Which classes of origin graphs are generated by transducers?*

Mikołaj Bojańczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle

University of Warsaw

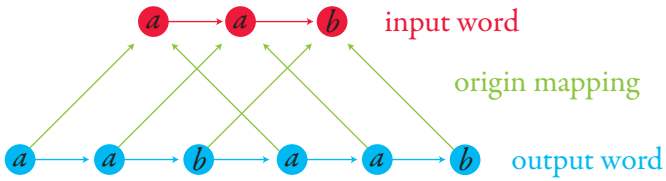
May 2017

Abstract

We study various models of transducers equipped with origin information. We consider the semantics of these models as particular graphs, called origin graphs, and we characterise the families of such graphs recognised by streaming string transducers.

1 Introduction

This paper is about string-to-string transductions with origin semantics. A string-to-string transduction is a binary relation between strings over fixed input and output alphabets. Examples include the *squaring* transduction $w \mapsto ww$, which is our running example, or the *subword* transduction, which is the set of pairs (u, v) such that v is a subword of u. Note that squaring is a function, while subword is a relation; both types will be studied. The origin semantics of a transduction (technically speaking, of a device computing it) consists not only of pairs (u, v) of input and output words, but also gives an *origin mapping* that specifies which positions of the input word were used to produce which positions of the output word. For example, suppose that we model the squaring transduction by a two-way automaton which does two consecutive left-to-right passes over the input word and copies the input word in each one. In this case, the origin semantics over a particular input word can be visualised as follows:



An object as in the above picture is called an *origin graph*, and we define an *origin string-to-string transduction* to be a set of origin graphs. Origin semantics is more fine-grained semantics than the usual semantics of transductions in the sense that even if two devices compute the same transduction, they might not have the same origin semantics. Origin semantics were introduced in [4], where it was shown that existing models of transducers, such as two-way transducers (also called two-way automata with outputs, e.g. [13]), MSO string-to-string transductions [9], or streaming string transducers [1, 3] can be equipped with origin semantics so that they generate not sets of pairs of words, but sets of origin graphs (the name origin graph is new in this paper). Furthermore, existing results on equivalence between models remain true when the origin semantics are used [4]. The goal of this paper is to study sets of origin graphs that are origin semantics of transducers. There are two parts.

In the first part, we study decision problems that involve MSO properties of origin graphs. The main result (not very hard) is that when given an MSO formula on origin graphs, and an origin string-to-string transduction realised by a non-deterministic streaming string transducer, one can decide if the formula is true in some origin graph from the transduction. This result gives a generic framework for deciding questions like: is some input position used at least twice in some output? is

*This paper is part of LIPA, a project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement N°683080).

the origin mapping order preserving? The result is proved by using techniques from the theory of MSO transductions [9].

In the second part, we study the structural properties of those classes of origin graphs that can be obtained by taking the origin semantics of some streaming string transducer (or any of the other equivalent models). Our goal is to describe them in a machine independent way. The principal result, Theorem 10, gives the following characterisation: a set of origin graphs is the origin semantics of some non-deterministic streaming string transducer if and only if it has three properties: *a*) it is MSO-definable as a set of coloured graphs; *b*) it has bounded degree; and *c*) it has bounded crossing, which means intuitively that the origin mapping does not oscillate too much. The idea to give a machine independent characterisation of origin semantics was already present in Theorem 1 from [4]. We believe however that origin graphs are a more intuitive and visual notion than the factorised words used in [4]. Furthemore, modelling the origin as a relational structure (the input word, the output word, and the origin information) makes it possible to use MSO logic, or to make the connection with structural notions such as clique width or tree width. That is why we hope that the machine independent characterisation from this paper can be seen as a natural complement to the results from [4], or possibly a clearer picture. Finally, in this paper we study more general models than [4], in particular we allow non-determinism and ε-transitions.

Important related work is the paper [11], which proposes to use logic to describe properties of origin graphs (they use the name *productions*). In particular [11] asks about the decidability of checking if a transducer, seen as a set of origin graphs, satisfies a specification given in some logic. This is the direct inspiration for our results in Section 3, in particular our Theorem 6 which says that it is decidable if a given MSO formula is true in some origin graph generated by a given transduction. In the case of [11], the logic used to express properties of origin graphs is not MSO, but a strict fragment called $\mathcal{L}_T$, a type of two-variable logic. Because $\mathcal{L}_T$ is weaker than MSO, our Theorem 6 is stronger than the model-checking result mentioned in [11, Section VI]. The reason why [11] uses a logic weaker than MSO is that they want to answer different questions than model-checking a given transducer; in particular the logic $\mathcal{L}_T$ is shown to have decidable satisfability when evaluated on the class of all origin graphs. For MSO, satisfiability is undecidable over the class of all origin graphs, and decidability requires additional assumptions, namely bounded tree width.

All the missing proofs can be found in the appendix.

2 Origin semantics

Define a *string-to-string transduction with input alphabet* Σ *and output alphabet* Γ to be a relation $R \subseteq \Sigma^* \times \Gamma^*$. A transduction is *functional* when it is a partial function.

Example 1 (Running example)**.** *Define the* squaring *functional transduction to be the function* $\{a,b\}^* \to \{a,b\}^*$ *defined by* $w \mapsto ww$.

2.1 Transductions recognised by streaming string transducers.

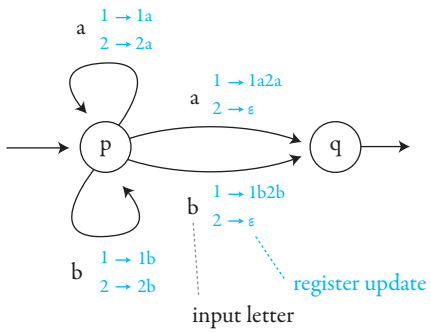
The main topic of this paper is the class of string-to-string transductions recognised by streaming string transducers [1]. Another equivalent presentation of this class is MSO string-to-string transductions, or a suitably defined non-deterministic version of two-way automata with output. We will mainly use the definition in terms of streaming string transducers, so we begin by defining that model.

Streaming string transducers. A streaming string transducer is a device which is used to transform (possibly non-deterministically) a word over an input alphabet into a word over an output alphabet. Because of non-determinism, one input might produce several outputs, possibly zero. The output is prepared by using registers. Before describing the device itself, let us explain how registers are used. Let Γ be an output alphabet and let $\mathcal{R}$ be a set of *register names*. Define a *register valuation* to be a function $\mathcal{R} \to \Gamma^*$ and define a *register update* to be a function $\mathcal{R} \to (\Gamma \cup \mathcal{R})^*$. A register update is viewed as a function from register valuations to register valuations in the following sense: if v is a register valuation and u is a register update, then applying u to v yields a register valuation which stores in register r the value $u(r)$ with each register name replaced by its contents under v. For example if $\mathcal{R}$ has only one register, and u is a register update defined by $r \mapsto ara$, then applying u to a register valuation simply adds the letter a to both the beginning and end of the word stored in the unique register r. A register update u is called *copyless* if for every register name r there is at most one register name s such that r appears in $u(s)$, and furthermore r appears at most once in $u(s)$.

Definition 2 (Streaming string transducer). *The syntax*[1] *of a streaming string transducer with input alphabet* Σ *and output alphabet* Γ *consists of:*

- *a non-deterministic automaton* $\mathcal{B}$ *with input alphabet* Σ*, called the* underlying automaton*;*
- *a finite set of* register names $\mathcal{R}$ *with a distinguished* output register $r_o \in \mathcal{R}$*;*
- *a labelling of transitions in* $\mathcal{B}$ *by copyless register updates.*

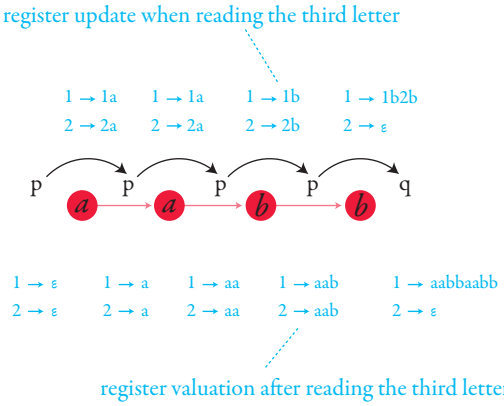
Example 3 (Running example). *The squaring function is recognised by a functional streaming string transducer which has two registers 1,2, with the output register being 1. The following picture shows this underlying automaton and its labelling by register updates.*



The automaton uses non-determinism to guess the last position so that the two registers are concatenated, however every non-empty input word admits exactly one run.

