Equivalence of Finite-Valued Streaming String Transducers Is Decidable

Anca Muscholl
LaBRI, University of Bordeaux, France

Gabriele Puppis
CNRS, LaBRI, Bordeaux, France

Abstract

In this paper we provide a positive answer to a question left open by Alur and and Deshmukh in 2011 by showing that equivalence of finite-valued copyless streaming string transducers is decidable.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases String transducers, equivalence, Ehrenfeucht conjecture

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.122

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.06973`.

Funding DeLTA project (ANR-16-CE40-0007)

1 Introduction

Finite transducers are simple devices that allow to reason about data transformations in an effective, and even efficient way. In their most basic form they transform strings using finite control. Unlike automata, their power heavily depends on various parameters, like non-determinism, the capability of scanning the input several times, or the kind of storage they may use. The oldest transducer model, known as generalized sequential machine, extends finite automata by outputs. Inspired by an approach that applies to arbitrary relational structures [9], logic-based transformations (also called transductions) were considered by Engelfriet and Hoogeboom [12]. They showed that two-way transducers and monadic-second order (MSO) definable transductions are equivalent in the deterministic case (and even if the transduction is single-valued, which is more general than determinism). This equivalence supports thus the notion of "regular" functions, in the spirit of classical results on regular word languages from automata theory and logics due to Büchi, Elgot, Trakhtenbrot, Rabin, and others. A one-way transducer model that uses write-only registers as additional storage was proposed a few years ago by Alur and Cerný [2], and called streaming string transducer (SST). SST were shown equivalent to two-way transducers and MSO definable transductions in the deterministic setting, and again, even in the single-valued case.

In the relational case the picture is less satisfactory, as expressive equivalence is only preserved for SST and non-deterministic MSO transductions [5], which extend the original MSO transductions by existentially quantified monadic parameters. On the other hand, two-way transducers and SST are incomparable in the relational case. Between functions and relations there is however one class of transductions that exhibits a better behavior, and this is the class of finite-valued transductions. Being finite-valued means that there exists some constant k such that every input belonging to the domain has at most k outputs.

Finite-valued transductions were intensively studied in the setting of one-way and two-way transducers. For one-way transducers, k-valuedness can be checked in PTIME [17]. In addition, every k-valued one-way transducer can be effectively decomposed into a union of k

unambiguous one-way transducers of exponential size [24, 23]. For both two-way transducers and SST, checking k-valuedness is in PSPACE.

Besides expressiveness, another fundamental question concerning transducers is the equivalence problem, that is, the problem of deciding whether two transducers define the same relation (or the same partial function if we consider the single-valued case). The equivalence problem turns out to be PSPACE-complete for deterministic two-way transducers [16], single-valued two-way transducers, as well as for single-valued SST [5]. For deterministic SST, equivalence is in PSPACE [3], but it is open whether this complexity upper bound is optimal. For arbitrary SST, and in fact even for non-deterministic one-way transducers over a unary output alphabet, equivalence is undecidable [13, 18]. The equivalence problem for k-valued one-way transducers was shown to be decidable by Culik and Karhumäki using an elegant argument based on Ehrenfeucht's conjecture [10], and the authors noted that the same proof goes through for two-way transducers as well. The decidability status for the equivalence problem for k-valued SST was first stated as an open problem in [5]. Another open problem is whether SST and two-way transducers are equivalent in the finite-valued case, like in the single-valued case. It is worth noting, however, that in the full relational case SST and two-way transducers are incomparable. Concerning this last open question, a partial positive answer was given in [14], by decomposing any finite-valued SST with only one register into a finite union of unambiguous SST. This decomposition result also entails the decidability of the equivalence problem for the considered class.

The main result of this paper is a positive answer to the first question left open in [5]:

▶ **Theorem 1.** *The equivalence problem for finite-valued SST is decidable.*

We show the above result with a proof idea due to Culik and Karhumäki [10], based on the Ehrenfeucht conjecture. Our proof is much more involved, because SST produce their outputs piece-wise, in contrast to one-way and two-way transducers, that produce output linearly while reading the input. We manage to overcome this obstacle using some (mild) word combinatorics and word equations, by introducing a suitable normalization procedure for SST. We believe that our technique will also allow to solve the second problem left open in [5], which is the expressive equivalence between finite-valued SST and two-way transducers.

Related work

The equivalence problem for transducers has recently raised interest for more complex types of transducers in the single-valued case: Filiot and Reynier showed that equivalence of copyful, deterministic SST is decidable by showing them equivalent to HDT0L systems and applying [10], which contains the above-mentioned result as a special case. Subsequently, Benedikt et al. showed that equivalence of copyful, deterministic SST has Ackerman complexity, with a proof based on polynomial automata and ultimately on Hilbert's basis theorem [6]. Interestingly, the use of Hilbert's basis theorem goes back to the proof of Ehrenfeucht's conjecture [1, 15]. A similar approach was used by Boiret et al. in [7] to show that bottom-up register automata over unordered forests have a decidable equivalence problem, see also the nice survey [8].

Overview

Section 2 introduces the transducer model, then Section 3 sets up the technical machinery that allows to normalize finite-valued SST. Section 4 shows the major normalization result, which holds for left quotients of SST. Finally Section 5 recalls the Ehrenfeucht-based proof for equivalence and the application to finite-valued SST. A full version of the paper is available at `https://arxiv.org/abs/1902.06973`.

2 Streaming string transducers

A *streaming string transducer* (*SST*) is a tuple $T = (\Sigma, \Gamma, X, Q, U, I, E, F, x_{\text{out}})$, where Σ and Γ are finite input and output alphabets, X is a finite set of registers (usually denoted x, x', x_1, x_2, etc.), Q is a finite set of states, U is a finite set of register updates, that is, functions from X to $(X \uplus \Gamma)^*$, $I, F \subseteq Q$ are subsets of states, defining the initial and final states, $E \subseteq Q \times \Sigma \times U \times Q$ is a transition relation, describing, for each state and input symbol, the possible register updates and target states, and finally $x_{\text{out}} \in X$ is a register for the output. Note that, compared to the original definition from [2], here we forbid for simplicity the use of final production rules, that perform ad additional register update after the end of the input. This simplification is immaterial with respect to the decidability of the equivalence problem. For example, it can be enforced, without loss of generality, by assuming that all well-formed inputs are terminated by a special marker, say $\dashv$, on which the transducer can apply a specific transition. We assume here that *all inputs of a transducer are non-empty and of the form $u \dashv$, with $\dashv$ not occurring in u.*

Copyless restriction and capacity

An SST as above is *copyless* if for all register updates $f \in U$, every register $x \in X$ appears at most once in the word $f(x_1) \dots f(x_m)$, where $X = \{x_1, \dots, x_m\}$. For a copyless SST, every output has length at most linear in the length of the input. More precisely, every output associated with an input u has length at most $c|u|$, where $c = \max_{f \in U} \sum_{x \in X} |f(x)|_\Gamma$ is the maximum number of letters that the SST can add to its registers along a single transition (this number c is called *capacity* of the SST).

