A Regular and Complete Notion of Delay for Streaming String Transducers

Emmanuel Filiot ✉ 🄔
Université libre de Bruxelles, Belgium

Ismaël Jecker ✉ 🄔
University of Warsaw, Poland

Christof Löding ✉
RWTH Aachen University, Germany

Sarah Winter ✉ 🄔
Université libre de Bruxelles, Belgium

──── **Abstract** ────

The notion of delay between finite transducers is a core element of numerous fundamental results of transducer theory. The goal of this work is to provide a similar notion for more complex abstract machines: we introduce a new notion of delay tailored to measure the similarity between streaming string transducers (SST). We show that our notion is *regular*: we design a finite automaton that can check whether the delay between any two SSTs executions is smaller than some given bound. As a consequence, our notion enjoys good decidability properties: in particular, while equivalence between non-deterministic SSTs is undecidable, we show that equivalence *up to fixed delay* is decidable. Moreover, we show that our notion has good *completeness* properties: we prove that two SSTs are equivalent if and only if they are equivalent up to some (computable) bounded delay. Together with the regularity of our delay notion, it provides an alternative proof that SSTs equivalence is decidable. Finally, the definition of our delay notion is machine-independent, as it only depends on the *origin semantics* of SSTs. As a corollary, the completeness result also holds for equivalent machine models such as deterministic two-way transducers, or MSO transducers.

1 Introduction

Transducers are another fundamental extension of finite automata for computing *functions*, and more generally *relations*, between structures. In this paper, we consider string-to-string transducers, which operate on (input) strings and produce (output) strings. The most basic, *finite transducers*, are obtained by adding output words on the transitions of a finite automaton [33, 21]. At the heart of several important results in the theory of finite transducers is a notion, called *delay*, allowing to finely compare, in a regular way (with a finite automaton), transducer executions. The goal of this paper is to provide such a notion for a strictly more powerful class of transducers, streaming string transducers [1], which have received a lot of attention in the recent years, due to its robustness and equivalence with many other formalisms to define string-to-string functions. In particular, this paper answers positively the following high-level question:

Is it possible to measure in a regular manner how differently executions of equivalent streaming string transducers produce their outputs?

Finite transducers. We first explain how the latter question is answered for finite transducers. Transitions of finite transducers over some alphabet Σ are tuples (p, σ, w, q) where p, q are states, $\sigma \in \Sigma$ is a symbol read on the input tape, and $w \in \Sigma^*$ is a word (possibly empty)
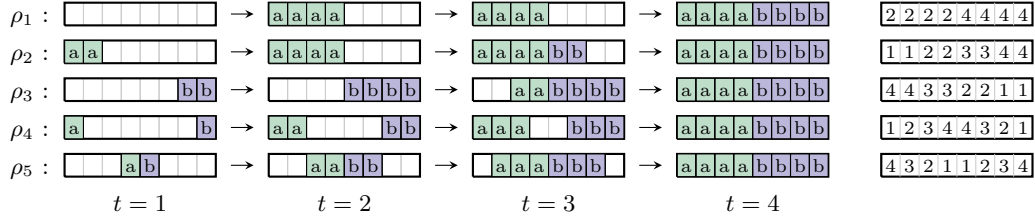
■ **Figure 1** Five ways of producing the output sequence $a^4 b^4$, and the corresponding origin functions.

written on the output tape. A finite transducer execution, i.e., a sequence of successive transitions $\rho = (p_1, \sigma_1, w_1, p_2) \ldots (p_{n-1}, \sigma_n, w_n, p_n)$, operates on the input word $\text{in}(\rho) = \sigma_1 \ldots \sigma_n$ and produces the output word $\text{out}(\rho) = w_1 \ldots w_n$. The semantics of a finite transducer is the set of pairs $(\text{in}(\rho), \text{out}(\rho))$ for all accepting executions ρ. Two different executions ρ_1 and ρ_2 might define the same input/output pair, but can produce the output in a different way because the output words w_i occurring on the transitions of the two runs may differ, although their whole concatenation is the same.

Origin information. Differences in the way transducers produce their output are best captured by the notion of *origin information*, initially proposed for streaming string transducers [10]. An origin function maps any position of the output word to the input position where it was produced. In an execution such as ρ above, any position of w_i has origin i. Figure 1 illustrates five different ways of producing the same output word $a^4 b^4$ (the input word is irrelevant here and not depicted). Consider for example the execution ρ_1. When reading the first input symbol (timestep $t = 1$), nothing is produced. At timestep $t = 2$, a^4 is produced, and its positions have origin 2, as depicted to the right of the figure. The sequence b^4 is produced at timestep $t = 4$ so its corresponding positions have origin 4. Note that ρ_3, ρ_4, ρ_5 do not correspond to finite transducer executions, as $a^4 b^4$ is not produced from left to right.

Delay for finite transducers. A natural way of comparing two finite transducer executions ρ_1, ρ_2 on the same input and producing the same output, is to compare the origin functions o_1, o_2 they induce, and in particular how much one is ahead of the other. In Figure 1, both ρ_1 and ρ_2 produce the same output from left to right, yet at different speed: ρ_1 produces bigger chunks of output at lower frequency. The *delay* between both executions is 2 for $t \in \{1, 3\}$ and 0 for $t \in \{2, 4\}$. The global delay $\text{delay}(\rho_1, \rho_2)$ is defined as the maximal delay along the executions, in this case 2. Two transducers T_1 and T_2 are said to be *k-delay equivalent* if for each run ρ_1 of T_1 or T_2 the other transducer has a run ρ_2 with same input and output satisfying $\text{delay}(\rho_1, \rho_2) \leq k$. This delay notion enjoys two important properties:

Regularity: While the set $E = \{(\rho_1, \rho_2) \mid \text{in}(\rho_1) = \text{in}(\rho_2) \wedge \text{out}(\rho_1) = \text{out}(\rho_2)\}$ is not regular[1] in general for finite transducer executions, its restrictions to pairs of executions with bounded delay is regular: For every $k \in \mathbb{N}$, the set $E_k = \{(\rho_1, \rho_2) \in E \mid \text{delay}(\rho_1, \rho_2) \leq k\}$ is regular. As finite automata compose well with other abstract machines, this allows to use the delay in all kinds of constructions while preserving good decidability properties. For instance k-delay equivalence is decidable [24].

---

[1] Regularity here is defined as classical regularity of word languages, where a pair (ρ_1, ρ_2), are encoded as a single word over a product alphabet of transitions.

Completeness: For any two finite transducers T_1, T_2 defining string-to-string *functions*, there exists a computable bound k such that T_1 and T_2 are equivalent iff they are k-delay equivalent. The completeness even holds in broader classes [24], such as *finite-valued* transducers, for which there is a global bound on the number of outputs mapped to a single input.

Applications. Due to its regularity and completeness, the notion of delay, while basic, proves to be very powerful, and is omnipresent in the study of finite transducers. It provides decidable approximations of various undecidable decision problems, which reveals to be exact for broad classes of transducers, such as finite-valued transducers [24]. Various patterns characterizing important subclasses of transducers are based on the delay notion: For instance, transducers which are *sequential* [14, 9] (i.e., definable by an input-deterministic transducer); equivalent to finite union of sequential functions [28, 20]; finite-valued [30, 32]. Moreover, it has been used to show that any finite-valued transducer can be decomposed as a union of 1-valued transducers [34, 31]. Canonical notions for finite transducers are also based on delay: for input-deterministic transducers [15] and 1-valued transducers [29].

Streaming string transducers and regular functions. *Streaming string transducers* (SST) are obtained by equipping deterministic finite automata with a finite set of registers that store output [1]. Those registers cannot be tested, but are updated by concatenating them, or prepending/appending some symbols to them. E.g., consider a single-state SST with a loop which, whatever it reads as input, updates its single register O by the operation $O := aOb$, and finally outputs the whole content of O. The execution ρ_5 in Figure 1 represents an execution of this SST. Consider an equivalent single-state SST with two registers X, Y which, whatever the input, executes $X := Xa$ and $Y := bY$, and finally outputs the concatenation XY (ρ_4 in Figure 1 produces the output in this way). While equivalent, those two transducers are not equivalent if the origin information is included in the semantics.

SSTs define a robust set of string-to-string functions, called *regular functions*, which can be equivalently defined by deterministic two-way transducers [1], monadic second-order transductions [16, 22, 4]), regular transducer expressions [5, 8, 18, 19], and a logic with origins [17]. Regular functions also enjoy a Krohn-Rhodes decomposition theorem [11]. SSTs have a decidable equivalence problem [27], have been applied to the verification of *list-processing programs* [2], and have been implemented for evaluating string transformations [3].

There are two natural ways of extending the delay defined for finite transducers to compare executions of SSTs with same input and output: we can either simply compare the *quantity* of output produced without caring about the positions where it is produced, or we can count the number of positions whose output has already been produced in one run but not in the other. Both methods match the previously defined delay if the output is produced from left to right, but unfortunately neither preserves conjointly the regularity and the completeness once other output production mechanisms, such as in SSTs, are considered[2]. As the basic notions of delay for SSTs fail to satisfy good properties, better ways of comparing SSTs were introduced, yet none that conjointly has good decidability *and* completeness properties. For instance, *bounded regular resynchronizers* can alter origin functions of SSTs in a controlled manner while preserving good decidability properties, but they are are not complete [13, 12]. In the restricted setting of *single-register* SSTs, a regular and complete notion of delay is defined based on word equations [26]. This approach was applied to prove a

---

[2] We refer the interested reader to the appendix for more details about those natural extensions and an explanation of why they are not satisfactory.

decomposition theorem for single-register SSTs, but unfortunately fails when more registers are allowed.

Contributions.   We define a delay notion for comparing SST executions, based on the origin functions they induce, that is both regular (Theorem 5) and complete for fundamental SST decision problems including equivalence (Theorem 3 and Corollaries 10 and 11). This delay notion is described in the next subsection. We first present our results. Since our notion is based on origins, it more generally applies to regular functions with origin information. SSTs are known to be equivalent to deterministic two-way and MSO transducers, via origin-preserving encodings [10], so we obtain as a corollary that our delay notion is also complete for equivalence of deterministic two-way transducers and MSO transducers (Corollary 4).

The origin semantics allows to recover decidability of several decision problems: While non-deterministic SST equivalence is undecidable, we can decide whether two non-deterministic SSTs define the same relations *with same origins*. However, asking for identical origins is very restrictive: it fails to detect equivalence if the origins are perturbed ever so slightly. This observation was already made in [24] in the context of finite transducers, where it is proposed to relax several decision problems, such as equivalence with same origins, to decision problems with *close* origins, such as equivalence up to a given delay bound k. We prove that the inclusion, equivalence and variable minimization problems up to a given delay bound are decidable for (non-deterministic) SSTs (Theorems 8 and 9), while in general those problems are undecidable. Since our delay notion is complete for equivalence of (deterministic) SSTs, the latter result provides an alternative proof that such SSTs have decidable equivalence problem, and sheds light on the intrinsic reasons why SSTs executions are equivalent.

Delay for SSTs.   Our notion of delay is based on the following observation: The order of production of each output segment that is a power of a small word should have no impact on the delay. For every $\ell \in \mathbb{N}$, we introduce the measure delay_ℓ, which first decomposes the output into blocks that are powers of words smaller than or equal to ℓ. Then, at any position j ending a block (instead of *any* position), we measure the maximal difference at any timestep t, between the number of output positions at the left of j produced before timestep t in the first and second executions. We then take the maximal such value for all j. Consider ρ_4 and ρ_5 for example. delay_1 splits the output $a^4 b^4$ of Figure 1 into two blocks, a^4 and b^4. Consider ending block position $j = 4$. At any timestep, the maximal difference of quantity of outputs produced at the left of j is always 0, because ρ_4, ρ_5 produce exactly one symbol per timestep before position $j = 4$. The same reasoning applies for ending block position $j = 8$, where here instead, exactly two symbols are produced by ρ_4, ρ_5 at any timestep, at the left of position $j = 8$. Hence $\mathsf{delay}_1(\rho_4, \rho_5) = 0$. We refer to Section 3 for a smooth introduction to our delay notion and the main results and to Section 4.1 for a proof of its completeness, and to Section 4.2 for a proof of its regularity. The completeness proof is perhaps the most involved. It is based on a key pumping lemma (Lemma 6), which intuitively states that for any SST, there exist computable bounds k and ℓ such that, if the delay delay_ℓ between two executions on the same input/output is greater than k, then those two executions can be pumped to construct two executions over the same input but with *different* outputs.

2   Preliminaries

We fix our notations. Let $\mathbb{N}$ denote the set of non-negative integers. The interval between integers a and b including resp. excluding a and b is denoted $[a, b]$ resp. (a, b).

Free monoid. Given a finite alphabet Σ, the *free monoid* over Σ is the monoid $(\Sigma^*, \cdot, \varepsilon)$ defined as follows. The set Σ^* is composed of finite sequences of elements of Σ, called *words*. The operation $\cdot$ is the usual word concatenation. The neutral element ε is the empty word.

A subset $L \subseteq \Sigma^*$ is called a *language*. Given a word $u = a_1 \cdots a_n \in \Sigma^*$, $|u|$ denotes its length, $u[i]$ denotes its ith letter a_i, $u[i,j]$ denotes its infix $a_i \cdots a_j$. Given an additional word $v = b_1 \cdots b_n \in \Sigma^*$, $u \otimes v$ denote the *convolution* $\binom{a_1 \cdots a_n}{b_1 \cdots b_n} \in (\Sigma \times \Sigma)^*$.

Automaton. A *(finite state) automaton* is a tuple $A = (\Sigma, Q, I, \Delta, F)$ composed of a finite alphabet Σ, a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a set of transitions $\Delta \subseteq Q \times \Sigma \times Q$. A *run* of A is a sequence $\rho = q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n \in Q(\Sigma Q)^*$ such that $(q_{i-1}, a_i, q_i) \in \Delta$ for every $1 \leq i \leq n$. The *input* of ρ is the word $a_1 a_2 \cdots a_n \in \Sigma^*$. The run ρ is called *accepting* if its first state is initial ($q_0 \in I$) and its last state is final ($q_n \in F$). The automaton A is *deterministic* if I is a singleton and Δ is expressed as a function $\delta : Q \times \Sigma \to Q$. The language recognized by A is the set $L(A) \subseteq \Sigma^*$ composed of the inputs of all its accepting runs.

Substitutions monoid. Given a finite alphabet Σ, a finite set $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ of *variables*, and a designated *output variable* $O \in \mathcal{X}$, the *(copyless) substitutions monoid* $(\mathcal{S}_{\mathcal{X},\Sigma}, \circ, \mathsf{Id}_{\mathcal{X}})$ is defined as follows. The set $\mathcal{S}_{\mathcal{X},\Sigma}$ contains all functions $\sigma : \mathcal{X} \to (\mathcal{X} \cup \Sigma)^*$ such that no variable appears twice in the image of a variable and no variable appears in the images of two distinct variables, i.e., the word $\sigma(X_1) \ldots \sigma(X_n)$ does not contain twice the same variable. The composition $\sigma_1 \circ \sigma_2$ of two substitutions $\sigma_1, \sigma_2 \in \mathcal{S}_{\mathcal{X},\Sigma}$ is obtained by first applying σ_2, and then σ_1. Formally, it is the function $\hat{\sigma}_1(\hat{\sigma}_2(\cdot))$, where each substitution σ is morphically extended to $\hat{\sigma}$ over $(\mathcal{X} \cup \Sigma)^*$ by letting $\hat{\sigma}(X) = \sigma(X)$ and $\hat{\sigma}(\alpha) = \alpha$ for $\alpha \in \Sigma$. We write σ in place of $\hat{\sigma}$. For instance, the substitution composition $(\sigma_1 : X \mapsto \varepsilon) \circ (\sigma_2 : X \mapsto aX) \circ (\sigma_3 : X \mapsto bXc)$ maps X to bac, since $\sigma_3(X) = bXc$, $\sigma_2(bXc) = baXc$ and $\sigma_1(baXc) = bac$. The neutral element $\mathsf{Id}_{\mathcal{X}}$ is the identity function mapping each variable $X \in \mathcal{X}$ to itself. Finally, we denote by σ_ε the substitution mapping each variable $X \in \mathcal{X}$ to ε. The *output* of a substitution σ is the word $\mathsf{out}(\sigma) = (\sigma_\varepsilon \circ \sigma)(O)$.

Streaming string transducer. A *streaming string transducer* (SST) is a tuple $T = (\Sigma, Q, I, \delta, F, \mathcal{X}, O, \kappa, \kappa_F)$ where $A_T = (\Sigma, Q, I, \delta, F)$ is a deterministic automaton, called *underlying automaton* of T, $\mathcal{X}$ is a finite set of variables (also called *registers*), $O \in \mathcal{X}$ is a final variable, $\kappa : \delta \to \mathcal{S}_{\mathcal{X},\Sigma}$ is an output function, and $\kappa_F : Q \to \mathcal{S}_{\mathcal{X},\Sigma}$ is a final output function. A run of T is a run $\rho = q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n \in Q(\Sigma Q)^*$ of A_T, we define $\kappa(\rho)$ as the substitution obtained by composing sequentially all substitutions occurring on the transitions and the final substitutions, i.e. $\kappa(\rho) = \kappa((q_0, a_1, q_1)) \circ \kappa((q_1, a_2, q_2)) \circ \cdots \circ \kappa((q_{n-1}, a_n, q_n)) \circ \kappa_F(q_n) \in \mathcal{S}_{\mathcal{X},\Sigma}$. The *output* of the run ρ is the word $(\sigma_\varepsilon \circ \kappa(\rho))(O)$. The *transduction $R(T)$* recognized by T is the set of pairs $(u,v) \in \Sigma^* \times \Sigma^*$ such that T has an accepting run with input u and output v. Non-deterministic are defined the same way except that A_T is non-deterministic.

▶ **Example 1.** Depicted in Figure 2 are deterministic SSTs T_1 and T_2 that realize the same sorting function $f : \{a, b\}^* \to \{a, b\}^*$ that maps $u \in \{a, b\}^*$ to $a^m b^n$, where m (resp. n) is the number of occurrences of a (resp. b) in u.

