s \overset{\langle\sigma}{\Longrightarrow} s_i$ $(i = 1, 2)$, *then* $s_1 = s_2$. *Also, for a well-matched nested word* w, *if* $(s, \Lambda_i, \alpha_i) \overset{w}{\Rightarrow} (s'_i, \Lambda'_i, \alpha'_i)$ $(i = 1, 2)$, *then* $s'_1 = s'_2$ *and* $\Lambda_i = \Lambda'_i$ *for each* $i = 1, 2$. *Especially, if* $w_1 \cdots w_n = w'_1 \cdots w'_{n'}$ *is a well-matched nested word and if* $s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n$ *and* $s_0 \overset{w'_1}{\Longrightarrow} s'_1 \overset{w'_2}{\Longrightarrow} \cdots \overset{w'_{n'}}{\Longrightarrow} s'_{n'}$, *then* $s_n = s'_{n'}$.

(iii) *If* $w_1 \cdots w_n$ *is a well-matched nested word and* $s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n$, *then for any* Λ_0 *and* α_0, *there exist* $\Lambda_1, \ldots, \Lambda_{n-1}$ *and* $\alpha_1, \ldots, \alpha_n$ *such that*

$$(s_0, \Lambda_0, \alpha_0) \overset{w_1}{\Longrightarrow} (s_1, \Lambda_1, \alpha_1) \overset{w_2}{\Longrightarrow} \cdots \overset{w_{n-1}}{\Longrightarrow} (s_{n-1}, \Lambda_{n-1}, \alpha_{n-1}) \overset{w_n}{\Longrightarrow} (s_n, \Lambda_0, \alpha_n).$$

(iv) *Let* w *be a well-matched nested word. Then,*

$$\begin{aligned}
& s \overset{\langle\sigma}{\Longrightarrow} s_0 \quad \textit{and} \quad s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \\
\textit{iff} \quad & s \overset{\langle\sigma}{\Longrightarrow}_p s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \overset{\sigma\rangle}{\Longrightarrow}_p s' \qquad \textit{for some (unique) } p, s' \\
\textit{iff} \quad & s \overset{\langle\sigma}{\Longrightarrow} s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \\
\textit{iff} \quad & s \overset{\langle\sigma}{\Longrightarrow} s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \overset{\sigma\rangle}{\Longrightarrow} s' \qquad \textit{for some (unique) } s'.
\end{aligned}$$

(v) *Let* $w_1 \cdots w_n$ *and* $w'_1 \cdots w'_{n'}$ *be well-matched nested words. Then,*

$$\begin{aligned}
& s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \overset{w'_1}{\Longrightarrow} s'_1 \overset{w'_2}{\Longrightarrow} \cdots \overset{w'_{n'}}{\Longrightarrow} s'_{n'} \\
\textit{iff} \quad & s_0 \overset{w_1}{\Longrightarrow} s_1 \overset{w_2}{\Longrightarrow} \cdots \overset{w_n}{\Longrightarrow} s_n \quad \textit{and} \quad s_n \overset{w'_1}{\Longrightarrow} s'_1 \overset{w'_2}{\Longrightarrow} \cdots \overset{w'_{n'}}{\Longrightarrow} s'_{n'}
\end{aligned}$$

Proof. (i) Clear.
(ii) The first one is clear by (i). The second one can be shown by induction on w, and the last one immediately follows from the second one.
(iii) This follows immediately from (ii) since δ^* is always defined for well-matched nested words.
(iv) The first "iff" is obvious by (i). The second "iff" follows from (iii). The last "iff" is obvious since δ^* is always defined for well-matched nested words.
(v) This follows from (iii).

Lemma 18. *(i) For any* $t = \sigma(t_1, \ldots, t_n)$ *and* $\pi \in \Pi$, *if* $t \in \mathcal{L}(\pi)$ *then* $\pi = (s^c, \sigma, s^r)$ *for some* s^c, s^r.

(ii) For $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ *and* $s^c, s^r \in S$, *there exists a transition* $s^c \overset{\langle\sigma}{\Longrightarrow} s_1 \overset{\lfloor t_1 \rfloor \cdots \lfloor t_n \rfloor}{\Longrightarrow} s^r \overset{\sigma\rangle}{\Longrightarrow} s'$ *iff* $(s^c, \sigma, s^r) \in \Pi$ *and* $t \in \mathcal{L}((s^c, \sigma, s^r))$.

(iii) For any $\pi \in \Pi$, $\mathcal{L}(\pi) \neq \emptyset$.

Proof. For any $t = \sigma(t_1, \ldots, t_n)$ and $\pi \in \Pi$, by the definition of NBTA and θ,

$$\begin{aligned}
& t \in \mathcal{L}(\pi) \\
& \Longleftrightarrow \\
& \pi \in \theta(\sigma, (\pi_1, \ldots, \pi_n)),\ t_1 \in \mathcal{L}(\pi_1),\ \ldots,\ t_n \in \mathcal{L}(\pi_n) \text{ for some } \pi_1, \ldots, \pi_n \in \Pi \\
& \Longleftrightarrow \\
& \pi = (s^c, \sigma, s^r),\ \mathrm{vst}((s^c, \sigma, s^r), \pi_1, \ldots, \pi_n),\ t_1 \in \mathcal{L}(\pi_1),\ \ldots,\ t_n \in \mathcal{L}(\pi_n) \\
& \text{for some } s^c, s^r \in S \text{ and } \pi_1, \ldots, \pi_n \in \Pi\ .
\end{aligned}$$

This proves the item (i) and is used below. We now prove the item (ii) by induction on t.

