Regular transductions with MCFG input syntax

Mark-Jan Nederhof
University of St Andrews

Heiko Vogler
Technische Universität Dresden

Abstract

We show that regular transductions for which the input part is generated by some multiple context-free grammar can be simulated by synchronous multiple context-free grammars. We prove that synchronous multiple context-free grammars are strictly more powerful than this combination of regular transductions and multiple context-free grammars.

1 Introduction

In machine translation, one is interested in automatically translating sentences of one natural language into sentences of another natural language. Such translations can be considered as string-to-string transductions by viewing the words of a natural language as symbols of a formal language, and viewing sentences as strings. Several formal models for such transductions have been proposed, e.g., syntax-directed translation schemata (Lewis and Stearns, 1968), also known as synchronous context-free grammars (Chiang, 2007), two-way generalized sequential machines (2gsm) (Sheperdson, 1959; Aho and Ullman, 1970), MSO definable string-to-string transductions (MSO-sst) (Courcelle and Engelfriet, 2012), and streaming string transducers (SST) (Alur and Černý, 2010).

It has been established that the deterministic versions of 2gsm, MSO-sst, and SST generate the same class of string-to-string transductions (Alur and Černý, 2010; Engelfriet and Hoogeboom, 2001); the same is true for the nondeterministic versions of MSO-sst and SST (Alur and Deshmukh, 2011). Due to these characterizations, the involved transducers and the corresponding transductions are called *regular transducers* and *regular transductions*, respectively.

In statistical machine translation (Lopez, 2008), one aims at automatically inferring a translation model from some bilingual corpus, where the translation model is chosen from some class of formal devices, e.g., the class of regular transducers. In the seminal paper by Brown et al. (1993), the inference is based on the concept of *alignment graph* (used as hidden random variable in the EM-algorithm (Dempster et al., 1977)); each such graph consists of an input sentence w, an output sentence v, and a binary relation between the set $\mathrm{pos}(v)$ of positions of v and the set $\mathrm{pos}(w)$ of positions of w. In the particular case of the IBM models each alignment graph is a partial mapping from $\mathrm{pos}(v)$ to $\mathrm{pos}(w)$. These have almost the same mathematical structure as the *origin graphs* of Bojańczyk (2013), except that in the latter, the mapping is total.

Bojańczyk (2013) and Bojańczyk et al. (2017a,b) investigated the concept of regular transductions with origin semantics, where the origin semantics of a regular transducer $\mathcal{A}$ is a set of the origin graphs that $\mathcal{A}$ can create: if $\mathcal{A}$ produces a portion v' of the output while reading the input symbol at position i, then each position of v' is aligned to i.

Since the domain of each regular transduction is a regular string language, it cannot capture non-regular syntactic phenomena on the source side of the translation. To enhance this capability, this paper investigates imposing additional syntactic restrictions on the input of a regular transducer, through intersection with a multiple context-free grammar (Seki et al., 1991) (MCFG). We prove that the resulting transduction can also be generated by a synchronous MCFG, which is a pair of MCFGs with synchronized nonterminals, much as in, e.g., synchronous context-free grammars. We further give an example of a synchronous MCFG whose transduction cannot be represented as the intersection of a regular transducer and a MCFG.

2 Preliminaries

We let $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$, and $[i, j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$ for every $i, j \in \mathbb{N}$. We abbreviate $[1, j]$ by $[j]$. We abbreviate sequences of objects, like $a_1 \cdots a_n$ and $a_1, \ldots, a_n$, by $a_{1,n}$. We denote the powerset of a set A by $\mathcal{P}(A)$. We abbreviate a set $\{a\}$ with one element by a. An alphabet A is a nonempty and finite set.

For functions $f : A \to B$ and $g : B \to C$, we denote their composition by $f \circ g$, i.e., $(f \circ g)(a) = g(f(a))$ for each $a \in A$.

We let $X = \{x_1, x_2, x_3, \ldots\}$ be a set of *variables* and $X_k = \{x_1, \ldots, x_k\}$ for each $k \in \mathbb{N}$.

Let Σ_1 and Σ_2 be alphabets. An *origin graph* (over Σ_1 and Σ_2) is a triple (w, v, g) where $w \in \Sigma_1^*$, $v \in \Sigma_2^*$, and g (*origin mapping*) maps each position j of v to a position i of w. Intuitively, the pair $(j, i) \in g$ indicates that the symbol at position j of v originated from position i of w. Let A be a set of origin graphs and L_1 and L_2 formal languages. Then we define

$$\begin{aligned}&A \Cap (L_1 \times L_2) = \\ &\quad \{(w, v, g) \mid (w, v, g) \in A, w \in L_1, v \in L_2\} \ .\end{aligned}$$

We generally refer to Σ_1 as the *input alphabet* and Σ_2 as the *output alphabet*. For a set $L \subseteq L_1 \times L_2$ we define the *input projection* as $\mathrm{proj}_1(L) = \{w \mid (w, v) \in L\}$ and the *output projection* as $\mathrm{proj}_2(L) = \{v \mid (w, v) \in L\}$.

3 Streaming String Transducers

Here we recall the definition of streaming transducer from Alur and Deshmukh (2011), with some slight modifications that refer to the final output of a string.

Let $\mathcal{R}$ be a finite set of *registers*, and let $\rho = |\mathcal{R}|$. Let Γ be an alphabet. A *copyless assignment to* $\mathcal{R}$ *over* Γ (or short: *assignment*) is a mapping $\alpha : \mathcal{R} \to (\mathcal{R} \cup \Gamma)^*$ such that any $r \in \mathcal{R}$ occurs at most once in the set $\{\alpha(r') \mid r' \in \mathcal{R}\}$. We assume there is a fixed total ordering $r_1, \ldots, r_\rho$ of the ρ registers in $\mathcal{R}$. This allows us to specify an assignment α in the form $(r_1, \ldots, r_\rho) := (\alpha(r_1), \ldots, \alpha(r_\rho))$. The *identity assignment* is the mapping $\mathrm{id} : \mathcal{R} \to (\mathcal{R} \cup \Gamma)^*$ such that $\mathrm{id}(r) = r$ for each $r \in \mathcal{R}$. The set of all copyless assignments to $\mathcal{R}$ over Γ is denoted by $\mathrm{Ass}(\mathcal{R}, \Gamma)$. The composition of two copyless assignments $\alpha_1, \alpha_2 \in \mathrm{Ass}(\mathcal{R}, \Gamma)$ is the mapping $\alpha_1 \circ \alpha_2' : \mathcal{R} \to (\mathcal{R} \cup \Gamma)^*$, where α_2' is the canonical extension of α_2 to a mapping of type $(\mathcal{R} \cup \Gamma)^* \to (\mathcal{R} \cup \Gamma)^*$. For convenience, we drop the prime and write $\alpha_1 \circ \alpha_2$ instead of $\alpha_1 \circ \alpha_2'$. Clearly, $\alpha_1 \circ \alpha_2$ is a copyless assignment to $\mathcal{R}$ over Γ, and $(\mathrm{Ass}(\mathcal{R}, \Gamma), \circ, \mathrm{id})$ is a monoid.