Origin information. A *word with origins* is a word $\tilde{u} := u \otimes o \in (\Sigma \times \mathbb{N})^*$ where each letter of u is annotated with a natural number signifying its origin in time. A *transduction with origins* is a relation $R \subseteq \Sigma^* \times (\Sigma \times \mathbb{N})^*$.

Naturally, we associate with a sequence $\lambda = \sigma_1 \sigma_2 \cdots \sigma_n \in \mathcal{S}^*_{\mathcal{X},\Sigma}$ of substitutions the output word with origins $\tilde{\mathsf{out}}(\lambda) = \mathsf{out}(\lambda) \otimes i_1 \cdots i_{|\mathsf{out}(\lambda)|}$ with $i_j = \mathsf{out}(\sigma'_1 \sigma'_2 \cdots \sigma'_n)[j]$ for

$1 \le j \le |\mathsf{out}(\lambda)|$ where for all $1 \le t \le n$, the substitution $\sigma'_t \in \mathcal{S}_{\mathcal{X},\mathbb{N}}$ is obtained by replacing each output letter in σ_t with the number t. Furthermore, we associate with an SST T the *transduction with origins* $R_{\mathsf{ori}}(T)$ that is the set of pairs $(u, \tilde{\mathsf{out}}(\lambda)) \in \Sigma^* \times (\Sigma \times \mathbb{N})^*$ such that T has an accepting run ρ with input u and sequence of substitutions $\kappa(\rho) = \lambda$. For example, regarding the SSTs T_1 and T_2 from Figure 2, we obtain that $(aaa, \binom{aaa}{123}) \in R_{\mathsf{ori}}(T_1)$ and $(aaa, \binom{aaa}{321}) \in R_{\mathsf{ori}}(T_2)$.

3 Delay measure

As mentioned in the introduction, the concept of delay has proven to be a useful tool in the understanding of finite transducers. Our goal is to introduce a robust delay measure suitable to gain a better understanding of streaming string transducers. Towards that, we in-

$a, O := aO$
$b, O := Ob$

$a, X := Xa, Y := Y, O := \varepsilon$
$b, X := X, Y := Yb, O := \varepsilon$

$T_1 \rightarrow \bigcirc\!\!\!\!\bigcirc \; O := O \rightarrow$

$T_2 \rightarrow \bigcirc\!\!\!\!\bigcirc \; O := XY \rightarrow$

Figure 2 Two deterministic SSTs that realize the same sorting function, cf. Example 1.

formally recall the notion of delay for finite transducers: The delay between two finite transducer computations that produce the same output in the end, is a measure for how much ahead one output is compared to the other during the computation. In other words, the difference between the length of the so-far produced output is measured.

A key difference between finite transducers and SSTs is that the former build their outputs from left to right while SSTs do not have this restriction. To design a notion of delay taking this into account, we use origin information to define a notion of *weight difference*. These notions are illustrated in Example 2.

Weight difference. Given a word with origins $\tilde{u} := u \otimes o \in (\Sigma \times \mathbb{N})^*$, a time $t \in \mathbb{N}$, and $j \in \mathbb{N}$, we define the *positional weight* $\mathsf{weight}_{j,t}(\tilde{u}) \in \mathbb{N}$ as the number of positions of u up to j whose origin is no later than t, i.e.,

$$\mathsf{weight}_{j,t}(\tilde{u}) = \big|\{i \in \{1, 2, \ldots, \min(j, |u|)\} \mid o[i] \le t\}\big|.$$

Note that $\mathsf{weight}_{j,t}(\tilde{u}) = \mathsf{weight}_{|u|,t}(\tilde{u})$ for all $j \ge |u|$. Given a second word with origins $\tilde{v} \in (\Sigma \times \mathbb{N})^*$, and $j_1, j_2 \in \mathbb{N}$, we define the *weight difference* as

$$\mathsf{diff}_{j_1,j_2,t}(\tilde{u}, \tilde{v}) = \big|\mathsf{weight}_{j_1,t}(\tilde{u}) - \mathsf{weight}_{j_2,t}(\tilde{v})\big| \text{ and } \mathsf{diff}_{j,t}(\tilde{u}, \tilde{v}) = \mathsf{diff}_{j,j,t}(\tilde{u}, \tilde{v}).$$

Furthermore, we define the *maximal weight difference* as

$$\mathsf{max\text{-}diff}_{j_1,j_2}(\tilde{u}, \tilde{v}) = \max_{t \in \mathbb{N}} \big(\mathsf{diff}_{j_1,j_2,t}(\tilde{u}, \tilde{v})\big) \text{ and } \mathsf{max\text{-}diff}_j(\tilde{u}, \tilde{v}) = \max_{t \in \mathbb{N}} \big(\mathsf{diff}_{j,t}(\tilde{u}, \tilde{v})\big).$$

We remark that the value of $\mathsf{max\text{-}diff}$ is bounded even though t takes infinitely many values. Also, we note that $\mathsf{max\text{-}diff}_0(\tilde{u}, \tilde{v}) = 0$. We illustrate these notions.

▶ **Example 2.** We base our example on the SSTs T_1, T_2 from Figure 2. On input $abaaa$ both SSTs produce output $aaaab$. The associated origins differ, we have $\tilde{u} = \binom{aaaab}{13452}$ and $v = \binom{aaaab}{54312}$. We obtain $\mathsf{diff}_{2,0}(\tilde{u}, \tilde{v}) = 0$, $\mathsf{diff}_{2,1}(\tilde{u}, \tilde{v}) = 1$, $\mathsf{diff}_{2,2}(\tilde{u}, \tilde{v}) = 1$, $\mathsf{diff}_{2,3}(\tilde{u}, \tilde{v}) = 2$, $\mathsf{diff}_{2,4}(\tilde{u}, \tilde{v}) = 1$, $\mathsf{diff}_{2,5}(\tilde{u}, \tilde{v}) = 0$, and $\mathsf{max\text{-}diff}_2(\tilde{u}, \tilde{v}) = 2$. More generally, on input aba^i the output is $a^{i+1}b$ and with origins we have $\tilde{u}_i = \big(\begin{smallmatrix} a & a & a & \cdots & a & a & b \\ 1 & 3 & 4 & \cdots & i+1 & i+2 & 2 \end{smallmatrix}\big)$ and $\tilde{v}_i = \big(\begin{smallmatrix} a & a & \cdots & a & a & a & b \\ i+2 & i+1 & \cdots & 4 & 3 & 1 & 2 \end{smallmatrix}\big)$ for all $i > 0$. Moreover, $\mathsf{max\text{-}diff}_{(i+1)/2}(\tilde{u}_i, \tilde{v}_i) = (i+1)/2$ for all odd i.
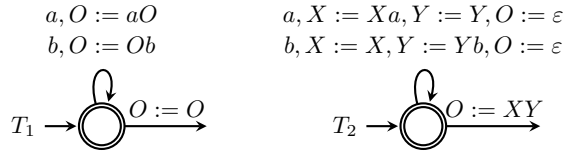
The first idea of a delay notion for SSTs similar to finite transducers is to consider the maximal weight difference that can occur. We go back to Example 2, since T_1 builds the a-output block from left to right and T_2 from right to left, their maximal weight difference is unbounded even though $T_1 \equiv T_2$. To avoid this problem, we would like our notion of delay to reflect that, for a periodic block, it is not important if a repetition of the period is appended or prepended: the result is the same. Therefore, we only measure the difference at the end of periodic blocks and not inside of them. To this end, we introduce a new notion.

Factors, cuts.   The *primitive root* of a word $u \in \Sigma^*$, denoted $\mathsf{root}(u)$, is the shortest word w such that $u = w^k$ for some positive integer k. We call a word $u \in \Sigma^*$ *primitive* if $\mathsf{root}(u) = u$.

Let $u \in \Sigma^*$ and $\ell > 0$. We cut u into factors such that each factor has a primitive root of length at most ℓ. The factors are chosen inductively from left to right in a way to maximize the size of each factor as follows. The first factor u_1 of u is the longest prefix of u such that $|\mathsf{root}(u_1)| \leq \ell$, the ith factor u_i of u is the longest prefix of u' such that $|\mathsf{root}(u_i)| \leq \ell$ where $u = u_1 \cdots u_{i-1} u'$. We refer to this unique ℓ-*factorization* as $\mathsf{factors}_\ell(u)$. Moreover, we denote by $\mathsf{cut}_\ell(u)$ the set that contains the end positions of the factors referred to as ℓ-*cuts*. For example, consider $u = aaababcbabaaaaa$ and $\ell = 2$, then the unique ℓ-factorization is $aaa|ba|bc|baba|aaaa|$ and its ℓ-cuts are $\{3, 5, 7, 11, 15\}$.

Delay.   We define delay for a word and two origin annotations (of said word) by considering the maximal weight difference *at the cut positions* which are obtained via the factorization into periodic words as introduced above. Formally, given a word $w \in \Sigma^*$ and two annotated versions $\tilde{u}, \tilde{v}$ with origin, i.e., $\tilde{u} := w \otimes o_1, \tilde{v} := w \otimes o_2 \in (\Sigma \times \mathbb{N})^*$, we define the ℓ-*delay* $\mathsf{delay}_\ell(\tilde{u}, \tilde{v})$ (delay for short) as

$$\mathsf{delay}_\ell(\tilde{u}, \tilde{v}) = \max_{j \in \mathsf{cut}_\ell(w)} \mathsf{max\text{-}diff}_j(\tilde{u}, \tilde{v}).$$

Going back to Example 2, we have $\mathsf{delay}_1(\tilde{u}_i, \tilde{v}_i) = 0$, because the 1-factorization of $a^{i+1}b$ is $a^{i+1}|b|$ and we have $\mathsf{max\text{-}diff}_{i+1}(\tilde{u}_i, \tilde{v}_i) = \mathsf{max\text{-}diff}_{i+2}(\tilde{u}_i, \tilde{v}_i) = 0$.

We apply the delay notion to transductions with origin. Intuitively, two such transductions are "close" if for every pair $(u, w \otimes o_1)$ (from one transduction) there is some pair $(u, w \otimes o_2)$ (from the other transduction) such that the delay between these outputs with origins is small. Let R_1, R_2 denote transductions with origin. We say that R_1 is (k, ℓ)-*included* in R_2 (written $R_1 \subseteq_{k,\ell} R_2$) if the following holds: for all $(u, w \otimes o_1) \in R_1$, there exists $(u, w \otimes o_2) \in R_2$ such that $\mathsf{delay}_\ell(w \otimes o_1, w \otimes o_2) \leq k$. We say that R_1 is (k, ℓ)-*equivalent* to R_2 (written $R_1 \equiv_{k,\ell} R_2$) if $R_1 \subseteq_{k,\ell} R_2$ and conversely $R_2 \subseteq_{k,\ell} R_1$. We are ready to state our first main result which illustrates the generality of our delay notion.

▶ **Theorem 3** (Completeness). *Given two SSTs T_1 and T_2, there exist computable integers k, ℓ such that $T_1 \equiv T_2$ iff $R_{ori}(T_1) \equiv_{k,\ell} R_{ori}(T_2)$.*

Section 4.1 is devoted to the proof of Theorem 3 and Section 5 illustrates some consequences of Theorem 3. As the delay between streaming string transducers only depends on their induced transductions with origin, we are able to state a more general result. Corollary 4 uses the fact that deterministic streaming string transducers, deterministic two-way transducers and MSO transductions are equally expressive – they characterize the so-called *regular functions* – and every regular function given in one formalism can be translated into every other one without changing its induced transduction with origins [16, 22, 4].

▶ **Corollary 4.** *Given deterministic two-way transducers resp. MSO transductions T_1 and T_2. Let R_1 and R_2 denote their induced transductions with origin. There exist computable integers k, ℓ such that $T_1 \equiv T_2$ iff $R_1 \equiv_{k,\ell} R_2$.*

We focus on the second main aspect of our delay measure, namely, regularity. Meaning that for every $k, \ell \in \mathbb{N}$, we would like to construct a finite automaton that accepts (suitable representations of) pairs $(w \otimes o_1, w \otimes o_2)$ with $\mathsf{delay}_\ell(w \otimes o_1, w \otimes o_2) \leq k$. As finite automata enjoy good closure properties, this yields a useful tool to solve for instance decision problems up to fixed delay, cf. Section 5. As mentioned in the paragraph before Corollary 4, our delay measure is applicable to transductions with origins and is complete for several transducer models. However, we need to pick some way to represent such transductions to show regularity. Hence, we prove our second main result for streaming string transducers.

▶ **Theorem 5** (Regularity). *Let $\mathcal{S} \subseteq \mathcal{S}_{\mathcal{X}, \Sigma}$ be a* finite *subset of $\mathcal{S}_{\mathcal{X}, \Sigma}$, let $k \geq 0$ and $\ell > 0$. The following set is a regular language:*

$$\mathbb{D}_{k, \ell, \mathcal{S}} = \{ \lambda \otimes \mu \in \mathcal{S}^* \times \mathcal{S}^* \mid \mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k \text{ and } |\lambda| = |\mu| \}.$$

Note that $\lambda \otimes \mu \in \mathbb{D}_{k, \ell, \mathcal{S}}$ implies $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$ because delay_ℓ is defined only for such substitution sequences. We write $\mathbb{D}_{k, \ell}$ instead of $\mathbb{D}_{k, \ell, \mathcal{S}}$ when $\mathcal{S}$ is clear from the context. We prove Theorem 5 in Section 4.2 and show some applications of this result in Section 5.

4   Completeness and regularity

4.1   Completeness of the delay notion

We now prove the completeness result for functional SSTs, as stated in Theorem 3. To this end, we show that whenever the delay between two sequences of substitutions is sufficiently large we can pump well-chosen factors to obtain two sequences of substitutions producing outputs that are distinct. Formally, given a sequence of substitutions $\lambda \in \mathcal{S}^*$ and $1 \leq s < t < |\lambda|$, we denote by $\mathsf{pump}_{(s,t]}(\lambda)$ the sequence of substitutions $\lambda[1, t]\lambda(s, t]\lambda(t, |\lambda|)$ obtained by pumping the interval $(s, t]$, and we prove the following:

▶ **Lemma 6.** *Let $\mathcal{S}$ be a finite set of substitutions. There exist computable integers $k, \ell \in \mathbb{N}$ such that for every integer $C \in \mathbb{N}$ and every pair $\lambda, \mu \in \mathcal{S}^*$ that satisfy $|\lambda| = |\mu|$, $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$, and $\mathsf{delay}_{C\ell}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k$, there exist $0 \leq t_1 < t_2 < \ldots < t_C < |\lambda|$ satisfying*

$$\mathsf{out}(\mathsf{pump}_{(t_i, t_j]}(\lambda)) \neq \mathsf{out}(\mathsf{pump}_{(t_i, t_j]}(\mu)) \text{ for every } 1 \leq i < j \leq C.$$

Moreover we can choose k and ℓ exponential with respect to the number of variables of $\mathcal{S}$. Before delving into the proof of Lemma 6, we argue that Theorem 3 follows as a corollary.

Proof of Theorem 3. Let T_1 and T_2 be two SSTs with set of states Q_1 and Q_2. By symmetry, it is sufficient to show that $R(T_1) \subseteq R(T_2)$ iff $R_{\mathsf{ori}}(T_1) \subseteq_{C^2 k, C\ell} R_{\mathsf{ori}}(T_2)$, where $C = |Q_1| \cdot |Q_2| + 1$, and k, ℓ are as in the statement of Lemma 6 with respect to the union $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ of the substitutions used by T_1 and T_2. The right to left direction of the 'iff' is immediate, as inclusion up to bounded delay is stronger than inclusion.

   We now apply Lemma 6 to prove the converse direction. Suppose that $R_{\mathsf{ori}}(T_1) \not\subseteq_{C^2 k, C\ell} R_{\mathsf{ori}}(T_2)$. Then there exists a pair with origins $(u, w \otimes o_1) \in R_{\mathsf{ori}}(T_1)$ such that all the pairs with origin $(u, w \otimes o_2) \in R_{\mathsf{ori}}(T_2)$ with matching input and output satisfy $\mathsf{delay}_{C\ell}(w \otimes o_1, w \otimes o_2) > C^2 k$. Two possible cases arise: either there is no pair of the form $(u, w \otimes o_2)$ in $R_{\mathsf{ori}}(T_2)$, or there exists (at least one) such pair $(u, w \otimes o_2) \in R_{\mathsf{ori}}(T_2)$, and it satisfies $\mathsf{delay}_{C\ell}(w \otimes o_1, w \otimes o_2) > C^2 k$. In the former case, we immediately get that $R(T_1) \not\subseteq R(T_2)$, since (u, w) is in $R(T_1)$ but not in $R(T_2)$. In the latter case, we get that there exists a run
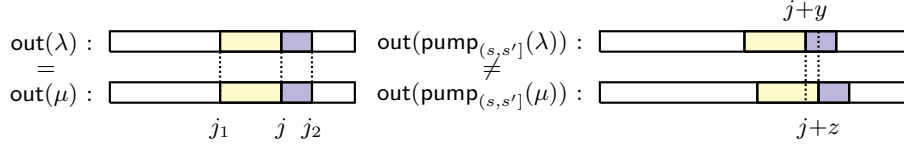
Figure 3 Illustration of the main idea used in the proof of Lemma 6.

ρ_1 of T_1 and a run ρ_2 of T_2 over the same input u that both produce the same output w, but with very different origins functions: $\mathsf{delay}_{C\ell}(\tilde{\mathsf{out}}(\kappa_1(\rho_1)), \tilde{\mathsf{out}}(\kappa_2(\rho_2))) > C^2 k$. Then Lemma 6 implies that we can find C intermediate points in $\kappa_1(\rho_1)$ and $\kappa_2(\rho_2)$ such that iterating the segment between any two points yields sequences of substitutions producing distinct outputs. As $C = |Q_1| \cdot |Q_2| + 1$, two of these points mark a loop in both ρ_1 and ρ_2, and pumping these loops creates runs of T_1 and T_2 with same input but different outputs. Since T_2 is deterministic, and therefore cannot map an input word to two distinct output words, this implies that $R(T_1)$ is not included in $R(T_2)$.   ◀

Proof overview of Lemma 6. We consider two substitution sequences $\lambda, \mu \in \mathcal{S}^*$ that have the same length, produce the same output, and satisfy $\mathsf{delay}_{C\ell}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k$ for some $C, k, \ell \in \mathbb{N}$. This implies the existence of a cut position $j \in \mathsf{cut}_{C\ell}(\mathsf{out}(\lambda))$ and a point $t \in [0, |\lambda|)$ for which the weight difference $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ is equal to $C^2 k$.