Hereafter, we assume that all SST are copyless.

Register updates and flows

Every register update, and in general every function $f : X \to (X \uplus \Gamma)^*$ is naturally extended to a morphism on $(X \uplus \Gamma)^*$, by defining it as identity over Γ. When reasoning with register updates, it is sometimes possible to abstract away the specific words over Γ, and only consider how the contents of the registers flows into other registers. Formally, the *flow* of an update $f : X \to (X \uplus \Gamma)^*$ is the bipartite graph that consists of two ordered sequences of nodes, one on the left and one on the right, with each node in a sequence corresponding to a specific register, and arrows that go from the node corresponding to register x to a right node corresponding to register x whenever x occurs in $f(x)$. For example, the flow of the update f defined by $f(x_1) = a\,x_1\,aa\,x_3$, $f(x_2) = b\,a$, and $f(x_3) = x_2\,b$ is the second bipartite graph in the figure on page 5.

Note that there are finitely many flows on a fixed number of registers. Moreover, flows can be equipped with a natural composition operation: given two flows F_1 and F_2, $F_1 \cdot F_2$ is the bipartite graph obtained by glueing the right nodes of F_1 with the left nodes of F_2, and by shortcutting pairs of consecutive arrows. We call *flow monoid* of an SST T the monoid of flows generated by the updates of T, with the composition operation as associative product.

Transitions, runs, and loops

A transition (q, a, f, q') of an SST T is conveniently denoted by the arrow $q \xrightarrow[T]{a/f} q'$, and the subscript T is often omitted when clear from the context. A *run* on $w = a_1 \dots a_n$ is a sequence of transitions of the form $q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \dots \xrightarrow{a_n/f_n} q_n$. Sometimes, a run as above is equally denoted by $q_0 \xrightarrow{w/f} q_n$, so as to highlight the underlying input w and the induced register update $f = f_1 \circ \dots \circ f_n$. A run is *initial* (resp. *final*) if it begins with an initial (resp. final) state; it is *successful* if it is both initial and final.

Given two registers x, x' and a run $\rho : q \xrightarrow{w/f} q'$, we say that x *flows into* x' *along* ρ if x occurs in $f(x')$. Note that this property depends only on the flow of the induced update f.

An SST is said to be *trimmed* is every state occurs in at least one successful run, so every state is reachable from the initial states and co-reachable from the final states. This property can be easily enforced with a polynomial-time preprocessing.

When reasoning with automata, it is common practice to use pumping arguments. Pumping will also be used here, but the notion of loop needs to be refined as to take into account the effect of register updates. Formally, a *loop* of a run ρ of an SST is any non-empty factor of ρ of the form $\gamma : q \xrightarrow{w/f} q$, that starts and ends in the same state q, and induces a *flow-idempotent* update, namely, an update f such that f and $f \circ f$ have the same flow.

Outputs and finite-valuedness

The *output* of a successful run $\rho : q_0 \xrightarrow{w/f} q_n$ is defined as $\mathrm{out}(\rho) = (f_0 \circ f)(x_{\mathrm{out}})$, where $f_0(x) = \varepsilon$ for all $x \in X$. Sometimes, we write $\mathrm{out}(f)$ in place of $\mathrm{out}(\rho)$. The *relation realized by an SST* is the set of pairs $(u, v) \in \Sigma^* \times \Gamma^*$, where u is a well-formed input (namely, terminating with $\dashv$) and v is the output associated with some successful run on u. An SST is *k-valued* if for every input u, there are at most k different outputs associated with u. It is *single-valued* (resp. *finite-valued*) if it is k-valued for $k = 1$ (resp. for some $k \in \mathbb{N}$). The domain an SST T, denoted $\mathrm{Dom}(T)$, is the set of input words that have some successful run in T. Two SST T_1, T_2 are *equivalent*, denoted as $T_1 \equiv T_2$, if they realize the same relation over $\Sigma^* \times \Gamma^*$.

Register valuations

A *register valuation* is a function from X to Γ^*. Given a successful run $\rho : q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \dots \xrightarrow{a_n/f_n} q_n$ and a position $i \in \{0, \dots, n\}$ in it, *the register valuation at position i in ρ* is the function $\mathrm{val}_{\rho,i}$ that is defined inductively on i as follows: $\mathrm{val}_{\rho,0}(x) = \varepsilon$, for all $x \in X$, and $\mathrm{val}_{\rho,i+1} = \mathrm{val}_{\rho,i} \circ f_i$. Note that $\mathrm{val}_{\rho,n}(x_{\mathrm{out}})$ coincides with the final output $\mathrm{out}(f_1 \circ \dots \circ f_n)$ produced by ρ.

3 Normalizations

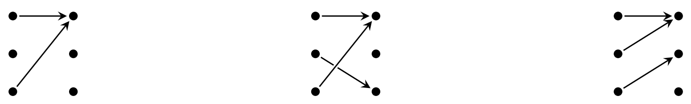
A major stumbling block in deciding equivalence of SST, as well as other crucial problems, lies in the fact that the same output can be produced by very different runs. This phenomenon already appears with much simpler transducers, e.g. with one-way transducers, where runs may produce the same output, but at different speeds. However, the phenomenon is more subtle for SST, as the output is produced piece-wise, and not sequentially: runs with same output may appear to be different in many ways, e.g. in terms of the flows of the register updates, or in terms of shifts of portions of the output. The goal of this section is to provide suitable normalization steps that remove, one at a time, the above mentioned degrees of freedom in producing the same output.

Another issue that we will be concerned with is the compatibility of the normalization steps with constructions on transducers that shortcut arbitrary long runs into a single transition. Essentially, we aim at having an effective notion of equivalence w.r.t. final outputs that works not only for transitions but also for runs.

Normalization of flows

In this section, *m will always denote the number of registers of an SST and $X = \{x_1, \dots, x_m\}$ the set of registers.* It is convenient to equip X with a total order, say $x_1 < \dots < x_m$. Accordingly, we let $\chi = x_1 \dots x_m$ be the juxtaposition of all register names, and $f(\chi) = f(x_1) \dots f(x_m)$ for every register update f.