A *nondeterministic streaming string transducer (over* Σ_1 *and* Σ_2*)* (for short: NSST) is a tuple $\mathcal{A} = (Q, \Sigma_1, \Sigma_2, \mathcal{R}, r_\mathrm{o}, T, q_0, F)$ where Q is a finite, nonempty set of *states*, Σ_1 and Σ_2 are the input alphabet and the output alphabet, respectively, $\mathcal{R}$ is a finite set of *registers*, $r_\mathrm{o} \in \mathcal{R}$ is the *output register*, $T \subseteq Q \times \Sigma_1 \times \mathrm{Ass}(\mathcal{R}, \Sigma_2) \times Q$ is a finite set of *transitions*, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is the set of *final states*.

The *summary of* $\mathcal{A}$ is the mapping

$$\Delta : (Q \times \Sigma_1^*) \to \mathcal{P}(\mathrm{Ass}(\mathcal{R}, \Sigma_2) \times Q)$$

defined inductively as follows.

$$\begin{aligned}\Delta(q, \varepsilon) &= \{(\mathrm{id}, q)\} \\ \Delta(q, wa) &= \{(\alpha \circ \alpha_w, q'') \mid \\ &\qquad (\exists q' \in Q) : (\alpha_w, q') \in \Delta(q, w), \\ &\qquad\qquad (q', a, \alpha, q'') \in T\}\end{aligned}$$

for each $q \in Q$, $w \in \Sigma_1^*$, and $a \in \Sigma_1$.

The string-to-string transduction *computed by* $\mathcal{A}$ is the set $[\![\mathcal{A}]\!] \subseteq \Sigma_1^* \times \Sigma_2^*$ defined by

$$\begin{aligned}[\![\mathcal{A}]\!] = \{&(w, (\alpha \circ \alpha_\varepsilon')(r_\mathrm{o})) \mid \\ &w \in \Sigma_1^*, (\exists q' \in F) : (\alpha, q') \in \Delta(q_0, w)\}\end{aligned}$$

where $\alpha_\varepsilon \in \mathrm{Ass}(\mathcal{R}, \Sigma_2)$ is defined by $\alpha_\varepsilon(r) = \varepsilon$. Clearly, $\mathrm{proj}_1([\![\mathcal{A}]\!])$ is a regular language. We note that $q_0 \in F$ if and only if $\{v \in \Sigma_2^* \mid (\varepsilon, v) \in [\![\mathcal{A}]\!]\} = \{\varepsilon\}$.

Each $(w, v) \in [\![\mathcal{A}]\!]$ is obtained by at least one sequence of transitions, and possibly more than one due to nondeterminism. For a given such sequence, each symbol occurrence in v is obtained by application of a transition (q', a, α, q''), and this links the index of that symbol occurrence in v to the index of the corresponding occurrence of a in w. Thereby the sequence of transitions corresponds in a natural way to an origin graph. The set of such origin graphs is denoted by $[\![\mathcal{A}]\!]^o$, and will be called the *origin semantics of* $\mathcal{A}$.

Example 3.1. Let $\Sigma = \{a, b, \#\}$. We consider the transformation

$$\begin{aligned}\tau = \{&(w, w\#w) \mid w \in \{a, b\}^*, \\ &|w| = 2 \cdot n \text{ for some } n \in \mathbb{N}_+\} \ .\end{aligned}$$

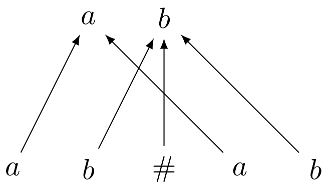


Figure 1: Origin graph of $(ab, ab\#ab)$ in Example 3.1.

For instance $(ab, ab\#ab) \in \tau$. The transformation τ can be computed by the NSST $\mathcal{A} = (Q, \Sigma, \Sigma, \mathcal{R}, r_1, T, q_0, F)$ where $Q = \{q_0, q_1, q_f\}$, $F = \{q_f\}$, $\mathcal{R} = \{r_1, r_2\}$, T contains, for every $i \in \{0, 1\}$ and $\gamma \in \{a, b\}$, the transitions

$$\big(q_i, \gamma, (r_1, r_2) := (r_1\gamma, r_2\gamma), q_{1-i}\big) \quad \text{and} \\ \big(q_1, \gamma, (r_1, r_2) := (r_1\gamma\#r_2\gamma, \varepsilon), q_f\big) \ .$$

For instance,

$$\begin{aligned} \Delta(q_0, \varepsilon) &= \{(\mathrm{id}, q_0)\} \\ \Delta(q_0, a) &= \{((r_1, r_2) := (r_1a, r_2a), q_1)\} \\ \Delta(q_0, ab) &= \{((r_1, r_2) := (r_1ab, r_2ab), q_0), \\ &\qquad ((r_1, r_2) := (r_1ab\#r_2ab, \varepsilon), q_f)\} \ . \end{aligned}$$

Let α denote $(r_1, r_2) := (r_1ab\#r_2ab, \varepsilon)$. Then

$$\begin{aligned} (\alpha \circ \alpha_\varepsilon)(r_1) = \alpha_\varepsilon(\alpha(r_1)) &= \alpha_\varepsilon(r_1ab\#r_2ab) \\ &= ab\#ab \ . \end{aligned}$$

Since q_0 is the initial state and q_f is the final state, we obtain $(ab, ab\#ab) \in [\![\mathcal{A}]\!]$. The corresponding origin graph is shown in Figure 1. □

We call a NSST *nondeleting* if for each assignment α occurring in a transition, each register r occurs exactly once in $\{\alpha(r') \mid r' \in \mathcal{R}\}$. For each NSST $\mathcal{A}$, there is a nondeleting NSST $\mathcal{A}'$ such that $[\![\mathcal{A}]\!] = [\![\mathcal{A}']\!]$. The proof is very similar to the proof of a similar result for MCFG by Seki et al. (1991), which we will mention again in Section 4. We outline how $\mathcal{A}' = (Q', \Sigma_1, \Sigma_2, \mathcal{R}', r_o, T', q_0', F')$ is constructed from $\mathcal{A} = (Q, \Sigma_1, \Sigma_2, \mathcal{R}, r_o, T, q_0, F)$.

First, Q' contains a new state q_0' plus states of the form q_D where $q \in Q$ and $D \subseteq \mathcal{R}$. The intuition is that the registers in D are those that must remain empty in $\mathcal{A}'$, as in a corresponding computation in $\mathcal{A}$ their contents would later appear as part of a register that is deleted (or that is not the output register when the end of the input is reached). By keeping those registers empty, they no longer need to be deleted, and instead can be added in an arbitrary way to assignments without changing the semantics. We let $q_D \in F'$ if and only if $q \in F$ and $r_o \notin D$, and $q_0' \in F'$ if and only if $q_0 \in F$.

For each $D \subseteq \mathcal{R}$ and $(q, a, \alpha, q') \in T$, we have $(q_{D'}, a, \alpha', q'_D) \in T'$, where D' and α' are defined as follows. The registers in D' are obtained in one of two ways. First, if $r \in D$, then every register in $\alpha(r)$ is in D', and secondly, if a register r' does not occur in $\alpha(r)$ for any r, then it is in D'.