Our proof is based on the following fact, illustrated by Figure 3: If we choose k such that $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ is sufficiently large, there are intervals $(s, s'] \subset [1, |\lambda|)$ that, once pumped, add distinct amount of output before position j, thus creating a misalignment between two copies of the letter $\mathsf{out}(\lambda)[j]$ in the output words generated by $\mathsf{pump}_{(s,s')}(\lambda)$ and $\mathsf{pump}_{(s,s')}(\mu)$. Moreover, we show that carefully identifying patterns occurring along the two substitution sequences also allows us to ensure that some neighborhood $\mathsf{out}(\lambda)[j_1, j_2] = \mathsf{out}(\mu)[j_1, j_2]$ of the position j is preserved in both $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))$, in a way that these two copies overlap, but not perfectly (this is the most complex part of the proof). At this point, we use the fact that j is a cut position of $\mathsf{out}(\lambda)$: since j marks the position where the period changes, and this position is not aligned properly in $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))$, we can derive the existence of a mismatch, proving that $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda)) \neq \mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))$.

Then, all that remains to do is to combine a few counting arguments to show how to choose k and ℓ sufficiently large with respect to the parameters of $\mathcal{S}$ (and, crucially, independently of C) so that the fact that $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k$ implies the existence of C consecutive intervals $(s, s'] \subset [1, |\lambda|)$, which, as described earlier, satisfy $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda)) \neq \mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))$, which concludes the proof of Lemma 6.   ◀

4.2   Regularity of the delay notion

Our goal is to prove that the delay notion is regular, as stated in Theorem 5. All over this section, k and ℓ are non-negative integers, and $\mathcal{S}$ is a finite set of substitutions over a finite set of variables $\mathcal{X}$ and an alphabet Σ. Lemma 7 characterizes the pairs of substitution sequences whose outputs end with a unique endmarker symbol $\dashv$ and that are *not* in $\mathbb{D}_{k,\ell,\mathcal{S}}$, using properties which are independently shown to be regular. We first start with the characterization, then give an overview on how to show its regularity.

A characterization. Note that a pair (λ, μ) of two substitution sequences of same length is not in $\mathbb{D}_{k,\ell,\mathcal{S}}$ if either $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$ or there is a cut witnessing a delay greater than k. Unfortunately, the first condition $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$ is not regular. So in order to characterize the complement of $\mathbb{D}_{k,\ell,\mathcal{S}}$ by regular properties, we somehow have to mix conditions on differences in the output and on positions witnessing a big delay. For the corresponding formal statement in Lemma 7 we need the following definition.

Given a sequence of substitutions λ and $i \geq 0$, let $\mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ be the smallest output position j such that $j > i$ and $j \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))$, if it exists. Note that, as the last output position is a cut, such a position j always exists unless i is the last output position. Formally, $\mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ denotes the set $\min(\mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \{j \mid j > i\})$. The result is either a singleton or the empty set, in the former case, we write $\mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda)) = j$ instead of $\mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda)) = \{j\}$.

▶ **Lemma 7.** *Let $\dashv \in \Sigma$ and $\lambda, \mu \in \mathcal{S}^*$ be sequences with $\mathsf{out}(\lambda), \mathsf{out}(\mu) \in (\Sigma \setminus \{\dashv\})^* \dashv$ and $|\lambda| = |\mu|$. Then $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$ iff there exists $i \in (\mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu))) \cup \{0\}$ such that $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$, and one of the following holds:*

1. *Both $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ and $j_2 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu))$ exist, and either $j_1 \neq j_2$ or $\mathsf{max\text{-}diff}_{j_1,j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$;*
2. *$\mathsf{out}(\lambda)[i+b] \neq \mathsf{out}(\mu)[i+b]$ for some $b \in [0, \ell^2]$.*

Proof sketch. Assume that λ, μ satisfy Item 1 or 2 from the above statement. If $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$, then clearly $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$. So assume that $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$. Then $\mathsf{cut}_\ell(\mathsf{out}(\lambda)) = \mathsf{cut}_\ell(\mathsf{out}(\mu))$. Hence Item 1 is satisfied with $j_1 = j_2 =: j$ and $\mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$. Since j is a cut, we obtain that $\mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$ and thus $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$.

Conversely, let $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$. First assume $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$. Since $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$, there exists some $j \in \mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu))$ such that $\mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$, and we can satisfy Item 1. Second, we assume $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$. Let m be the position of the earliest mismatch (that is, $\mathsf{out}(\lambda)[m] \neq \mathsf{out}(\mu)[m]$), and i be the nearest common cut to the left of the mismatch. If $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$, we have a common cut with a too large difference before a mismatch occurs. We can treat this situation as if $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$ and satisfy Item 1 as before. If $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$, we show that either the mismatch is close to i (that is, $m \leq i + \ell^2$) and Item 2 is satisfied, or the mismatch causes $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ and $j_2 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu))$ to be different and Item 1 is satisfied for $j_1 \neq j_2$. ◀

Proof overview for the regularity of the delay notion. Let us denote by $\mathbb{C}_{k,\ell,\mathcal{S}}^\dashv$ the set of words of the form $\lambda \otimes \mu$ such that $\lambda, \mu \in \mathcal{S}^*$ satisfy the properties of the characterization given in Lemma 7. The main technical part is to show that $\mathbb{C}_{k,\ell,\mathcal{S}}^\dashv$ is regular. Then regularity of $\mathbb{D}_{k,\ell,\mathcal{S}}$ follows by complementation and end-marker removal, which preserve regularity.

First, note that the definition of $\mathbb{C}_{k,\ell,\mathcal{S}}^\dashv$ is existential in nature: it asks for the existence of positions in $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$ satisfying some properties. A classical way of dealing with positions quantified existentially in automata theory is to mark some positions in the input by using an extended alphabet, construct an automaton over the extended alphabet, and then project this automaton over the original alphabet. Here, the positions needed in $\mathbb{C}_{k,\ell,\mathcal{S}}^\dashv$ are positions of $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$, while the automata we want to construct read λ and μ as input. So, instead of marking input positions, we rather mark positions in right-hand sides of updates occurring in the substitutions of λ and μ. Let us make this more precise. First, for $n \geq 0$, words u over the alphabet Σ are extended into *n-marked words*, i.e., words over the alphabet $\Sigma \times 2^{\{1,\ldots,n\}}$, such that the additional information in $\{1,\ldots,n\}$ precisely

corresponds to an n-tuple of positions $\overline{x}$ of u (position x_i is marked with label i for all $i \in \{1, \ldots, n\}$, and we consider sets because the same position can correspond to different components of $\overline{x}$). By extension, we also define an operation $\triangleright$ which marks any substitution sequence $\lambda \in \mathcal{S}^*_{\mathcal{X}, \Sigma}$ by a tuple $\overline{x}$ of positions of $\mathsf{out}(\lambda)$, resulting in a substitution sequence $(\lambda \triangleright \overline{x}) \in \mathcal{S}^*_{\mathcal{X}, \Sigma \times 2^{\{1, \ldots, n\}}}$ such that $\mathsf{out}(\lambda \triangleright \overline{x}) = \mathsf{out}(\lambda) \triangleright \overline{x}$.

We show (Lemma 18 in Appendix) that the set of substitution sequences $\lambda \triangleright i$ satisfying $i \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))$ is regular, and similarly for $\mathsf{next\text{-}cut}_\ell$. To do so, we prove that the set $\{u \triangleright i \mid i \in \mathsf{cut}_\ell(u)\}$ is regular (and similarly for $\mathsf{next\text{-}cut}_\ell$) and then transfer this result to marked substitution sequences, as regular languages are preserved under inverse of SSTs.

Then, we show regularity results for predicates of the form $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$. In the end, all parts of the property of the characterization of Lemma 7 are shown to be regular, so that the whole property can be checked by a synchronized product of automata. Perhaps the most interesting part is how to show that the predicate $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ is regular (Lemma 19 in Appendix D.2). More precisely, it is shown that the set of $(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2)$ such that $i_1 = i_2 = i$ and $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ is smaller than k (which is a given constant), is regular. Let us intuitively explain why. In general, checking whether two marked positions i_1 (in $\mathsf{out}(\lambda)$) and i_2 (in $\mathsf{out}(\mu)$) are equal cannot be done in a regular way (recall that the automaton reads $\lambda \otimes \mu$ and not their outputs). However, if additionally, one has to check that $\mathsf{max\text{-}diff}_{i_1, i_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ is smaller than k, it is actually regular. To do so, a finite automaton needs to monitor the difference in the outputs produced in λ and μ before positions i_1 and i_2 resp., and check that it is bounded by k (otherwise it rejects). The difference must eventually reach 0 when the whole inputs λ and μ have been read, to ensure $i_1 = i_2$.

We also prove that once a position i in $\mathsf{out}(\lambda)$ is marked, then the next cut $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ can be identified in a regular way by an automaton (Lemma 18 in Appendix D.2). Similarly, given a constant d, the output position $i + d$ can also be identified in a regular way (Lemma 20 in Appendix D.2). This allows us to check the properties of Item 1 and Item 2 respectively of Lemma 7. This concludes the overview of the proof of Theorem 5. A complexity analysis yields that the set $\mathbb{D}_{k, \ell, \mathcal{S}}$ is recognizable by a DFA with a number of states doubly exponential in ℓ^3 and in $|\mathcal{X}|$, and singly exponential in k.

5 Applications of delay completeness and regularity

Decision problems up to fixed delay. Instead of comparing the transductions defined by non-deterministic SSTs for inclusion or equivalence, which are undecidable problems already for finite (variable-free) transducers, we show here that the strengthening of those problems up to fixed delay, $\subseteq_{k,l}$ and $\equiv_{k,l}$, are decidable. The key to decide inclusion and equivalence up to fixed delay is the regularity of the delay notion as stated in Theorem 5. Hence, those problems can be reduced to classical inclusion and equivalence problems of regular languages.

▶ **Theorem 8.** *Given integers k, ℓ, the (k, ℓ)-inclusion problem for non-deterministic SSTs is decidable. It is* PSPACE-*complete if k, ℓ are constants and the number of variables $|\mathcal{X}|$ is a constant. The same results hold for the (k, ℓ)-equivalence problem.*

Proof sketch. We sketch the result for the inclusion problem. For a non-deterministic SST T over input alphabet Σ, we denote by $L(T)$ its language, defined as the set of words of the form $u \otimes \tau(\rho)$, where $u \in \Sigma^*$, ρ is an accepting run of T over u, and $\tau(\rho)$ is the sequence of substitutions occurring on ρ. Let T_1 and T_2 be two non-deterministic SSTs over two finite sets of variables $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively, both with output variable O. Let $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) be

the finite set of substitutions occurring in T_1 (resp. T_2). Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$. Let $\ell, k \in \mathbb{N}$.

We let $\mathbb{D}_{k,\ell,\mathcal{S}}^{in} = \{u \otimes \lambda \otimes \mu \mid u \in \Sigma^* \wedge \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}\}$. The automaton recognizing $\mathbb{D}_{k,\ell,\mathcal{S}}$ from the proof of Theorem 5 can be easily extended into an automaton which recognizes $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}$. Now, observe that T_1 is (k,ℓ)-included in T_2 iff $L(T_1) \subseteq \mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$, where $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$ denotes the set $\{u \otimes \lambda \mid \exists u \otimes \mu \in L(T_2) \colon u \otimes \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}^{in}\}$. ◀

We turn to the variable minimization problem which is open for SSTs and undecidable for non-deterministic SSTs. We prove decidability for variable minimization up to fixed delay in both cases.

▶ **Theorem 9.** *Given integers k, ℓ, m and a non-deterministic SST T, it is decidable whether $T \equiv_{k,\ell} T'$ for some (non-deterministic resp. deterministic) SST T' that uses at most m variables.*

Proof sketch. First, we sketch the result for non-deterministic SST. Since we are looking for some non-deterministic SST T' with m variables such that $T \equiv_{k,\ell} T'$, we need to consider SSTs that produce at most $r := 2k + p$ letters (where p is the maximal number of letters produced by T in one step) per computation step in order to not violate the delay bound. The reasoning is that the difference between the output of the computations can be at most k letters, then in the next computation step, the computation that was k letters ahead may produce p letters, the other computation must recover the difference by producing at least p letters and at most $2k + p$ letters to keep the difference at most k. Thus, let $\mathcal{S} = \mathcal{S}_T \cup \mathcal{S}_{r,m}$ and $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_m$, where $\mathcal{S}_T$ (resp. $\mathcal{X}_T$) are the substitutions (resp. variables) occurring in T, $\mathcal{X}_m = \{X_1, \cdots, X_m\}$, and $\mathcal{S}_{r,m}$ are substitutions over $\mathcal{X}_m$ producing at most r letters.

As as in the proof of Theorem 8, $L(T)$ is the set of words of the form $u \otimes \tau(\rho)$, where $u \in \Sigma^*$, ρ is an accepting run of T over u, and $\tau(\rho)$ is the sequence of substitutions occurring on ρ. Furthermore, as in the proof above, we let $\mathbb{D}_{k,\ell,\mathcal{S}}^{in} = \{u \otimes \lambda \otimes \mu \mid u \in \Sigma^* \wedge \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}\}$. Now, observe that there exists some non-det SST T' with m variables such that $T \equiv_{k,\ell} T'$ iff $L(T) \subseteq \mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L)$ where L is the set of all words $u \otimes \mu$ such that $u \in \Sigma^*$ and $\mu \in \mathcal{S}_{r,m}^*$. As in the above proof, $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L)$ denotes the set $\{u \otimes \lambda \mid \exists u \otimes \mu \in L \colon u \otimes \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}^{in}\}$.

If the goal is to have a deterministic SST, we need to adapt the above procedure. Let M be the projection of $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}$ onto its first and third component. Hence, M contains all $u \otimes \mu$ such that there is some $u \otimes \lambda \in L(T)$ with $\mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$. We reduce the problem to a safety game played on a DFA for M. In alternation, one player provides an input letter, the other chooses a matching transition in the DFA. If the input player has provided a sequence $u \in \mathrm{dom}(R(T))$ then the play must be in an accepting state of the DFA, otherwise the output player has lost. We show that the output player has a winning strategy iff there exists an equivalent deterministic SST with at most m variables. ◀

Comparison with delay for finite transducers. We compare our notion of delay for streaming string transducers with the previously existing delay notion for finite transducers [24]. A *rational* SST is a non-deterministic SST with only one variable O, and updates all of the form $O := Ou$. In other words, a rational SST T is simply a finite transducer. Applying our delay notion to rational SSTs yields that if T_1 is (k,ℓ)-included in T_2 for two rational SSTs T_1, T_2, then T_1 is k-included in T_2 for the definition according to [24] and vice versa. The k-inclusion problem for finite transducers is PSPACE-complete for fixed k [24]. Hence, the complexity obtained in Theorem 8 matches this bound. Details can be found in Appendix B, including a discussion on conceptual differences between those notions.

Consequences of completeness. We now turn to some consequences of our completeness result (Theorem 3). Inclusion for SSTs is known to be decidable [2]. It is undecidable for non-deterministic SSTs, but (k, ℓ)-inclusion is decidable (Theorem 8). Although Theorem 3 provides a new decision procedure for the inclusion problem for SSTs its value lies in showing that our notion of delay is a sensible approach to gain a better understanding of streaming string transducers. The following two corollaries are easy consequences of Theorem 3.

▶ **Corollary 10.** *Given an SST T and an integer m, there exist integers k, ℓ such that there exists an SST T' with m variables such that $T \equiv T'$ iff there exists an SST T'' with m variables such that $T \equiv_{k,\ell} T''$.*

Note that the above corollary does not imply that k and ℓ are *computable*. This would entail a solution for the variable minimization problem for SSTs which is open (and decidable for concatenation-free SSTs [6]). The next result is about rational functions, that is, functions recognizable by finite transducers.

▶ **Corollary 11.** *Given an SST T, there exist integers k, ℓ such that there exists a rational SST T' such that $T \equiv T'$ iff there exists a rational SST T'' such that $T \equiv_{k,\ell} T''$.*

It was shown that it decidable whether a deterministic two-way transducer (which is effectively equivalent to an SST) recognizes a rational function [23]. The decision procedure is effective, i.e., a finite transducer is constructed if possible. A procedure with improved complexity was given that yields a finite transducer of doubly exponential size in [7]. While computability is not implied by Corollary 11, note that one could compute k, ℓ satisfying the statement of Corollary 11 using Theorem 3 from an equivalent rational SST (if it exists) that has been obtained using the decision procedure from [7].

Other applications. We mention other potential applications of our delay notion that ought to be investigated. For instance, a *decomposition theorem* for regular relations is still only conjectured: Can every finite-valued regular relation be decomposed into a finite union of regular functions? In other settings where the corresponding statement holds (finite transducers [34], single-register SST [26]), the main ingredients of the proof is the regularity and completeness of the appropriate notion of delay. So, having a good delay notion seems necessary to obtain such a result, but solving the decomposition theorem for SSTs does *not* seem to be a low-hanging fruit of our present study.

The notion of delay might also help to solve the *register minimization* problem: Can we determine the minimal number of registers needed to define a given regular function? Corollary 10 makes some progress towards a positive answer, yet it remains to prove that the integers k and ℓ of the statement are computable, which is likely a complex problem. Another interesting research direction is to study how our notion of delay fares beyond SST. For *copyfull* SST (where the copyless restriction of the substitutions is dropped), our notion of delay can be defined in the same manner, but its properties are unclear: our proofs of regularity and completeness both crucially rely on the copyless assumption.

References

---

1 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2010*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. `doi:10.4230/LIPIcs.FSTTCS.2010.1`.
2 Rajeev Alur and Pavol Cerný. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *POPL 2011*, pages 599–610. ACM, 2011.

3   Rajeev Alur, Loris D'Antoni, and Mukund Raghothaman. Drex: A declarative language for efficiently evaluating regular string transformations. In Sriram K. Rajamani and David Walker, editors, *POPL 2015*, pages 125–137. ACM, 2015. `doi:10.1145/2676726.2676981`.

4   Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011. `doi:10.1007/978-3-642-22012-8\_1`.