The "if" part: Let $t \in \mathcal{L}((s^c, \sigma, s^r))$; then $\mathrm{vst}((s^c, \sigma, s^r), \pi_1, \ldots, \pi_n)$ and $t_i \in \mathcal{L}(\pi_i)$ for some $\pi_1, \ldots, \pi_n \in \mathit{\Pi}$. By the vst condition, for some $s'_1, \ldots, s'_n \in S$ and $p, p_1, \ldots, p_n \in P$ we have a transition:

$$s^c \overset{\langle\sigma}{\Longrightarrow}_p s_1 \overset{\langle\sigma_1}{\Longrightarrow}_{p_1} s'_1, \quad s^r_1 \overset{\sigma_1\rangle}{\Longrightarrow}_{p_1} s_2 \overset{\langle\sigma_2}{\Longrightarrow}_{p_2} s'_2, \quad s^r_2 \overset{\sigma_2\rangle}{\Longrightarrow}_{p_2} s_3 \cdots$$
$$\cdots s_n \overset{\langle\sigma_n}{\Longrightarrow}_{p_n} s'_n, \quad s^r_n \overset{\sigma_n\rangle}{\Longrightarrow}_{p_n} s^r$$

where s_i and s^r_i are the first and third components of π_i, respectively. Let $t_i = \sigma_i(t^i_1, \ldots, t^i_{l_i})$. Since $t_i \in \mathcal{L}(\pi_i) = \mathcal{L}((s_i, \sigma_i, s^r_i))$, by induction hypothesis, for each i there exists s''_i and transitions:

$$s_i \overset{\langle\sigma_i}{\Longrightarrow} s''_i \quad \text{and} \quad s''_i \overset{\lfloor t^i_1 \rfloor \cdots \lfloor t^i_{l_i} \rfloor}{\Longrightarrow} s^r_i \ .$$

Since $\overset{\langle\sigma_i}{\Longrightarrow}$ is deterministic, $s'_i = s''_i$. Thus we have obtained

$$s^c \overset{\langle\sigma}{\Longrightarrow}_p s_1 \overset{\langle\sigma_1}{\Longrightarrow}_{p_1} s'_1 \overset{\lfloor t^1_1 \rfloor \cdots \lfloor t^1_{l_1} \rfloor}{\Longrightarrow} s^r_1 \overset{\sigma_1\rangle}{\Longrightarrow}_{p_1} s_2 \overset{\langle\sigma_2}{\Longrightarrow}_{p_2} s'_2 \overset{\lfloor t^2_1 \rfloor \cdots \lfloor t^2_{l_2} \rfloor}{\Longrightarrow} s^r_2 \overset{\sigma_2\rangle}{\Longrightarrow}_{p_2} s_3 \cdots$$
$$\cdots s_n \overset{\langle\sigma_n}{\Longrightarrow}_{p_n} s'_n \overset{\lfloor t^n_1 \rfloor \cdots \lfloor t^n_{l_n} \rfloor}{\Longrightarrow} s^r_n \overset{\sigma_n\rangle}{\Longrightarrow}_{p_n} s^r$$

and hence by Lemma 17 - (iv), (v), we have

$$s^c \overset{\langle\sigma}{\Longrightarrow} s_1 \overset{\lfloor t_1 \rfloor \cdots \lfloor t_n \rfloor}{\Longrightarrow} s^r \overset{\sigma\rangle}{\Longrightarrow} s' \ .$$

The "only if" part just proceeds in the converse way to the above.

The item (iii) is clear from the above argument; π contains the root symbol of the required tree and the definition of $\mathit{\Pi}$ (and $\mathit{\Pi}_i$) ensures the existence of some "child" states $\pi_1, \ldots, \pi_n$, recursively. In fact we can show by induction on i that, for any $i \in \mathbb{N}$ and any $\pi \in \mathit{\Pi}_i$ there is a tree $t \in \mathcal{L}(\pi)$ whose height is at most i. □

Remark 19. Above, the item (iii) ensures that all the states of the NBTA A generate non-empty regular languages, as required always in the current paper. If we change the definition of $\mathit{\Pi}$ to $\mathit{\Pi} \triangleq S \times \mathit{\Sigma} \times S$, still we obtain an NBTA and the item (ii) holds; the current definition of $\mathit{\Pi}$ is nothing but the set of states $\pi \in S \times \mathit{\Sigma} \times S$ such that $\mathcal{L}(\pi) \neq \emptyset$.

Remark 20. Our construction of the NBTA A from the state-transitions δ_c, δ_r of T given in Section 4.2 and that of the state-transitions δ_c, δ_r of T from a DBTA A given in Section 4.1 are basically similar to the construction given in [2] except for the design of a state $\pi = (s^c, \sigma, s^r)$ in our constructions of the NBTA A: (i) Our style of a state π contains the information of σ, which is used in the definition of rules of T. (ii) We use s^r, which is a state *before* a return-transition δ_r while the construction in [2] uses a state *after* a return-transition δ_r. Since δ_r is a function, our information is "richer" than the other (i.e., given s^r, we can obtain "a state *after* a return-transition" simply as $\delta_r(s^r, p, \sigma)$), and we use s^r as an argument of ρ_r in the definitions of rules and *Init* of T.