The semantics of a streaming string transducer is defined as follows. Suppose that ρ is a run of the underlying automaton $\mathcal{B}$, i.e., a sequence of transitions. Consider the empty register valuation which maps all registers to the empty word, and then apply all the register updates that label the transitions in ρ, beginning with the first transition and ending with the last transition. The output of ρ is defined to be the word over the output alphabet contained in the output register in the valuation described this way. Finally, the semantics of a streaming string transducer is defined to be the string-to-string transduction which consists of pairs (v, w) such that v is a word over the input alphabet and w is the output of some accepting run over the input word.

Example 4 (Running example). *Here is a picture of a run of the transducer from Example 3:*



A streaming string transducer is called *unambiguous* if the automaton $\mathcal{B}$ admits at most one accepting run on every input word[2]. For an unambiguous streaming string transducer, its semantics is a partial function $\Sigma^* \to \Gamma^*$. The transducer in Example 3 is unambiguous.

[1] Our syntax for streaming string transducers is different than the one used in [1], but it is routine to show that the expressive power is the same, as long as we allow non-determinism.

[2] One could also consider a streaming string transducer where the underlying automaton is deterministic. In this case, we would need to slightly modify the semantics, by adding a final function which performs a register update after reading the last input letter (e.g. concatenating the two registers as in the running example). After adding such final function, the deterministic model would have the same expressive power as the unambiguous variant used in this paper, see [1].

2.2 Origin semantics and origin graphs.

We now turn to the origin semantics of transducers. The idea is to give not just the output word, but also say which input positions were used to produce which output positions. The formalisation we use in this paper is *origin graphs*. Let fix an input alphabet Σ and an output alphabet Γ. For $w \in \Sigma^*$ and $v \in \Gamma^*$, an *origin graph* with input w and output v is defined to be a relational structure of the following form:

- the universe is the disjoint union of positions in w and positions in v;
- there are two binary predicates for the successor relations in w and v;
- there is a binary predicate, called the *origin mapping*, which is a total function from output positions to input positions;
- for each $a \in \Sigma \cup \Gamma$ there is a unary predicate which identifies positions with label a.

Note that the vocabulary of the relational structure depends on the choice of input and output alphabets; therefore a more formal definition would require talking about origin graphs over (Σ, Γ) where Σ is the input alphabet and Γ is the output alphabet. An origin graph can also be viewed as a directed graph, with vertices coloured by letters of the input and output alphabets, and edges coloured by three possible colours: successor edge in the input word, successor edge in the output word, and origin edge. Not every directed graph coloured this way is an origin graph; in an origin graph the input successors form a path, the output successors form a path on the remaining vertices, and the origin edges give a total function from the second path to the first one.

Definition 5. *An* origin string-to-string transduction *(origin transduction for short) consists of an input alphabet, an output alphabet, and a set of origin graphs over these alphabets.*

Note that an origin transduction might contain origin graphs which differ only on the origin mapping. Here is an example picture, for the (not necessarily connected) subword relation equipped with the natural origin semantics:

An origin transduction is called *functional* if every input word appears in at most one origin graph. An example of a functional origin transduction is the squaring in our running example, when equipped with the natural origin information. A non-example is the subword relation.

Let us define the origin semantics of a streaming string transducer. When reading an input position x, the transducer executes a register update. Such a register update creates some new letters which are added to registers, and also moves the contents between registers. We assume that the origin of these created letters is the input position x, and remains this way even if the position is moved to different registers in subsequent transitions. Using this description, we can associate an origin graph to each run of the transducer. We say that an origin transduction is the *origin semantics of* (or to use an alternative name, *recognised by*) a streaming string transducer if it is the set of origin graphs corresponding to its successful runs. Note that, when the automaton is non-deterministic, different successful runs over the same input word and producing the same output word might generate different origin graphs.

3 MSO on origin graphs

In this section, we discuss properties of origin graphs that can be defined in monadic second-order logic MSO. This is the logic which extends first-order logic by allowing quantification over sets of elements in the universe (but not sets of pairs, nor sets of sets, etc.). For a definition of the syntax and semantics of MSO, see [9].

MSO on origin graphs. An origin graph is a special case of a relational structure. If the input and output alphabets are Σ and Γ, then the vocabulary of the relational structure consists of three binary relations (input edge, output edge, origin edge) as well as one unary predicate for each letter in $\Sigma \cup \Gamma$. We use the name *origin vocabulary of* (Σ, Γ) for this vocabulary. An MSO formula over this

vocabulary defines a set of origin graphs, namely those origin graphs where it is true. Note that the structures over the origin vocabulary which are origin graphs are a set definable in MSO, essentially because one can axiomatise in MSO (but not, e.g. in first-order logic) that a directed graph is a single finite directed path. Therefore when talking about MSO-definable sets of origin graphs, it makes no difference whether or not we require the MSO formula to check if a structure is actually an origin graph.

The following result shows that satisfiability of MSO over origin graphs produced by a given streaming string transducer is decidable.

Theorem 6. *The following problem is decidable:*

Input: *A non-deterministic streaming string transducer $\mathcal{A}$ and an* MSO *formula φ over the origin vocabulary corresponding to $\mathcal{A}$;*

Question: *Is φ true in some origin graph in the origin semantics of $\mathcal{A}$?*

Proof sketch. To prove this result it is convenient to use MSO transductions in the sense of Courcelle and Engelfriet, see [9]. We first convert $\mathcal{A}$ into a non-deterministic MSO transduction, which can be done while preserving origin semantics [4] [3]. Given an MSO representation of $\mathcal{A}$, we can easily get an MSO transduction which inputs a word over the input alphabet, and outputs (non-deterministically) an origin graph that corresponds to some possible output of $\mathcal{A}$. Consider the following language

$$L = \{w \in \Sigma^* : \varphi \text{ is true in an origin graph produced by } \mathcal{A} \text{ on } w\}$$

By the Backward Translation Theorem (e.g. [9], p.66) the language L is definable in MSO, as the inverse image of an MSO-definable property under an MSO transduction. Therefore, L is regular, since MSO defines only regular word languages. □

Example 7. *An origin graph is called* order preserving *if the origin mapping gives a non-decreasing function from output positions to input positions. The set of order preserving origin graphs is clearly definable in* MSO*. Theorem 3 in [4] says that if an origin transduction has only order preserving origin graphs and is recognised by a streaming string transducer, then it is already recognised by a non-deterministic one-way transducer. Therefore Theorem 6 gives an algorithm for deciding if a streaming string transducer is equivalent, in terms of origin semantics, to a one-way non-deterministic transducer (such decidability, and even a polynomial time algorithm, although with inputs represented in a different way, was already given in [4]).*

Example 8 (Running example)**.** *Consider the squaring transduction. Every origin graph in this transduction satisfies the following property, which can be formalised in* MSO*: the output positions can be partitioned into two connected blocks, such that the origin mapping is order preserving when restricted to each of the blocks. For functional origin transductions, this property corresponds to being recognised by a deterministic two-way transducer which does two left-to-right passes on the input.*

A corollary of the proof of Theorem 6 is that when the transducer $\mathcal{A}$ is fixed, then there is a linear time algorithm (which simply runs a finite automaton) for checking if a given input word can produce an origin graph satisfying φ.

Proposition 9. *If an origin transduction is recognised by a streaming string transducer, then it is definable in* MSO *as a set of origin graphs.*

Note that the converse of the above proposition is false, even assuming that there is a bound on the number output positions which can originate in the same input position, see Examples 13 and 15. The issue is that it is more difficult to produce an origin graph (using the origin semantics of an MSO transduction) than it is to check if a given origin graph is correct.