We say that a register update f is *non-erasing* if for every register x, $f(\chi)$ contains at least an occurrence of x (in fact, exactly one, since T is copyless). This can be rephrased as a property of the flow of f, where every node on the left must have an outgoing arrow. In a similar way, we say that f is *non-permuting* if registers appear in $f(\chi)$ with their natural order and without jumps, that is, $f(\chi) \in \Gamma^* x_1 \Gamma^* \dots \Gamma^* x_k \Gamma^*$, for some $k \leq m$. As before, this can be rephrased by saying that the arrows in the flow of f must not be crossing, and the target nodes to the right must form a prefix of χ. Below are some examples of updates with their flows: the first update f is erasing, the second update g is non-erasing but permuting, and the third update h is non-erasing and non-permuting.



$$f\colon \begin{array}{l} x_1 \mapsto \_\, x_1 \,\_\, x_3 \,\_ \\ x_2 \mapsto \_ \\ x_3 \mapsto \_ \end{array} \qquad g\colon \begin{array}{l} x_1 \mapsto \_\, x_1 \,\_\, x_3 \,\_ \\ x_2 \mapsto \_ \\ x_3 \mapsto \_\, x_2 \,\_ \end{array} \qquad h\colon \begin{array}{l} x_1 \mapsto \_\, x_1 \,\_\, x_2 \,\_ \\ x_2 \mapsto \_\, x_3 \,\_ \\ x_3 \mapsto \_ \end{array}$$

We say that T is *flow-normalized* if all its register updates are non-erasing and non-permuting. Note that a flow-normalized SST with m registers can have at most 2^m different flows.

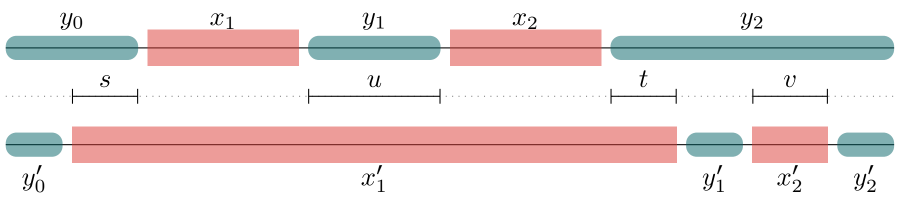
▶ **Proposition 2.** *One can transform any SST into an equivalent flow-normalized one.*

Recall that a register valuation is a function from X to Γ^*. With a flow-normalized SST, one can also define a dual notion of valuation, representing "gaps" between registers that shrink along the run. For this we introduce $m + 1$ fresh variables $y_0, y_1, \dots, y_m$, called *gaps*. Hereafter, $Y = \{y_0, y_1, \dots, y_m\}$ *will always denote the set of gaps.* We use the term *valuation* to generically denote a register/gap valuation, that is, a function from $X \uplus Y$ to Γ^*.

The idea is that a gap y_j represents a word that is inserted between register x_j (if $j > 0$) and register x_{j+1} (if $j < n$) so as to form the final output. Formally, given a word $w \in \Gamma^* x_1 \Gamma^* \dots \Gamma^* x_k \Gamma^*$, with $k \leq m$, and given two registers x_i, x_j, with $i < j$, we denote by $w\langle x_i, x_j \rangle$ the maximal factor of w strictly between the unique occurrence of x_i and the unique occurrence of x_j, using the following conventions for the degenerate cases: if $i = 0$, then $w\langle x_i, x_j \rangle$ is a maximal prefix of w; if $i > 0$ but there is no occurrence of x_i, then $w\langle x_i, x_j \rangle = \varepsilon$; finally, if there is an occurrence of x_i but no occurrence of x_j in w, then $w\langle x_i, x_j \rangle$ is a maximal suffix. Given a run $\rho : q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \dots \xrightarrow{a_n/f_n} q_n$ and a position i in it, *the valuation at position i of ρ* is the function $\mathrm{val}_{\rho,i} : X \uplus Y \to \Gamma^*$ such that

- $\mathrm{val}_{\rho,i}$ restricted to X is the register valuation at position i of ρ,
- $\mathrm{val}_{\rho,i}$ maps every gap y_j to the word $(f_{i+1} \circ \dots \circ f_n)(\chi)\langle x_j, x_{j+1} \rangle$.

By definition, the image of the word $\zeta = y_0\, x_1\, y_1 \dots x_m\, y_m$ via the valuation $\mathrm{val}_{\rho,i}$ is always equal to the final output $\mathrm{out}(\rho)$, for all positions i. In this sense, the sequence of valuations $\mathrm{val}_{\rho,0}, \mathrm{val}_{\rho,1}, \dots, \mathrm{val}_{\rho,n}$ can be identified with a sequence of factorizations of $\mathrm{out}(\rho)$. For example, below are the factorizations of the output before and after a transition with register update f such that $f(x_1) = s\, x_1\, u\, x_2\, t$ and $f(x_2) = v$, for $s, u, t, v \in \Gamma^*$:



This also suggests the principle that gaps, like registers, are updated along transitions via suitable morphisms, but in a symmetric way, that is, from right to left. For instance, in the above picture, the gaps y_0, y_1, y_2 are updated by the function $f^\star$ such that $f^\star(y_0) = y_0\, s$, $f^\star(y_1) = u$, and $f^\star(y_2) = t\, y_1\, v\, y_2$. In general, the function $f^\star$, called *gap update*, is uniquely determined by the register update f, and vice versa, f is uniquely determined by the gap update $f^\star$. Another perhaps interesting phenomenon is that the gap update $f^\star$ is also non-erasing and non-permuting (the notion of non-permuting gap assignment is defined w.r.t. the reverse order $y_m < \cdots < y_0$).

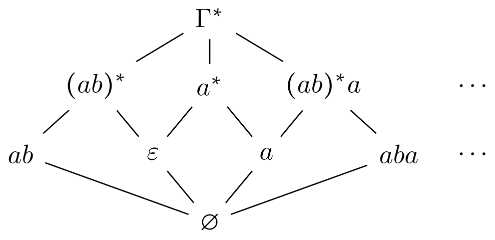
Normalization of states

The next normalization step splits the states of an SST in such a way that it becomes possible to associate with each state an over-approximation of the possible register/gap valuations witnessed when the state is visited along a successful run. These over-approximations are very simple languages over the output alphabet Γ, e.g. singleton languages like $\{aba\}$ and periodic languages like $\{ab\}^*\{a\}$ (often denoted $(ab)^*a$ to improve readability). Basically our over-approximations refer to length and period constraints. The *period* of a word w is the least number $0 < p \le |w|$ such that w is a prefix of $(w[1,p])^\omega$. For example, the period of $w = abcab$ is 3.

For a given parameter $\alpha \in \mathbb{N}$ we define the family $\mathcal{L}_\alpha$ that contains:

- the empty language $\varnothing$,
- the singleton languages $\{u\}$, with $u \in \Gamma^*$ and $|u| \le \alpha$,
- the periodic languages u^*v, with $u \in \Gamma^+$ *primitive* (i.e. $u = w^k$ only if $k = 1$), $|u| \le \alpha$, and $v \in \Gamma^*$ strict prefix of u,
- the universal language Γ^*.