Lemma 21. *For $s^c \in S$ and $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$, there exists exact one state $s^r \in S$ such that $(s^c, \sigma, s^r) \in \Pi$ and $t \in \mathcal{L}((s^c, \sigma, s^r))$.*

Proof. The uniqueness follows from Lemma 18 - (ii) and Lemma 17 - (ii). The existence also follows from Lemma 18 - (ii) since δ^* is always defined for well-matched nested words. □

Lemma 22. *For $(s^c, \sigma, s^r) \in \Pi$, $t \in \mathcal{L}((s^c, \sigma, s^r))$, and $\gamma \in \Gamma$, $[\![(s^c, s^r, \gamma)]\!]_M(t)$ is always defined. Hence, for any $(\tau, \pi) \in \mathit{Init}$ and $t \in \mathcal{L}(\pi)$, $[\![\tau]\!]_M(t)$ is always defined.*

Proof. The latter follows immediately from the former. We prove the former by induction on t.

Let $t = \sigma(t_1, \ldots, t_n)$. Since $t \in \mathcal{L}((s^c, \sigma, s^r))$, there exist $\pi_i = (s^c_i, \sigma_i, s^r_i) \in \Pi$ $(i = 1, \ldots, n)$ such that: $(s^c, \sigma, s^r) \in \theta(\sigma, (\pi_1, \ldots, \pi_n))$; $t_i \in \mathcal{L}(\pi_i)$ for each $i = 1, \ldots, n$; and $\mathrm{vst}((s^c, \sigma, s^r), (\pi_1, \ldots, \pi_n))$ holds. Then by the definition of R, there exists $\tau \triangleq \mathrm{rhs}((s^c, s^r, \gamma), \sigma, (\pi_1, \ldots, \pi_n))$, and $[\![(s^c, s^r, \gamma)]\!]_M(t) = [\![\tau]\!]_M(t_1, \ldots, t_n)$ if $[\![\tau]\!]_M(t_1, \ldots, t_n)$ is defined. Recall the definition of τ after (3) in the proof of Lemma 7. In τ, an occurrence of an element in $Q(X_n)$ is of the form $(s^c_i, s^r_i, \gamma')(x_i)$—given in η_i—for some $\gamma' \in \Gamma$, and then $[\![\tau]\!]_M(t_1, \ldots, t_n)$ involves $[\![(s^c_i, s^r_i, \gamma')]\!]_M(t_i)$, which is defined by induction hypothesis. Thus $[\![\tau]\!]_M(t_1, \ldots, t_n)$ is defined, and so is $[\![(s^c, s^r, \gamma)]\!]_M(t)$. □

Lemma 23. *The domain of M is isomorphic to the domain of T by $\lceil - \rceil$.*

Proof. Let $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$. It holds that

$$
\begin{aligned}
&\langle \sigma \ \lfloor t_1 \rfloor \cdots \lfloor t_n \rfloor \ \sigma \rangle \in \mathrm{Dom}(T) \\
\iff &(s_0, \varepsilon, \alpha_\varepsilon) \overset{\langle\sigma}{\Longrightarrow} (s_1, (p_1, \alpha_1), \alpha_\varepsilon) \xrightarrow{\lfloor t_1 \rfloor \ldots \lfloor t_n \rfloor} (s^r, (p_1, \alpha_1), \alpha) \overset{\sigma\rangle}{\Longrightarrow} (s', \varepsilon, \alpha') \\
&\wedge s' \in \mathrm{Dom}(F) \quad \text{where } s_1, p_1, \alpha_1, s^r, \alpha, s', \alpha' \text{ are given by the transitions} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by the definition of } [\![T]\!]) \\
\iff &(\tau, (s_0, \sigma, s^r)) \in \mathit{Init} \wedge t \in \mathcal{L}((s_0, \sigma, s^r)) \ \text{ for some } \tau, s^r \\
&\qquad\qquad \text{(by the definition of } \mathit{Init}\text{, Lemma 18 - (ii), and Lemma 17 - (iii))} \\
\iff &(\tau, (s^c, \sigma', s^r)) \in \mathit{Init} \wedge t \in \mathcal{L}((s^c, \sigma', s^r)) \wedge [\![\tau]\!]_M(t) \text{ is defined} \\
&\text{for some } \tau, s^c, \sigma', s^r \\
&\qquad\qquad\qquad\qquad \text{(by Lemma 22, the definition of } \mathit{Init}\text{, and Lemma 18-(i))} \\
\iff &t \in \mathrm{Dom}(M). \qquad\qquad\qquad\qquad \text{(by the definitions of } [\![M]\!] \text{ and } \mathit{Init})
\end{aligned}
$$

□

Lemma 24. *M is well-defined as a* yDT$^{\mathrm{R}}$*: i.e.,*

(i) for any $(\tau, \pi), (\tau', \pi') \in \mathit{Init}$, if $(\tau, \mathcal{L}(\pi)) \neq (\tau', \mathcal{L}(\pi'))$, then $\mathcal{L}(\pi) \cap \mathcal{L}(\pi') = \emptyset$,
(ii) for any $(q, \sigma, \tau, (\pi_1, \ldots, \pi_n)), (q, \sigma, \tau', (\pi'_1, \ldots, \pi'_n)) \in R$, if

$$(q, \sigma, \tau, (\mathcal{L}(\pi_1), \ldots, \mathcal{L}(\pi_n))) \quad \neq \quad (q, \sigma, \tau', (\mathcal{L}(\pi'_1), \ldots, \mathcal{L}(\pi'_n)))$$

then $\mathcal{L}(\pi_i) \cap \mathcal{L}(\pi'_i) = \emptyset$ for some $i \in [n]$.