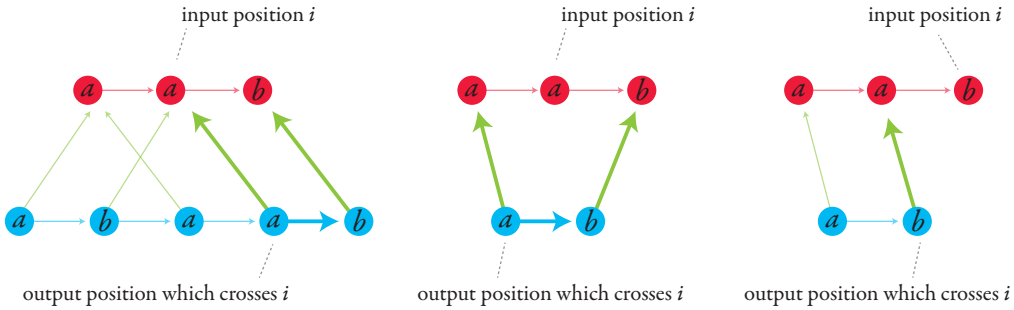
4 Which sets of origin graphs are recognised by streaming string transducers?

Which sets of origin graphs are recognised by streaming string transducers (equivalently, MSO transductions)? What about functional streaming string transducers? The main goal of this paper is

[3] In [4] the conversion is done in the deterministic case, but it can be easily extended to the non-deterministic case.

to give machine independent characterisations of such sets. This is Theorem 10 below, which says that a set of origin graphs is recognised by a streaming string transducer if and only if it is MSO-definable, has bounded origin (i.e., each input position is the origin of a bounded number of output positions), and it has bounded crossing, as explained below.

An output position j in the graph is said to *cross* an input position i if the position j has origin at most i and the successor of j either does not exist (i.e. j is the last output position) or has origin greater than i. Intuitively, to go from the origin of position j to the origin position $j+1$ on the input word, a reading head needs to cross position i. Here is a picture:



We are now ready to state the main result of this paper.

Theorem 10. *Let $\mathcal{G}$ be an origin transduction, i.e., an input alphabet, an output alphabet, and a set of origin graphs over these alphabets. Then $\mathcal{G}$ is recognised by a streaming string transducer with k registers if and only if it satisfies all of the following conditions:*

bounded origin: *there is some $m \in \mathbb{N}$ such that in every origin graph from $\mathcal{G}$, every input position is the origin of at most m output positions;*

k-crossing: *in every origin graph from $\mathcal{G}$, every input position is crossed by at most k output positions;*

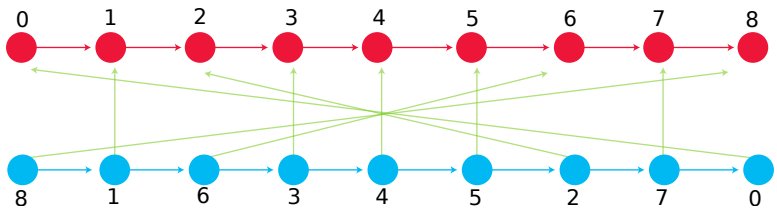
mso-*definable:* *there is an* MSO *formula which is true in exactly the origin graphs from $\mathcal{G}$.*

The proof of the theorem is sketched in the next section. A corollary of the theorem is that an origin transduction is recognised by a streaming string transducer if and only if it has bounded origin, is MSO definable, and has bounded crossing (i.e., k-crossing for some k). The following three examples show how the three conditions in Theorem 10 are minimal, i.e., none of the conditions is implied by the remaining ones.

Example 11 (MSO definable)**.** *Consider the identity origin transduction with its domain restricted to some non-regular subset of inputs, e.g. words of prime length. This origin transduction satisfies all conditions in Theorem 10 except for* MSO *definability.*

Example 12 (Bounded origin)**.** *Consider an origin transduction (with one letter in both the input and output alphabets) which is only defined on inputs with one letter and then copies the unique output letter an arbitrary number of times (therefore every output position originates in the unique input position). This origin transduction satisfies all conditions in Theorem 10 except for bounded origin. Streaming string transducers with ε-transitions, as discussed in Section 5, will be able to recognise this example.*

Example 13 (Bounded crossing)**.** *Consider the following origin transduction, which is functional. The input and output alphabets have only one letter each. The domain is words of odd length. A word of length $2n+1$ is mapped to a word of same length, but the origins are shuffled so that the origins of odd numbered positions are the same position, while the origins for even numbered positions are reversed. More precisely, if the positions are $0, \ldots, 2n$ then the origin of an odd numbered position $2i+1$ is $2i+1$, while the origin of an even numbered position $2i$ is $2n-2i$. Here is the picture of an origin graph in this origin transduction:*



We claim that this origin transduction has unbounded crossing, but satisfies the remaining conditions in Theorem 10. To show unbounded crossing, observe that if the length of the input word is $2n+1$, then the middle input position n is crossed by all even numbered output positions greater than n. Clearly every origin graph in the transduction has bounded origin, because each input position is the origin of exactly one output position. For MSO *definability, we observe that an origin graph belongs to the transduction if and only if it satisfies all the following conditions which are definable in* MSO *(in fact, first-order logic):*

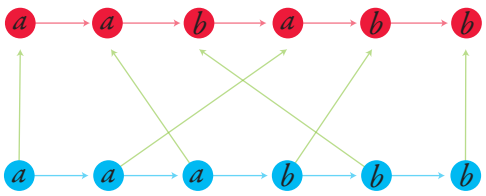
1. *the origin of the first output position is the last input position;*
2. *the origin of the second output position is the second input position;*
3. *if $j > 1$ is an odd output position with origin i, then the origin of $j-2$ is $i-2$;*
4. *if $j > 1$ is an even output position with origin i, then the origin of $j-2$ is $i+2$.*

Tree width. In Theorem 10, we use bounded crossing as one of the conditions. Another candidate for a structural property on origin graphs is that they have bounded tree width (see e.g. [9] for a definition). The following result shows that bounded tree width (and even bounded path width, which corresponds to the width of path decompositions, i.e., the special case of tree decompositions where the tree is a path) is a necessary condition for being recognised by a streaming string transducer. Indeed, we show that any origin transduction which is bounded crossing has bounded path width. Moreover, we can express the crossing boundedness property in terms of a particular path decomposition of bounded width.

Proposition 14. *Every bounded crossing origin transduction has bounded path width.*

We now show that bounded path width is not a sufficient condition, in the sense that the bottom-up implication of Theorem 10 would fail if we would replace bounded crossing by bounded path width. One example is the origin transduction from Example 13, which can be shown to have bounded path width. Here is another example, which also shows that bounded crossing and recognisability by streaming string transducers are both notions that are not closed under reversing origin edges.

Example 15. *Consider a variant of the squaring function, which is defined only on words of even length, and maps a word w to uv where u (resp. v) is the subword consisting of the odd-numbered (resp. even-numbered) positions of w. This transduction is easily seen to be recognised by a (deterministic) streaming string transducer, and hence the underlying set $\mathcal{G}$ of origin graphs has bounded tree width by Proposition 14. Define $\mathcal{G}'$ to be the set of origin graphs which are obtained from $\mathcal{G}$ by reversing the origin edges. Here is a picture of an origin graph in $\mathcal{G}'$:*



Since the origin mapping is bijective, $\mathcal{G}'$ is also a set of origin graphs. It has bounded origin (bounded by 1). By Theorem 10 and Proposition 14, $\mathcal{G}$ is MSO *definable and has bounded path width. Path width and* MSO *definability are not changed by reversing arrows, and therefore $\mathcal{G}'$ has bounded origin, bounded path width and is also* MSO*-definable. Nevertheless, $\mathcal{G}'$ is not the origin semantics of any streaming string transducer. Indeed, if $\mathcal{A}$ would be a streaming string transducer recognising $\mathcal{G}'$, then the pre-image under $\mathcal{A}$ of the regular set $(aa+bb)^*$ would be the non-regular set of words of the form ww over the alphabet $\{a,b\}$, contradicting the fact that regular word languages are preserved under taking pre-images of streaming string transducers (essentially the Backwards Translation Theorem from [9]).*

Recognisability. In Theorem 10, the conditions used are bounded origin, bounded crossing and MSO definability. While bounded origin and bounded crossing are purely combinatorial properties of graphs, MSO definability has a more syntactic character. A less syntactic alternative to MSO definability would be to use *recognisability* in the sense of Definition 4.29 in [8]. Intuitively speaking, a class of relational structures is called recognisable if it has finite index for a certain naturally defined equivalence relation à la Myhill Nerode. In [5] it is shown that if a class of relational structures has bounded tree width, then recognisability is the same as thing as definability in MSO. Therefore, in the

statement of Theorem 10 we could replace MSO definability by recognisability, and the theorem would still be true.