The languages in $\mathcal{L}_\alpha$, partially ordered by containment, form a finite meet semi-lattice, where the meet is the intersection $\cap$. We depict here part of the lattice $\mathcal{L}_\alpha$ for a parameter $\alpha \ge 3$:



The semi-lattice structure allows to derive a best over-approximation in $\mathcal{L}_\alpha$ of any language $L \subseteq \Gamma^*$, that is: $L^{\uparrow\alpha} = \bigcap\{L' \in \mathcal{L}_\alpha : L' \supseteq L\}$. We will mostly use the approximation operator ${}^{\uparrow\alpha}$ on singleton languages. For example, for $\alpha = 3$, we have $\{aba\}^{\uparrow\alpha} = \{aba\}$, $\{ababa\}^{\uparrow\alpha} = (ab)^*a$, and $\{abbb\}^{\uparrow\alpha} = \Gamma^*$. Note also that if $|w| \le \alpha$ then $w^{\uparrow\alpha} = \{w\}$. A useful property is the compatibility of ${}^{\uparrow\alpha}$ with concatenation, which immediately extends to compatibility with word morphisms:

▶ **Lemma 3.** $(L_1 \cdot L_2)^{\uparrow\alpha} = (L_1{}^{\uparrow\alpha} \cdot L_2{}^{\uparrow\alpha})^{\uparrow\alpha}$ *for every* $\alpha \in \mathbb{N}$ *and* $L_1, L_2 \subseteq \Gamma^*$.

Recall that X, Y denote, respectively, the sets of registers and gaps of a flow-normalized SST. Given a valuation $\nu : X \uplus Y \to \Gamma^*$, its *$\alpha$-approximant* is the function $\nu^{\uparrow\alpha} : X \uplus Y \to \mathcal{L}_\alpha$ that maps any $z \in X \uplus Y$ to the language $\{\nu(z)\}^{\uparrow\alpha}$. The set of α-approximants is denoted $\mathcal{L}_\alpha^{X \uplus Y}$, and consists of all maps from $X \uplus Y$ to $\mathcal{L}_\alpha$. Further let

$$\mathrm{Val}_q = \{\mathrm{val}_{\rho,i} \,:\, \rho \text{ successful run visiting } q \text{ at any position } i\}$$

be the set of possible valuations induced by an arbitrary successful run when visiting state q.

A first desirable property is that all valuations in Val_q have the *same* α-approximant, which is thus determined by the state q. Formally, given a flow-normalized SST T with trimmed state space Q, we say that T *admits α-approximants* if every state $q \in Q$ can be effectively annotated with an α-approximant $A_q \in \mathcal{L}_\alpha^{X \uplus Y}$ in such a way that

$$\forall \nu \in \mathrm{Val}_q : \qquad \nu^{\uparrow\alpha} = A_q. \tag{1}$$

This condition is best understood as an invariant on lengths and periods that can be enforced on valuations of registers and gaps when visiting a particular state. For instance, when the approximant A_q guarantees a certain period, then this period will be the same for all valuations occurring at q, independently of the specific initial run that may lead to q (for registers), and of the run that may lead from q to an accepting state (for gaps).

The proposition below shows that it is always possible to refine any SST T so as to admit α-approximants, for any parameter α. The proof for α-approximants that concern only registers could be understood as unfolding the SST T, and merging nodes corresponding to any two inputs u and v, with u prefix of v, whenever the induced α-approximants at u and v are the same for every register x. In general, the resulting SST can be seen a covering of the original SST T, in the sense formalized by Sakarovitch and de Souza in [22]: T' is a *covering* of T if the states of T' can be mapped homomorphically to states of T, while preserving transitions and the distinction into initial and final states, and, moreover, the outgoing transitions of every state of T' map one-to-one to outgoing transitions of a corresponding state of T. This implies that the successful runs of T and those of T' are in one-to-one correspondence.

▶ **Proposition 4.** *Let T be a flow-normalized SST, and let $\alpha \in \mathbb{N}$. One can construct an equivalent flow-normalized SST T' that admits α-approximants and that is a covering of T.*

Notation. Whenever an SST admits α-approximants A_q as above, it is convenient to denote its states by triples of the form (q, A_X, A_Y), where A_X (resp. A_Y) is the restriction of the α-approximant A_q of state q to registers (resp. gaps).

Note that the smaller the parameter α, the weaker is the property required for α-approximants (in particular, for $\alpha = 0$ the lattice $\mathcal{L}_\alpha$ collapses to $\varnothing$ and Γ^*). Choosing α to be at least the capacity of the SST is already a reasonable choice, as it gives a nice characterization of equivalence of transitions w.r.t. the produced outputs (cf. Lemma 5 below). However, we will see that it is desirable to have even finer approximants, in such a way that our results will be compatible with left quotients of SST, that shortcut arbitrary long runs into single transitions. We postpone the technical details to Section 4, and only provide a rough intuition underlying the choice of the appropriate parameter α. We will choose α much larger than the capacity of the SST, so that, by pumping arguments, one can show that, for every state q and every parameter $\beta \geq \alpha$, the β-approximant cannot be strictly smaller than the α-approximant on *all* valuations from Val_q.

Normalization of transitions

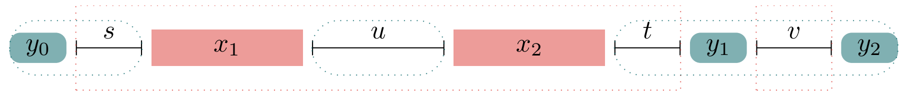
We finally turn to studying a notion of equivalence on transitions that is similar to the two-sided Myhill-Nerode equivalence on words. We will only compare transitions that consume the same input letter and link the same pair of states. Instead of using words as two-sided contexts, we will use initial and final runs that can be attached to the considered transitions in order to form successful runs, and instead of comparing membership in a language, we will compare the effect on the produced outputs.

Consider two transitions $\tau_1 : q \xrightarrow{a/f_1} q'$ and $\tau_2 : q \xrightarrow{a/f_2} q'$. We say that τ_1 and τ_2 are *equivalent* if for every initial run ρ leading to q and every final run σ starting in q', the outputs $\mathrm{out}(\rho\,\tau_1\,\sigma)$ and $\mathrm{out}(\rho\,\tau_2\,\sigma)$ are equal. We often refer to (ρ,σ) as a *context* for τ_1,τ_2.