In the first instance, $\alpha'(r)$ is a copy of $\alpha(r)$ for each $r \notin D$, and $\alpha'(r)$ is obtained from $\alpha(r)$ by omitting all output symbols for each $r \in D$. However, each register r' that does not occur in $\alpha(r)$, for any r, is added to $\alpha'(r'')$ in an arbitary place for an arbitrary r''. Moreover, if $q = q_0$, then T' also contains (q_0', a, α', q'_D).

For example, if

$$(q, a, (r_1, r_2, r_3, r_4) := (r_2br_3, c, d, er_1), q')$$

is in T, then for $D = \{r_1, r_3\}$, we have $D' = \{r_2, r_3, r_4\}$, where D' contains r_2 and r_3 because r_2br_3 is assumed to be deleted later, and D' contains r_4 because it is deleted here. Further, T' would include

$$(q_{D'}, a, (r_1, r_2, r_3, r_4) := (r_2r_3, cr_4, \varepsilon, er_1), q'_D)$$

where we have added r_4 to the right-hand side of the assignment in an arbitrary place.

4 Synchronous Multiple Context-Free Grammars

A *multiple context-free grammar (over Σ)* (for short: MCFG) is a tuple $\mathcal{G} = (N, S, \Sigma, P)$ where N is an alphabet of *nonterminals*, each nonterminal A has a *fanout* in $\mathbb{N}$ (denoted by $\mathrm{fo}(A)$), $S \in N$ is an *initial nonterminal* with $\mathrm{fo}(S) = 1$, Σ is an alphabet of *terminals*, and P is a finite set of *rules*, where each rule has the form

$$A_0(w_{1,\ell_0}) \to A_1(x^{(1)}_{1,\ell_1}) \cdots A_n(x^{(n)}_{1,\ell_n})$$

where $n \in \mathbb{N}$, $A_0, A_1, \ldots, A_n$ are nonterminals with $\mathrm{fo}(A_i) = \ell_i$ for each $i \in [0, n]$; for each $i \in [n]$, $x^{(i)}_{1,\ell_i}$ is a sequence of ℓ_i variables in X such that the set of all variables occurring in $x^{(1)}_{1,\ell_1}, \ldots, x^{(n)}_{1,\ell_n}$ is X_m where $m = \sum_{i=1}^{n} \ell_i$; for each $j \in [\ell_0]$, each w_j is in $(\Sigma \cup X_m)^*$; finally, the rule is *linear in* X, i.e., each variable in X occurs at most once in

$w_1 \cdots w_{\ell_0}$. The *rank* of this rule is n. The rank of $\mathcal{G}$ is the maximal rank of its rules.

Rules can be instantiated by consistent substitution of variables. The derivation relation $\Rightarrow$ of $\mathcal{G}$ is defined in the usual way, by applying instantiated rules. The language over Σ generated by $\mathcal{G}$ is defined to be the set of strings w such that $S(w) \Rightarrow^* \varepsilon$, and is denoted by $L(\mathcal{G})$. Two MCFGs are *equivalent* if they generate the same language.

A MCFG is called *uni-lexicalized* if there is exactly one terminal in each rule. For each MCFG there is an equivalent uni-lexicalized MCFG. A MCFG is called *nondeleting* if each variable that occurs in the right-hand side occurs exactly once in the left-hand side. For each MCFG there is an equivalent nondeleting MCFG (Seki et al., 1991).

A *synchronous multiple context-free grammar* (for short: synchronous MCFG) is a tuple $\mathcal{G} = (N, S, \Sigma_1, \Sigma_2, P)$ such that $\mathcal{G}' = (N, S, \Sigma_1 \cup \Sigma_2, P)$ is an MCFG (called *underlying MCFG*) except that S has fanout 2. Moreover, for each nonterminal A we split its fanout ℓ into an *input fanout* ℓ_1 and an *output fanout* ℓ_2 such that $\ell = \ell_1 + \ell_2$, and denote this by $\mathrm{fo}(A) = (\ell_1, \ell_2)$. In particular, we let $\mathrm{fo}(S) = (1, 1)$. We call the first ℓ_1 arguments of A its *input arguments* and the remaining ℓ_2 arguments its *output arguments*, and we separate these two blocks by a semicolon. We require that elements of Σ_1 and Σ_2 may only occur in input arguments and output arguments, respectively. Finally, we require that no variable may simultaneously occur in an input and in an output argument. We implement this requirement by choosing X as set of input variables and $Y = \{y_1, y_2, \ldots\}$ as set of output variables. Hence, a rule of a synchronous MCFG has the form

$$A_0(w_{1,\ell_0}; v_{1,m_0}) \rightarrow A_1(x^{(1)}_{1,\ell_1}; y^{(1)}_{1,m_1}) \cdots A_n(x^{(n)}_{1,\ell_n}; y^{(n)}_{1,m_n})$$

where $n \in \mathbb{N}$, $A_0, A_1, \ldots, A_n$ are nonterminals and each A_i has fanout (ℓ_i, m_i); for each $i \in [n]$, $x^{(i)}_{1,\ell_i}$ and $y^{(i)}_{1,m_i}$ are sequences of variables in X and Y, respectively; for each $j \in [\ell_0]$, string w_j is in $(\Sigma_1 \cup \{x^{(1)}_{1,\ell_1}, \ldots, x^{(n)}_{1,\ell_n}\})^*$, and for each $j \in [m_0]$, string v_j is in $(\Sigma_2 \cup \{y^{(1)}_{1,m_1}, \ldots, y^{(n)}_{1,m_n}\})^*$; finally, the rule is linear in X and Y.

The MCFG $\mathcal{G}_1 = (N, S, \Sigma_1, P_1)$ is the *input component of* $\mathcal{G}$, where the fanout of each nonterminal of N is its input fanout in $\mathcal{G}$, and P_1 is the set of all rules of P in which the output arguments are dropped. Similary, we define the *output component of* $\mathcal{G}$.

Let $\mathcal{G}$ be a synchronous MCFG. We define the derivation relation $\Rightarrow_{\mathcal{G}}$ of $\mathcal{G}$ to be the derivation relation of its underlying MCFG. The *string-to-string transduction computed by* $\mathcal{G}$ is the set

$$[\![\mathcal{G}]\!] = \{(w, v) \in \Sigma_1^* \times \Sigma_2^* \mid S(w; v) \Rightarrow^* \varepsilon\} \ .$$

A *uni-lexicalized* synchronous MCFG is a synchronous MCFG in which each rule either contains exactly one input symbol or contains neither input symbols nor output symbols. In a straightforward way, we can associate with each uni-lexicalized synchronous MCFG $\mathcal{G}$ a set $[\![\mathcal{G}]\!]^o$ of origin graphs by linking each occurrence of an output terminal of a rule to the unique input terminal of that rule.

5 Intersecting the Input of NSST with MCFG

Lemma 5.1. For every NSST $\mathcal{A}$ over Σ_1 and Σ_2 and every MCFG $\mathcal{G}$ over Σ_1, there is a uni-lexicalized synchronous MCFG $\mathcal{G}'$ over Σ_1 and Σ_2 such that $[\![\mathcal{A}]\!]^o \Cap ([\![\mathcal{G}]\!] \times \Sigma_2^*) = [\![\mathcal{G}']\!]^o$.