The functional case. Theorem 10 gives a characterisation of origin transductions recognised by streaming string transducers. Recall that a streaming string transducer was called unambiguous if its underlying automaton had at most one successful run for each input word. Such transducers are equivalent to the deterministic model in [3], also when using origin semantics [4]. They can only recognise functional origin transductions. As it turns out, this is the only restriction, i.e., if an origin transduction is functional and recognised by a (possibly ambiguous) streaming string transducer, then it is recognised by an unambiguous one. Furthermore, in the functional case the condition on bounded origin becomes superfluous.

Theorem 16. *Let $\mathcal{G}$ be an origin transduction. Then $\mathcal{G}$ is recognised by an unambiguous streaming string transducer with k registers if and only if it is functional, k-crossing and* MSO*-definable.*

Unambiguous streaming string transducers can moreover be simulated by deterministic ones (in the sense of Alur and Černý [1]) but at the cost of adding registers [2].

5 Sketch of the proof

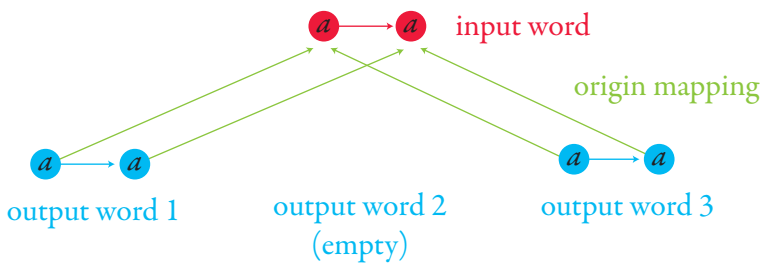
To prove Theorems 10 and 16, we first characterise the origin graphs which can be generated by streaming string transducers with ε-transitions. A streaming string transducer with ε-transitions is the generalisation of the model described in Definition 2, where ε-transitions are allowed in the underlying automaton. The origin mapping is defined so that if an output position is created (i.e., added to some register) by an ε-transition, then its origin is the most recently read input letter. To make this well defined, we make the syntactic restriction that no ε-transitions can be used when in an initial state, and therefore the first transition in each run must consume an input letter.

Example 17. *Consider the set of origin graphs where the input word has only one letter, and hence this letter is the origin of all output positions, and the output word is $a^n b^n$ for some $n \in \mathbb{N}$. To recognise this string transduction, we use two registers. After reading the unique input position, the automaton enters a loop of ε-transitions. Each one appends a to one register, and b to the other. At the end, the transducer does an ε-transition to the accepting state which concatenates both registers.*

The above example shows that when ε-transitions are allowed, the origin semantics of a streaming string transducer needs no longer to be MSO-definable. Theorem 18 below shows that if we additionally assume that a set of origin graphs is MSO-definable, then being the semantics of a streaming string transducer with ε-transitions is equivalent to having bounded crossing. The theorem is the main step in our proof of Theorems 10 and 16.

Theorem 18. *Let $\mathcal{G}$ be an origin transduction which is* MSO*-definable. Then, $\mathcal{G}$ is recognised by a k-register streaming string transducer with ε-transitions if and only if $\mathcal{G}$ is k-crossing.*

We sketch the proof here. The left-to-right implication is straightforward, and does not need the assumption on MSO definability. Below we discuss the converse implication. The idea is that if an origin graph has bounded crossing, then it can be constructed by applying a sequence of elementary operations to the empty graph. Intermediate objects produced by these elementary operations are going to be like origin graphs, except that the output word might be in several pieces, corresponding intuitively to the register contents. Define a *k-block origin graph* to be the extension of origin graphs where there are exactly k output words, which are called *blocks*, some of which may be empty, as in the following picture for $k = 3$:



To transform k-block origin graphs, we use the following toolkit of operations, which corresponds intuitively to the registers in a streaming string transducer.

Input. This operation takes one parameter, a letter a of the input alphabet. The result of the operation is that a new position with letter a is added to the end of the input word;

Output. This operation takes three parameters: a *target block* $i \in \{1, \ldots, k\}$, a *content* c which is either a letter of the output alphabet or a number $j \in \{1, \ldots, k\}$ different than i, and a *side* $s \in \{\text{left,right}\}$. The result of the operation is that the content (i.e., either an output letter of or the contents of register c, depending on the type of c) is concatenated to the left/right (depending on the side s) of the target block i. If the content is a letter, then its origin is set to be the last input position (if there is no input position and the content is a letter, then the operation fails).

We write Ω_k for the above set of operations (assuming that the alphabets are implicit from the context), which is finite. Define *k-folding* to be the function from $(\Omega_k)^*$ to k-block origin graphs which maps a sequence of operations to the k-block origin graph obtained by successively applying the sequence of operations starting from the empty k-block origin graph. Note that k-folding is partial, because the output operation can fail.

Lemma 19. *The k-folding operation is (a) surjective; and (b) an* MSO *interpretation.*

Proof of Theorem 18. We only show the right-to-left implication. Suppose then that $\mathcal{G}$ is an origin transduction which is MSO definable and is k-crossing. Define $\mathcal{G}'$ to be the set of k-block origin graphs such that: (i) the i-th output word is empty for $i \neq 1$; and (ii) if only the input and 1-st output word are kept, the resulting origin graph belongs to $\mathcal{G}$. If $\mathcal{G}$ is MSO definable, then so is $\mathcal{G}'$. Let $L \subseteq (\Omega_k)^*$ be those sequences of operations whose k-folding is in $\mathcal{G}'$. By Lemma 19 (b) and the Backwards Translation Theorem [9], L is definable in MSO. By Lemma 19 (a), $\mathcal{G}'$ is equal to the image of L under k-folding. By Büchi-Elgot-Trakhtenbrot's Theorem [7], L is recognised by a finite automaton. We transform this finite automaton into a k-register non-deterministic streaming string transducer with ε-transitions by translating any letter σ of Ω_k into:

- a transition reading an input symbol without updating the registers, if σ is of type input;
- an ε-transition with an appropriate register operation if σ is of type output. □

To complete the proof of Theorem 10, we finally show that if the origin semantics of an εNSST has bounded origin then ε-transitions can be eliminated.