In general, two transitions of an SST having the same source, target and Σ-label might turn out to be non-equivalent, and still produce the same output within specific contexts. However, Lemma 5 below shows that this is not the case with α-approximants at hand, provided that α is at least the capacity of the SST. More precisely, we will show that the equivalence of two transitions $\tau_1 : (q, A_X, A_Y) \xrightarrow{a/f_1} (q', A'_X, A'_Y)$ and $\tau_2 : (q, A_X, A_Y) \xrightarrow{a/f_2} (q', A'_X, A'_Y)$, where f_1, f_2 have the same flow, only depends on the α-approximants A_X, A'_Y that annotate the source and target states. This will imply that τ_1,τ_2 either always produce the same output or always produce different outputs, independently of the surrounding contexts. To prove the statement, we have to consider register valuations induced by initial runs, and symmetrically gap valuations induced by final runs. It helps to introduce the following:

Notation. Given an initial run ρ, $\mathrm{val}_{\rho\bullet}$ is the register valuation induced at the end of ρ; symmetrically, $\mathrm{val}_{\bullet\sigma}$ is the gap valuation induced at the beginning of a final run σ.

We also recall a consequence of the flow normalization: the effect on the final output of an update f that occurs in a successful run can be described by a word $\mathrm{Effect}(f)$ over the alphabet $X \uplus Y \uplus \Gamma$, defined as $\mathrm{Effect}(f) = y_0\, f(x_1)\, y_1 \dots f(x_m)\, y_{m+1}$. Note that in $\mathrm{Effect}(f)$ each register (resp. gap) occurs exactly once, according to the order $x_1 < \cdots < x_m$ (resp. $y_0 < \cdots < y_m$) – the occurrences of registers and gaps, however, may not be strictly interleaved. In $\mathrm{Effect}(f)$, an occurrence of $x_i \in X$ represents an abstract valuation for register x_i *before* applying the update f, while an occurrence of $y_j \in Y$ represents an abstract valuation for the gap y_j *after* applying f. In particular, note that the x's and the y's refer to valuations induced at different positions of a run. The maximal factors of $\mathrm{Effect}(f)$ that are entirely over Γ represent the words that need to be added in order to get the register valuation *after* f, or equally the gap valuation *before* f. For instance, by reusing the example update f from page 5, where $f(x_1) = s\,x_1\,u\,x_2\,t$ and $f(x_2) = v$, the effect of f is described by the word $\mathrm{Effect}(f) = y_0\,s\,x_1\,u\,x_2\,t\,y_1\,v\,y_2$, suggestively depicted as

$\mathrm{Effect}(f)$ =


(here the lengths of the blocks labeled with variables are immaterial). Note that the y_j above are in fact the y'_j from the picture at page 5. In the above figure we have also highlighted with dotted rectangles the factors that represent gap valuations before the update (e.g. $y_0\,s$), and register valuations after the update (e.g. $s\,x_1\,u\,x_2\,t$).

Given a valuation ν and two approximants $A \in \mathcal{L}^X_\alpha$ and $A' \in \mathcal{L}^Y_\alpha$, one for register valuations and the other for gap valuations, we write $\nu \in A \uplus A'$ to mean that $\nu(x) \in A(x)$ and $\nu(y) \in A'(y)$ for all $x \in X$ and $y \in Y$.

▶ **Lemma 5.** *Let T be a trimmed flow-normalized SST. Given two transitions $\tau_i : q \xrightarrow{a/f_i} q'$, with $i \in \{1,2\}$, a context (ρ,σ) for them, and the α-approximants $A = ((val_{\rho\bullet})^{\uparrow\alpha})|_X$ and $A' = ((val_{\bullet\sigma})^{\uparrow\alpha})|_Y$, with $\alpha \in \mathbb{N}$, the following holds:*

1. *If $Effect(f_1) = Effect(f_2)$ holds on all valuations $\nu \in A \uplus A'$, then $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$.*
2. *If τ_1, τ_2 have the same flow, $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$, and $\alpha \geq c$, where c is the capacity of T, then $Effect(f_1) = Effect(f_2)$ holds on all valuations $\nu \in A \uplus A'$.*

Recall that in an SST that admits α-approximants, states are of the form (q, A_X, A_Y), and we have $((\mathrm{val}_{\rho\bullet})^{\uparrow\alpha})|_X = A_X$ (resp. $((\mathrm{val}_{\bullet\sigma})^{\uparrow\alpha})|_Y = A_Y$) for every initial run ρ that ends in (q, A_X, A_Y) (resp. for every final run σ that starts in (q, A_X, A_Y)). By pairing this with Lemma 5, we immediately obtain the following corollary:

▶ **Corollary 6.** *Let T be a trimmed flow-normalized SST. One can decide in polynomial time whether two given transitions τ_1, τ_2 of T with the same flow are equivalent. Moreover, if T has capacity c and admits α-approximants for some $\alpha \geq c$, then only two cases can happen:*

1. *either $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$ for every context (ρ,σ) (so τ_1, τ_2 are equivalent),*
2. *or $out(\rho\,\tau_1\,\sigma) \neq out(\rho\,\tau_2\,\sigma)$ for* every *context (ρ,σ) (so τ_1, τ_2 are not equivalent).*

Another important consequence is the following theorem, that normalizes finite-valued SST in order to bound the maximum number of transitions linking the same pair of states and consuming the same input letter. This number is called *edge ambiguity* for short.

▶ **Theorem 7.** *Let T be a k-valued, flow-normalized SST that has m registers, capacity c, and that admits α-approximants, for some $\alpha \geq c$. One can construct an equivalent SST T', with the same states and the same registers as T, that has edge ambiguity at most $k \cdot 2^m$.*

Proof. By Corollary 6, T has at most k pairwise non-equivalent transitions with the same input letter, the same source and target states, and the same flow. Moreover equivalence of such transitions can be decided. We can then normalize T by removing in each equivalence class all but one transitions with the same flow. Since T has at most 2^m flows, the normalization results in an equivalent SST T' with edge ambiguity at most $k \cdot 2^m$. ◀

The next section is devoted to prove a very similar result as above, but for all SST that can be obtained by shortcutting runs into single transitions, and that thus have arbitrary large capacity. This will be the main technical ingredient for establishing the decidability of the equivalence problem for k-valued SST.

4 Shortcut construction

Here we focus on a transformation of relations that absorbs the first input letter when this is equal to a specific element, say $a \in \Sigma \setminus \{\dashv\}$. Such a transformation maps any relation R to the relation $R_a = \{(u,v) : (au,v) \in R\}$. Observe that $R = R_\varepsilon \cup \bigcup_{a\in\Sigma} R_a$, where $R_\varepsilon = R \cap (\{\dashv\} \times \Gamma^*)$.

It is easy to see that the class of relations realized by SST is effectively closed under the transformation $R \mapsto R_a$. To prove this closure property, it is convenient to restrict, without loss of generality, to SST with *transient initial states*, namely, SST where no transition reaches an initial state. Under this assumption, the closure property also preserves the state space (though some states may become useless), the set of registers, the property of being k-valued, as well as the α-approximants, if they are admitted by the original SST. However, the transformation does not preserves the capacity, which may increase.