5   Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 9:1–9:10. ACM, 2014. `doi:10.1145/2603088.2603151`.

6   Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. Minimizing resources of sweeping and streaming string transducers. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 114:1–114:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

7   Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. Untwisting two-way transducers in elementary time. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017.

8   Nicolas Baudru and Pierre-Alain Reynier. From two-way transducers to regular function expressions. *Int. J. Found. Comput. Sci.*, 31(6):843–873, 2020. `doi:10.1142/S0129054120410087`.

9   Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theor. Comput. Sci.*, 292(1):45–63, 2003. `doi:10.1016/S0304-3975(01)00214-6`.

10   Mikolaj Bojanczyk. Transducers with origin information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014. `doi:10.1007/978-3-662-43951-7\_3`.

11   Mikolaj Bojanczyk. Polyregular functions. *CoRR*, abs/1810.08760, 2018. URL: `http://arxiv.org/abs/1810.08760`.

12   Sougata Bose. *On decision problems on word transducers with origin semantics. (Sur les problèmes de décision concernant les transducteurs de mots avec la sémantique d'origine)*. PhD thesis, University of Bordeaux, France, 2021. URL: `https://tel.archives-ouvertes.fr/tel-03216029`.

13   Sougata Bose, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. Origin-equivalence of two-way word transducers is in PSPACE. *CoRR*, abs/1807.08053, 2018. URL: `http://arxiv.org/abs/1807.08053`, `arXiv:1807.08053`.

14   Christian Choffrut. Une caracterisation des fonctions sequentielles et des fonctions soussequentielles en tant que relations rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977. `doi:10.1016/0304-3975(77)90049-4`.

15   Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003. `doi:10.1016/S0304-3975(01)00219-5`.

16   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. URL: `http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR`.

17   Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Logics for word transductions with synthesis. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 295–304. ACM, 2018.

18   Luc Dartois, Paul Gastin, R. Govind, and Shankara Narayanan Krishna. Efficient construction of reversible transducers from regular transducer expressions. In Christel Baier and

Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 50:1–50:13. ACM, 2022.

19 Luc Dartois, Paul Gastin, and Shankara Narayanan Krishna. Sd-regular transducer expressions for aperiodic transformations. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. URL: `https://doi.org/10.1109/LICS52264.2021.9470738`.

20 Laure Daviaud, Ismaël Jecker, Pierre-Alain Reynier, and Didier Villevalois. Degree of sequentiality of weighted automata. In Javier Esparza and Andrzej S. Murawski, editors, *FOSSACS 2017*, volume 10203 of *Lecture Notes in Computer Science*, pages 215–230, 2017. `doi:10.1007/978-3-662-54458-7\_13`.

21 Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, 1965. `doi:10.1147/rd.91.0047`.

22 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. `doi:10.1145/371316.371512`.

23 Emmanuel Filiot, Olivier Gauwin, Pierre-Alain Reynier, and Frédéric Servais. From two-way to one-way finite state transducers. In *LICS 2013*, pages 468–477. IEEE Computer Society, 2013.

24 Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 125:1–125:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

25 N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. URL: `http://www.jstor.org/stable/2034009`.

26 Paul Gallot, Anca Muscholl, Gabriele Puppis, and Sylvain Salvati. On the decomposition of finite-valued streaming string transducers. In Heribert Vollmer and Brigitte Vallée, editors, *STACS 2017*, volume 66 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.STACS.2017.34`.

27 Eitan M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM J. Comput.*, 11(3):448–452, 1982. `doi:10.1137/0211035`.

28 Ismaël Jecker and Emmanuel Filiot. Multi-sequential word relations. In Igor Potapov, editor, *DLT 2015*, volume 9168 of *Lecture Notes in Computer Science*, pages 288–299. Springer, 2015. `doi:10.1007/978-3-319-21500-6\_23`.

29 Christophe Reutenauer and Marcel Paul Schützenberger. Minimization of rational word functions. *SIAM J. Comput.*, 20(4):669–685, 1991. `doi:10.1137/0220042`.

30 Jacques Sakarovitch and Rodrigo de Souza. On the decidability of bounded valuedness for transducers. In Edward Ochmanski and Jerzy Tyszkiewicz, editors, *MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 588–600. Springer, 2008. `doi:10.1007/978-3-540-85238-4\_48`.

31 Jacques Sakarovitch and Rodrigo de Souza. On the decomposition of k-valued rational relations. In Susanne Albers and Pascal Weil, editors, *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, volume 1 of *LIPIcs*, pages 621–632. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008. `doi:10.4230/LIPIcs.STACS.2008.1324`.

32 Jacques Sakarovitch and Rodrigo de Souza. Lexicographic decomposition of *k*-valued transducers. *Theory Comput. Syst.*, 47(3):758–785, 2010. `doi:10.1007/s00224-009-9206-6`.

33 Marcel Paul Schützenberger. A remark on finite transducers. *Inf. Control.*, 4(2-3):185–196, 1961. `doi:10.1016/S0019-9958(61)80006-5`.

34 Andreas Weber. Decomposing A *k*-valued transducer into *k* unambiguous ones. *RAIRO Theor. Informatics Appl.*, 30(5):379–413, 1996. `doi:10.1051/ita/1996300503791`.

A   Discussion on natural delay notions for SSTs

We illustrate here two natural delay notions for SSTs and their flaws, using the executions of Figure 1.

- The measure delay_∞ computes the maximal difference between the sizes of the partial outputs occurring along two runs. For instance, since the runs ρ_2, ρ_3, ρ_4 and ρ_5 all steadily produce two output symbols (albeit different ones) at each step, $\mathsf{delay}_\infty(\rho_i, \rho_j) = 0$ for all $i, j \in \{2, 3, 4, 5\}$. This measure is complete for SSTs, but is not regular: as witnessed by Figure 1, a small delay does not imply any similarities between the partial output produced, thus unbounded memory might be required to ensure that the final outputs are equal. This causes the undecidability of the related decision problems.

- The measure delay_0 computes the maximal size of the *symmetric difference* between the partial outputs occurring along two runs. For instance $\mathsf{delay}_0(\rho_2, \rho_4) = \mathsf{delay}_0(\rho_3, \rho_5) = 4$ and $\mathsf{delay}_0(\rho_2, \rho_3) = \mathsf{delay}_0(\rho_4, \rho_5) = 8$, witnessed at $t = 2$. While the high value of $\mathsf{delay}_0(\rho_2, \rho_3)$ is desirable (ρ_2 starts by producing only a's while ρ_3 produces b's), the high value of $\mathsf{delay}_0(\rho_4, \rho_5)$ is problematic: the partial outputs of the two runs are not *so* different, as they commute in the final word. This behavior leads to the incompleteness of delay_0 for functional SSTs: for instance, over a single-letter alphabet the SST that copies the input and the SST that reverses it are equivalent, yet their delay is unbounded.

We refer the reader to Appendix B for a discussion of the differences between delay notions for finite transducers and streaming string transducers.

B   Comparison with delay for finite transducers

Finite Transducers.   Formally, a *(real-time finite state) transducer* (NFT) is a tuple $T = (\Sigma, Q, I, \Delta, F, \kappa)$ where $A_T = (\Sigma, Q, I, \Delta, F)$ is an automaton, called *underlying automaton* of T, and $\kappa \colon \Delta \to \Sigma^*$ is an output function. A run of T is a run

$$\rho = q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n \in Q(\Sigma Q)^*$$

of A_T, and the *output* of ρ is the word

$$\kappa(\rho) = \kappa((q_0, a_1, q_1))\kappa((q_1, a_2, q_2)) \cdots \kappa((q_{n-1}, a_n, q_n)) \in \Sigma^*.$$

The *transduction* $R(T)$ recognized by T is the set of pairs $(u, v) \in \Sigma^* \times \Sigma^*$ such that T has an accepting run with input u and output v. If $R(T)$ is a (partial) function, T is called *functional*. If $\kappa : \Delta \to \Sigma$, then T is called *letter-to-letter*.

Rational SSTs.   As already said in the main body of the paper, any rational SST T can be seen as a finite transducer, that we denote by $\mathsf{NFT}(T)$, and which is defined as expected. As mentioned before, the delay between two finite transducer computations with the same input and output is a measure of how ahead one output is compared to the other. More formally[3], let T_1 and T_2 be NFTs, given runs $\rho_1 = q_0 a_1 q_1 \cdots a_n q_n$ and $\rho_2 = p_0 a_1 p_1 \cdots a_n p_n$ of T_1 resp. T_2 with the same input and output, then $\mathsf{delay}(\rho_1, \rho_2)$ according to [24] is defined

---

[3] In [24], delay is defined between synchronizations instead of runs, for ease of presentation, we do not formally introduce this notion here.

as

$$\max_{0 \leq i \leq n} ||\kappa(q_0, a_1, q_1) \cdots \kappa(q_{i-1}a_i, q_i)| - |\kappa(p_0, a_1, p_1) \cdots \kappa(p_{i-1}a_i, p_i)||. \tag{1}$$

Note that this definition of delay pays no attention to periodic factors. In the proof of the following proposition, we show that this indeed plays no role for NFTs. The reason is that NFTs build their output from left to right; there are no gaps in the output (as, e.g., depicted in Figure 4) during the computation as for SSTs. As we explained in the introduction and in Section 3, is does not play a role how periodic factors are built (appending or prepending a period yields the same result), cutting the output into periodic factors avoids including irrelevant intermediate gaps in the output into the delay computation. Since computations of NFTs do not have these gaps, there is no need to cut the word into periodic factors.

▶ **Proposition 12.** *Let T_1, T_2 be two rational SSTs, and $k \in \mathbb{N}$. Then, $\mathsf{NFT}(T_1)$ is k-included in $\mathsf{NFT}(T_2)$ (definition of [24]) iff T_1 is (k, ℓ)-included in T_2 for any $\ell \geq 1$.*

Proof. Let $\rho_1 = q_0 a_1 q_1 \cdots a_n q_n$ denote a run of $\mathsf{NFT}(T_1)$, and $\rho_2 = p_0 a_1 p_1 \cdots a_n p_n$ denote a run of $\mathsf{NFT}(T_2)$ with the same input and output. Furthermore, let λ and μ denote the corresponding sequences of substitutions of T_1 and T_2, respectively.

First, we express $\mathsf{delay}(\rho_1, \rho_2)$ using our notations. For all $0 \leq i \leq n$, let j_i denote

$$\max\{|\kappa(q_0, a_1, q_1) \cdots \kappa(q_{i-1}a_i, q_i)|, |\kappa(p_0, a_1, p_1) \cdots \kappa(p_{i-1}a_i, p_i)|\}.$$

We express

$$||\kappa(q_0, a_1, q_1) \cdots \kappa(q_{i-1}a_i, q_i)| - |\kappa(p_0, a_1, p_1) \cdots \kappa(p_{i-1}a_i, p_i)||$$

as $\mathsf{diff}_{j_i,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$. Note that $\mathsf{diff}_{j_i,i}(\lambda, \mu) = \mathsf{diff}_{j_k,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ for all $i \leq k \leq n$ by definition of diff, because there is no output right of j_i after the ith computation step, as the output is build left to right. So,

$$
\begin{aligned}
\mathsf{delay}(\rho_1, \rho_2) &= \max_{0 \leq i \leq n} ||\kappa(q_0, a_1, q_1) \cdots \kappa(q_{i-1}a_i, q_i)| - |\kappa(p_0, a_1, p_1) \cdots \kappa(p_{i-1}a_i, p_i)|| \\
&= \max_{0 \leq i \leq n} \mathsf{diff}_{j_i,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \\
&= \max_{0 \leq i \leq n} \mathsf{diff}_{j_n,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \\
&= \mathsf{max\text{-}diff}_{j_n}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)).
\end{aligned}
$$

Recall that by definition, we have

$$\mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = \max_{j \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))} \mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)).$$

Furthermore, note that $j_n \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))$, because the last output position is always a cut.

Assume $\mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = d$ for some $\ell \geq 1$, we show $\mathsf{delay}(\rho_1, \rho_2) \leq d$. Since $j_n \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))$, we have $\mathsf{max\text{-}diff}_{j_n}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq d$. As stated above, $\mathsf{delay}(\rho_1, \rho_2) = \mathsf{max\text{-}diff}_{j_n}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$. Hence, $\mathsf{delay}(\rho_1, \rho_2) \leq d$.

For the other direction, assume $\mathsf{delay}(\rho_1, \rho_2) = d$. We show $\mathsf{delay}_\ell(\lambda, \mu) \leq d$ for any $\ell \geq 1$, i.e., we prove that $\mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq d$ for any $j \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))$. For each such j there is some $i < n$ such that $j_i < j \leq j_{i+1}$. This means at least one of λ and μ produces output at position j in the $i+1$th computation step and the other computation produces this output later (or also in the $i+1$th step). After the ith step, we have $\mathsf{diff}_{j_i,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = d_i \leq d$, and after the $i+1$th step, we have $\mathsf{diff}_{j_{i+1},i+1}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = d_{i+1} \leq d$. Since after the ith step the outputs have length at most $j_i < j$, we can conclude that $\min\{d_i, d_{i+1}\} \leq \mathsf{diff}_{j,i+1}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq \max\{d_i, d_{i+1}\}$.
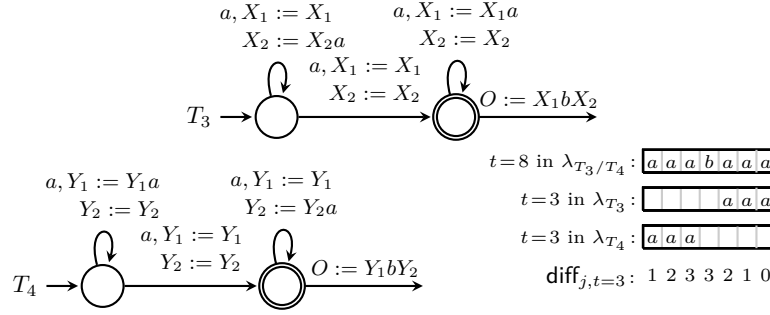
Figure 4 Two SSTs that realize the same relation, λ_{T_3} resp. λ_{T_4} is the sequence of substitutions of the run of T_3 resp. T_4 on *aaaaaaa* where each loop is taken 3 times, cf. Example 13.

As a last step, note that $\mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ is the maximum of

$$\max_{0 \leq i} \mathsf{diff}_{j_i,i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \text{ and } \mathsf{diff}_{j,i+1}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)).$$

Hence, we obtain that $\mathsf{max\text{-}diff}_j(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq d$.
We have proven that $\mathsf{delay}_\ell(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq d$ for any $\ell \geq 1$. ◀

We now turn our attention to some subtleties of the delay notion for streaming string transducers. We start with an example.

▶ **Example 13.** Depicted in Figure 4 are SSTs T_3 and T_4 that realize the transduction $R(T_3) = R(T_4) = \{(a^{m_1+m_2+1}, a^{m_1}ba^{m_2}) \mid m_1, m_2 \geq 0\}$. Let ρ_{T_3} resp. ρ_{T_4} be the run of T_3 resp. T_4 on the input word a^7 where both loops are taken three times, let λ_{T_3} resp. λ_{T_4} denote the associated substitution sequence of length 8 (because we have a final substitution). We have $\mathsf{out}(\lambda_{T_3}) = \mathsf{out}(\lambda_{T_4}) = aaabaaa$. In Figure 4, on the right-hand side, is a visualization of the output produced by λ_{T_3} resp. λ_{T_4} after 3 computation steps. The output letters are placed at the positions they will have in the final output word. The intermediate step shows that the common output word is built quite differently. The figure displays $\mathsf{diff}_{j,3}(\tilde{\mathsf{out}}(\lambda_{T_3}), \tilde{\mathsf{out}}(\lambda_{T_4}))$ for all output positions j. Computing the maximal difference over all computation steps, we have, e.g, $\mathsf{max\text{-}diff}_4(\tilde{\mathsf{out}}(\lambda_{T_3}), \tilde{\mathsf{out}}(\lambda_{T_4})) = 3$, $\mathsf{max\text{-}diff}_6(\tilde{\mathsf{out}}(\lambda_{T_3}), \tilde{\mathsf{out}}(\lambda_{T_4})) = 1$ and $\mathsf{max\text{-}diff}_8(\tilde{\mathsf{out}}(\lambda_{T_3}), \tilde{\mathsf{out}}(\lambda_{T_4})) = 0$. All in all, we obtain $\mathsf{delay}_1(\tilde{\mathsf{out}}(\lambda_{T_3}), \tilde{\mathsf{out}}(\lambda_{T_4})) = 3$, because the 1-cuts of *aaabaaa* are at $3, 4, 8$.

We recall that we defined our delay notion (for streaming string transducers) with two properties in mind: Completeness (Theorem 3) in order to have generality and regularity (Theorem 5) in order to ensure good decidability properties. The delay notion for finite transducers introduced in [24] also has these qualities.

We have explained in Section 3 why we introduced the notion of cuts. These ensure to measure delay at relevant output positions, a key ingredient to achieve completeness (while maintaining regularity which we explain below). We have explained at the beginning of this section and shown in the proof of Proposition 12 that the notion of cuts is irrelevant for a good notion of delay for finite transducers.

We now want to highlight a key difference between the delay notion for SSTs and the delay notion for finite transducers that is needed to obtain regularity. Recall the definition of delay between two finite transducer runs (Equation (1)): the delay is measured between *intermediate* outputs (produced on the so far processed input), then the maximum of these delays is taken. This is in stark contrast to our delay notion: The delay is measured once

between the *complete* outputs. We come back to Example 13 to explain why this is necessary to obtain regularity.

▶ **Example 14.** We generalize Example 13. Given $m, n \in \mathbb{N}$, let $\alpha_{m,n}$, resp. $\beta_{m,n}$, be the sequence of substitutions of T_3, resp. T_4, over the input a^{m+n+1} that takes the first loop m times and the second loop n times. The output $a^n b a^m$ of $\alpha_{m,n}$ and the output $a^m b a^n$ of $\beta_{m,n}$ are equal if and only if $m = n$, hence the set $M := \{\alpha_{m,n} \otimes \beta_{m,n} \mid \mathsf{out}(\alpha_{m,n}) = \mathsf{out}(\beta_{m,n})\}$ is not regular: an unbounded memory is required to check that the first loop and the second loop are taken the same number of times.