Proof. Let $\mathcal{A} = (Q, \Sigma_1, \Sigma_2, \mathcal{R}, r_\mathrm{o}, T, q_0, F)$, where $\mathcal{R}$ consists of the registers $r_1, \ldots, r_\rho$, and let $\mathcal{G} = (N, S, \Sigma_1, P)$ be an MCFG. Without loss of generality we may assume that $\mathcal{A}$ is nondeleting and that $\mathcal{G}$ is uni-lexicalized.

The intuition behind the construction of $\mathcal{G}'$ covers two aspects. Starting from the MCFG $\mathcal{G}$, we impose the state behaviour of $\mathcal{A}$ onto the nonterminal behaviour of $\mathcal{G}$ by a type of construction that can be traced back to Bar-Hillel et al. (1964). This aspect of the construction achieves $\mathrm{proj}_1([\![\mathcal{G}']\!]) = \mathrm{proj}_1([\![\mathcal{A}]\!]) \cap [\![\mathcal{G}]\!]$. The second aspect concerns the manipulation of the registers of $\mathcal{A}$. We let $\mathcal{G}'$ simulate the assignments in its output component, while its input component processes the input string.

The number of relevant assignments is in general infinite. In order to be able to simulate these assignments using a finite set of rules of $\mathcal{G}'$, we split up each assignment into a finite part, called "pattern", and a potentially infinite part, called "residue". The pattern represents ρ register occurrences in the image of the assignment, while the residue consists of the 2ρ strings that are interlaced with the register occurrences. The patterns are maintained as annotations of the nonterminals of $\mathcal{G}'$ and the residues appear in the output arguments of $\mathcal{G}'$. The residues of the left-hand side of a rule will be expressed in terms of residues that appear as output variables in the right-hand side. For this purpose, we introduce assignments that have output variables in their image.

For instance, let $\mathcal{G}$ contain the rule $A(ax_1^{(2)}x_1^{(1)}) \to B(x_1^{(1)})\ C(x_1^{(2)})$ and $\mathcal{A}$ have states $q_1, \ldots, q_5$. Assume that there is a transition (q_5, a, α, q_3), and that there are strings $w_1^{(2)}, w_1^{(1)} \in \Sigma_1^*$ such that $(\alpha_2, q_1) \in \Delta(q_3, w_1^{(2)})$ and $(\alpha_1, q_2) \in \Delta(q_1, w_1^{(1)})$. Then $\mathcal{G}'$ will have a rule of the form

$$A_{(q_5,p_1^{(0)},q_2)}(ax_1^{(2)}x_1^{(1)};\ldots) \to B_{(q_1,p_1^{(1)},q_2)}(x_1^{(1)};\ldots)\ C_{(q_3,p_1^{(2)},q_1)}(x_1^{(2)};\ldots)$$

where $p_1^{(1)}$ and $p_1^{(2)}$ are patterns corresponding to α_1 and α_2, respectively, and $p_1^{(0)}$ corresponds to $\alpha_1 \circ \alpha_2 \circ \alpha$. Hence there is a corresponding pattern for each argument in the right-hand sided and in the left-hand side.

Formally, a *pattern over* $\mathcal{R}$ is an assignment $p \in \mathrm{Ass}(\mathcal{R}, \emptyset)$. Obviously, $\mathrm{Ass}(\mathcal{R}, \emptyset)$ is finite. Now assume a rule $A_0(w_{1,\ell_0}) \to A_1(x_{1,\ell_1}^{(1)}) \cdots A_n(x_{1,\ell_n}^{(n)})$. Let $\vec{\ell} = (\ell_1, \ldots, \ell_n)$, $X_{\vec{\ell}} = \{x_i^{(k)} \mid k \in [n], i \in [\ell_k]\}$, and $Y_{\vec{\ell}} = \{y_i^{(k)} \mid k \in [n], i \in [2\rho\ell_k]\}$. For each $\alpha \in \mathrm{Ass}(\mathcal{R}, \Sigma_2 \cup Y_{\vec{\ell}})$ we define the *pattern* $\mathrm{p}(\alpha)$ and the *residue* $\mathrm{r}(\alpha) \in ((\Sigma_2 \cup Y_{\vec{\ell}})^*)^*$ as follows. Assume that for each $j \in [\rho]$ the string $\alpha(r_j)$ has the form $v_{j,0}r_{j,1}v_{j,1} \cdots r_{j,\mu_j}v_{j,\mu_j}$ for some $\mu_j \in [0,\rho]$, $v_{j,k} \in (\Sigma_2 \cup Y_{\vec{\ell}})^*$ ($k \in [0,\mu_j]$), and $\{r_{j,1}, \ldots, r_{j,\mu_j}\} \subseteq \mathcal{R}$. Then

$$\begin{aligned}\mathrm{p}(\alpha) &= ((r_1, \ldots, r_\rho) := \\ &\qquad (r_{1,1} \cdots r_{1,\mu_1},\ \ldots,\ r_{\rho,1} \cdots r_{\rho,\mu_\rho}))\\ \mathrm{r}(\alpha) &= (v_{1,0}, v_{1,1}, \ldots, v_{1,\mu_1},\ \ldots,\\ &\qquad v_{\rho,0}, v_{\rho,1}, \ldots, v_{\rho,\mu_\rho})\ .\end{aligned}$$

For example, let α be the assignment

$$(r_1, r_2, r_3, r_4) := (ay_3br_4,\ r_2y_2cr_3,\ dey_4,\ fy_1r_1)\ .$$

Then

$$\begin{aligned}\mathrm{p}(\alpha) &= ((r_1, r_2, r_3, r_4) := (r_4,\ r_2r_3,\ \varepsilon,\ r_1))\\ \mathrm{r}(\alpha) &= (ay_3b, \varepsilon,\ \varepsilon, y_2c, \varepsilon,\ dey_4,\ fy_1, \varepsilon)\ .\end{aligned}$$

For a pattern $p_i^{(k)}$ corresponding to $x_i^{(k)}$, we define an assignment $[p_i^{(k)}] \in \mathrm{Ass}(\mathcal{R}, Y_{\vec{\ell}})$, which introduces output variables next to register occurrences. Assume that

$$p_i^{(k)} = ((r_1, \ldots, r_\rho) := (s_1 \cdots s_{\mu_1},\ s_{\mu_1+1} \cdots s_{\mu_2},\ \ldots,\ s_{\mu_{\rho-1}+1} \cdots s_{\mu_\rho}))$$

for some $\mu_1, \ldots, \mu_\rho \in [0,\rho]$, $\mu_i \le \mu_{i+1}$ ($i \in [\rho - 1]$), and $s_j \in \mathcal{R}$ ($j \in [\mu_\rho]$). Let $\mu_0 = 0$ and $\kappa_j = 2\rho(i-1) + \mu_j + j$ ($j \in [0, \mu_\rho]$). Then

$$[p_i^{(k)}](r_j) = y_{\kappa_{j-1}+1}^{(k)}\ s_{\mu_{j-1}+1}\ y_{\kappa_{j-1}+2}^{(k)}\ \cdots\ s_{\mu_j}\ y_{\kappa_j}^{(k)}\ .$$