Proof. On (i): Let us assume that there exists $t = \sigma(t_1, \ldots, t_n) \in \mathcal{L}(\pi) \cap \mathcal{L}(\pi')$ and show the contradiction. Since $t \in \mathcal{L}(\pi) \cap \mathcal{L}(\pi')$ and $(\tau, \pi), (\tau', \pi') \in \mathit{Init}$, each of π and π' is of the form (s_0, σ, s). By Lemma 21 there exists exactly one state of the form $(s_0, \sigma, s) \in \Pi$ such that $t \in \mathcal{L}((s_0, \sigma, s))$. Hence $\pi = \pi'$. Then $\tau = \tau'$ by the definition of *Init*, which contradicts $(\tau, \mathcal{L}(\pi)) \neq (\tau', \mathcal{L}(\pi'))$.

On (ii): Let us assume that there exists $t_i \in \mathcal{L}(\pi_i) \cap \mathcal{L}(\pi'_i)$ for each $i \in [n]$, and show the contradiction. First note that we have $\pi_i \neq \pi'_i$ for some $i \in [n]$—since otherwise by the definition of R we have $\tau = \tau'$, which contradicts the assumption—and let i_0 be the least such i; thus $\pi_{i_0} \neq \pi'_{i_0}$ and $\pi_i = \pi'_i$ for $i < i_0$. Let $\pi_i = (s^c_i, \sigma_i, s^r_i)$, $\pi'_i = (s'^c_i, \sigma_i, s'^r_i)$, and $t_i = \sigma_i(t^i_1, \ldots, t^i_{l_i})$, for each $i = 1, \ldots, n$, and $q = (s^c, s^r, \gamma)$. By the definition of R, $\mathrm{vst}((s^c, \sigma, s^r), \pi_1, \ldots, \pi_n)$ and $\mathrm{vst}((s^c, \sigma, s^r), \pi'_1, \ldots, \pi'_n)$ hold, and also we have $\pi_i = \pi'_i \in \mathcal{L}(t_i)$ for $i < i_0$ with Lemma 18 - (ii); therefore, we have

$$s^c \xrightarrow{\langle\sigma} s^c_1 \xrightarrow{\lfloor t_1 \rfloor} s^c_2 \xrightarrow{\lfloor t_2 \rfloor} \cdots s^c_{i_0-1} \xrightarrow{\lfloor t_{i_0-1} \rfloor} s^c_{i_0} \xrightarrow{\langle\sigma_{i_0} \lfloor t^{i_0}_1 \rfloor \cdots \lfloor t^{i_0}_{l_{i_0}} \rfloor} s^r_{i_0}$$

and

$$s^c \xrightarrow{\langle\sigma} s^c_1 \xrightarrow{\lfloor t_1 \rfloor} s^c_2 \xrightarrow{\lfloor t_2 \rfloor} \cdots s^c_{i_0-1} \xrightarrow{\lfloor t_{i_0-1} \rfloor} s'^c_{i_0} \xrightarrow{\langle\sigma_{i_0} \lfloor t^{i_0}_1 \rfloor \cdots \lfloor t^{i_0}_{l_{i_0}} \rfloor} s'^r_{i_0} \ .$$

Note that the above two sequences start from the same state s^c even if $i_0 = 1$. Hence, by using Lemma 17 - (ii) twice for $\langle\sigma$ and for $\lfloor t_1 \rfloor \cdots \lfloor t_{i_0-1} \rfloor$, we obtain $s^c_{i_0} = s'^c_{i_0}$, and then similarly we obtain $s^r_{i_0} = s'^r_{i_0}$, which contradicts $\pi_{i_0} \neq \pi'_{i_0}$. □

We extend $[\![-]\!]_M$ to $\mathcal{A}(\Gamma, Q(X_n), \Delta)$: for $\rho \in \mathcal{A}(\Gamma, Q(X_n), \Delta)$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$, we define

$$[\![\rho]\!]_M(t_1, \ldots, t_n) \quad \triangleq \quad \big[\gamma := [\![\rho(\gamma)]\!]_M(t_1, \ldots, t_n)\big]_{\gamma \in \Gamma} \quad \in \quad \mathcal{A}(\Gamma, \emptyset, \Delta)$$

if $[\![\rho(\gamma)]\!]_M(t_1, \ldots, t_n)$ is defined for every $\gamma \in \Gamma$, and $[\![\rho]\!]_M(t_1, \ldots, t_n)$ is not defined otherwise.

Lemma 25. *(i) For $e \in E(\Gamma, \Delta)$, $\alpha \in \mathcal{A}(\Gamma', \Gamma, \Delta)$, $\rho \in \mathcal{A}(\Gamma, Q(X_n), \Delta)$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$, if $[\![\rho]\!]_M(t_1, \ldots, t_n)$ is defined, then so are $[\![e\rho]\!]_M(t_1, \ldots, t_n)$ and $[\![\alpha\rho]\!]_M(t_1, \ldots, t_n)$, and*

$$\begin{array}{rcl} [\![e\rho]\!]_M(t_1, \ldots, t_n) & = & e\,[\![\rho]\!]_M(t_1, \ldots, t_n) \quad \in \quad \Delta^* \\ [\![\alpha\rho]\!]_M(t_1, \ldots, t_n) & = & \alpha\,[\![\rho]\!]_M(t_1, \ldots, t_n) \quad \in \quad \Delta^*. \end{array}$$

(ii) For $\rho_i \in \mathcal{A}(\Gamma_i, Q(X_n), \Delta)$ $(i = 1, 2)$ such that $\Gamma_1 \cap \Gamma_2 = \emptyset$ holds, and for $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$, if $[\![\rho_1]\!]_M(t_1, \ldots, t_n)$ and $[\![\rho_2]\!]_M(t_1, \ldots, t_n)$ are defined, then so is $[\![\rho_1 \uplus \rho_2]\!]_M(t_1, \ldots, t_n)$, and

$$[\![\rho_1 \uplus \rho_2]\!]_M(t_1, \ldots, t_n) \quad = \quad [\![\rho_1]\!]_M(t_1, \ldots, t_n) \ \uplus \ [\![\rho_2]\!]_M(t_1, \ldots, t_n).$$

Proof. (i) By induction on e and α, respectively.
(ii) By case analysis on an argument $\gamma \in \Gamma_1 \uplus \Gamma_2$.