▶ **Lemma 8.** *Given a flow-normalized SST T with transient initial states, and given a letter $a \in \Sigma \setminus \{\dashv\}$, one can construct an SST T_a with transient initial states such that*

- *$Dom(T_a) = \{u \in \Sigma^* : au \in Dom(T)\}$,*
- *T_a on input u produces the same outputs as T on input au.*

Moreover, T_a has the same states and the same registers as T; if T has capacity c, then T_a has capacity $2c$; if T admits α-approximants, then so does T_a (via the same annotation).

The above construction can be applied inductively to compute an SST for any *left quotient* $R_u = \{(v,w) : (uv,w) \in R\}$ of R. For $u = a_1 \dots a_n \in (\Sigma \setminus \{\dashv\})^*$, we let $T_u = (\dots(T_{a_1})_{a_2} \dots)_{a_n}$.

The last and most technical step consists in proving that edge ambiguity can be uniformly bounded in every SST T_u, provided that the initial SST T is finite-valued, flow-normalized, and admits α-approximants for a large enough α. More precisely, we aim at establishing that, for α much larger than the capacity of T, the notion of α-approximant, besides satisfying Equation (1), also satisfies the following property:

$$\begin{aligned} \forall \beta \geq \alpha \quad \exists \rho :& \qquad (\mathrm{val}_{\rho\bullet})^{\uparrow\beta} = (\mathrm{val}_{\rho\bullet})^{\uparrow\alpha} \\ \forall \beta \geq \alpha \quad \exists \sigma :& \qquad (\mathrm{val}_{\bullet\sigma})^{\uparrow\beta} = (\mathrm{val}_{\bullet\sigma})^{\uparrow\alpha}. \end{aligned} \tag{2}$$

In this case we say that the α-approximants are *tight*.

Intuitively, the above property can be explained as follows. When considering an SST with states annotated with α-approximants, it may happen that for some larger parameter β some initial runs induce register valuations at a state (q, A_X, A_Y) whose β-approximants are strictly included in A_X (e.g. possibly entailing new periodicities). These runs should be thought of as exceptional cases, and there is a way of pumping them so as to restore the equality between A_X and the induced β-approximant.

Let us first see how tight approximants are used. The theorem below assumes that there is an SST T' with tight approximants (later we will show how to compute such an SST), and bounds the edge ambiguity of the SST T'_u that realizes a left quotient of T'.

▶ **Theorem 9.** *Let T' be a k-valued, flow-normalized SST realizing R, with transient initial states and tight α-approximants. For every $u \in \Sigma^*$, one can construct an SST T'_u realizing R_u, with the same states and the same registers as T', and with edge ambiguity at most $k \cdot 2^m$.*

Proof. The crux is to show that the SST T'_u obtained from Lemma 8 has at most k pairwise non-equivalent transitions with the same flow (for any given source/target state and label). Once this is proven, one can proceed as in the proof of Theorem 7, by removing all but one transition with the same flow in each equivalence class. By way of contradiction, assume that T'_u has $k+1$ pairwise non-equivalent transitions $\tau_1, \dots, \tau_{k+1}$ with the same flow. Since T' admits tight α-approximants, by Lemma 8 we know that the source and target state, respectively, of the previous transitions are annotated with tight α-approximants, say (A_X, A_Y) and (A'_X, A'_Y), respectively.

We begin by applying the first claim of Lemma 5, implying that the equation $\mathrm{Effect}(f_i) = \mathrm{Effect}(f_j)$ is violated for some valuation $\nu \in A_X \uplus A'_Y$. Then, we let $\beta = \max(\alpha, |u|c)$ and use Equations (1) and (2) to get a context (ρ, σ) such that $(\mathrm{val}_{\rho\bullet})^{\uparrow\beta} = A_X$ and $(\mathrm{val}_{\bullet\sigma})^{\uparrow\beta} = A'_Y$. Finally, knowing that β is at least the capacity of T'_u, we apply the second claim of Lemma 5 to get $\mathrm{out}(\rho\,\tau_i\,\sigma) \neq \mathrm{out}(\rho\,\tau_j\,\sigma)$, thus witnessing non-equivalence of all pairs of transitions τ_i, τ_j at the same time. This contradicts the assumption that T'_u (and hence T') is k-valued. ◀

Now, let T be a flow-normalized SST with m registers, capacity c, and trimmed state space Q. Below, we show how to compute, with the help of Proposition 4, an SST T' equivalent to T that admits tight approximants. For simplicity, we will mostly focus on register valuations induced by initial runs, even though similar results can be also stated for gap valuations induced by final runs. We begin by giving a few technical results based on pumping arguments. We say that register x is *productive* along ρ if the update induced by ρ maps x to a word that contains at least one letter from Γ. We also recall that a loop of a run needs to induce a flow-idempotent update.

▶ **Lemma 10.** *If $\rho = \rho_1\,\gamma\,\rho_2$ is an initial run of T, with γ loop, then for every $n > 0$ the pumped run $\rho^{(n)} = \rho_1\,\gamma^n\,\rho_2$ induces valuations $val_{\rho^{(n)}\bullet}$ mapping any register x to a word of the form $u_0\,v_1^{n-1}\,u_1\,\dots\,v_{2m}^{n-1}\,u_{2m}$, where $u_0,\dots,u_{2m},v_1,\dots,v_{2m} \in \Gamma^*$ depend on ρ and x, but not on n. Moreover, we have $v_i \neq \varepsilon$ for some i if there is a register x' that is productive along γ and that flows into x along ρ_2.*

Given a tuple of pairwise disjoint loops $\overline{\gamma} = \gamma_1,\dots,\gamma_\ell$ in a run ρ, we write $\rho' \trianglerighteq_{\overline{\gamma}} \rho$ when ρ' is obtained from ρ by *simultaneously* pumping n times every loop γ_i, for some $n > 0$. When using this notation, we often omit the subscript $\overline{\gamma}$; in this case we tacitly assume that $\overline{\gamma}$ is *uniquely determined* from ρ. In this way, when writing, for instance, $\rho',\rho'' \trianglerighteq \rho$, we will know that ρ',ρ'' are obtained by pumping the same loops of ρ. We also say that a property on runs holds *for all but finitely many* $\rho' \trianglerighteq \rho$ if it holds on runs ρ' that are obtained from ρ by pumping n times the loops in a fixed tuple $\overline{\gamma}$, for all $n > n_0$ and for a sufficiently large n_0.