For instance, if $\rho = 4$ and $p_3^{(2)}$ is the pattern

$$(r_1, r_2, r_3, r_4) := (r_4,\ r_2r_3,\ \varepsilon,\ r_1)$$

then $\mu_1 = 1$, $\mu_2 = 3$, $\mu_3 = 3$, $\mu_4 = 4$, $\kappa_0 = 2\rho(i-1) = 16$, $\kappa_1 = \kappa_0 + \mu_1 + 1 = 18$, $\kappa_2 = \kappa_0 + \mu_2 + 2 = 21$, $\kappa_3 = \kappa_0 + \mu_3 + 3 = 22$, $\kappa_4 = \kappa_0 + \mu_4 + 4 = 24$, and

$$[p_3^{(2)}] = ((r_1, r_2, r_3, r_4) := (y_{17}^{(2)}r_4y_{18}^{(2)}, y_{19}^{(2)}r_2y_{20}^{(2)}r_3y_{21}^{(2)}, y_{22}^{(2)}, y_{23}^{(2)}r_1y_{24}^{(2)}))\ .$$

We construct the synchronous MCFG $\mathcal{G}' = (N', S', \Sigma_1, \Sigma_2, P')$ as follows. We let

$$N' = \{S'\} \cup \{A_c \mid A \in N, \mathrm{fo}_{\mathcal{G}}(A) = \ell,\ c \in (Q \times \mathrm{Ass}(\mathcal{R}, \emptyset) \times Q)^\ell\}$$

where $\mathrm{fo}_{\mathcal{G}'}(S') = (1,1)$ and $\mathrm{fo}_{\mathcal{G}'}(A_c) = (\mathrm{fo}_{\mathcal{G}}(A), 2\rho \cdot \mathrm{fo}_{\mathcal{G}}(A))$.

Let $A_0(w_{1,\ell_0}) \to A_1(x_{1,\ell_1}^{(1)}) \cdots A_n(x_{1,\ell_n}^{(n)})$ be a rule in P. For every $k \in [0,n]$ and $i \in [\ell_k]$ let $q_{i,1}^{(k)}, q_{i,2}^{(k)} \in Q$ and $p_i^{(k)} \in \mathrm{Ass}(\mathcal{R}, \emptyset)$. We abbreviate $(q_{i,1}^{(k)}, p_i^{(k)}, q_{i,2}^{(k)})$ by $c_i^{(k)}$.

We extend Δ to a function

$$\Delta' : Q \times (\Sigma_1 \cup X_{\vec{\ell}})^* \to \mathcal{P}(\mathrm{Ass}(\mathcal{R}, \Sigma_2 \cup Y_{\vec{\ell}}) \times Q)$$

by defining

$$\begin{aligned}\Delta'(q, \varepsilon) &= \{(\mathrm{id}, q)\}\\ \Delta'(q, wa) &= \{(\alpha \circ \alpha_w, q'') \mid\\ &\qquad (\exists q' \in Q) : (\alpha_w, q') \in \Delta'(q, w),\\ &\qquad (q', a, \alpha, q'') \in T\}\\ \Delta'(q, wx_i^{(k)}) &= \{([p_i^{(k)}] \circ \alpha_w, q_{i,2}^{(k)}) \mid\\ &\qquad (\alpha_w, q_{i,1}^{(k)}) \in \Delta'(q, w)\}\end{aligned}$$

Then the set P' contains the rule

$$\begin{aligned}&(A_0)_{c^{(0)}_{1,\ell_0}}(w_{1,\ell_0}; v_{1,2\rho\ell_0}) \to \\ &\quad (A_1)_{c^{(1)}_{1,\ell_1}}(x^{(1)}_{1,\ell_1}; y^{(1)}_{1,2\rho\ell_1}) \cdots \\ &\quad (A_n)_{c^{(n)}_{1,\ell_n}}(x^{(n)}_{1,\ell_n}; y^{(n)}_{1,2\rho\ell_n})\end{aligned}$$

for each choice of α_{1,ℓ_0} such that $(\alpha_i, q^{(0)}_{i,2}) \in \Delta'(q^{(0)}_{i,1}, w_i)$ and $\mathrm{p}(\alpha_i) = p^{(0)}_i$, for each $i \in [\ell_0]$, where $\mathrm{r}(\alpha_i) = (v_{2\rho(i-1)+1,2\rho i})$. This is illustrated in Figure 2.

In addition, for each $c \in \{(q_0, p^{(1)}_1, q) \mid p \in \mathrm{Ass}(\mathcal{R}, \emptyset), q \in F\}$ the rule

$$S'(x_1; \alpha_\varepsilon([p^{(1)}_1](r_\mathrm{o}))) \to S_c(x_1; y^{(1)}_{1,2\rho})$$

is in P'. Note that if $\mathcal{G}$ is uni-lexicalized, then so is $\mathcal{G}'$.

We can prove the following invariant. For every $A \in N$, $\ell = \mathrm{fo}(A)$, $w_1, \dots, w_\ell \in \Sigma_1^*$, $v_1, \dots, v_{2\rho\ell} \in \Sigma_2^*$, $c = (c_1, \dots, c_\ell)$ with $c_i = (q_{i1}, p_i, q_{i2}) \in Q \times \mathrm{Ass}(\mathcal{R}, \emptyset) \times Q$ for each $i \in [\ell]$, we have

$$\begin{aligned}&A_c(w_{1,\ell}; v_{1,2\rho\ell}) \Rightarrow^*_{\mathcal{G}'} \varepsilon \quad \text{if and only if} \\ &A(w_{1,\ell}) \Rightarrow^*_{\mathcal{G}} \varepsilon \wedge \\ &(\forall i \in [\ell]) : (\exists \alpha) : (\alpha, q_{i2}) \in \Delta(q_{i1}, w_i) \wedge \\ &\qquad\qquad\qquad \mathrm{r}(\alpha) = v_{2\rho(i-1)+1,2\rho i} \ .\end{aligned}$$

This invariant implies that for every $w \in \Sigma_1^*$ and $v \in \Sigma_2^*$: $S'(w; v) \Rightarrow^*_{\mathcal{G}'} \varepsilon$ if and only if $S(w) \Rightarrow^*_{\mathcal{G}} \varepsilon \wedge (w, v) \in [\![\mathcal{A}]\!]$. Thus $[\![\mathcal{G}']\!] = [\![\mathcal{A}]\!] \cap ([\![\mathcal{G}]\!] \times \Sigma_2^*)$. By the assumption that $\mathcal{G}$ is uni-lexicalized, furthermore $[\![\mathcal{G}']\!]^o = [\![\mathcal{A}]\!]^o \Cap ([\![\mathcal{G}]\!] \times \Sigma_2^*)$. □

Example 5.2. We consider the NSST $\mathcal{A}$ of Example 3.1 and the MCFG $\mathcal{G} = (N, A, \Sigma, P)$ with $N = \{A\}$, $\mathrm{fo}(A) = 1$, and for each $\gamma \in \Sigma$, P contains the rules

$$A(\gamma x_1) \to A(x_1) \quad \text{and} \quad A(\gamma) \to \varepsilon \ .$$