Lemma 26. *For $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$, if*

$$(s^c, \Lambda, \alpha) \xrightarrow{\langle\sigma\, \lfloor t_1 \rfloor \ldots \lfloor t_n \rfloor} (s^r, (p, \alpha')\Lambda, \alpha^r)$$

then $[\![(s^c, s^r, \gamma)]\!]_M(t) = \alpha^r(\gamma)$ for every $\gamma \in \Gamma$. □

Proof. The proof proceeds by induction on t.

Let $t_i = \sigma_i(t_1^i, \ldots, t_{l_i}^i)$ and suppose that we have

$$\begin{aligned}
(s^c, \Lambda, \alpha) \xRightarrow{\langle\sigma} (s_1^c, \Lambda_1, \alpha_1) &\xrightarrow{\langle\sigma_1\, \lfloor t_1^1 \rfloor \ldots \lfloor t_{l_1}^1 \rfloor} (s_1^r, (p_1, \alpha_1')\Lambda_1, \alpha_1^r) \xRightarrow{\sigma_1\rangle} \\
(s_2^c, \Lambda_2, \alpha_2) &\xrightarrow{\langle\sigma_2\, \lfloor t_1^2 \rfloor \ldots \lfloor t_{l_2}^2 \rfloor} (s_2^r, (p_2, \alpha_2')\Lambda_2, \alpha_2^r) \xRightarrow{\sigma_2\rangle} \\
&\vdots \\
(s_n^c, \Lambda_n, \alpha_n) &\xrightarrow{\langle\sigma_n\, \lfloor t_1^n \rfloor \ldots \lfloor t_{l_n}^n \rfloor} (s_n^r, (p_n, \alpha_n')\Lambda_n, \alpha_n^r) \xRightarrow{\sigma_n\rangle} (s^r, (p, \alpha')\Lambda, \alpha^r)\,.
\end{aligned}$$

Then we have

$$\begin{array}{rcl}
\alpha_1 & = & \alpha_\varepsilon^\Gamma \\
\alpha_i' & = & \rho_c(s_i^c, \sigma_i)\alpha_i \\
\overline{\alpha_i'} & = & \overline{\rho_c(s_i^c, \sigma_i)\alpha_i} \quad = \quad \overline{\rho_c(s_i^c, \sigma_i)}\alpha_i \quad = \quad \beta_i^c\alpha_i
\end{array}$$

and hence, by the definition of T-transition,

$$\begin{aligned}
\alpha^r &= \rho_r(s_n^r, p_n, \sigma_n)\big(\alpha_n^r \uplus \overline{\alpha_n'}\big) \\
&= \beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\alpha_n)\big) \\
&= \beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\Big(\rho_r(s_{n-1}^r, p_{n-1}, \sigma_{n-1})\big(\alpha_{n-1}^r \uplus \overline{\alpha_{n-1}'}\big)\Big))\big) \\
&= \beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\Big(\beta_{n-1}^r\big(\alpha_{n-1}^r \uplus (\beta_{n-1}^c\alpha_{n-1})\big)\Big))\big) \\
&= \cdots \\
&= \beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\Big(\cdots\Big(\beta_1^r\big(\alpha_1^r \uplus (\beta_1^c\alpha_1)\big)\Big)\cdots\Big))\big) \\
&= \beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\Big(\cdots\Big(\beta_1^r\big(\alpha_1^r \uplus (\beta_1^c\alpha_\varepsilon^\Gamma)\big)\Big)\cdots\Big))\big).
\end{aligned}$$

Let $\pi_i \triangleq (s_i^c, \sigma_i, s_i^r)$. By Lemma 18 - (ii),

$$t_i \in \mathcal{L}(\pi_i). \tag{7}$$

Also, on (3) in the proof of Lemma 7, recall that

$$\eta_i \quad \triangleq \quad [\gamma := (s_i^c, s_i^r, \gamma)(x_i)]_{\gamma\in\Gamma} \quad \in \quad \mathcal{A}(\Gamma, Q(X_n), \Delta).$$

By induction hypothesis, $[\![\eta_i]\!]_M(t_1, \ldots, t_n)$ is defined and

$$[\![\eta_i]\!]_M(t_1, \ldots, t_n) = \alpha_i^r. \tag{8}$$

Then, for each $\gamma \in \Gamma$,

$$\begin{aligned}
&[\![(s^c, s^r, \gamma)]\!]_M(t) \\
&= [\![\mathrm{rhs}((s^c, s^r, \gamma), \sigma, (\pi_1, \ldots, \pi_n))]\!]_M(t_1, \ldots, t_n) && \text{(by (7))} \\
&= \left[\!\!\left[\gamma\left(\beta_n^r\big(\eta_n \uplus (\beta_n^c\Big(\cdots\Big(\beta_1^r\big(\eta_1 \uplus (\beta_1^c\alpha_\varepsilon^\Gamma)\big)\Big)\cdots\Big)\big)\big)\right)\right]\!\!\right]_M(t_1, \ldots, t_n) \\
&= \gamma\left(\beta_n^r\big(\alpha_n^r \uplus (\beta_n^c\Big(\cdots\Big(\beta_1^r\big(\alpha_1^r \uplus (\beta_1^c\alpha_\varepsilon^\Gamma)\big)\Big)\cdots\Big)\big)\big)\right) \\
&\qquad\qquad \text{(by Lemma 25, (8), and } [\![\alpha_\varepsilon^\Gamma]\!]_M(t_1, \ldots, t_n) = \alpha_\varepsilon^\Gamma) \\
&= \alpha^r(\gamma). && \square
\end{aligned}$$