▶ **Lemma 11.** *Let ρ be an initial run and x a register. If $val_{\rho\bullet}(x)$ has length (resp. period) larger than $\alpha = m\,c\,|Q|\,2^{3\cdot 2^m}$, then for every $\beta \geq \alpha$ and for all but finitely many $\rho' \trianglerighteq \rho$, $val_{\rho'\bullet}(x)$ has length (resp. period) larger than β.*

Using the previous lemmas and the fact that the type of quantification "for all but finitely many runs" commutes with conjunctions (e.g. those used to enforce properties on each register $x \in X$), we obtain that α-approximants are tight for sufficiently large α:

▶ **Proposition 12.** *Let T' be the SST admitting α-approximants that is obtained from T using Proposition 4, for any $\alpha \geq m\,c\,|Q|\,2^{3\cdot 2^m}$, where Q is the set of states of T. The α-approximants of T' are tight.*

5 Equivalence algorithm

The equivalence algorithm for k-valued SST follows a classical approach of Culik and Karhumäki [10] that is based on so-called *test sets*. A test set for two SST T_1, T_2 over input alphabet Σ is a set $F \subseteq \Sigma^*$ such that T_1, T_2 are equivalent if and only if they are equivalent over F. The main contribution of [10] is to show that *finite* test sets exist and be computed effectively for k-valued one-way transducers.The key ingredient of their proof is to show the existence of a test set that works for *all* transducers with fixed number of states. An essential observation is that for k-valued one-way, or even two-way, transducers one can assume that the edge ambiguity is at most k. The reason for this is simply that the output is generated sequentially. For SST the situation is far more complex because the output is generated piecewise. The purpose of the normalizations performed in Section 3 was precisely to restore the property of bounded edge ambiguity.

In a nutshell, the existence of a test set for transducers is a consequence of Ehrenfeucht's conjecture, whereas the effectiveness is based on the resolution of word equations due to Makanin (see e.g. the survey [11]).

Ehrenfeucht's conjecture was originally stated as a conjecture about formal languages: for every language $L \subseteq \Sigma^*$, there is a finite subset $F \subseteq L$ such that for all morphisms $f, g : \Sigma^* \to \Delta^*$, $f(w) = g(w)$ for every $w \in L$ if and only if $f(w) = g(w)$ for every $w \in F$. Such a set F is called a *test set* for L.

There is an equivalent formulation of Ehrenfeucht's conjecture in terms of a compactness property of word equations [20]. Let Σ and Ω be two alphabets, where the elements in Ω are called unknowns. A word equation is a pair $(u, v) \in \Omega^* \times \Omega^*$, and a solution is a morphism $\sigma : \Omega^* \to \Sigma^*$ such that $\sigma(u) = \sigma(v)$. Ehrenfeucht's conjecture is equivalent to saying that any system of equations over a finite set Ω of unknowns has a finite, equivalent subsystem, where equivalence means that the solution sets are the same. The latter compactness property was proved in [1, 15] by encoding words by polynomials and using Hilbert's basis theorem.

In view of Propositions 2, 4, and 12, we can restrict without loss of generality to SST that are flow-normalized and that admit tight approximants. Hereafter, we shall tacitly assume that all transducers are of this form. Given some integers k, n, m, and e, let $\mathcal{C}_k(n,m,e)$ be the class of k-valued SST with at most n states, m registers, and edge-ambiguity at most e. Note that if T is k-valued, then by Theorem 7 it belongs to $\mathcal{C}_k(n,m,e)$, where n, m are the number of states and registers of T and $e = k \cdot 2^m$. Similarly, by Lemma 8 and Theorem 9, every left quotient T_u also belongs to $\mathcal{C}_k(n,m,e)$.

Now, let us fix k, n, m, e and consider an arbitrary SST T from $\mathcal{C}_k(n,m,e)$. Following [10] we first build an abstraction of T by replacing each maximal factor from Γ^* occurring in some update function of T, by a distinct unknown from Ω. The SST $\Delta(T)$ obtained in this way is called a *schema*; its outputs are words over Ω. Note that the assumption of bounded edge ambiguity is essential here to get a uniform bound on the number of unknowns required for a schema. Clearly, there are only finitely many schemas of SST in $\mathcal{C}_k(n,m,e)$. We denote by $\phi_T : \Omega \rightharpoonup \Gamma^*$ the partial mapping (concretization) that associates with each unknown the corresponding word from Γ^* as specified by the updates of T.

We can rephrase the equivalence $T_1 \equiv T_2$ of two arbitrary SST from $\mathcal{C}_k(n,m,e)$ as an infinite "system" of word equations[1] $\mathcal{S} = \bigwedge_{u \in \Sigma^*} \bigvee_\pi S_\pi$ over set of unknowns $\Omega \uplus \Omega'$. The unknowns from Ω are used for the schema $\Delta(T_1)$, whereas those from Ω' are used for $\Delta(T_2)$; in particular, $\phi_{T_1} : \Omega \to \Gamma^*$ and $\phi_{T_2} : \Omega' \to \Gamma^*$. The disjunctions in $\mathcal{S}$ are finite, with π ranging over the possible schemas Δ_1, Δ_2 (for T_1 and T_2, respectively) and the possible partitions of the set of runs of Δ_1 and Δ_2 over the input u, into at most k groups (one for each possible output). Finally, S_π is a (finite) system of word equations, stating the equality of the words from $\Omega^* \cup \Omega'^*$ that belong to the same group according to π.

The following lemma was stated in [10] for k-valued one-way transducers, but it holds as well for two-way transducers and for SST (even copyful, with a proper definition for $\mathcal{C}_k(n,m,e)$):

▶ **Lemma 13.** *Given two SST T_1, T_2 from $\mathcal{C}_k(n,m,e)$, the system $\mathcal{S} = \bigwedge_{u \in \Sigma^*} \bigvee_\pi S_\pi$ has $\phi_{T_1} \uplus \phi_{T_2}$ as solution if and only if $T_1 \equiv T_2$.*

As shown in [10], the Ehrenfeucht conjecture can be used to show that any infinite system $\mathcal{S}$ as in Lemma 13 is equivalent to some *finite* sub-system $\mathcal{S}_N = \bigwedge_{u \in \Sigma^{\leq N}} \bigvee_\pi S_\pi$. This gives:

▶ **Lemma 14.** *Given $n, m, e \in \mathbb{N}$, there is $N \in \mathbb{N}$ such that $\Sigma^{\leq N}$ is a test set for every pair of SST T_1, T_2 from $\mathcal{C}_k(n,m,e)$.*

Using Theorem 7 and Lemma 14 we can derive immediately the existence of a finite test set for any two k-valued SST. The last question is how to compute such a test set effectively. For this we will use the shortcut construction provided in Section 4.

[1] Formally, $\mathcal{S}$ depends on n and k, but for simplicity we leave out the indices.