6 Classes of origin transductions and perspectives

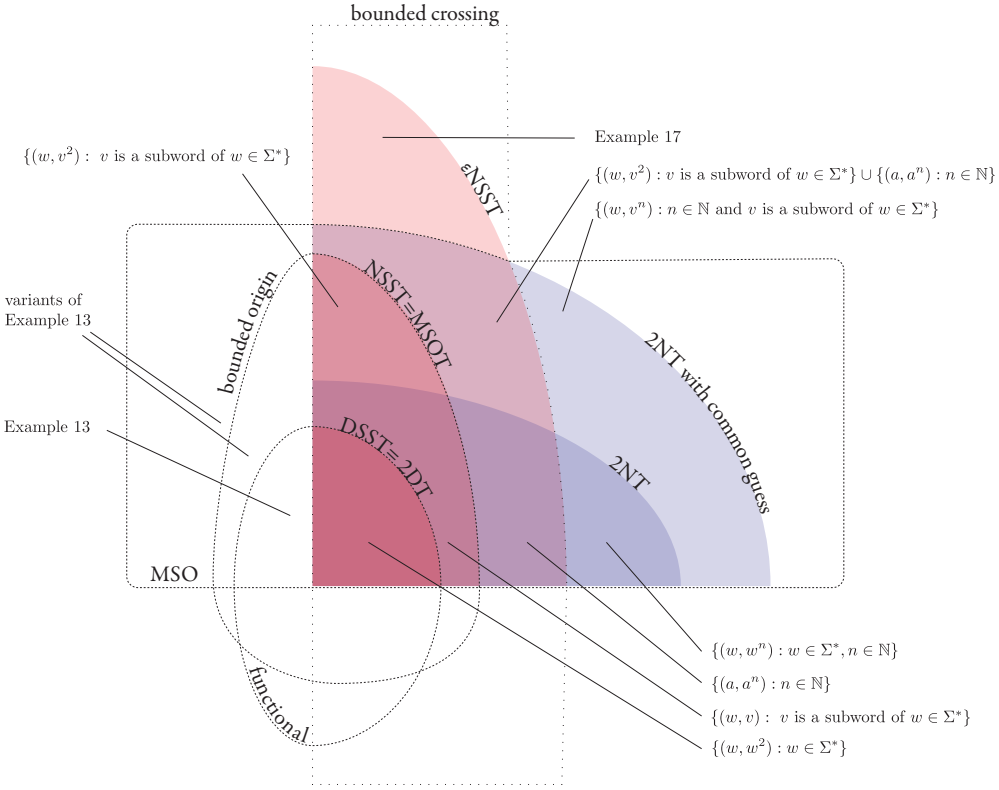
Our main contribution is a characterisation of the origin semantics of streaming string transducers (deterministic; non-deterministic; with ε-transitions providing MSO-definability), using properties of the origin graphs such as functionality, origin boundedness, crossing boundedness and MSO-definability. The origin transductions recognised by these transducers form a hierarchy depicted in red in the figure below, where DSST (resp. NSST, εNSST) denotes the family of origin transductions recognised by a deterministic (resp. non-deterministic, non-deterministic with ε-transitions) streaming string transducer.

The figure also includes two-way transducers, which define an orthogonal hierarchy, depicted in blue. We consider deterministic and non-deterministic variants as well as those with *common guess*. A two-way transducer (deterministic or not) is equipped with a common guess if, before starting the computation, a finite colouring of the input positions is performed, and this colouring is the same each time the head revisits a position. This strictly increases the expressivity, e.g. the relation $\{(u, vv) \mid v \text{ is a subword of } u\}$ is recognised by a two-way transducer with common guess but not without. The origin semantics of a two-way transducer is defined in a natural way, i.e., an output letter originates in the input position which was scanned by the input head when the letter has been produced. In the figure, the classes of origin transductions recognised by two-way automata are denoted respectively by 2DT, 2NT, and 2NT with common guess.

The class of functions recognised by deterministic two-way transducers (resp. with common guess) is known to be the same as the one recognised by deterministic streaming string transducers [13, 1] (resp. non-deterministic streaming string transducers [3]), even when considering the origin semantics [4].

Finally, we denote by MSOT the family of origin transductions recognised by a non-deterministic MSO-transduction. It is equal to NSST [3, 13]. The transductions recognised by deterministic MSO-transductions are the same as for DSST [1], this remains true with origin semantics [4].

We can prove that all the MSO-definable (resp. bounded origin) origin transductions in εNSST are recognised by a non-deterministic two-way transducer with common guess (resp., are in NSST). Moreover, all the transductions that have bounded origin and which are recognised by a non-deterministic two-way transducer with common-guess, are recognised by a streaming string tranducer (and have therefore bounded crossing). All the other intersections are non-empty and can be populated with some origin transductions (with their natural origin information) as depicted in the figure.



References

[1] Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, pages 1–12, 2010.

[2] Rajeev Alur and Loris D'Antoni. Streaming tree transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 42–53. Springer, 2012.

[3] Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 1–20. Springer, 2011.

[4] Mikołaj Bojańczyk. Transducers with origin information. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 26–37, 2014.

[5] Mikołaj Bojańczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 407–416, 2016.

[6] Mikołaj Bojańczyk and Michal Pilipczuk. Optimizing tree decompositions in mso. 2017.

[7] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960.

[8] Bruno Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theor. Comput. Sci.*, 80(2):153–202, 1991.

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

[10] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Cham, 2015.

[11] Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Decidable logics for transductions and data words. *CoRR*, abs/1701.03670, 2017.

[12] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

[13] 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.

[14] Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, (140):326–329, 1961.

A MSO Transductions

In this part of the appendix we formally define MSO transductions and prove the results from Section 3.

A.1 MSO transductions and their origin semantics

This part, Appendix A.1 is mainly copied from [6], but included here to make the paper self-contained. Define a *vocabulary* to be set of *relation names*, each one with associated arity in $\mathbb{N}$. A *relational structure* over the vocabulary Σ consists of a set called the *universe*, and for each relation name in the vocabulary, an associated relation of the same arity over the universe. To describe properties of relational structures, we use logics, mainly monadic second-order logic (MSO). This logic allows quantification both over single elements of the universe and also over subsets of the universe. For a precise definition of MSO, see [9].

We use MSO to describe properties on words. A word $w \in \Sigma^*$ is viewed as a relational structure, where the universe is the positions, there is a binary predicate for the successor relation, and a unary predicate for every label $a \in \Sigma$ which selects positions with that label.

Transductions between relational structures. Suppose that Σ and Γ are vocabularies. Define a *transduction* with input vocabulary Σ and output vocabulary Γ to be a set of pairs

$$\underbrace{\text{structure over } \Sigma}_{\text{input}} \qquad \underbrace{\text{structure over } \Gamma}_{\text{output}}$$

which is invariant under isomorphism of relational structures, i.e., applying an isomorphism to either the input or the output does not change membership in the transduction. A string-to-string transduction is the special case when each pair in the transduction is a pair of words, modelled as relational structures in the way described below. Note that a transduction is a relation and not necessarily a function, thus it can have many different possible outputs for the same input. A transduction is called *deterministic* if it is a function. For example, the (not necessarily connected) subword relation $sub \subseteq \Sigma^* \times \Sigma^*$ is a transduction from words to words, but it is not deterministic since a word can have many subwords.

MSO transductions. This paper uses MSO transductions, as defined in the book of Courcelle and Engelfriet [9], which are a special case of transductions that can be defined using the logic MSO.

We use the terminology and definitions of transductions from [5]; the main idea is that an MSO transduction is a finite composition of transductions of the following types: copy the input a fixed number of times, non-deterministically colour the universe of the input, and add new predicates to the vocabulary with interpretations given by MSO formulas over the input vocabulary. We refer to Courcelle and Engelfriet [9] for a broader discussion of the role of MSO transduction in the theory of formal languages. More formally, an MSO transduction is any transduction that can be obtained by composing a finite number of transductions of the following kinds. Note that kind 1 is a partial function, kinds 2, 3, 4 are functions, and kind 5 is a relation.

1. **Filtering.** For every MSO sentence φ over the input vocabulary, there is a transduction that filters out structures where φ is satisfied. Formally, the transduction is the partial identity whose domain consists of the structures that satisfy φ. The input and output vocabularies are the same.

2. **Universe restriction.** For every MSO formula $\varphi(x)$ over the input vocabulary with one free first-order variable, there is a transduction which restricts the universe to those elements that satisfy φ. The input and output vocabularies are the same, the interpretation of each relation in the output structure is defined as the restriction of its interpretation in the input structure to tuples of elements that remain in the universe.

3. **MSO interpretation.** This kind of transduction changes the vocabulary of the structure while keeping the universe intact. For every relation name R of the output vocabulary, there is an MSO formula $\varphi_R(x_1, \ldots, x_k)$ over the input vocabulary which has as many free first-order variables as the arity of R. The output structure is obtained from the input structure by keeping the same universe, and interpreting each relation R of the output vocabulary as the set of those tuples $(x_1, \ldots, x_k)$ that satisfy φ_R.

4. **Copying.** For $k \in \{1, 2, \ldots\}$, define k-copying to be the transduction which inputs a structure and outputs a structure consisting of k disjoint copies of the input. Precisely, the output universe consists of k copies of the input universe. The output vocabulary is the input vocabulary enriched with a binary predicate copy that selects copies of the same element, and unary predicates $\mathsf{layer}_1, \mathsf{layer}_2, \ldots, \mathsf{layer}_k$ which select elements belonging to the first, second, etc. copies of the universe. In the output structure, a relation name R of the input vocabulary is interpreted as the set of all those tuples over the output structure where the original elements of the copies were in relation R in the input structure.