Lemma 27. *$[\![M]\!](\lceil w \rceil) = [\![T]\!](w)$ for every $w \in \mathrm{Dom}(T)$.*

Proof. Let $t = \sigma(t_1, \ldots, t_n) = \lceil w \rceil \in \mathrm{Dom}(M) = \lceil \mathrm{Dom}(T) \rceil$, and let the transition by $\lfloor t \rfloor$ be

$$(s_0, \varepsilon, \alpha_\varepsilon^\Gamma) \xrightarrow{\langle \sigma \lfloor t_1 \rfloor \ldots \lfloor t_n \rfloor} (s^r, (p, \rho_c(s_0, \sigma)\alpha_\varepsilon^\Gamma), \alpha^r) \xRightarrow{\sigma\rangle} (s', \varepsilon, \alpha').$$

By Lemma 18 - (ii), $t \in \mathcal{L}((s_0, \sigma, s^r))$. Since $t \in \mathrm{Dom}(M)$, we have:

$$\begin{aligned}
&[\![M]\!](t) \\
&= [\![\mathit{Init}((s_0, \sigma, s^r))]\!]_M(t) \\
&= [\![F(s')\beta^r_{s^r,p,\sigma}\big(\eta_{s^r} \uplus (\beta_\sigma^c\alpha_\varepsilon^\Gamma)\big)]\!]_M(t) \\
&= F(s')\beta^r_{s^r,p,\sigma}\big([\![\eta_{s^r}]\!]_M(t) \uplus (\beta_\sigma^c\alpha_\varepsilon^\Gamma)\big) && \text{(by Lemma 25 and } [\![\alpha_\varepsilon^\Gamma]\!]_M(t) = \alpha_\varepsilon^\Gamma) \\
&= F(s')\beta^r_{s^r,p,\sigma}\big(\alpha^r \uplus (\beta_\sigma^c\alpha_\varepsilon^\Gamma)\big) && \text{(by Lemma 26)} \\
&= F(s')\rho_r(s^r, p, \sigma)\big(\alpha^r \uplus \overline{\rho_c(s_0, \sigma)}\alpha_\varepsilon^\Gamma\big).
\end{aligned}$$

On the other hand, by the definition of T-transition, we have

$$\begin{aligned}
[\![T]\!](w) &= F(s')\alpha' \\
&= F(s')\rho_r(s^r, p, \sigma)\big(\alpha^r \uplus \overline{\rho_c(s_0, \sigma)\alpha_\varepsilon^\Gamma}\big) \\
&= F(s')\rho_r(s^r, p, \sigma)\big(\alpha^r \uplus \overline{\rho_c(s_0, \sigma)}\alpha_\varepsilon^\Gamma\big) \\
&= [\![M]\!](t). && \square
\end{aligned}$$

D Examples

The following yDT$^{\mathrm{R}}$ is semantically equivalent to the SRTST in Example 5.

Example 28. Let us consider a DBTA $A \triangleq (\Pi, \Sigma, \theta)$ where $\Sigma \triangleq \{\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}\}$, $\Pi \triangleq \{\pi_\mathtt{a}, \pi_\mathtt{b}\}$, and θ is defined by

$$\theta(\mathtt{f}, (\pi, \pi')) \triangleq \{\pi\} \text{ for } \pi, \pi' \in \Pi, \qquad \theta(\mathtt{a}, ()) \triangleq \{\pi_\mathtt{a}\}, \qquad \theta(\mathtt{b}, ()) \triangleq \{\pi_\mathtt{b}\}.$$

For $d \in \{\mathtt{a}, \mathtt{b}\}$, π_d accepts just a tree whose leftmost leaf is d.

With this regular look-ahead automaton A, let us consider a yDT$^{\mathrm{R}}$ $M \triangleq (Q, \Sigma, \Delta, \mathit{Init}, R)$ where

$$Q \triangleq \{q\}, \quad \Delta \triangleq \{\mathtt{a}, \mathtt{b}\}, \quad \mathit{Init} \triangleq \{(q(x_1), \pi_{\mathtt{a}}), (q(x_1), \pi_{\mathtt{b}})\}$$

and the set of rules R are:

$$\begin{array}{llll} q(\mathtt{f}(x_1, x_2)) \to q(x_1)q(x_2) & \langle \pi_{\mathtt{a}}, \pi_d \rangle, & q(d) \to d, & \\ q(\mathtt{f}(x_1, x_2)) \to q(x_2)q(x_1) & \langle \pi_{\mathtt{b}}, \pi_d \rangle, & & (d \in \{\mathtt{a}, \mathtt{b}\}). \end{array}$$