Let us measure the delay between $\alpha_{m,n}$ and $\beta_{m,n}$ for all even $m, n \in \mathbb{N}$ using the delay notion introduced for finite transducers: Until the last substitution, the so far produced outputs cannot be differentiated (both are a-blocks of the same length), there is no delay. Finally, the last substitutions insert a b, cutting the a-blocks into two. In order to know if the outputs are still equal, it would have been necessary to keep track of the number of times each loop is taken.

The fact that one needs to keep track of the number of times each loop is taken is reflected by our delay notion: We obtain $\mathsf{delay}_1(\mathsf{o\tilde{u}t}(\alpha_{m,n}), \mathsf{o\tilde{u}t}(\beta_{m,n})) = m$ for all even $m = n \in \mathbb{N}$, because we consider the 1-factorization of the *complete* output, namely, $a^m|b|a^m|$. Thus, measuring the difference between outputs at the *cuts* yields that $\mathsf{max\text{-}diff}_m(\mathsf{o\tilde{u}t}(\alpha_{m,n}), \mathsf{o\tilde{u}t}(\beta_{m,n})) = m$ for all even $m = n \in \mathbb{N}$. So, it is easy to see that there is no $k \in \mathbb{N}$ such that the regular (according to Theorem 5) set $\{\alpha_{m,n} \otimes \beta_{m,n} \mid \mathsf{delay}_1(\mathsf{o\tilde{u}t}(\alpha_{m,n}), \mathsf{o\tilde{u}t}(\beta_{m,n})) \leq k\}$ is equal to M.

C   Completeness of the delay notion

This section is devoted to the proof of Lemma 6. We begin by setting the notation used throughout the proof. Let $\mathcal{S}$ be a finite substitution alphabet with finite alphabet Σ and set of variables $\mathcal{X}$. We fix three integers $C, k, \ell \in \mathbb{N}$ and two sequences of substitutions $\lambda = \sigma_1 \sigma_2 \ldots \sigma_n \in \mathcal{S}^*$ and $\mu = \tau_1 \tau_2 \ldots \tau_n \in \mathcal{S}^*$ satisfying $|\lambda| = |\mu|$, $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$, and $\mathsf{delay}_{C\ell}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > C^2 k$. We preemptively modify λ_n and μ_n to add 2ℓ padding symbols $\# \notin \Sigma$ at the start and end of $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$: this ensures that the cuts of $\mathsf{out}(\lambda)$ are not too close to its extremities (and does not change the delay). By definition, $\mathsf{delay}_{C\ell}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > C^2 k$ implies the existence of positions $j \in \mathsf{cut}_{C\ell}(\mathsf{out}(\lambda))$ and $t \in [1, n]$ such that $\mathsf{diff}_{j,t}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > C^2 k$. We fix such a pair j and t. The proof revolves around a careful study of what happens to the infix $\mathsf{out}(\lambda)[j - 2C\ell, j + C\ell]$ when some interval is pumped. To simplify the notation we let $j_1 = j - 2C\ell$ and $j_2 = j + C\ell$ (see Figure 5).

Structure of the proof of Lemma 6.   We show that if k and ℓ are sufficiently large, then $\mathsf{diff}_{j,t}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > C^2 k$ implies the existence of $0 \leq t_1 < t_2 < \ldots < t_C < |\lambda|$ such that $\mathsf{out}(\mathsf{pump}_{(t_i, t_j]}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(t_i, t_j]}(\mu))$ are distinct for every $1 \leq i < j \leq C$. This is done in two steps. First, we establish a property P for intervals $(s, s'] \subset [1, |\lambda|)$ that is sufficient for the words $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$ to be distinct (Claim 15). It can be summarized as follows: If both $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$ contain a copy of the word $\mathsf{out}(\lambda)[j_1, j_2]$ but these copies do not overlap properly, the fact that $j \in \mathsf{cut}_{C\ell}(\mathsf{out}(\lambda))$ implies the existence of a mismatch between $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$. Second, we show how to ensure property P by carefully regulating the quantity of output produced and the arrangement of the variables with respect to the neighborhood $[j_1, j_2]$ of $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$. Formally, we establish three conditions C_1, C_2 and C_3 sufficient to get property P
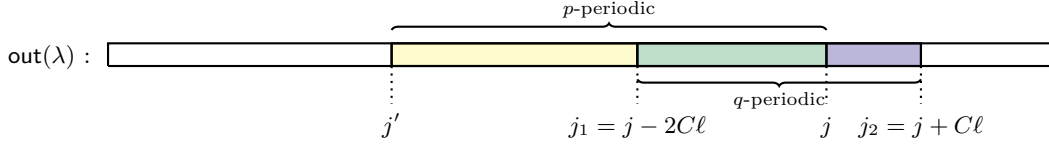
Figure 5 The different factors of $\mathsf{out}(\lambda)$ studied in the proof of Claim 15.

(Claim 16). We then conclude by showing that, since $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k$, choosing k and ℓ sufficiently large with respect to the parameters of $\mathcal{S}$ (and, crucially, independently of C) guarantees the existence of C consecutive intervals satisfying C_1, C_2 and C_3 (Claim 17). Combining the three claims yields the proof of Lemma 6.

One sufficient property.   An interval $(s, s') \subset [1, |\lambda|)$ *satisfies Property P* if the output words generated by $\mathsf{pump}_{(s,s']}(\lambda)$ and $\mathsf{pump}_{(s,s']}(\mu)$ both contain a copy of the word $\mathsf{out}(\lambda)[j_1, j_2] = \mathsf{out}(\mu)[j_1, j_2]$ such that these copies overlap partially, with a shift of at most $C\ell/2$. Formally:

P. There exist $x, y \in \mathbb{N}$ such that:
$$\begin{cases} \mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda))[j_1 + x, j_2 + x] = \mathsf{out}(\lambda)[j_1, j_2]; \\ \mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))[j_1 + y, j_2 + y] = \mathsf{out}(\mu)[j_1, j_2]; \\ 0 < |x - y| \leq C\ell. \end{cases}$$

Since $j \in \mathsf{cut}_{C\ell}(\mathsf{out}(\lambda))$ we can derive from Property P the existence of a mismatch between $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$.

$\triangleright$ Claim 15.   For each interval (s, s') satisfying P, $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda)) \neq \mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$.

Proof. The proof relies on the Fine-Wilf theorem [25], a fundamental result in the study of combinatorics on words, that we now remind: For each $p > 0$, we say that a word $w \in \Sigma^*$ is *p-periodic* if $w[i] = w[i + p]$ for all $1 \leq i \leq |w| - p$. Given $p, q > 0$ and $u, v, w \in \Sigma^*$, if uv is p-periodic, vw is q-periodic and $|v| \geq p + q - gcd(p, q)$, then uvw is $gcd(p, q)$-periodic.

Let us now prove the claim. First, remark that since $j \in \mathsf{cut}_{C\ell}(\mathsf{out}(\lambda))$, by definition of $\mathsf{factors}_{C\ell}(\mathsf{out}(\lambda))$ there exists an integer $j' < j$ such that $\mathsf{out}(\lambda)[j', j]$ is the longest prefix of $\mathsf{out}(\lambda)[j', |\mathsf{out}(\lambda)|]$ that has a primitive root of length smaller than $C\ell$. In particular, this implies that $\mathsf{out}(\lambda)[j', j]$ is p-periodic for some $p \leq C\ell$. Now let $(s, s') \subset [0, n]$ be an interval satisfying P. To prove the claim we suppose, towards building a contradiction, that $\mathsf{out}(\mathsf{pump}_{(s,s']}(\lambda)) = \mathsf{out}(\mathsf{pump}_{(s,s']}(\mu))$. Combining Property P with this equality implies that $\mathsf{out}(\lambda)[j_1, j_2]$ is q-periodic for some $1 \leq q \leq C\ell$. We show that this implies that $\mathsf{out}(\lambda)[j', j_2]$ is also q-periodic. To this end, we differentiate two cases:

1. If $j' \geq j_1$ then $[j', j_2] \subseteq [j_1, j_2]$, and we directly get that $\mathsf{out}(\lambda)[j', j_2]$ is q-periodic.
2. If $j' < j_1$ then $[j', j] \cap [j_1, j_2] = [j_1, j]$ (See Figure 5). Since $j - j_1 = 2C\ell \geq p + q$, the Fine-Wilf theorem yields that $\mathsf{out}(\lambda)[j', j_2]$ is $gcd(p, q)$-periodic (hence it is also q-periodic).

In both cases we showed that the interval $\mathsf{out}(\lambda)[j', j_2]$ is q-periodic. Since $q \leq C\ell$ there exists $j'' \in (j, j_2] = (j, j + C\ell]$ such that $j'' - j'$ is a multiple of q. Then $\mathsf{out}(\lambda)[j', j'']$ is a prefix of $\mathsf{out}(\lambda)[j', |\mathsf{out}(\lambda)|]$ longer than $\mathsf{out}(\lambda)[j', j]$ that has a primitive root of length at most q, thus smaller than $C\ell$. This contradicts the definition of j as the next cut after j'.   ◀

Three sufficient conditions.   We establish three conditions on intervals $(s, s') \subset [1, |\lambda|)$ whose conjunction implies Property P. Condition C_1 limits the growth of the delay from s to s' in order to ensure that the shift between the two copies of $\mathsf{out}(\lambda)[j_1, j_2]$ is appropriate.

C_1. The delay between λ and μ at position j_1 increases between s and s', but *not too much*:
$$\mathsf{diff}_{j_1,s}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) < \mathsf{diff}_{j_1,s'}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) < \mathsf{diff}_{j_1,s}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) + C\ell;$$

Conditions C_2 and C_3 are more technical. To guarantee property P we need to ensure that $\mathsf{out}(\lambda)[j_1, j_2] = \mathsf{out}(\mu)[j_1, j_2]$ still appears after (s, s') is pumped. To this end, we study how the production of $\mathsf{out}(\lambda)[j_1, j_2]$ and $\mathsf{out}(\mu)[j_1, j_2]$ evolves between s and s'. We begin by introducing some notation concerning the sequences of substitutions $\lambda = \sigma_1 \sigma_2 \ldots \sigma_n \in \mathcal{S}^*$ and $\mu = \tau_1 \tau_2 \ldots \tau_n \in \mathcal{S}^*$. We only define the notation explicitly for λ: it is defined analogously for μ by replacing each occurrence of λ with μ, and each occurrence of σ with τ.

We set $\sigma_0 = \sigma_\varepsilon$ and for every interval $[s, s'] \subseteq [0, n]$ we denote by $\sigma_{[s,s']}$ the composition $\sigma_s \circ \sigma_{s+1} \circ \ldots \circ \sigma_{s'}$. Using this notation, we remark that the words $\mathsf{out}(\lambda) = \sigma_{[0,s']}(\sigma_{(s',n]}(O))$ and $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda)) = \sigma_{[0,s']}(\sigma_{(s,n]}(O))$ are obtained by applying the same substitution $\sigma_{[0,s']}$ to either $\sigma_{(s',n]}(O)$ or $\sigma_{(s,n]}(O)$. Since Property P requires that the infix $\mathsf{out}(\lambda)[j_1, j_2]$ still appears after pumping (s, s'), we are specifically interested in the parts of $\sigma_{(s,n]}(O)$ and $\sigma_{(s',n]}(O)$ that are responsible for the creation of $\mathsf{out}(\lambda)[j_1, j_2]$: For all $s \in [1, n]$ we define a decomposition $\alpha_\lambda(s)\beta_\lambda(s)\gamma_\lambda(s)$ of $\sigma_{(s,n]}(O)$ by setting $\beta_\lambda(s)$ as the *minimal* infix such that

$$\sigma_{[0,s]}(\alpha_\lambda(s)) = \mathsf{out}(\lambda)[1, j_1'], \quad \sigma_{[0,s]}(\beta_\lambda(s)) = \mathsf{out}(\lambda)[j_1', j_2'], \quad \sigma_{[0,s]}(\gamma_\lambda(s)) = \mathsf{out}(\lambda)(j_2', \lambda],$$

for some $j_1' \leq j_1$ and $j_2' \geq j_2$. In other words, $\beta_\lambda(s)$ is composed of the parts of $\mathsf{out}(\lambda)[j_1, j_2]$ appearing in $\sigma_{(s,n]}(O)$ together with the variables that will contain fragments of $\mathsf{out}(\lambda)[j_1, j_2]$ once $\sigma_{[0,s)}$ is applied. We now have all the tools to define the remaining two conditions:

C_2. The parts of $\sigma_{(s,n]}(O)$ and $\sigma_{(s',n]}(O)$ responsible for the creation of $\mathsf{out}(\lambda)[j_1, j_2]$ and $\mathsf{out}(\mu)[j_1, j_2]$ are identical: $\beta_\lambda(s) = \beta_\lambda(s')$ and $\beta_\mu(s) = \beta_\mu(s')$.

C_3. The sequence of variables containing fragments of $\mathsf{out}(\lambda)[1, j_1]$ and $\mathsf{out}(\mu)[1, j_1]$ are the same at s and s': $\eta_{\mathcal{X}}(\alpha_\lambda(s)) = \eta_{\mathcal{X}}(\alpha_\lambda(s'))$ and $\eta_{\mathcal{X}}(\alpha_\mu(s)) = \eta_{\mathcal{X}}(\alpha_\mu(s'))$, where $\eta_{\mathcal{X}}$ is the erasing morphism that preserves the symbols of $\mathcal{X}$ and erases the symbols of $\Sigma \cup \{\#\}$.

We show that the conjunction of the three conditions implies Property P.

$\triangleright$ Claim 16. Every interval $(s, s']$ that satisfies C_1, C_2 and C_3 also satisfies P.

Proof. Let us consider an interval $(s, s']$ that satisfies the three conditions. We first show that Condition C_2 directly implies that both $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))$ and $\mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))$ contain a copy of the word $\mathsf{out}(\lambda)[j_1, j_2] = \mathsf{out}(\mu)[j_1, j_2]$, and then we combine C_1 and C_3 to show that the overlap between these two copies has the required size.

We focus on the sequence of substitutions λ by comparing $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))$. Since $\beta_\lambda(s) = \beta_\lambda(s')$ by Condition C_2, there exist $j_1'' \leq j_1$ and $j_2'' \geq j_2$ satisfying

$$
\begin{aligned}
\mathsf{out}(\lambda) &= (\sigma_{[0,s']} \circ \sigma_{(s',n]})(O) \\
&= \sigma_{[0,s']}(\alpha_\lambda(s') \cdot \beta_\lambda(s') \cdot \gamma_\lambda(s')) \\
&= \sigma_{[0,s']}(\alpha_\lambda(s')) \cdot \mathsf{out}(\lambda)[j_1'', j_2''] \cdot \sigma_{[0,s']}(\gamma_\lambda(s')); \\
\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda)) &= (\sigma_{[0,s']} \circ \sigma_{(s,n]})(O) \\
&= \sigma_{[0,s']}(\alpha_\lambda(s) \cdot \beta_\lambda(s) \cdot \gamma_\lambda(s)) \\
&= \sigma_{[0,s']}(\alpha_\lambda(s)) \cdot \sigma_{[0,s']}(\beta_\lambda(s')) \cdot \sigma_{[0,s']}(\gamma_\lambda(s)) \\
&= \sigma_{[0,s']}(\alpha_\lambda(s)) \cdot \mathsf{out}(\lambda)[j_1'', j_2''] \cdot \sigma_{[0,s']}(\gamma_\lambda(s)).
\end{aligned}
$$

As an immediate consequence, we have that

$$\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))[j_1''+x, j_2''+x] = \mathsf{out}(\lambda)[j_1'', j_2''], \text{ with } x = |\sigma_{[0,s']}(\alpha_\lambda(s))| - |\sigma_{[0,s']}(\alpha_\lambda(s'))|.$$

Since $j_1'' \leq j_1$ and $j_2'' \geq j_2$, this implies that, as required, $\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))$ contains a copy of $\mathsf{out}(\lambda)[j_1, j_2]$. Let us now focus on the value of the shift x. First, remark that by definition of the function α_λ, the letters of Σ that appear in $\alpha_\lambda(s)$, respectively $\alpha_\lambda(s')$, are exactly the letters of $\mathsf{out}(\lambda)$ whose position is smaller than j_1 and origin is greater than s, respectively s', hence:

$$
\begin{aligned}
x &= |\sigma_{[0,s']}(\alpha_\lambda(s))| - |\sigma_{[0,s']}(\alpha_\lambda(s'))| \\
&= |\eta_\Sigma(\alpha_\lambda(s))| + |\sigma_{[0,s']}(\eta_\mathcal{X}(\alpha_\lambda(s)))| - |\eta_\Sigma(\alpha_\lambda(s'))| - |\sigma_{[0,s']}(\eta_\mathcal{X}(\alpha_\lambda(s')))| \\
&= (|\mathsf{out}(\lambda)| - \mathsf{weight}_{j_1,s}(\tilde{\mathsf{out}}(\lambda))) + |\sigma_{[0,s']}(\eta_\mathcal{X}(\alpha_\lambda(s)))| \\
&\quad -(|\mathsf{out}(\lambda)| - \mathsf{weight}_{j_1,s'}(\tilde{\mathsf{out}}(\lambda))) - |\sigma_{[0,s']}(\eta_\mathcal{X}(\alpha_\lambda(s')))|
\end{aligned}
$$

Moreover, Condition C_3 implies that the second and fourth terms cancel each other. To summarize, we showed that

$$
\mathsf{out}(\mathsf{pump}_{(s,s')}(\lambda))[j_1 + x, j_2 + x] = \mathsf{out}(\lambda)[j_1, j_2],
$$

with $x = \mathsf{weight}_{j_1,s'}(\tilde{\mathsf{out}}(\lambda)) - \mathsf{weight}_{j_1,s}(\tilde{\mathsf{out}}(\lambda))$. Since Conditions C_2 and C_3 treat λ and μ identically, we get similarly that

$$
\mathsf{out}(\mathsf{pump}_{(s,s')}(\mu))[j_1 + y, j_2 + y] = \mathsf{out}(\mu)[j_1, j_2],
$$

with $y = \mathsf{weight}_{j_1,s'}(\tilde{\mathsf{out}}(\mu)) - \mathsf{weight}_{j_1,s}(\tilde{\mathsf{out}}(\mu))$. To prove that $(s, s']$ satisfies Property P, it only remains to apply C_1:

$$
\begin{aligned}
|x - y| &= \big|\mathsf{weight}_{j_1,s'}(\tilde{\mathsf{out}}(\lambda)) - \mathsf{weight}_{j_1,s}(\tilde{\mathsf{out}}(\lambda)) \\
&\quad -\big(\mathsf{weight}_{j_1,s'}(\tilde{\mathsf{out}}(\mu)) - \mathsf{weight}_{j_1,s}(\tilde{\mathsf{out}}(\mu))\big)\big| \\
&= |\mathsf{diff}_{j_1,s}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) - \mathsf{diff}_{j_1,s'}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))| \in [1, C\ell/2].
\end{aligned}
$$

This concludes the proof. ◀

To conclude, we show that if k and ℓ are large enough with respect to the parameters of $\mathcal{S}$, the fact that $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k$ implies through a combination of counting arguments the existence of C consecutive segments that satisfy the three conditions.