Obviously, $[\![\mathcal{G}]\!] = \Sigma^*$. We apply the construction of Lemma 5.1 to $\mathcal{A}$ and $\mathcal{G}$ and we obtain the uni-lexicalized synchronous MCFG $\mathcal{G}'$ which contains for each $\gamma \in \Sigma$ and $i \in \{0, 1\}$ at least the following rules.

$$\begin{aligned}&S'(x_1;\ y_1y_2y_3) \to A_{(q_0,p,q_f)}(x_1;\ y_{1,4}) \\ &A_{(q_i,p,q_f)}(\gamma x_1;\ y_1, \gamma y_2, \gamma y_3, y_4) \to \\ &\qquad\qquad\qquad A_{(q_{1-i},p,q_f)}(x_1;\ y_{1,4}) \\ &A_{(q_1,p,q_f)}(\gamma;\ \varepsilon, \gamma\#, \gamma, \varepsilon) \to \varepsilon\end{aligned}$$

where $p = ((r_1, r_2) := (r_1r_2, \varepsilon))$. For instance, $\Delta'(q_0, \gamma x_1)$ contains $([p^{(1)}_1] \circ \alpha, q_f)$ where $\alpha = ((r_1, r_2) := (r_1\gamma, r_2\gamma))$ is the assignment in the transition $(q_0, \gamma, \alpha, q_1)$ of $\mathcal{A}$. The calculation of $[p^{(1)}_1] \circ \alpha$ is

$$\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \xrightarrow{[p^{(1)}_1]} \begin{pmatrix} y_1r_1y_2r_2y_3 \\ y_4 \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} y_1r_1\gamma y_2r_2\gamma y_3 \\ y_4 \end{pmatrix}$$

hence $\mathrm{p}([p^{(1)}_1] \circ \alpha) = ((r_1, r_2) := (r_1r_2, \varepsilon))$ and $\mathrm{r}([p^{(1)}_1] \circ \alpha) = (y_1, \gamma y_2, \gamma y_3, y_4)$.

An example of a derivation is

$$\begin{aligned}S'(ab; ab\#ab) &\Rightarrow_{\mathcal{G}'} A_{(q_0,p,q_f)}(ab; \varepsilon, ab\#, ab, \varepsilon) \\ &\Rightarrow_{\mathcal{G}'} A_{(q_1,p,q_f)}(b; \varepsilon, b\#, b, \varepsilon) \\ &\Rightarrow_{\mathcal{G}'} \varepsilon \ .\end{aligned}$$

□

This example can be easily generalized:

Lemma 5.3. For every NSST $\mathcal{A}$ there is a uni-lexicalized synchronous MCFG $\mathcal{G}'$ such that $[\![\mathcal{A}]\!]^o = [\![\mathcal{G}']\!]^o$.

Proof. Let $\mathcal{A}$ be a NSST over Σ_1 and Σ_2. As illustrated by Example 5.2, we can construct a MCFG $\mathcal{G}$ such that $[\![\mathcal{G}]\!] = \Sigma_1^*$. The result then follows from Lemma 5.1. □

On the basis of Lemma 5.3, one can obtain complexity bounds on typical tasks involving NSST, such as deciding whether $(w, v) \in [\![\mathcal{A}]\!]$ for given strings w and v and NSST $\mathcal{A}$, relying on known complexity results for synchronous MCFG, and related formalisms such as synchronous LCFRS (Kaeshammer, 2013).

However, the relation between NSST and synchronous MCFG does not in any obvious way suggest a practical algorithm to do inference of NSST on the basis of sets of origin graphs, and this problem must remain outside the scope of the present paper.[1]

6 A Proper Subclass of Synchronous MCFG

In the light of Lemma 5.1 one may ask whether for *every* synchronous MCFG $\mathcal{G}$ one may find NSST $\mathcal{A}$

[1]We thank an anonymous reviewer for the suggestion to consider the problem of inference of NSST.

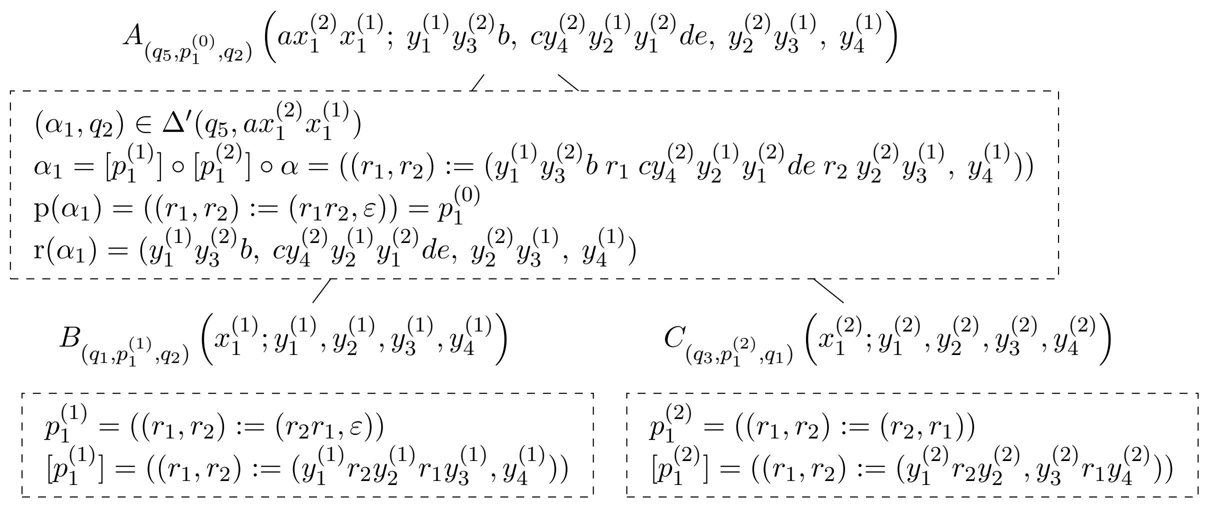

$A_{(q_5,p_1^{(0)},q_2)}\left(ax_1^{(2)}x_1^{(1)};\ y_1^{(1)}y_3^{(2)}b,\ cy_4^{(2)}y_2^{(1)}y_1^{(2)}de,\ y_2^{(2)}y_3^{(1)},\ y_4^{(1)}\right)$
$(\alpha_1, q_2) \in \Delta'(q_5, ax_1^{(2)}x_1^{(1)})$
$\alpha_1 = [p_1^{(1)}] \circ [p_1^{(2)}] \circ \alpha = ((r_1, r_2) := (y_1^{(1)}y_3^{(2)}b\, r_1\, cy_4^{(2)}y_2^{(1)}y_1^{(2)}de\, r_2\, y_2^{(2)}y_3^{(1)},\ y_4^{(1)}))$
$\mathrm{p}(\alpha_1) = ((r_1, r_2) := (r_1r_2, \varepsilon)) = p_1^{(0)}$
$\mathrm{r}(\alpha_1) = (y_1^{(1)}y_3^{(2)}b,\ cy_4^{(2)}y_2^{(1)}y_1^{(2)}de,\ y_2^{(2)}y_3^{(1)},\ y_4^{(1)})$
$B_{(q_1,p_1^{(1)},q_2)}\left(x_1^{(1)}; y_1^{(1)}, y_2^{(1)}, y_3^{(1)}, y_4^{(1)}\right)$
$C_{(q_3,p_1^{(2)},q_1)}\left(x_1^{(2)}; y_1^{(2)}, y_2^{(2)}, y_3^{(2)}, y_4^{(2)}\right)$
$p_1^{(1)} = ((r_1, r_2) := (r_2r_1, \varepsilon))$
$[p_1^{(1)}] = ((r_1, r_2) := (y_1^{(1)}r_2y_2^{(1)}r_1y_3^{(1)}, y_4^{(1)}))$
$p_1^{(2)} = ((r_1, r_2) := (r_2, r_1))$
$[p_1^{(2)}] = ((r_1, r_2) := (y_1^{(2)}r_2y_2^{(2)}, y_3^{(2)}r_1y_4^{(2)}))$