The rules are self-explanatory: given $t \in \mathcal{T}_\Sigma$, $[\![M]\!] = [\![q]\!]_M$ recursively swaps the two subtrees of the root if the leftmost leaf is $\mathtt{b}$, and skip the swapping otherwise; and then produces the leaves as the output. For instance,

$$\begin{aligned} &[\![M]\!](\mathtt{f}(\mathtt{f}(\mathtt{b}, \mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{b}))) = [\![q]\!]_M(\mathtt{f}(\mathtt{f}(\mathtt{b}, \mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{b}))) \\ &= [\![q]\!]_M(\mathtt{f}(\mathtt{a}, \mathtt{b}))[\![q]\!]_M(\mathtt{f}(\mathtt{b}, \mathtt{a})) = [\![q]\!]_M(\mathtt{a})[\![q]\!]_M(\mathtt{b})[\![q]\!]_M(\mathtt{a})[\![q]\!]_M(\mathtt{b}) = \mathtt{abab}. \end{aligned}$$

Example 29. By applying the construction in the proof of Lemma 6 to the yDT$^{\mathrm{R}}$ in Example 28, we obtain an SRTST $T = (S, \Sigma, \Delta, P, s_0, \Gamma, F, \delta_c, \delta_r, \rho_c, \rho_r)$ as follows. Recall that $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}\}$ and $\Delta = \{\mathtt{a}, \mathtt{b}\}$.

The set S of states, the set P of stack symbols and the set Γ of variables are defined by

$$\begin{aligned} S &\triangleq \{(), (\pi_{\mathtt{a}}), (\pi_{\mathtt{b}}), (\pi_{\mathtt{a}}, \pi_{\mathtt{a}}), (\pi_{\mathtt{a}}, \pi_{\mathtt{b}}), (\pi_{\mathtt{b}}, \pi_{\mathtt{b}}), (\pi_{\mathtt{b}}, \pi_{\mathtt{a}})\} &&= \Pi^{(\le 2)}, \\ P &\triangleq \{(), (\pi_{\mathtt{a}}), (\pi_{\mathtt{b}}), (\pi_{\mathtt{a}}, \pi_{\mathtt{a}}), (\pi_{\mathtt{a}}, \pi_{\mathtt{b}}), (\pi_{\mathtt{b}}, \pi_{\mathtt{b}}), (\pi_{\mathtt{b}}, \pi_{\mathtt{a}})\} &&= \Pi^{(\le 2)}, \\ \Gamma &\triangleq \{q(x_1), q(x_2)\} &&= Q(X_2), \end{aligned}$$

and $s_0 \triangleq ()$. Also, $F((\pi_{\mathtt{a}})) \triangleq q(x_1)$ and $F((\pi_{\mathtt{b}})) \triangleq q(x_1)$. The call state-transition function δ_c is:

$$\delta_c((), \sigma) \triangleq ((), ()), \qquad \delta_c((\pi_d), \sigma) \triangleq ((), (\pi_d))$$

for every $d \in \{\mathtt{a}, \mathtt{b}\}$. The return state-transition function δ_r is defined by

$$\begin{aligned} \delta_r((), (), d_1) &\triangleq (\pi_{d_1}), \\ \delta_r((\pi_{d_1}, \pi), (), \mathtt{f}) &\triangleq (\pi_{d_1}), \\ \delta_r((), (\pi_{d_1}), d_2) &\triangleq (\pi_{d_1}, \pi_{d_2}), \\ \delta_r((\pi_{d_2}, \pi), (\pi_{d_1}), \mathtt{f}) &\triangleq (\pi_{d_1}, \pi_{d_2}) \end{aligned}$$

for every $\pi \in \Pi$ and $d_1, d_2 \in \{\mathtt{a}, \mathtt{b}\}$. The call variable-update function ρ_c is defined by

$$\rho_c(s, \sigma) \triangleq [\gamma := \gamma]_{\gamma \in \Gamma}.$$

The return variable-update function ρ_r is defined as follows:

$$\begin{aligned}
\rho_r((\quad &, () \quad , d) \triangleq [q(x_1) := d],\\
\rho_r((\quad &, (\pi_1), d) \triangleq [q(x_1) := \overline{q(x_1)}, q(x_2) := d],\\
\rho_r((\pi_{\mathtt{a}}, \pi_1), () \quad &, \mathtt{f}) \triangleq [q(x_1) := q(x_1)q(x_2)],\\
\rho_r((\pi_{\mathtt{b}}, \pi_1), () \quad &, \mathtt{f}) \triangleq [q(x_1) := q(x_2)q(x_1)],\\
\rho_r((\pi_{\mathtt{a}}, \pi_1), (\pi_2), \mathtt{f}) &\triangleq [q(x_1) := \overline{q(x_1)}, q(x_2) := q(x_1)q(x_2)],\\
\rho_r((\pi_{\mathtt{b}}, \pi_1), (\pi_2), \mathtt{f}) &\triangleq [q(x_1) := \overline{q(x_1)}, q(x_2) := q(x_2)q(x_1)].
\end{aligned}$$

for every $\pi_1, \pi_2 \in \Pi$ and $d \in \{\mathtt{a}, \mathtt{b}\}$.

Example 30. By applying the construction in the proof of Lemma 7 to the SRTST in Example 5, we obtain a yDT$^{\mathrm{R}}$ $M = (Q, \Sigma, \Delta, \mathit{Init}, R)$ with a DBTA $A = (\Pi, \Sigma, \theta)$ as follows. Recall that $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}\}$ and $\Delta = \{\mathtt{a}, \mathtt{b}\}$.