▷ **Claim 17.** Let M_1 be the size of the largest substitution occurring in $\mathcal{S}$, and let $M_2 = (|\mathcal{X}| + 1)^{|\mathcal{X}|+1} \cdot |\mathcal{X}|!$. If $\ell = M_1 M_2^2$ and $k > (6\ell + 3)M_1 M_2^2$, then there exist $0 \leq t_1 < t_2 < \ldots < t_C \leq t$ such that $(t_i, t_{i'})$ satisfies C_1, C_2 and C_3 for all $1 \leq i < i' \leq C$.

Proof. If we consider a variable s ranging from 0 to t, then the value of $\mathsf{diff}_{j,s}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ increases by at most M_1 whenever s increases by 1. Since $\mathsf{diff}_{j,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > C^2 k = C^2 \cdot (6\ell+3)M_1 M_2^2$, this implies that we can find $C' = C^2 \cdot (6\ell+3) \cdot M_2^2$ points $s_1 < s_2 < \ldots < s_{C'}$ between 0 and t along which the value of $\mathsf{diff}_{j,s}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$ increases: We set $s_0 = 0$, and for every $1 \leq i \leq C'$ we let s_i be the minimal integer greater than s_{i-1} that satisfies $\mathsf{diff}_{j,s_i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > \mathsf{diff}_{j,s_{i-1}}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$. This satisfies the following variation of condition C_1:

C_1'. $\mathsf{diff}_{j,s_i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) < \mathsf{diff}_{j,s_{i+1}}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq \mathsf{diff}_{j,s_i}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) + M_1$ for every $0 \leq i \leq C' - 1$.

Now, in order for C_2 to be satisfied, in particular we need intervals along which no output at the positions $[j_1, j_2]$ of $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$ is produced. Let $\tilde{\mathsf{out}}(\lambda) = \mathsf{out}(\lambda) \otimes o_\lambda$ and $\tilde{\mathsf{out}}(\mu) = \mathsf{out}(\mu) \otimes o_\mu$. The set $U = \{o_\lambda[i] \mid i \in [j_1, j_2]\} \cup \{o_\mu[i] \mid i \in [j_1, j_2]\}$ contains at most $6C\ell + 2$ elements, thus whenever we pick $6C\ell + 3$ disjoint intervals in $[0, n]$, at least

one of them contains no element from U. As $C' = C^2 \cdot (6\ell + 3) \cdot M_2^2 > C \cdot (6C\ell + 3) \cdot M_2^2$, there exist $0 \leq x < y \leq C'$ such that $y - x = C \cdot M_2^2$ and for all $i \in [j_1, j_2]$, $\mathsf{ori}_\lambda(i) \notin (s_x, s_y)$ and $\mathsf{ori}_\mu(i) \notin (s_x, s_y)$. This property greatly restricts the possible values of $\beta_\lambda(s_i)$ when i ranges from x to y: if we set $\beta_\lambda(s_y) = u_0 X_1 u_1 X_2 u_2 \ldots X_m u_m$ with $X_1, \ldots, X_m \in \mathcal{X}$ and $u_0, \ldots, u_m \in \Sigma^*$, then for every $x \leq i \leq y$, since $\sigma_{(i,n]}(0) = (\sigma_{(i,y]} \circ \sigma_{(y,n]})(O)$, there exists $H_1, H_2, \ldots, H_m \in \mathcal{X}^*$ such that $\beta_\lambda(s_i) = u_0 H_1 u_1 H_2 u_2 \ldots H_m u_m$. In other words $\beta_\lambda(s_i)$ is entirely determined by the tuple composed of the m sequences of variables $H_1, H_2, \ldots, H_m \in \mathcal{X}^*$ that occur in it, and if we set on top of that $H_0 = \eta_\mathcal{X}(\alpha_\lambda(s_i))$, we get that the pair $(\eta_\mathcal{X}(\alpha_\lambda(s_i)), \beta_\lambda(s_i))$ is entirely determined by a tuple of $m + 1$ sequences of $\mathcal{X}^*$. Similarly, the pair $(\eta_\mathcal{X}(\alpha_\mu(s_i)), \beta_\mu(s_i))$ is also determined by $m + 1$ sequences of $\mathcal{X}^*$. Finally, since we are dealing with copyless substitutions, at most $M_2 = (|\mathcal{X}| + 1)^{|\mathcal{X}|+1} \cdot |\mathcal{X}|!$ tuples of $m + 1$ sequences of $\mathcal{X}^*$ can occur: there are $|\mathcal{X}|!$ possible ways of concatenating all the variables in $\mathcal{X}$, each such sequence has $|\mathcal{X}| + 1$ distinct prefixes, and there are (at most) $(|\mathcal{X}| + 1)^{|\mathcal{X}|}$ manners to cut each prefix in a tuple of $m + 1$ (possibly empty) parts since $m \leq |X|$ (as m is the number of variables appearing in $\beta_\lambda(s_y)$).

Therefore, since $y - x = CM_2^2$, we can find C indices $t_1, t_2, \ldots, t_C \in \{s_x, s_{x+1}, \ldots, s_y\}$ such that for every $1 \leq i < i' \leq C$, C_2 and C_3 are satisfied: $(\beta_\lambda(t_i), \eta_\mathcal{X}(\alpha_\lambda(t_i))) = (\beta_\lambda(t_{i'}), \eta_\mathcal{X}(\alpha_\lambda(t_{i'})))$ and $(\beta_\mu(t_i), \eta_\mathcal{X}(\alpha_\lambda(t_i))) = (\beta_\mu(t_{i'}), \eta_\mathcal{X}(\alpha_\lambda(t_{i'})))$. Finally, C_1 is obtained by combining C_1' with the facts that $y - x = CM_2^2$ and $\ell = M_1 M_2^2$. ◄

D Regularity of the delay notion

The following sections are devoted to the missing proofs of Section 4.2.

D.1 Characterization lemma

▶ **Lemma 7.** *Let* $\dashv \in \Sigma$ *and* $\lambda, \mu \in \mathcal{S}^*$ *be sequences with* $\mathsf{out}(\lambda), \mathsf{out}(\mu) \in (\Sigma \setminus \{\dashv\})^* \dashv$ *and* $|\lambda| = |\mu|$. *Then* $\lambda \otimes \mu \notin \mathbb{D}_{k,\ell,\mathcal{S}}$ *iff there exists* $i \in (\mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu))) \cup \{0\}$ *such that* $\mathsf{max\text{-}diff}_i(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) \leq k$, *and one of the following holds:*

1. *Both* $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ *and* $j_2 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu))$ *exist, and either* $j_1 \neq j_2$ *or* $\mathsf{max\text{-}diff}_{j_1, j_2}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > k$;
2. $\mathsf{out}(\lambda)[i + b] \neq \mathsf{out}(\mu)[i + b]$ *for some* $b \in [0, \ell^2]$.

Proof. First assume that λ, μ satisfy Item 1 or 2 from the statement of the lemma. We show that $(\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}$. If $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$, then clearly $(\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}$. So assume that $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$. Then $\mathsf{cut}_\ell(\mathsf{out}(\lambda)) = \mathsf{cut}_\ell(\mathsf{out}(\lambda))$. Hence it is not possible that Item 2 is satisfied or Item 1 with $j_1 \neq j_2$. The remaining case is Item 1 with $j_1 = j_2 =: j$ and $\mathsf{max\text{-}diff}_j(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > k$. Since j is a cut, we obtain that $\mathsf{delay}_\ell(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > k$ and thus $(\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}$.

For the other direction, consider $(\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}$. Firstly, we assume that $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$ which entails that $\mathsf{cut}_\ell(\mathsf{out}(\lambda)) = \mathsf{cut}_\ell(\mathsf{out}(\mu))$. Since $(\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}$, there exists some $j \in \mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu))$ such that $\mathsf{max\text{-}diff}_j(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > k$. Let j be the smallest such cut position. If j is the first cut, let $i = 0$, otherwise let $i \in \mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu))$ such that $\mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda)) = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu)) = j$. Clearly, Item 1 is satisfied for this i.

Secondly, we assume that $\mathsf{out}(\lambda) \neq \mathsf{out}(\mu)$. Since $\mathsf{out}(\lambda), \mathsf{out}(\mu) \in (\Sigma - \dashv)^* \dashv$, there exists some m such that $\mathsf{out}(\lambda)[m] \neq \mathsf{out}(\mu)[m]$. Consider the smallest such m. Let $M = ((\mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap [1, m - 1]) \cap (\mathsf{cut}_\ell(\mathsf{out}(\mu)) \cap [1, m - 1])) \cup \{0\}$. Pick the position $i \in M$

closest to m (the maximal position in M). If $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $m \leq i + \ell^2$, we are done since Item 2 is satisfied.

Otherwise, we consider two cases. Either $\mathsf{max\text{-}diff}_i(\lambda, \mu) > k$ or $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $m > i + \ell^2$.

Assume that $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$. Note that $\mathsf{out}(\lambda)[1, i] = \mathsf{out}(\mu)[1, i]$ since $i < m$. Thus, $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$ and $\mathsf{max\text{-}diff}_0(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = 0$ implies that there exist $i', i'' \in M$ with $0 \leq i' < i'' \leq i$ and $i'' = \mathsf{next\text{-}cut}_\ell(i', \mathsf{out}(\lambda)) = \mathsf{next\text{-}cut}_\ell(i', \mathsf{out}(\mu))$ such that $\mathsf{max\text{-}diff}_{i'}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $\mathsf{max\text{-}diff}_{i'', i''}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$. Hence, Item 1 is satisfied (with i' for i and i'' for j_1 and j_2).

Assume that $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $m > i + \ell^2$. Recall that $i < m$ so $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$ and $j_2 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu))$ exist. If $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$, Item 1 is clearly satisfied. If $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$, we have to prove that $j_1 \neq j_2$ in order to satisfy Item 1. Towards a contradiction, assume $j_1 = j_2 =: j$. Since i is maximal in M, we obtain that $j \geq m$. Let $\mathsf{root}(\mathsf{out}(\lambda)[i, j]) = u$ and $\mathsf{root}(\mathsf{out}(\mu)[i, j]) = v$ and h_1, h_2 such that $\mathsf{out}(\lambda)[i, j] = u^{h_1}$ and $\mathsf{out}(\mu)[i, j] = v^{h_2}$. Because of the mismatch at $m \leq j$, we have $u^{h_1} \neq v^{h_2}$. Since further $|u^{h_1}| = |v^{h_2}|$, the difference between u^{h_1}, v^{h_2} must be on the first $|u||v|$ positions. This contradicts the fact that m is the first mismatch between $\mathsf{out}(\lambda)$ and $\mathsf{out}(\mu)$ and $j \geq m \geq i + \ell^2$ and $|u||v| \leq \ell^2$. ◄

D.2 Formal details for the regularity of the characterization

We provide here more details for the proof that $\mathbb{C}_{k,\ell,\mathcal{S}}^{\dashv}$ is regular (Lemma 24), and prove that it implies the regularity of $\mathbb{D}_{k,\ell,\mathcal{S}}$ in Lemma 21. First, we introduce some useful notions.

Let Γ be an arbitrary set and $\pi_\Sigma : (\Sigma \times \Gamma)^* \to \Sigma^*$ be the projection morphism which projects labels on Σ. We extend π_Σ naturally to substitution sequences ν in $\mathcal{S}_{\mathcal{X}, \Sigma \times \Gamma}^*$: π_Σ projects on Σ the labels from $\Sigma \times \Gamma$ occurring in the substitutions of ν. In particular, $\mathsf{out}(\pi_\Sigma(\nu)) = \pi_\Sigma(\mathsf{out}(\nu))$.

Let $n \geq 0$ and $u \in \Sigma^*$. For any n-tuple of positions $\overline{x}$ of u, we let $u \triangleright \overline{x}$ be the word in $(\Sigma \times 2^{\{1,\ldots,n\}})^*$ of length $|u|$ such that for all $p \in \{1, \ldots, |u|\}$, $(u \triangleright \overline{x})[p] = (u[p], \{i \mid p = x_i\})$. An n-marking is any word of the form $u \triangleright \overline{x}$. E.g., $abc \triangleright (1, 3, 1) = (a, \{1, 3\})(b, \varnothing)(c, \{2\})$. An n-marked substitution sequence is a sequence of substitutions ν in $(\mathcal{S}_{\mathcal{X}, \Sigma \times 2^{\{1,\ldots,n\}}})^*$ such that $\mathsf{out}(\nu)$ is an n-marking. Given $\lambda \in (\mathcal{S}_{\mathcal{X}, \Sigma})^*$ and an n-tuple of positions $\overline{x}$ of $\mathsf{out}(\lambda)$, we denote by $(\lambda \triangleright \overline{x})$ the unique n-marked substitution sequence such that $\mathsf{out}(\lambda \triangleright \overline{x}) = \mathsf{out}(\lambda) \triangleright \overline{x}$, $\pi_\Sigma(\lambda \triangleright \overline{x}) = \pi_\Sigma(\lambda)$ and each mark appears exactly once in $\lambda \triangleright \overline{x}$. For example, if $\mathcal{X} = \{X, O\}$ and $\lambda = ((X, O) \mapsto (Xc, Ob)).((X, O) \mapsto (\varepsilon, aOX))$, then $\lambda \triangleright (1, 3, 1) = ((X, O) \mapsto (X(c, \{2\}), O(b, \varnothing))).((X, O) \mapsto (\varepsilon, (a, \{1, 3\})OX))$.

▶ **Lemma 18.** *The following sets of 1- and 2-marked substitution sequences* $\lambda \triangleright i$ *and* $\lambda \triangleright (i, j)$ *where* $\lambda \in \mathcal{S}^*$ *and* $1 \leq i \leq j \leq |\mathsf{out}(\lambda)|$, *are regular, recognizable by NFA with both* $O(\ell|\Sigma|^{4|\mathcal{X}|\ell^3})$ *states:*

$$Cut = \{(\lambda \triangleright i) \mid i \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))\} \qquad NextCut = \{(\lambda \triangleright (i, j)) \mid \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda)) = j\}$$

Proof sketch. The proof is done in two steps:

1. we prove that $\{u \triangleright i \mid u \in \Sigma^*, i \in \mathsf{cut}_\ell(u)\}$ and $\{u \triangleright (i, j) \mid u \in \Sigma^*, \mathsf{next\text{-}cut}_\ell(i, u) = j\}$ are recognizable by NFA with $O(\ell.|\Sigma|^{\ell + \ell^3})$ states,
2. then we transfer the latter result to marked substitution sequences.

For step 1, we actually prove a somewhat stronger result: the set of words u such that all positions in $\mathsf{cut}_\ell(u)$ are marked by 1 is regular, recognizable by an NFA C. Intuitively, the

two main ingredients of the construction of C are: (1) checking that a word is periodic with period *at most* ℓ, and (2) checking that a word is *not* periodic for any period at most ℓ. The first property is needed to check that words in between two marked positions are periodic of period $\ell' \leq \ell$, and it is doable by a finite automaton as ℓ is fixed (using $O(\ell.|\Sigma|^\ell)$ states, where the factor ℓ comes from the fact that a period $\ell' \leq \ell$ must be non-deterministically guessed). The second property is needed to check that the factor in between marked positions are maximal, in the sense that they cannot be extended to longer periodic factors of u, for any period at most ℓ. This step is slightly more difficult but doable with $O(|\Sigma|^{\ell^3})$. The full proof is given in Appendix (Lemma 22).

For step 2, we prove that given a regular language P of n-markings of words over Σ, that we call regular predicate, the set of n-marked substitution sequences ν such that $\pi_\Sigma(\nu) \in \mathcal{S}^*$ and $\mathsf{out}(\nu) \in P$ is itself regular, recognizable by an NFA with $O(q^{2m})$ states, where $m = |\mathcal{X}|$ and q is the number of states of an NFA recognizing P. This result is actually an easy consequence of the fact that given a regular language L and some non-deterministic SST T, the inverse of L by T is regular as well. It seems to be a folklore result but we could not find a reference where this result is explicitly mentioned with the right complexity, we reprove it in Appendix (Lemma 23) for the sake of completeness. ◀

We now prove some useful regularity results about substitutions sequences marked with positions satisfying some particular properties of their maximal difference $\mathsf{max\text{-}diff}$.