5. **Colouring.** We add a new unary predicate to the input structure. Precisely, the universe as well as the interpretations of all relation names of the input vocabulary stay intact, but the output vocabulary has one more unary predicate. For every possible interpretation of this unary predicate, there is a different output with this interpretation implemented.

Origin semantics of MSO transductions Define an *origin transduction* with input vocabulary Σ and output vocabulary Γ to be a set of triples

$$\underbrace{\text{structure over } \Sigma}_{\text{input}} \quad \underbrace{\text{structure over } \Gamma}_{\text{output}} \quad \underbrace{\text{function from output universe to input universe}}_{\text{origin mapping}}$$

which is invariant under isomorphisms. When applying an isomoprhism to either the input or output, the origin mapping is updated accordingly. A triple as above is called an *origin graph*. An origin graph can be represented as a relational structure, which is the disjoint union of the input and output structures, together with a binary relation representing the origin mapping.

Each of the five types of transductions used in the definition of MSO transduction comes with a natural origin semantics: the identity function (up to isomoprhism) is used for filtering, universe restriction and colouring, while copying and interpretations do the natural thing. Origin transductions are composed in the natural way, by composing the origin mappings. Hence, we can talk about the origin semantics of an MSO transduction. In [4] it is remarked that, with origin semantics, MSO transductions and streaming string transducers have the same expressive power.

A.2 MSO satisfiability over origin graphs

We now give a more formal proof of Theorem 6.

Theorem 6. *The following problem is decidable:*

Input: *A non-deterministic streaming string transducer $\mathcal{A}$ and an* MSO *formula φ over the origin vocabulary corresponding to $\mathcal{A}$;*

Question: *Is φ true in some origin graph in the origin semantics of $\mathcal{A}$?*

The following lemma is easy to see from the definition of MSO transductions and their origin semantics.

Lemma 20. *Let φ be an* MSO *transduction. Define $\hat{\varphi}$ to be the set of pairs $(\mathfrak{A}, \mathfrak{B})$ such that $\mathfrak{B}$ is an origin graph in the origin semantics of φ (represented as a relational structure) and $\mathfrak{A}$ is the input structure of $\mathfrak{B}$. Then $\hat{\varphi}$ is an* MSO *transduction.*

Since for origin string-to-string semantics, MSO transductions have the same expressive power as streaming string transducers, we get the following result.

Lemma 21. *If $\mathcal{A}$ is a streaming string transducer, then the following set is an* MSO *transduction from strings to origin graphs: the set of pairs (w, g) such that w is an input word and g is an origin graph in the origin semantics of $\mathcal{A}$ which has w as the input word.*

Proof of Theorem 6. Suppose that $\mathcal{A}$ and φ are as in the assumptions of the theorem. Apply Lemma 21, yielding an MSO transduction from strings to origin graphs. Apply the Backwards Translation Theorem from [9], to this transduction and the set of origin graphs satisfying φ, yielding an MSO sentence, say ψ, which defines those input strings which can produce at least one origin graph satisfying φ. The constructions in Lemmæ 20 and 21 as well as the Backwards Translation Theorem are effective, which means that one can compute ψ based on $\mathcal{A}$ and φ. Over words, MSO has effectively the same expressive power as automata, and therefore ψ can be effectively converted into a finite state automaton. To finish the proof of Theorem 6, it suffices to test this automaton for emptiness. For the linear time algorithm mentioned after Theorem 6, it suffices to run the automaton on the input word. □

Proposition 9. *If an origin transduction is recognised by a streaming string transducer, then it is definable in* MSO *as a set of origin graphs.*

Proof. Given a streaming string transducer, we fix the MSO-transduction ϕ from strings to origin graphs obtained by Lemma 21. We call $\mathcal{G}$ the set of images of ϕ. We will exhibit an MSO-transduction τ from origin graphs to strings which inverts ϕ and then use the Backward Translation Theorem [9] to draw the conclusion. Given an origin graph g, τ decides whether g belongs to $\mathcal{G}$, and if so, outputs the input word of g.

Take $g \in \mathcal{G}$ and call u its input word. We first construct g' by applying ϕ to g, i.e., to its input word u, and obtain a so-called "origin bi-graph", i.e., an input word and two output words v_1, v_2 whose positions originate in the input word. This operation can be done by an (origin graph-to-origin bi-graph) MSO transduction ρ_1. (This is possible because there are distinct input- and output-successor predicates in the origin vocabulary as defined in Section 3.)

Consider that ϕ is m-origin bounded. We duplicate m times each letter of u and separate them by a fresh symbol #, and redefine the origin edges so that given a letter u_i of u, the first letter of v_1 (resp. v_2) originating in u_i in g', originates in the first copy of u_i, the second in the second copy, etc. We call u' the new input word and g'' the obtained origin bi-graph. As m is finite and v_1 and v_2 are ordered, this transformation can be done with an MSO-transduction ρ_2.

We now define a formula in first order logic ρ_3 which checks that $v_1 = v_2$. This is done by checking that for any vertex x of v_1 (resp. v_2), there exists exactly one vertex y of v_2 (resp. v_1) which originates in the same vertex of u', has the same label as x, and such that its left (resp. right) neighbour has the same label and same origin as the left (resp. right) neighbour of x, or both x and y have no left (resp. right) neighbour. As each letter of v_1 and v_2 has exactly one origin, this formula is true only when both words are equal.

Finally, we apply an MSO-transduction ρ_4 dropping v_1 and v_2, identifying all nodes between two # and dropping the #, thus leaving us with u.

Taken together, $\tau = \rho_1 \circ \rho_2 \circ \rho_3 \circ \rho_4$ is an MSO-transduction with domain $\mathcal{G}$ which maps an element of $\mathcal{G}$ to its input word. The image of this transduction is thus the inverse image of $\mathcal{G}$ by ϕ. As the domain of ϕ is MSO-definable, we get that the image of $\mathcal{G}$ by τ is MSO-definable, and therefore, by the Backward Translation Theorem, $\mathcal{G}$ is MSO-definable. □

B Proof of Theorem 18

Theorem 18. *Let $\mathcal{G}$ be an origin transduction which is* MSO*-definable. Then, $\mathcal{G}$ is recognised by a k-register streaming string transducer with ε-transitions if and only if $\mathcal{G}$ is k-crossing.*

In the rest of the appendix, we use the shortcut εNSST for streaming string transducer with ε-transitions.

B.1 Proof of the if direction of Theorem 18

Proposition 22. *For all positive integers k, every origin transduction recognised by a εNSST with k registers is k-crossing.*

Proof. First, consider an εNSST $\mathcal{A}$ and a run ρ of $\mathcal{A}$ on some input word u which produces an output word v. Let i be a position of u and C the set of output positions crossing i (as defined in Section 4). We consider the valuation ν of registers obtained after reading the position i of u in ρ. Suppose $|C| \geq k+1$. As there are k registers, there is a register r such that $\nu(r) = xaybz$ for some words x, y and z and some letters a and b corresponding to some positions $j < j' \in C$ in v. Since j crosses i, the position $j+1$ of v originates in a position strictly greater than i. Thus, y is necessarily non-empty and its first letter corresponds to the position $j+1$ in v and therefore originates to a position strictly greater than i. A contradiction. Thus $|C| \leq k$, and the origin semantics of $\mathcal{A}$ is k-crossing. □

B.2 Proof of the only-if direction of Theorem 18

Proposition 23. *Let $\mathcal{G}$ be an* MSO*-definable origin transduction. For all positive integers k, if $\mathcal{G}$ is k-crossing, then it is recognised by a εNSST with k registers.*

This direction is much more intricate than the previous one and involves the use of a generalisation of origin graphs: block origin graphs.

B.2.1 Block origin graphs

A key point of our proof is to decompose origin graphs into a sequence of elementary operations applied to the empty graph. This decomposition is described thanks to the notion of *block origin graphs*. Intuitively, a *block origin graph* is an origin graph deprived of a suffix of the input word, and the positions of the output word that originate in this suffix. In the following, we will see a word as a relational structure.