Figure 2: The construction in the proof of Lemma 5.3, for an NSST with two registers, $A(ax_1^{(2)}x_1^{(1)}) \to B(x_1^{(1)})\, C(x_1^{(2)})$, and transition (q_5, a, α, q_3) with $\alpha = ((r_1, r_2) := (br_1c, der_2))$.

and MCFG $\mathcal{G}'$ such that $[\![\mathcal{G}]\!] = [\![\mathcal{A}]\!] \cap ([\![\mathcal{G}']\!] \times \Sigma_2^*)$, for the shared output alphabet Σ_2, and perhaps even that $[\![\mathcal{G}]\!]^o = [\![\mathcal{A}]\!]^o \Cap ([\![\mathcal{G}']\!] \times \Sigma_2^*)$. In this section we show the answer to the former question is negative, whereby it is negative for the latter question as well. This holds even if the rank of $\mathcal{G}$ is restricted to 1 and $[\![\mathcal{G}]\!]$ is a function, that is, if $(w, v_1), (w, v_2) \in [\![\mathcal{G}]\!]$ implies $v_1 = v_2$.

To see this, consider the synchronous MCFG $\mathcal{G}$ of rank 1 with $N = \{S, A\}$, $\Sigma_1 = \Sigma_2 = \{a, b, a', b'\}$, and the following rules.

$$\begin{aligned} S(x_1x_2;\ y) &\to A(x_1, x_2;\ y) \\ A(x_1a, x_2a';\ yaa') &\to A(x_1, x_2;\ y) \\ A(x_1a, x_2b';\ yab') &\to A(x_1, x_2;\ y) \\ A(x_1b, x_2a';\ yba') &\to A(x_1, x_2;\ y) \\ A(x_1b, x_2b';\ ybb') &\to A(x_1, x_2;\ y) \\ A(\varepsilon, \varepsilon;\ \varepsilon) &\to \varepsilon \end{aligned}$$

For two strings $w = a_1 \cdots a_n$ and $w' = a'_1 \cdots a'_n$ of identical length n, we define $\mathrm{shuffle}(w, w')$ to be the string $a_1a'_1 \cdots a_na'_n$. The string-to-string transduction computed by $\mathcal{G}$ can now be written as

$$[\![\mathcal{G}]\!] = \{(ww', \mathrm{shuffle}(w, w')) \mid (\exists n \in \mathbb{N}) : w \in \{a, b\}^n \wedge w' \in \{a', b'\}^n\}$$

which is clearly a function. Suppose $[\![\mathcal{G}]\!]$ were $[\![\mathcal{A}]\!] \cap ([\![\mathcal{G}']\!] \times \Sigma_2^*)$ for some NSST $\mathcal{A} = (Q, \Sigma_1, \Sigma_2, \mathcal{R}, c_o, T, q_0, F)$ and MCFG $\mathcal{G}'$. Then

$$[\![\mathcal{G}]\!] = [\![\mathcal{A}]\!] \cap (\{ww' \mid (\exists n \in \mathbb{N}) : w \in \{a, b\}^n \wedge w' \in \{a', b'\}^n\} \times \Sigma_2^*) \ .$$

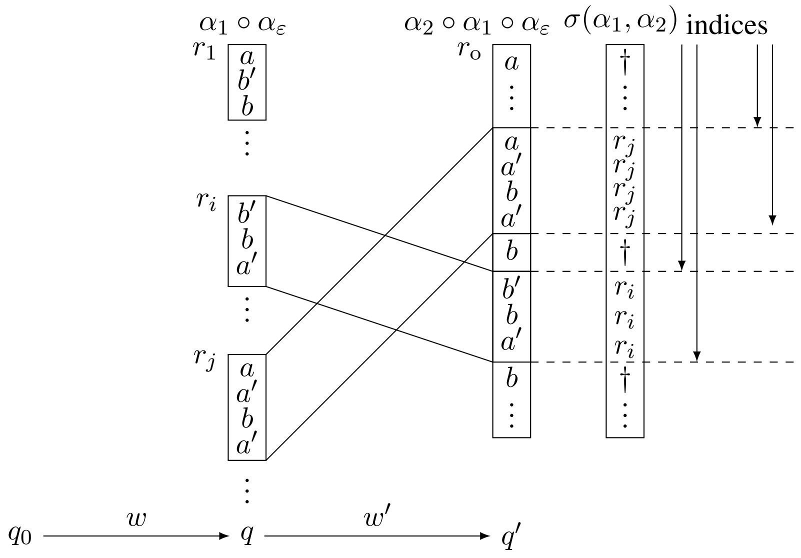
For each $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$, there are $q \in Q$, $q' \in F$, and assignments α_1 and α_2 such that $(\alpha_1, q) \in \Delta(q_0, w)$ and $(\alpha_2, q') \in \Delta(q, w')$. We then have $(\alpha_2 \circ \alpha_1 \circ \alpha_\varepsilon)(r_o) = \mathrm{shuffle}(w, w')$.

As illustrated in Figure 3, we want to capture how the contents that the registers have just after reading w eventually become substrings of register r_o, after also w' has been read. To formalize this, we first define $\Phi_{\alpha_1}(r) = r^{|(\alpha_1 \circ \alpha_\varepsilon)(r)|}$ for $r \in \mathcal{R}$. In words, each register r is mapped to $|(\alpha_1 \circ \alpha_\varepsilon)(r)|$ copies of its own name, to encode the size of its contents after reading w. Secondly, we introduce a new symbol †, and define Ψ to be the assignment such that $\Psi(r) = r$ for $r \in \mathcal{R}$ and $\Psi(c) = \dagger$ for $c \in \Sigma_2$. We now define $\sigma(\alpha_1, \alpha_2) = (\alpha_2 \circ \Psi \circ \Phi_{\alpha_1})(r_o) \in (\{\dagger\} \cup \mathcal{R})^{2n}$. We call $\sigma(\alpha_1, \alpha_2)$ the *schema* of α_1 and α_2.

If we fix $n > 0$, to be determined later, then the number of possible schemas $\sigma(\alpha_1, \alpha_2)$ is bounded by $(2n)^{2\rho}$, where $\rho = |\mathcal{R}|$ as before. This follows from the fact that each schema is determined by a set of pairs of indices. There is one such pair for each register r, consisting of the index in the schema where the substring $r^{|(\alpha_1 \circ \alpha_\varepsilon)(r)|}$ starts, and another index where it ends. If this substring is empty, this can be encoded by a starting index that is greater than the ending index.