▶ **Lemma 19.** *For all $\alpha \in \mathbb{N}$, the following sets of pairs of 2-marked substitution sequences $(\lambda \rhd i_1) \otimes (\mu \rhd i_2)$ where $\lambda, \mu \in \mathcal{S}^*$, $1 \leq i_1 \leq \mathsf{out}(\lambda)$ and $1 \leq i_2 \leq \mathsf{out}(\mu)$ are regular and recognizable by NFA with $O(\alpha.8^{|\mathcal{X}|})$ states:*

$$
\begin{aligned}
\mathsf{MD}_{\leq\alpha} &= \{(\lambda \rhd i_1) \otimes (\mu \rhd i_2) \mid \mathsf{max\text{-}diff}_{i_1,i_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq \alpha\} \\
\mathsf{MD}_{>\alpha} &= \{(\lambda \rhd i_1) \otimes (\mu \rhd i_2) \mid \mathsf{max\text{-}diff}_{i_1,i_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > \alpha\} \\
\mathsf{eMD}_{\leq\alpha} &= \mathsf{MD}_{\leq\alpha} \cap \{(\lambda \rhd i_1) \otimes (\mu \rhd i_2) \mid i_1 = i_2\} \\
\mathsf{dMD}_{\leq\alpha} &= \mathsf{MD}_{\leq\alpha} \cap \{(\lambda \rhd i_1) \otimes (\mu \rhd i_2) \mid i_1 \neq i_2\}
\end{aligned}
$$

Proof sketch. Let us sketch the proof for $\mathsf{MD}_{\leq\alpha}$. We construct a finite automaton A checking the conditions of the definition of $\mathsf{MD}_{\leq\alpha}$. For a substitution s in λ and an occurrence of a Σ-symbol in s, we say that this occurrence is *relevant* if in the final output, it will occur before i_1. A similar notion can be defined for μ. When reading $(\lambda \rhd i_1) \otimes (\mu \rhd i_2)$, the automaton must count, incrementally, the number of relevant occurrences of Σ-symbols in substitutions of $\lambda \rhd i_1$ minus the number of relevant occurrences of Σ-symbols in substitutions of $(\mu \rhd i_2)$. Note that the absolute value of this difference after having read t input pairs of substitutions is exactly $\mathsf{diff}_{i_1,i_2,t}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu))$. The automaton updates this difference incrementally while reading pairs of substitutions, and as soon as its absolute value exceeds α, the automaton rejects (or go to a sink accepting state if one wants to recognize $\mathsf{MD}_{>\alpha}$). It is correct since $\mathsf{max\text{-}diff}$ is a max of diff over all input positions t, by definition. There is a little technicality due to the fact it is unknown, when reading a pair of substitutions of $(\lambda \rhd i_1) \otimes (\mu \rhd i_2)$, which Σ-symbol occurrences are relevant and which are not. To overcome this, we first apply a rational relabelling into a more informative marking which also indicates which symbol occurrences are relevant or not, and obtain the result since rational relabelings preserve regularity under inverse. Note that if at the end of the computation, the difference is 0 iff $i_1 = i_2$, allowing to recognize the languages $\mathsf{dMD}_{\leq\alpha}$ and $\mathsf{eMD}_{\leq\alpha}$. The proof is detailed in Appendix D.4. ◀

Finally, the following result is useful to prove the regularity of the second property (Item 2) of the characterization of Lemma 7:

▶ **Lemma 20.** *The following set of pairs of 2-marked substitution sequences $(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangleright (i_2, j_2))$ where $\lambda, \mu \in \mathcal{S}^*$ and $1 \leq i_1, j_1 \leq |\mathsf{out}(\lambda)|$, $1 \leq i_2, j_2 \leq |\mathsf{out}(\mu)|$, is regular, recognizable by an NFA with $O((|\Sigma|\ell)^{8|\mathcal{X}|})$ states:*

$$\mathsf{Mism}_\ell = \{(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangleright (i_2, j_2)) \mid 0 \leq j_1 - i_1 = j_2 - i_2 \leq \ell^2, \mathsf{out}(\lambda)[j_1] \neq \mathsf{out}(\lambda)[j_2]\}$$

Proof sketch. As for Lemma 18, we first prove the regularity for the output language using the fact that the distance between j_1 and i_1 (and between j_2 and i_2) is bounded by a constant, and then transfer it to substitution sequences while preserving regularity. The full proof is given in Appendix D.5. ◀

Finally we prove the main lemma of this section: regularity of the delay notion.

▶ **Lemma 21.** *For every $k, \ell \in \mathbb{N}$ and any finite subset of substitutions $\mathcal{S}$ over a set of variables $\mathcal{X}$ and alphabet Λ, the set $\mathbb{D}_{k,\ell,\mathcal{S}}$ is regular, recognizable by a DFA with a number of states doubly exponential in ℓ^3 and in $|\mathcal{X}|$, and singly exponential in k.*

Proof sketch. It is done in two steps: (i) first, for any finite subset of substitutions $\mathcal{S}'$, we prove the regularity of the set $\mathbb{C}^\dashv_{k,\ell,\mathcal{S}'}$, and then (ii) we show how to use it to prove Lemma 21.

For step (i), the proof is based on the characterization of Lemma 7, which states sub-properties which can all be checked individually by the automata constructed in Lemmas 18–20. Taking a synchronized product of those automata yield the result. The technical details are in Appendix (Lemma 24).

For step (ii), observe that the characterization of Lemma 7,
(1) holds for pairs of substitution sequences whose output ends with some endmarker $\dashv$, and
(2) characterizes the pairs of such substitution sequences which are *not* in $\mathbb{D}_{k,\ell,\mathcal{S}}$.

The restriction of observation (1) can be removed as follows: first, we define $\mathcal{S} = \mathcal{S}' \cup \mathcal{S}_\dashv$, where $\mathcal{S}_\dashv$ is the set of substitutions of $\mathcal{S}$ where an endmarker has been added in the update of the output variable O. The statement (i) tells us that $\mathbb{C}^\dashv_{k,\ell,\mathcal{S}'}$ is regular. This yields regularity of the complement of $\mathbb{D}_{k,\ell,\mathcal{S}'}$, as noticed in the second observation), complement which is restricted to substitutions sequences whose outputs end with an endmarker. The latter set is then complemented into a DFA. Finally, we apply a morphism which removes the endmarker to the substitutions of substitution sequences. This preserves the regularity (and the determinism of the automaton), as well as the delay. The full details are given in Appendix D.6. ◀

D.3   Regularity of Cut and NextCut

▶ **Lemma 18.** *The following sets of 1- and 2-marked substitution sequences $\lambda \triangleright i$ and $\lambda \triangleright (i, j)$ where $\lambda \in \mathcal{S}^*$ and $1 \leq i \leq j \leq |\mathsf{out}(\lambda)|$, are regular, recognizable by NFA with both $O(\ell|\Sigma|^{4|\mathcal{X}|\ell^3})$ states:*

$$\mathsf{Cut} = \{(\lambda \triangleright i) \mid i \in \mathsf{cut}_\ell(\mathsf{out}(\lambda))\} \qquad \mathsf{NextCut} = \{(\lambda \triangleright (i, j)) \mid \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda)) = j\}$$

The proof of this lemma is done in two steps, as explained in the main body of the paper. Those two steps are respectively given by Lemma 22 and Lemma 23 below, whose combination immediately yields the desired result. In what follows, we call n-ary regular predicate any regular set of n-markings.

▶ **Lemma 22.** *The sets $\{u \triangleright i \mid u \in \Sigma^*, i \in \mathsf{cut}_\ell(u)\}$ and $\{u \triangleright (i, j) \mid u \in \Sigma^*, \mathsf{next\text{-}cut}_\ell(i, u) = j\}$ are unary and binary regular predicates respectively. They can be both recognized by NFA with $O(\ell.|\Sigma|^{\ell+\ell^3})$ states.*

Proof. For all $h \in \mathbb{N}$, let $P_h = \bigcup_{v \in \Sigma^h} v^*$ be the set of periodic words of period h. Note that P_h is recognizable by a deterministic finite automaton D_h with $O(|\Sigma|^h)$ states. Now, for all words $u \in \Sigma^*$ and set X of positions of u, let $u \triangleright X \in (\Sigma \times \{\varnothing, \{1\}\})^*$ be such that for all positions i, we have $(u \triangleright X)[i] = (u[i], \{1\})$ if $i \in X$, otherwise it is $(u[i], \varnothing)$. We prove that the language

$$L_{cut} = \{u \triangleright \mathsf{cut}_\ell(u) \mid u \in \Sigma^*\}$$

is regular. To that end, consider a word u and an arbitrary subset $X = \{i_1 < i_2 < \cdots < i_n\}$ of positions of u. Let $i_0 = 0$. It defines a factorization $u = u(i_0{:}i_1]u(i_1{:}i_2] \ldots u(i_n{:}|u|]$ (note that the last factor is empty if $i_n = |u|$). We let $u_k = u(i_k{:}i_{k+1}]$. An automaton C recognizing L_{cut} needs to check several properties in parallel (via a synchronized product construction):

(i) the maximal element of X is $|u|$,

(ii) for all $0 \leq k < n$, $u_k \in P_{\ell'}$ for some $1 \leq \ell' \leq \ell$. To do so , C successively for each k, non-deterministically guesses a period ℓ' and checks that u_k is accepted by $D_{\ell'}$. In total, this needs $O(\ell.|\Sigma|^\ell)$ states.

(iii) for all $0 \leq k < n$, the automaton C must check that there is no factor of u which starts at position $i_k + 1$, belongs to some $P_{\ell'}$, and is strictly longer than u_k. In other words, C must check that for all periods $1 \leq \ell' \leq \ell$, if $\alpha_{\ell'} > i_{k+1}$ is the smallest integer such that $\alpha_{\ell'} - i_k \bmod \ell' = 0$, then either $\alpha_{\ell'} > |u|$ or $u(i_k{:}\alpha_{\ell'}] \notin P_{\ell'}$. To do so, for all $0 \leq k < n$ and all $1 \leq \ell' \leq \ell$, C simulates a copy of the automaton $\overline{D_{\ell'}}$ (complement of $D_{\ell'}$) running on the factor $u(i_k{:}\alpha_{\ell'}]$ (C also makes sure that $\alpha_{\ell'}$ is indeed the smallest integer such that $\alpha_{\ell'} - i_k \bmod \ell' = 0$ and $\alpha_{\ell'} \leq |u|$, by slightly modifying $\overline{D_{\ell'}}$). The different copies of the (modified) automata $\overline{D_{\ell'}}$ are run in parallel via a synchronized product, and must all accept. There is at most ℓ^2 copies running in parallel, and each copy has $O(|\Sigma|^\ell)$ states, so, C needs $O(|\Sigma|^{\ell^3})$ states for this verification.

Overall, to check the three properties above in parallel, C needs $O(\ell.|\Sigma|^{\ell+\ell^3})$ states.

It is not difficult, from an automaton A recognizing L_{cut}, to define two automata recognizing the languages of the lemma statement. For the first language, it is a kind of projection of A: its transition on labels $(\sigma, \{1\})$ are duplicated as transitions on labels $(\sigma, \varnothing)$ and it is additionally checked that there exists exactly one position marked $\{1\}$ on the input.

An automaton for the second language can be constructed using similar ideas. ◄

▶ **Lemma 23.** *Let $n \geq 0$ be fixed. Given a regular n-ary predicate P recognized by an NFA with q states, the set of n-marked substitution sequences λ over $\mathcal{S}_n$ such that $\mathsf{out}(\lambda) \in P$ is regular, recognized by an NFA with $O(q^{2m})$ states, where $m = |\mathcal{X}|$.*

Proof. Suppose that T uses m variables and α states, and L is defined by a NFA A_L with n_q states. The main idea is to construct an NFA B which simulates T and in parallel non-deterministically computes for each of its variables, a pair of states (p, q), with the following invariant: for each variable X and at each moment there is a run of A_L from p to q on the content of X. So, states of B are of the form $Q_T \times (Var_T \to Q_L \times Q_L)$ where Q_T are the states of T, Var_T its variables, and Q_L the states of A_L. In a state (p, f), reading a symbol a, the automaton B goes to any state of the form (p', f') such that T goes from p to p' on a, producing a substitution σ, and f' satisfies the following: for all $X \in Var_T$, if $\sigma(X)$ is of the form $u_1 X_1 u_2 X_2 \ldots u_n X_n u_{n+1}$ where the u_i do not contain variables, then there exists states $p_1, q_1, p_2, q_2, \ldots, p_{n+1}, q_{n+1} \in Q_L$ such that $f'(X) = (p_1, q_{n+1})$, $f(X_i) = (q_i, p_{i+1})$ and $p_i \xrightarrow{u_i}_{A_L} q_i$ for all $i = 1 \ldots n$. The automaton accepts if at the end of the computation, the function f computed satisfies that $f(O) \in I_L \times F_L$, where O is the output variable of

T, I_L the set of initial states of A_L and F_L its set of accepting states. So, the automaton B has $|Q_L|^{2|Var_T|}$ states.

To get the statement of the lemma, it suffices to apply the latter result to a single initial-accepting state SST T whose input alphabet is $\mathcal{S}_n$, with set of variables $\mathcal{X}$, and which does nothing but apply the substitution it reads as input. For L, it suffices to take the predicate P.   ◄

D.4   Regularity results for max-diff

▶ **Lemma 19.** *For all $\alpha \in \mathbb{N}$, the following sets of pairs of 2-marked substitution sequences $(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2)$ where $\lambda, \mu \in \mathcal{S}^*$, $1 \leq i_1 \leq \mathsf{out}(\lambda)$ and $1 \leq i_2 \leq \mathsf{out}(\mu)$ are regular and recognizable by NFA with $O(\alpha.8^{|\mathcal{X}|})$ states:*

$$
\begin{aligned}
\mathsf{MD}_{\leq \alpha} &= \{(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2) \mid \textit{max-diff}_{i_1,i_2}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) \leq \alpha\} \\
\mathsf{MD}_{> \alpha} &= \{(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2) \mid \textit{max-diff}_{i_1,i_2}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) > \alpha\} \\
\mathsf{eMD}_{\leq \alpha} &= \mathsf{MD}_{\leq \alpha} \cap \{(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2) \mid i_1 = i_2\} \\
\mathsf{dMD}_{\leq \alpha} &= \mathsf{MD}_{\leq \alpha} \cap \{(\lambda \triangleright i_1) \otimes (\mu \triangleright i_2) \mid i_1 \neq i_2\}
\end{aligned}
$$

Proof (formal details). Before proving the lemma, we introduce some useful notions. Given a word $u \in \Sigma^*$, an n-*labelling* of u is a word in $v \in (\Sigma \times 2^{\{1,\ldots,n\}})^*$ such that $\pi_1(v) = u$ and for all positions $p_1 \leq p_2$ of u, $\pi_2(v[p_1]) \supseteq \pi_2(v[p_2])$. For example, the word $v = (a, \{1,2,3\})(b, \{1,2,3\})(c, \{1,3\})(d, \varnothing)$ is a 3-labelling of $u = abcd$. Observe that there is a bijection between the n-labelings of u and the n-markings of u: the n-marking of u associated with an n-labelling of u is obtained by keeping only the right-most occurrence of $i \in \{1,\ldots,n\}$, and conversely, any occurrence of i in some n-marking is propagated leftward to all sets. This bijection from n-labelings to n-markings, denoted mk, is a rational function, defined by a finite transducer of constant size if n is fixed. For example, we have $\mathsf{mk}(v) = (a, \varnothing)(b, \{2\})(c, \{1,3\})(d, \varnothing)$.

Similarly, an n-*labelled substitution sequence* τ is a substitution sequence such that $\mathsf{out}(\tau)$ is an n-labelling (of its projection onto Σ). Any n-labelled substitution sequence τ can be mapped injectively to an n-marked substitution sequence we denote $\mathsf{mks}(\tau)$ such that $\mathsf{out}(\mathsf{mks}(\tau)) = \mathsf{mk}(\mathsf{out}(\tau))$. The function mks is defined by a finite transducer T with $O(2^{|\mathcal{X}|})$ states (if n is fixed) we now describe. Suppose T processes an n-labelled substitution sequence λ and reads a substitution s of τ. It must replace any set $I \subseteq \{1,\ldots,n\}$ occurring in some update of some variable X, by $J \subseteq I$ such that J contains $i \in I$ if this is the last occurrence of i in $\mathsf{out}(\tau)$. To that end, the transducer guesses $J \subseteq I$ for all $j \in J$, verifies that j will not occur again in $\mathsf{out}(\tau)$. The latter requires T to use non-determinism to guess the order in which the content of the variables in the substitutions occur in $\mathsf{out}(\tau)$. This requires to keep $O(2^{|\mathcal{X}|})$ states.

Now, observe that any 1-labelled substitution sequence τ uniquely defines a substitution sequence λ and some position i in $\mathsf{out}(\lambda)$ such that $\mathsf{mks}(\tau) = \lambda \triangleright i$. Let $G_\alpha = \{\tau_1 \otimes \tau_2 \mid \exists \lambda.\mathsf{mks}(\tau_1) = \lambda \triangleright i_1, \mathsf{mks}(\tau_2) = \lambda \triangleright i_2, \textsf{max-diff}_{i_1,i_2}(\mathsf{o\tilde{u}t}(\lambda), \mathsf{o\tilde{u}t}(\mu)) \leq \alpha\}$. We have

$$\mathsf{MD}_{\leq \alpha} = \{\mathsf{mks}(\tau_1) \otimes \mathsf{mks}(\tau_2) \mid \tau_1 \otimes \tau_2 \in G_\alpha\}$$

Therefore, if G_α is regular, definable by an NFA with q states, $\mathsf{MD}_{\leq \alpha}$ is regular and definable by an NFA with $O(q.2^{|\mathcal{X}|})$ states, because mks is a letter-to-letter rational function defined by a finite transducer with $O(2^{|\mathcal{X}|})$ states.

We show that G_α is regular, definable by an NFA A with $O(\alpha.2^{|\mathcal{X}|})$ states. The automaton A is obtained via a synchronized product of three finite automata which respectively check that:

1. for each pair of substitutions (σ_1, σ_2) read as input, their Σ-projection are equal, i.e., $\pi_\Sigma(\sigma_1) = \pi_\Sigma(\sigma_2)$. This ensures that $\mathsf{mks}(\tau_1) = \lambda \triangleright i_1$ and $\mathsf{mks}(\tau_2) = \lambda \triangleright i_2$ for the same λ, and some i_1, i_2. Only one state is needed.

2. it reads pairs of 1-labelled substitution sequences as inputs. The set L_1 of words v such that v is a 1-labelling of some word $u \in \Sigma^*$ is easily seen to be regular, definable by some NFA with 2 states. We can then apply Lemma 23 (for $n = 0$) and $P = L_1$, and get that using $O(4^{|\mathcal{X}|})$ states, A can verify that its input consists in pairs of 1-labelled substitution sequences.