Definition 24 (block origin graphs)**.** *Given a positive integer k, a k-block origin graph (k-*BLOG *for short) over (Σ, Γ) is a tuple $g = ((V_{in}, E_{in}), (V_1, E_1), \dots, (V_k, E_k), \mathrm{Origin})$ where (V_{in}, E_{in}) is a word on Σ (called the* input word*), for all $1 \leq m \leq k$, (V_m, E_m) is a word on Γ (called the* block of index m*), and* Origin *is an application from $V_{out} = \bigcup_{1 \leq m \leq k} V_m$ to V_{in}.*

Remark that if a k-BLOG is non-empty, then it has a non-empty input word (otherwise the function Origin cannot be defined). Given a k-BLOG g, the first (resp. last) position of the block (V_m, E_m) is called the *interface of side* L (resp. R) of index m. The set of the k-BLOGs is denoted by $\mathcal{H}_k$.

Remark 25. *Let us point out that two different k-*BLOG*s may have the same graph representation. Indeed, the block numbering is lost when considering a k-*BLOG *as a graph and thus by permuting the blocks, the graph representation is not changed.*

Link with origin graphs. For all positive integers k, any origin graph can be seen as a k-BLOG with at most one non-empty block. However, this may lead to several representations of the same origin graph, depending on which block is chosen non-empty. In what follows, we will identify an origin graph g with the k-BLOG with same input word as g, first block the output word of g, other blocks empty, and same origin function as g. Conversely, a k-BLOG with at most one non-empty block can be seen as an origin graph where the information about the index of the non-empty block is lost.

Operations on k-blogs. Let k be a positive integer. We now define operations on k-BLOGs that is to say partial functions from $\mathcal{H}_k$ to itself.

Definition 26 (elementary operations on $\mathcal{H}_k$)**.** *Given a k-*BLOG

$$g = ((V_{in}, E_{in}), (V_1, E_1), \dots, (V_k, E_k), \mathrm{Origin})$$

*for each $a \in \Sigma$, $b \in \Gamma$ and $m, n \in \{1, \dots, k\}$, the following operations result in a k-*BLOG

$$g' = ((V'_{in}, E'_{in}), (V'_1, E'_1), \dots, (V'_k, E'_k), \mathrm{Origin}')$$

defined as follows:

$g \cdot \texttt{input}(a)$**:** *If g is the empty graph, g' is the k-*BLOG *with input word a and all blocks empty. Otherwise, it is defined as the k-*BLOG *with input word ua where u is the input word of g and same blocks and origin function as g.*

All the following operation are undefined if (V_{in}, E_{in}) is empty.

$g \cdot \texttt{create}_m(b)$**:** *If (V_m, E_m) is non-empty, g' is undefined. Otherwise, $(V'_{in}, E'_{in}) = (V_{in}, E_{in})$, for all $n \neq m$, $(V'_n, E'_n) = (V_n, E_n)$ and (V'_m, E'_m) is the word b. Moreover,* Origin$'$ *extends* Origin *such that for the only element t in V'_m,* Origin(t) *is the last position of the input word of g'.*

$g \cdot \texttt{L-append}_m(b)$**:** *If (V_m, E_m) is empty, g' is undefined. Otherwise, $(V'_{in}, E'_{in}) = (V_{in}, E_{in})$, for all $n \neq m$, $(V'_n, E'_n) = (V_n, E_n)$ and (V'_m, E'_m) is the word bv where v is the word represented by the graph (V_m, E_m). Moreover,* Origin$'$ *extends* Origin *such that the interface of side* L *of index m in g' (the newly created vertex) originates in the last position of the input word of g'.*

$g \cdot \texttt{R-append}_m(b)$**:** *If (V_m, E_m) is empty, g' is undefined. Otherwise, $(V'_{in}, E'_{in}) = (V_{in}, E_{in})$, for all $n \neq m$, $(V'_n, E'_n) = (V_n, E_n)$ and (V'_m, E'_m) is the word vb where v is the word represented by the graph (V_m, E_m). Moreover,* Origin$'$ *extends* Origin *such that the interface of side* R *of index m in g' (the newly created vertex) originates in the last position of the input word of g'.*

$g \cdot \texttt{concat}_{m,n}(b)$**:** *If (V_m, E_m) or (V_n, E_n) is empty, g' is undefined. Otherwise, $(V'_{in}, E'_{in}) = (V_{in}, E_{in})$, for all $\ell \notin \{n, m\}$, $(V'_\ell, E'_\ell) = (V_\ell, E_\ell)$, (V'_n, E'_n) is the empty word and (V'_m, E'_m) is the word vbw where v (resp. w) is the word represented by the graph (V_m, E_m) (resp. (V_n, E_n)). Moreover,* Origin$'$ *extends* Origin *such that the newly created vertex labelled by b originates in the last position of the input word of g'.*

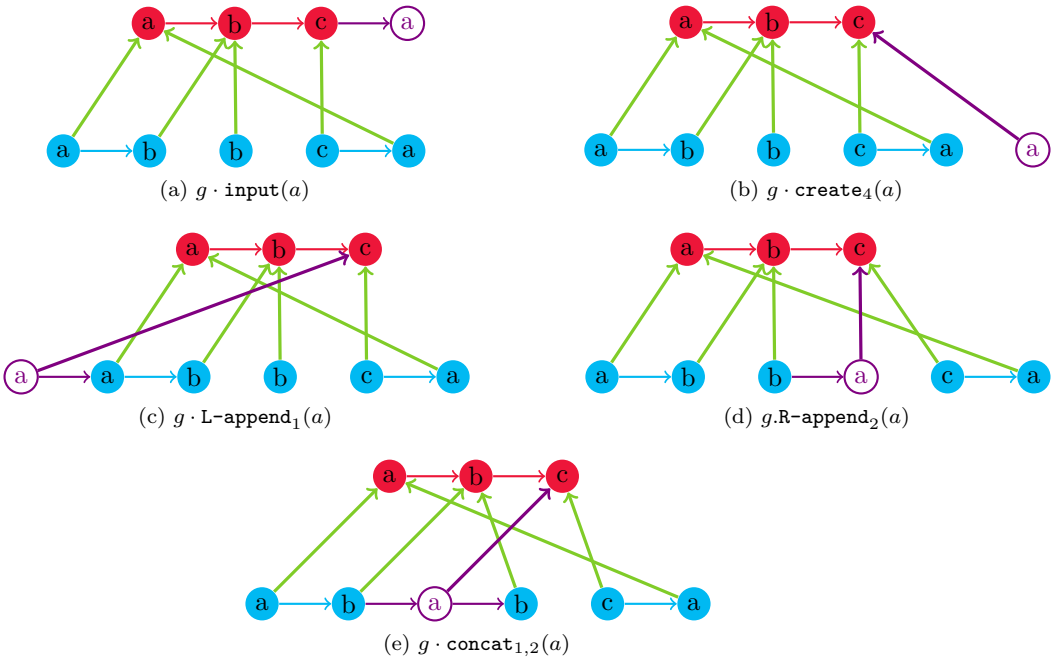


Figure 1: Operations applied to a BLOG.

On Figure 1, we represent the application of these operations to a 4-BLOG.

For a fixed positive integer k, the set of these operations is finite and denoted by Ω_k. Then, we can view a finite sequence of operations as a finite word in Ω_k^*. The set of operations given in the main part of the paper slightly differs from this one to properly match with the register operations. However, we choose here this equivalent set of operations because it is easier to use in the proofs.

Folding of a word. Given a word w in Ω_k^*, the *folding* of w, denoted $\alpha_k(w)$, is the k-BLOG obtained by successively applying the sequence of operations in w (from left to right) to the empty graph, denoted $\emptyset$. Since the operations are partially defined, so is the folding.

Let us formally define it. It is the partial function α_k from Ω_k^* to $\mathcal{H}_k$ defined inductively by $\alpha_k(\varepsilon) = \emptyset$ and $\alpha_k(w\tau) = \alpha_k(w) \cdot \tau$ for $w \in \Omega_k^*$ and $\tau \in \Omega_k$.