The set Π of states is obtained as below:

$$\begin{aligned}
\Pi_0 &= \{(s_?, \mathtt{a}, s_?), (s_?, \mathtt{b}, s_?), (s_{\mathtt{a}}, \mathtt{a}, s_?), (s_{\mathtt{a}}, \mathtt{b}, s_?), (s_{\mathtt{b}}, \mathtt{a}, s_?), (s_{\mathtt{b}}, \mathtt{b}, s_?)\},\\
\Pi_1 &= \Pi_0 \cup \{(s_?, \mathtt{f}, s_{\mathtt{a}}), (s_?, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{a}}, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{a}}, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{b}}, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{b}}, \mathtt{f}, s_{\mathtt{b}})\},\\
\Pi_2 &= \Pi_1 \cup \{(s_?, \mathtt{f}, s_{\mathtt{a}}), (s_?, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{a}}, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{a}}, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{b}}, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{b}}, \mathtt{f}, s_{\mathtt{b}})\}\\
&= \Pi_1.
\end{aligned}$$

So we use Π_2 as Π. The transition function θ is defined as follows:

$$\begin{aligned}
\theta(d_1) &\triangleq \{(s_?, d_1, s_?), (s_{\mathtt{a}}, d_1, s_?), (s_{\mathtt{b}}, d_1, s_?)\},\\
\theta(\mathtt{f}, (s_?, d_1, s_?), (s_{d_1}, d_2, s_?)) &\triangleq \{(s_?, \mathtt{f}, s_{d_1}), (s_{\mathtt{a}}, \mathtt{f}, s_{d_1}), (s_{\mathtt{b}}, \mathtt{f}, s_{d_1})\},\\
\theta(\mathtt{f}, (s_?, \mathtt{f}, s_{d_1}), (s_{d_1}, d_2, s_?)) &\triangleq \{(s_?, \mathtt{f}, s_{d_1}), (s_{\mathtt{a}}, \mathtt{f}, s_{d_1}), (s_{\mathtt{b}}, \mathtt{f}, s_{d_1})\},\\
\theta(\mathtt{f}, (s_?, d_1, s_?), (s_{d_1}, \mathtt{f}, s_{d_2})) &\triangleq \{(s_?, \mathtt{f}, s_{d_1}), (s_{\mathtt{a}}, \mathtt{f}, s_{d_1}), (s_{\mathtt{b}}, \mathtt{f}, s_{d_1})\},\\
\theta(\mathtt{f}, (s_?, \mathtt{f}, s_{d_1}), (s_{d_1}, \mathtt{f}, s_{d_2})) &\triangleq \{(s_?, \mathtt{f}, s_{d_1}), (s_{\mathtt{a}}, \mathtt{f}, s_{d_1}), (s_{\mathtt{b}}, \mathtt{f}, s_{d_1})\}
\end{aligned}$$

for every $d_1, d_2 \in \{\mathtt{a}, \mathtt{b}\}$. Then

$$Q \quad \triangleq \quad S \times S \times \Gamma \quad = \quad \{s_?, s_{\mathtt{a}}, s_{\mathtt{b}}\} \times \{s_?, s_{\mathtt{a}}, s_{\mathtt{b}}\} \times \{\gamma\}.$$

The set R of rules consists of the following:

$$\begin{aligned}
&(s, s_{\mathtt{a}}, \gamma)(\mathtt{f}(x_1, x_2)) \to \mathtt{a}d && \langle (s_?, \mathtt{a}, s_?), (s_{\mathtt{a}}, d, s_?) \rangle\\
&(s, s_{\mathtt{b}}, \gamma)(\mathtt{f}(x_1, x_2)) \to d\mathtt{b} && \langle (s_?, \mathtt{b}, s_?), (s_{\mathtt{b}}, d, s_?) \rangle\\
&(s, s_{\mathtt{a}}, \gamma)(\mathtt{f}(x_1, x_2)) \to (s_?, s_{\mathtt{a}}, \gamma)(x_1)d && \langle (s_?, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{a}}, d, s_?) \rangle\\
&(s, s_{\mathtt{b}}, \gamma)(\mathtt{f}(x_1, x_2)) \to d(s_?, s_{\mathtt{b}}, \gamma)(x_1) && \langle (s_?, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{b}}, d, s_?) \rangle\\
&(s, s_{\mathtt{a}}, \gamma)(\mathtt{f}(x_1, x_2)) \to (s_?, s_{\mathtt{a}}, \gamma)(x_1)(s_{\mathtt{a}}, s_d, \gamma)(x_2) && \langle (s_?, \mathtt{f}, s_{\mathtt{a}}), (s_{\mathtt{a}}, \mathtt{f}, s_d) \rangle\\
&(s, s_{\mathtt{b}}, \gamma)(\mathtt{f}(x_1, x_2)) \to (s_{\mathtt{b}}, s_d, \gamma)(x_2)(s_?, s_{\mathtt{b}}, \gamma)(x_1) && \langle (s_?, \mathtt{f}, s_{\mathtt{b}}), (s_{\mathtt{b}}, \mathtt{f}, s_d) \rangle\\
&(s, s_?, \gamma)(d \qquad) \to \varepsilon && \langle\rangle
\end{aligned}$$

for every $s \in S$ and $d \in \{\mathtt{a}, \mathtt{b}\}$. The set *Init* of initial sequences is defined as follows:

$$\mathit{Init} \triangleq \left\{ \big(d, (s_?, d, s_?)\big) \,\middle|\, d \in \{\mathtt{a}, \mathtt{b}\} \right\} \cup \left\{ \big((s_?, s_d, \gamma)(x_1), (s_?, \mathtt{f}, s_d)\big) \,\middle|\, d \in \{\mathtt{a}, \mathtt{b}\} \right\}.$$