▶ **Lemma 15.** *Assume that the formulas $\mathcal{S}_N$ and $\mathcal{S}_{N+1}$ are equivalent, i.e., they have the same solutions. Then $\Sigma^{\le N}$ is a test set for any pair of SST from $\mathcal{C}_k(n,m,e)$.*

Proof. Let $T_1 \equiv_r T_2$ denote equivalence of T_1 and T_2 relativized to $\Sigma^{\le r}$. The goal is to prove that $\Sigma^{\le N}$ is a test set, namely, for all $r > N$ and all $T_1, T_2 \in \mathcal{C}_k(n,m,e)$, $T_1 \equiv_r T_2$ holds if and only if $T_1 \equiv_N T_2$. Clearly, for any $r \ge 0$, $T_1 \equiv_{r+1} T_2$ is equivalent to $T_{1,a} \equiv_r T_{2,a}$ for every $a \in \Sigma$, and $T_1 \equiv_0 T_2$ (the latter being abbreviated as $(*)$ below). Moreover, by Theorem 9, we have $T_{1,a}, T_{2,a} \in \mathcal{C}_k(n,m,e)$. This enables the following proof by induction on r:

$$\begin{array}{ccccc} T_1 \equiv_{r+1} T_2 & \Leftrightarrow & T_{1,a} \equiv_r T_{2,a} \;\; (\forall a \in \Sigma) \;\text{ and } (*) & & T_1 \equiv_N T_2. \\ & & \Updownarrow \text{(ind. hyp.)} & & \Updownarrow \\ & & T_{1,a} \equiv_N T_{2,a} \;\; (\forall a \in \Sigma) \;\text{ and } (*) & \Leftrightarrow & T_1 \equiv_{N+1} T_2 \end{array}$$ ◀

Using Makanin's algorithm for solving word equations (and even for deciding the existential theory of word equations, see e.g. [11] for a modern presentation) we obtain:

▶ **Proposition 16.** *Given $n, m, e \in \mathbb{N}$, there is $N \in \mathbb{N}$ such that $\Sigma^{\le N}$ is a test set for every pair of SST from $\mathcal{C}_k(n,m,e)$, and such an N can be effectively computed.*

Proof. By Lemma 14 we know that N exists, and Makanin's algorithm allows to determine whether $\mathcal{S}_N, \mathcal{S}_{N+1}$ are equivalent, so to determine N by Lemma 15. ◀

We finally obtain the main result:

▶ **Theorem 1.** *The equivalence problem for finite-valued SST is decidable.*

Of course, Theorem 1 does not come with any complexity upper bound, mainly because of the Ehrenfeucht conjecture. The only known lower bound is PSPACE-hardness, which holds even for single-valued SST over unary output alphabets, and follows from a simple reduction from universality of NFA.

Quite surprisingly, the exact complexity of equivalence is not known even for *deterministic* SST, where the problem is known to be between NLOGSPACE and PSPACE [3]. We also recall that equivalence of deterministic SST with *unary output* can be checked in PTIME using invariants [4]. Finally, we recall that the currently best upper bound for solving word equations is PSPACE [21] (with even linear space requirement, as shown in [19]).

6 Conclusions

Our paper answers to a question left open in [5], showing that the equivalence problem for finite-valued SST is decidable. We followed a proof for one-way transducers due to Culik and Karhumäki [10], that is based on the Ehrenfeucht conjecture. The main contribution of the paper is to provide the technical development that allows to follow the proof scheme of [10]. We believe that this development will also allow to obtain stronger results. We conjecture that finite-valued SST can be effectively decomposed into finite unions of unambiguous SST. This would entail that in the finite-valued setting, two-way transducers and SST have the same expressive power, as is it the case for single-valued transducers. If this holds with elementary complexity, then the equivalence of single-valued SST (or two-way transducers) could also be solved with elementary complexity. We believe that the complexity is indeed elementary, and leave this for future work.

References

1 M.H. Albert and J. Lawrence. A proof of Ehrenfeucht's conjecture. *Theor. Comput. Sci.*, 41(1):121–123, 1985.

2 Rajeev Alur and Pavel Cerný. Expressiveness of streaming string transducer. In *IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'10)*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

3 Rajeev Alur and Pavol Cerný. Streaming Transducers for Algorithmic Verification of Single-pass List-processing Programs. In *POPL'11*. ACM, 2011.

4 Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *Proc. of Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013)*, pages 13–22. IEEE, 2013.

5 Rajeev Alur and Jyotirmoy Deshmukh. Nondeterministic streaming string transducers. In *International Colloquium on Automata, Languages and Programming (ICALP'11)*, volume 6756 of *LNCS*. Springer, 2011.

6 Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. Polynomial automata: Zeroness and applications. In *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'17)*, pages 1–12. IEEE, 2017.

7 Adrien Boiret, Radoslaw Piórkowski, and Janusz Schmude. Reducing Transducer Equivalence to Register Automata Problems Solved by "Hilbert Method". In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'18)*, volume 122 of *LIPIcs*, pages 48:1–48:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

8 Mikolaj Bojańczyk. The Hilbert Method for Transducer Equivalence. *ACM SIGLOG News*, January 2019.

9 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

10 Karel Culik II and Juhani Karhumäki. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theor. Comput. Sci.*, 47:71–84, 1986.

11 Volker Diekert. Makanin's Algorithm. In M. Lothaire, editor, *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of mathematics and its applications*, chapter 12, pages 387–442. Cambridge University Press, 2002.

12 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

13 Patrick C. Fischer and Arnold L. Rosenberg. Multi-tape one-way nonwriting automata. *J. Comput. and System Sci.*, 2:88–101, 1968.

14 Paul Gallot, Anca Muscholl, Gabriele Puppis, and Sylvain Salvati. On the Decomposition of Finite-Valued Streaming String Transducers. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS'17)*, volume 66 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

15 Victor S. Guba. Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems. *Mat. Zametki*, 40(3):688—-690, 1986.

16 Eitan M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal of Computing*, 448–452, 1982.

17 Eitan M. Gurari and Oscar H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Math. Syst. Theory*, 16(1):61–66, 1983.

18 Oscar H. Ibarra. The unsolvability of the equivalence problem for e-free NGSM's with unary input (output) alphabet and applications. *SIAM J. of Comput.*, 7(4):524–532, 1978.

19 Artur Jez. Word Equations in Nondeterministic Linear Space. In *Proc. International Colloquium on Automata, Languages, and Programming (ICALP'17)*, volume 80 of *LIPIcs*, pages 95:1–95:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

20 Juhani Karhumäki. The Ehrenfeucht conjecture: a compactness claim for finitely generated free monoids. *Theor. Comput. Sci.*, 29:285–308, 1984.

21 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *JACM*, 51(3):483–496, 2004.

22 Jacques Sakarovitch and Rodrigo de Souza. On the decomposition of k-valued rational relations. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS'08)*, volume 1 of *LIPIcs*, pages 621–632. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2008.

23 Jacques Sakarovitch and Rodrigo de Souza. Lexicographic decomposition of K-valued transducers. *Theory Comput. Sci.*, 47:758–785, 2010.

24 Andreas Weber. Decomposing A k-Valued Transducer into k Unambiguous Ones. *RAIRO-ITA*, 30(5):379–413, 1996.