3. $\mathsf{max\text{-}diff}_{i_1,i_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq \alpha$: to do so, it needs $2.\alpha + 2$ states in the $D = \{-\alpha, -\alpha + 1, \ldots, \alpha - 1, \alpha, \infty\}$ to count the differences of the weights in the interval $[-\alpha, \alpha] \cap \mathbb{N}$ (and state ∞ if the difference exits that interval). Let (s_1, s_2) be a pair of substitutions and $d \in D$. The state ∞ is a non-accepting sink state while the other states from D are accepting. State 0 is the unique initial state. Suppose that $d \neq \infty$. Let n_1 be the number of symbols from $\Sigma \times \{\{1\}\}$ occurring in the right hand-sides of s_1, and n_2 be defined similarly but for s_2. When reading (s_1, s_2) from state d, A goes to ∞ if $d + n_1 - n_2 < -\alpha$ or $d + n_1 - n_2 > \alpha$, otherwise it goes to state $d + n_1 - n_2$.

So, overall, A has $O(\alpha.8^{|\mathcal{X}|})$ states.

To recognize $\mathsf{eMD}_{\leq \alpha}$, we just modify the third automaton: all states of D are rejecting but state 0. Indeed, $i_1 = i_2$ iff $\mathsf{max\text{-}diff}_{i_1,i_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) = 0$. So, if the automaton does not reach the sink state ∞ and ends up in state 0, we are guaranteed that (i) the maximal absolute difference is lesser than α and (ii) $i_1 = i_2$. For $\mathsf{dMD}_{\leq \alpha}$, it suffices to set states $0, \infty$ to be rejecting and all states in $D \backslash \{0, \infty\}$ to be accepting. For $\mathsf{eMD}_{> \alpha}$, only ∞ is set to be accepting. ◀

D.5 Regularity of mismatches

▶ **Lemma 20.** *The following set of pairs of 2-marked substitution sequences $(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangleright (i_2, j_2))$ where $\lambda, \mu \in \mathcal{S}^*$ and $1 \leq i_1, j_1 \leq |\mathsf{out}(\lambda)|$, $1 \leq i_2, j_2 \leq |\mathsf{out}(\mu)|$, is regular, recognizable by an NFA with $O((|\Sigma|\ell)^{8|\mathcal{X}|})$ states:*

$$\mathsf{Mism}_\ell = \{(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangleright (i_2, j_2)) \mid 0 \leq j_1 - i_1 = j_2 - i_2 \leq \ell^2, \mathsf{out}(\lambda)[j_1] \neq \mathsf{out}(\lambda)[j_2]\}$$

Proof. We prove that the language

$$H = \{(u \triangleright (i_1, i_1 + d)) \otimes (v \triangleright (i_2, i_2 + d)) \mid \lambda, \mu \in \mathcal{S}^*, 0 \leq d \leq \ell^2, u[i_1 + d] \neq v[i_2 + d]\}$$

is recognizable by an NFA with $O(|\Sigma|^2 \ell^4)$ states using the fact that ℓ is a given constant. Then, it suffices to apply Lemma 23 to conclude.

For all $\sigma \in \Sigma$ and $0 \leq d \leq \ell^2$, let $P_{\sigma,d}$ be the following binary predicate:

$$P_{\sigma,d} = \{u \triangleright (i, i + d) \mid u \in \Sigma^*, 1 \leq i \leq i + d \leq |u|, \ u[i + d] = \sigma\}$$

This predicate is easily seen to be regular, recognized by a DFA $A_{\sigma,d}$ with $O(d)$ states. The operator $\otimes$ is naturally extended to languages, and we get:

$$H = \bigcup_{\sigma_1, \sigma_2 \in \Sigma, \sigma_1 \neq \sigma_2} \bigcup_{0 \leq d \leq \ell^2} P_{\sigma_1,d} \otimes P_{\sigma_2,d}$$

Moreover, for all σ_1, σ_2, d, one can construct a DFA recognizing $P_{\sigma_1,d} \otimes P_{\sigma_2,d}$ with $O(d^2)$ states, and take the union of all these DFA to obtain an NFA recognizing H with $O(|\Sigma|^2 \ell^4)$ states. ◀

D.6 Regularity of the delay notion

In order to prove Lemma 21, we need one more auxiliary lemma.

▶ **Lemma 24.** *Let $\mathcal{S}$ be a finite set of substitutions over a finite set of variables $\mathcal{X}$ and an alphabet Σ. Let $\dashv \in \Sigma$ and let $T_\dashv = \{\lambda \in \mathcal{S}^* \mid \mathsf{out}(\lambda) \in (\Sigma \backslash \{\dashv\})^* \dashv\}$. Let $k, \ell \in \mathbb{N}$. Then, the set*

$$\mathbb{C}^\dashv_{k,\ell,\mathcal{S}} = \{\lambda \otimes \mu \mid \lambda, \mu \in T_\dashv \text{ and } (\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}}\}$$

is regular, recognized by an NFA with a number of states exponential in ℓ^3 and in $|\mathcal{X}|$, and polynomial in k.

Proof. First, for $x \in \{1, \ldots, 5\}$, we define the set C_x of convolutions of marked substitution sequences $(\lambda \triangleright (i, j_1)) \otimes (\mu \triangleright (i, j_2))$ satisfying the xth property below:

1. $i \in (\mathsf{cut}_\ell(\mathsf{out}(\lambda)) \cap \mathsf{cut}_\ell(\mathsf{out}(\mu)))$ and $\mathsf{max\text{-}diff}_i(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$
2. $j_1 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\lambda))$, $j_2 = \mathsf{next\text{-}cut}_\ell(i, \mathsf{out}(\mu))$
3. $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$
4. $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $j_1 = j_2$
5. $\mathsf{out}(\lambda)[j_1] \neq \mathsf{out}(\mu)[j_2]$, $j_1 - i \leq \ell^2$ and $j_2 - i \leq \ell^2$.

Similarly, we define languages corresponding to the various properties of Lemma 7 but in the case $i = 0$. So, for $x \in \{1, \ldots, 4\}$, we define the set Z_x of convolutions of marked substitution sequences $(\lambda \triangleright j_1) \otimes (\mu \triangleright j_2)$ satisfying the xth property below:

1. $j_1 = \mathsf{next\text{-}cut}_\ell(0, \mathsf{out}(\lambda))$, $j_2 = \mathsf{next\text{-}cut}_\ell(0, \mathsf{out}(\mu))$
2. $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) > k$
3. $\mathsf{max\text{-}diff}_{j_1, j_2}(\tilde{\mathsf{out}}(\lambda), \tilde{\mathsf{out}}(\mu)) \leq k$ and $j_1 = j_2$
4. $\mathsf{out}(\lambda)[j_1] \neq \mathsf{out}(\mu)[j_2]$, $j_1 \leq \ell^2$ and $j_2 \leq \ell^2$.

Let π_Σ be the projection of marked substitutions on their Σ component. So, π_Σ substitutes any label (α, M) occurring in the right-hand side of a substitution, where $\alpha \in \Sigma$ and M is a marking, by just α. According to Lemma 7, we have:

$$\mathbb{C}^\dashv_{k,\ell,\mathcal{S}} = \pi_\Sigma(T_\dashv \cap (C_1 \cap C_2 \cap C_3 \cup C_1 \cap C_2 \cap C_4 \cup C_1 \cap C_5 \cup Z_1 \cap Z_2 \cup Z_1 \cap Z_3 \cup Z_4))$$

Since π_Σ preserves regular languages (in linear time for regular sets given as NFA), and regular languages are closed under Boolean operation, it suffices to prove that each set in the above expression is regular. For C_1, C_2, C_3, C_4, it is an immediate consequence of the various results we have previously shown: Lemma 18, Lemma 19 and Lemma 20. For example, for C_1, it suffices to simulate twice the automaton recognizing the language *Cut* of Lemma 18, on inputs of the form $(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangle (i_2, j_2))$, and intersect it with the automaton recognizing $\mathsf{MD}^=_{\leq k}$ of Lemma 19. More precisely, using a product construction:

- the automaton for *Cut* is simulated both twice on $(\lambda \triangleright (i_1, j_1)) \otimes (\mu \triangle (i_2, j_2))$: the first simulation ignores all components but the one corresponding to λ and the marking of i_1, while the other ignores all components but the one corresponding to μ and the marking of i_2.
- the automaton for $\mathsf{MD}^=_{\leq k}$ is simulated by ignoring the markings corresponding to j_1 and j_2.

This yields an automaton for C_1 with $O(k8^{|\mathcal{X}|}\ell^2|\Sigma|^{4|\mathcal{X}|\ell^3})$ states.

The regularity of $Z_1, \ldots, Z_4$ can be shown using similar ideas. Observe they are particular cases of $C_2, \ldots, C_5$ for $i = 0$. Since i is not a position of the input, it cannot be marked. However, modulo adding a starting symbol, say, $\vdash$, occurring uniquely, the regularity of $C_2, \ldots, C_5$ and this starting symbol removed while preserving regularity.

Based on the number of states of each automata obtained in Lemma 18, Lemma 19 and Lemma 20, a careful analysis yields an NFA for $\mathcal{C}_{k,\ell,\mathcal{S}}^{\dashv}$ with a number of states which is:

- exponential in ℓ^3
- exponential in $|\mathcal{X}|$
- polynomial in k.

◄

▶ **Lemma 21.** *For every $k, \ell \in \mathbb{N}$ and any finite subset of substitutions $\mathcal{S}$ over a set of variables $\mathcal{X}$ and alphabet Λ, the set $\mathbb{D}_{k,\ell,\mathcal{S}}$ is regular, recognizable by a DFA with a number of states doubly exponential in ℓ^3 and in $|\mathcal{X}|$, and singly exponential in k.*

Proof (formal details). Let $k, \ell \in \mathbb{N}, \mathcal{X}, \Lambda, \mathcal{S}$ as in the statement of the lemma. Let $\dashv \notin \Lambda$ and $\Sigma = \Lambda \cup \{\dashv\}$. Let $\Phi : \mathcal{S}_{\mathcal{X},\Lambda} \to \mathcal{S}_{\mathcal{X},\Sigma}$ defined for all $s \in \mathcal{S}_{\mathcal{X},\Lambda}$ by $\Phi(s)(X) = s(X)$ if $X \neq O$ and $\Phi(s)(O) = s(X).\dashv$. Note that Φ is a bijection. We let $\mathcal{S}' = \mathcal{S} \cup \Phi(\mathcal{S})$. Note that $|\mathcal{S}'| = 2|\mathcal{S}|$. We let $\Psi : \mathcal{S}' \to \mathcal{S}$ defined by $\Psi(s) = s$ if $s \in \mathcal{S}$ and $\Psi(s) = \Phi^{-1}(s)$ if $s \in \Phi(\mathcal{S})$. So, Ψ removes the occurrence of the endmarker, if it exists.

Lemma 24 precisely proves that the following set is regular:

$$\mathbb{C}_{k,\ell,\mathcal{S}'}^{\dashv} = \{\lambda \otimes \mu \mid \lambda, \mu \in T_\dashv \text{ and } (\lambda, \mu) \notin \mathbb{D}_{k,\ell,\mathcal{S}'}\}$$

From that, we obtain that the following set:

$$\mathbb{D}_{k,\ell,\mathcal{S}'}^{\dashv} = \{\lambda \otimes \mu \mid \lambda, \mu \in T_\dashv \text{ and } (\lambda, \mu) \in \mathbb{D}_{k,\ell,\mathcal{S}'}\}$$

is regular. Indeed, we have

$$\mathbb{D}_{k,\ell,\mathcal{S}'}^{\dashv} = ((\mathcal{S}' \times \mathcal{S}')^* \setminus \mathbb{C}_{k,\ell,\mathcal{S}'}^{\dashv}) \cap (T_\dashv \otimes T_\dashv)$$

where here $\otimes$ is extended to languages. Since $\mathbb{C}_{k,\ell,\mathcal{S}'}^{\dashv}$ is recognizable by an NFA with a number of states which is exponential in ℓ^3 and $|\mathcal{X}|$, and polynomial in k, $((\mathcal{S}' \times \mathcal{S}')^* \setminus \mathbb{C}_{k,\ell,\mathcal{S}'}^{\dashv})$ can be recognized by a DFA D_1 with a number of states doubly exponential in both ℓ^3 and $|\mathcal{X}|$, and exponential in k. The set $T_\dashv$ is recognizable by a DFA D_2 with $4^{|\mathcal{X}|}$ states by an immediate application of Lemma 23 to 2-states automaton accepting words with a unique occurrence of $\dashv$ at the end. From the DFA D_2, it is easy to get a DFA D_3 recognizing $T_\dashv \otimes T_\dashv$, with $4^{2|\mathcal{X}|}$ states. Taking the product of D_1 and D_3, one obtains a DFA recognizing $\mathbb{D}_{k,\ell,\mathcal{S}'}^{\dashv}$ with a number of states doubly exponential in ℓ^3 and in $|\mathcal{X}|$, and exponential in k.

Let us now explain how to obtain a DFA for $\mathbb{D}_{k,\ell,\mathcal{S}}$. The mapping Ψ which removes occurrences of endmarkers in substitutions is first extended two pairs of substitutions by $\Psi(s, s') = (\Psi(s), \Psi(s'))$ and then morphically to sequences of the form $\lambda \otimes \mu$. Then, we claim that

$$\mathbb{D}_{k,\ell,\mathcal{S}} = \Psi(\mathbb{D}_{k,\ell,\mathcal{S}'}^{\dashv}) \qquad (\star)$$

Indeed, let $\lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}$. Let $\lambda = \lambda_1 s$ for $\lambda_1 \in \mathcal{S}^*$ and $s \in \mathcal{S}$. Similarly, decompose μ into $\mu = \mu_1 t$. Then, let $\lambda' = \lambda_1.\Phi(s)$ and $\mu' = \mu_1.\Phi(t)$. We have $\Psi(\lambda' \otimes \mu') = \lambda \otimes \mu$.

Moreover, $\lambda', \mu' \in T_\dashv$, $\mathsf{out}(\lambda') = \mathsf{out}(\lambda).\dashv = \mathsf{out}(\mu).\dashv = \mathsf{out}(\mu')$, because $\mathsf{out}(\lambda) = \mathsf{out}(\mu)$ since $\lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}$. It can be seen that $d_\ell(\lambda', \mu') = d_\ell(\lambda, \mu)$ (adding an endmarker does not modify the delay). So, $\lambda' \otimes \mu' \in \mathbb{D}_{k,\ell,\mathcal{S}'}^\dashv$. We have shown inclusion $\subseteq$ in equality $(\star)$. The other inclusion is shown similarly: removing the endmarker from two equal outputs does not modify the delay.

To conclude, the mapping Ψ is an alphabetic morphism, so, by the regularity of $\mathbb{D}_{k,\ell,\mathcal{S}'}^\dashv$, we also get the regularity of $\mathbb{D}_{k,\ell,\mathcal{S}}$, and its recognizability by an DFA of the same size as the DFA recognizing $\mathbb{D}_{k,\ell,\mathcal{S}'}^\dashv$: doubly exponential in ℓ^3 and in $|\mathcal{X}|$, and exponential in k.   ◄

E   Inclusion and equivalence up to fixed delay

▶ **Theorem 8.** *Given integers k, ℓ, the (k, ℓ)-inclusion problem for non-deterministic SSTs is decidable. It is* PSPACE-*complete if k, ℓ are constants and the number of variables $|\mathcal{X}|$ is a constant. The same results hold for the (k, ℓ)-equivalence problem.*

Proof. To begin with, we introduce some notation. For a non-deterministic SST T over input alphabet Σ, we denote by $L(T)$ its language, defined as the set of words of the form $u \otimes \tau(\rho)$, where $u \in \Sigma^*$, ρ is an accepting run of T over u, and $\tau(\rho)$ is the sequence of substitutions occurring on ρ. Let T_1 and T_2 be two non-deterministic SSTs over two finite sets of variables $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively, both with output variable O. Let $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) be the finite set of substitutions occurring in T_1 (resp. T_2). Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$. Let $\ell, k \in \mathbb{N}$. Let n_1 and n_2 be the number of states of T_1 and T_2, respectively.

We prove the result for the inclusion problem, i.e. $T_1 \subseteq_{k,\ell} T_2$. We let $\mathbb{D}_{k,\ell,\mathcal{S}}^{in} = \{u \otimes \lambda \otimes \mu \mid u \in \Sigma^* \wedge \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}\}$. The automaton recognizing $\mathbb{D}_{k,\ell,\mathcal{S}}$ in Lemma 21 can be easily extended into a DFA A_D which recognizes $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}$, using the same number of states, i.e., doubly exponential in ℓ^3 and $|\mathcal{X}|$, and singly exponential in k.

Now, observe that T_1 is (k, ℓ)-included in T_2 iff $L(T_1) \subseteq \mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$, where $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$ denotes the set $\{u \otimes \lambda \mid \exists u \otimes \mu \in L(T_2) : u \otimes \lambda \otimes \mu \in \mathbb{D}_{k,\ell,\mathcal{S}}^{in}\}$. Let A_1 be an NFA recognizing $L(T_1)$. It is easily obtained by simulating T_1 and has n_1 states, where n_1 is the number of states of T_1. Similarly, let A_2 be an NFA recognizing $L(T_2)$ with n_2 states.

Clearly, $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$ is definable by restricting the first and third component of A_D to $L(T_2)$ and then projecting onto the first and second component. This can be done via a product construction with A_2. This yields a NFA B for $\mathbb{D}_{k,\ell,\mathcal{S}}^{in}(L(T_2))$ with doubly exponentially many states in ℓ^3 and $|\mathcal{X}|$, singly exponentially many states in k, and polynomially many states in n_2. Note the B has polynomially many transitions in the size of T_1 and size of T_2, because the number of transitions is polynomial in $|\mathcal{S}|$ which depends on the size of T_1 and T_2.

So, the problem $T_1 \subseteq_{k,\ell} T_2$ boils down to checking the inclusion of the NFA A_1 of polynomial size in n_1 into the NFA B, which is well-known to be decidable in PSPACE in the size of both NFA. This yields a 2EXPSPACE procedure.

Now, if k, ℓ and $|\mathcal{X}|$ are fixed constants, note that A_D has a constant number of states and a polynomial number of transitions in the size of T_1 and T_2, because it depends on $\mathcal{S}$. Thus, B has a polynomial number of states in n_2 and polynomially many transitions. Therefore, the latter procedure is PSPACE.

The PSPACE lower bound is a direct consequence of the fact that the language inclusion problem for NFA is PSPACE-hard. This problem can be encoded to the $(0, 0)$-inclusion problem for non-deterministic SSTs which uses only a single variable O which is always updated by $O := \varepsilon$. So, those SSTs do not produce anything and are basically NFA.   ◄