We define the predicate G as

$$\begin{aligned} G(w, w', q, s) \equiv\ & \\ (\exists \alpha_1, \alpha_2, q' \in F) :\ & (\alpha_1, q) \in \Delta(q_0, w) \wedge \\ & (\alpha_2, q') \in \Delta(q, w') \wedge \\ & \sigma(\alpha_1, \alpha_2) = s \end{aligned}$$



$\alpha_1 \circ \alpha_\varepsilon$
$\alpha_2 \circ \alpha_1 \circ \alpha_\varepsilon$
$\sigma(\alpha_1, \alpha_2)$
indices
r_1
r_i
r_j
r_0
a b' b
b' b a'
a a' b a'
a a a' b a' b b' b a' b
† r_j r_j r_j r_j † r_i r_i r_i †
q_0
w
q
w'
q'

Figure 3: The processing of ww' and corresponding schema $\sigma(\alpha_1, \alpha_2)$. A set of indices, one pair per register, determines a schema.

for $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$, $q \in Q$ and schema s. For each $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$, there is at least one combination of q and s such that $G(w, w', q, s)$.

For each $q \in Q$, let $C(q)$ be the number of pairs $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$ such that $G(w, w', q, s)$ for some s. Now fix q to be such that $C(q)$ is maximal among the κ states of $\mathcal{A}$. This means that there are at least $2^{2n}/\kappa$ pairs $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$ such that $G(w, w', q, s)$ for some s.

For each schema s, let $C(q, s)$ be the number of pairs $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$ such that $G(w, w', q, s)$. Now fix s to be such that $C(q, s)$ is maximal among the at most $(2n)^{2\rho}$ schemas. This means that there are at least $\frac{2^{2n}}{\kappa \cdot (2n)^{2\rho}}$ pairs $(w, w') \in \{a, b\}^n \times \{a', b'\}^n$ such that $G(w, w', q, s)$.

There is a string $w' \in \{a', b'\}^n$ such that there are at least $\frac{2^{2n}}{\kappa \cdot (2n)^{2\rho}}/2^n = \frac{2^n}{\kappa \cdot (2n)^{2\rho}}$ strings $w \in \{a, b\}^n$ such that $G(w, w', q, s)$. For this w' and α_2 fixed, there are at least $\log_2(\frac{2^n}{\kappa \cdot (2n)^{2\rho}}) = n - \log_2 \kappa - 2\rho(1 + \log_2 n) \geq n - \log_2 \kappa - 2\rho n$ positions in $\mathrm{shuffle}(w, w')$ where we may find both a and b, depending on the choice of $w \in \{a, b\}^n$. This means the schema s contains at least $2n - 2\log_2 \kappa - 4\rho n - \rho$ occurrences of symbols from $\mathcal{R}$; note that in the output string, symbols from $\{a', b'\}$ are interlaced with symbols from $\{a, b\}$.

Similarly, there is a string $w \in \{a, b\}^n$ such that there are at least $\frac{2^n}{\kappa \cdot (2n)^{2\rho}}$ strings $w' \in \{a', b'\}^n$ such that $G(w, w', q, s)$. For this w and α_1 fixed, there are at least $n - \log_2 \kappa - 2\rho n$ positions in $\mathrm{shuffle}(w, w')$ where we may find both a' and b', depending on the choice of $w' \in \{a, b\}^n$. This means the schema s contains at least $2n - 2\log_2 \kappa - 4\rho n - \rho$ occurrences of †.

Altogether, this requires the length of the schema, and thereby of the output string, to be at least $4n - 4\log_2 \kappa - 8\rho n - 2\rho$. We now obtain the contradiction $4n - 4\log_2 \kappa - 8\rho n - 2\rho > 2n$ by choosing $n > \frac{\rho + 2\log_2 \kappa}{1 - 4\rho}$.

The transduction $[\![\mathcal{G}]\!]$ above is almost the same as the transduction called *merge* by Alur and Černý (2010), who also present a proof that this is beyond the power of deterministic SST (DSST). Because this transduction is a function, and because functional NSSTs are equivalent to DSSTs (Alur and Deshmukh, 2011), this could be used to produce an alternative to our proof above. However, the proof by Alur and Černý (2010) appears to contain at least one mistake, which is why we chose to present our own.[2]

[2]The proof by Alur and Černý (2010) considers "short configurations". These represent the contents of the registers after reading the first half of an input string, but replacing the contents of a register by a special symbol $*$ if its length is greater than some number t such that $2^t > \rho$. (Here we use our own variable names rather than those of op. cit.) It is then argued that the number of distinct short configurations is bounded by $\kappa\rho 2^t$. It appears to us this should have been

7 Conclusions

Motivated by potential applications of origin graphs for machine translation, we have considered NSSTs. We have shown that when their input languages are restricted by MCFGs, then transducers with origin semantics are obtained that can also be generated by synchronous MCFGs. We have further shown that not every synchronous MCFG can be obtained by such a combination of a NSST and a MCFG.

Acknowledgments

Anonymous reviewers provided valuable comments.

References

A.V. Aho and J.D. Ullman. 1970. A characterization of two-way deterministic classes of languages. *J. Comput. Syst. Sci.*, 4(6):523—538.

R. Alur and P. Černý. 2010. Expressiveness of streaming string transducers. In *FSTTCS, volume 8 of LIPIcs*, pages 1—12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

R. Alur and J.V. Deshmukh. 2011. Nondeterministic streaming transducers. In *International Colloquium on Automata, Languages, and Programming (ICALP 2011)*, volume 6756 of *LNCS*, pages 1–20. Springer-Verlag.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley, Reading, Massachusetts.

M. Bojańczyk. 2013. Transducers with origin semantics. Technical report, arXiv:1309.6124, 24. Sep 2013.

M. Bojańczyk, L. Daviaud, B. Guillon, and V. Penelle. 2017a. Which classes of origin graphs are generated by transducers? In *44th Int. Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 114:1–114:13.

M. Bojańczyk, L. Daviaud, B. Guillon, and V. Penelle. 2017b. Which classes of origin graphs are generated by transducers? http://guillon.di.unimi.it/files/automatic/pdf/gb_2017_oxford.pdf.

P.F. Brown, V.J. della Pietra, S.A. della Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(3)(2):263–311.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

B. Courcelle and J. Engelfriet. 2012. *Graph structure and monadic second-order logic*. Cambridge University Press.

A.P. Dempster, N.M. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

J. Engelfriet and H.-J. Hoogeboom. 2001. MSO definable string transductions and two-way finite state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254.

M. Kaeshammer. 2013. Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.

P. Lewis and R. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

A. Lopez. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3):8:1–8:9.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

J.C. Sheperdson. 1959. The reduction of two-way automata to one-way automata. IBM J. Res. Development 3.

$\kappa(1+\sum_{t'\leq t} 4^{t'})^\rho$, as for each register, the short configuration contains either $*$ or a string of length up to t over an alphabet of 4 symbols; their assumption that registers at that point cannot contain symbols a' or b' is insufficiently motivated, and even if it were, there is a discrepancy between $\kappa(1+\sum_{t'\leq t} 2^{t'})^\rho$ and $\kappa\rho 2^t$, which places the correctness of the proof as a whole in doubt.