We say that a BLOG is k-crossing if every input position is crossed by at most k output positions. In particular, k is always greater than or equal to the number of non-empty blocks of g.

Lemma 27. *Given a positive integer k, for all k-crossing origin graphs g, there exists a word w in Ω_k^* such that $\alpha_k(w) = g$.*

Proof. We are going to prove the result by induction on the number of vertices in a k-BLOG.

H_d**:** For every k-BLOG g with at most d vertices and crossing at most k, there exists a word w in Ω_k^* such that $\alpha_k(w) = g$.

The empty graph is the folding of the empty word so H_0 is immediately true. Suppose now that H_d holds for some $d \in \mathbb{N}$ and consider a k-crossing k-BLOG g and $d+1$ vertices. Let i be the last position of the input word of g (that we are going to consider indifferently as a position or as a vertex). We proceed by cases.

Case 1 *(input).* If no position in the blocks originates in i, then the graph $g' = g - \{i\}$ (i.e., the graph g without the vertex i) is a k-crossing k-BLOG and d vertices. By induction hypothesis, there is a word w' in Ω_k^* such that $\alpha_k(w') = g'$. We conclude by observing that $g = g' \cdot \texttt{input}(a)$ where a is the label of the position i in the input word of g. Thus, $\alpha_k(w' \cdot \texttt{input}(a)) = g$.

Otherwise, there exists a vertex j labelled by some $b \in \Gamma$ which originates in i.

Case 2 *(create).* Suppose j is the unique vertex of a block of length 1, and let us denote m the index of this block. We consider the graph $g' = g - \{j\}$ which has crossing at most k and d vertices.

By induction hypothesis, there is a word w' in Ω_k^* such that $\alpha_k(w') = g'$. We conclude by observing that $g = g' \cdot \mathtt{create}_m(b)$. Thus, $\alpha_k(w' \cdot \mathtt{create}_m(b)) = g$.

Case 3 *(L-append).* Suppose j is an interface of side L of index m of a block of length greater than 1. We consider the graph $g' = g - \{j\}$ which has crossing at most k and d vertices (by removing an interface, the crossing can only decrease). By induction hypothesis, there is a word w' in Ω_k^* such that $\alpha_k(w') = g'$. We conclude by observing that $g = g' \cdot \mathtt{L\text{-}append}_m(b)$. Thus, $\alpha_k(w' \cdot \mathtt{L\text{-}append}_m(b)) = g$.

Case 4 *(R-append).* Similarly, if j is an interface of side R of index m of a block of length greater than 1, we obtain the result by using an operation $\mathtt{R\text{-}append}_m(b)$.

Case 5 *(concat).* Suppose now that we are not in one of the previous cases, i.e. j is a position in a block of index m but not an interface (and none of the interfaces originates in i). In particular, $i \geq 1$. Let us denote the block of index m by vbv' where b is the label of the position j and $v, v' \in \Gamma^*$.

Since the two interfaces of index m originate in a position smaller than or equal to $i-1$, and j originates in i, then there are at least two output positions in the block m that cross $i-1$. Since every last position of a non-empty block crosses every input position (and in particular $i-1$) and there are by hypothesis at most k positions crossing $i-1$, there is at least one block of g which is empty. Let us denote its index by n.

We consider the k-BLOG g' with same input word as g, for $\ell \notin \{n, m\}$ same block of index ℓ as g, the word v (resp. v') as block of index m (resp. n) and origin function that restricts naturally the origin function of g. g' has d vertices. Let us prove that its crossing is bounded by k. For z a position in the input word of g, if $\ell \notin \{n, m\}$, the number of output positions in the block ℓ crossing z in g' is the same as the number of output positions in the block ℓ crossing z in g. Let us prove that $c'_{z,m} + c'_{z,n} \leq c_{z,m}$ where $c'_{z,m}$ (resp. $c'_{z,n}$, $c_{z,m}$) stands for the number of output positions in the block of index m (resp. n,m) of g' (resp. g', g) crossing z. This will conclude the proof by definition of the crossing (and since the block of index n in g is empty, it does not contribute to the crossing). Consider j in the block of index m or in the block of index n in g' such that j is not the interface of side R of index m. Then, if j crosses z in g', then j was also crossing z in g. The only position we need to take care of is the interface of side R of index m. If it originates in a position greater than z then it does not cross z in g'. Otherwise, it crosses z in g'. But, in g, this position was the position $j-1$ in the block of index m. And since j originates in i that is greater than z, this position was crossing z in g. Thus, g' is also k-crossing.

By induction hypothesis, there is a word w' in Ω_k^* such that $\alpha_k(w') = g'$. We conclude by observing that $g = g' \cdot \mathtt{concat}_{m,n}(b)$. Thus, $\alpha_k(w' \cdot \mathtt{concat}_{m,n}(b)) = g$. □

B.2.2 MSO-interpretation

We first prove that the class of origin graphs is MSO-definable in the class of k-BLOG.

Proposition 28. *For all positive integers k, there exists an* MSO *formula ϕ over $\mathcal{H}_k$ such that for all g in $\mathcal{H}_k$, ϕ holds over g if and only if g is an origin graph (i.e. g is a k-*BLOG *with at most one non-empty block).*

Proof. It is sufficient to check that there is at most one non-empty block or equivalently that there are not two different positions in the blocks that have no predecessor. This is expressible in MSO. □

MSO-interpretation from Ω_k^* to $\mathcal{H}_k$. In what follows, we exhibit an MSO interpretation which is aimed to simulate α_k. The formal definition of MSO interpretation is given in Appendix A.

The following theorem is a consequence of the Backward Translation Theorem [9] combined with the Büchi, Elgot and Trakhtenbrot Theorem [7, 12, 14].

Theorem 29. *Let $\mathcal{I}$ be an* MSO *interpretation from Ω_k^* to $\mathcal{H}_k$. The pre-image by $\mathcal{I}$ of an* MSO*-definable subset of $\mathcal{H}_k$ is a regular language of Ω_k^*.*

We now define an MSO interpretation $\mathcal{I}_k$ from Ω_k^* to $\mathcal{H}_k$ which is aimed to simulate α_k.

Let w be a word of Ω_k^*. All the operations of Ω_k applied to some k-BLOG g add exactly one vertex to g and possibly link it to some (at most two) vertices of g. The MSO interpretation $\mathcal{I}_k$ will thus interpret each vertex of w labelled by the letter w_i as a vertex i of $\mathcal{I}_k(w)$ (the one created by the operation w_i) in which the edges will be defined thanks to the w_i's. In particular, if $w_i = \mathtt{input}(a)$ for some $a \in \Sigma$, then i will be a vertex of the input word of $\mathcal{I}_k(w)$, otherwise, it will be a vertex of a block.

Let us fix notations for the origin vocabulary (as defined in Section 2). It is composed of the predicates In for the edges of the input word, Out for the edges of the output word, Orig for the edges representing the origin function ($\mathrm{Orig}(x, y)$ holds if x is the origin of y), and P_a for the labelling predicates.

We define $\mathcal{I}_k$ with:

- ϕ_{P_a} ensures that each i is labelled by the letter labelling the vertex created by w_i, i.e. for all $a \in \Sigma \cup \Gamma$, $w \models \phi_{P_a}(i)$ if and only if

$$w_i \in \{\mathtt{input}(a), \mathtt{create}_m(a), \mathtt{L\text{-}append}_m(a), \mathtt{R\text{-}append}_m(a), \mathtt{concat}_{m,n}(a) \mid 1 \leq m, n \leq k\}.$$

- for positions $i < j$ in w, $\phi_{\mathrm{In}}(i, j)$ holds if i and j are neighbouring vertices in the input word, i.e. $w_i = \mathtt{input}(a)$ and $w_j = \mathtt{input}(a')$ for some a, a' in Σ, and if there is no position $i < \ell < j$ such that $w_\ell = \mathtt{input}(a'')$ for some $a'' \in \Sigma$.

- ϕ_{Orig} ensures that the origin of a vertex j which is a vertex of the output word is the rightmost vertex of the input word at the left of j. This corresponds to the definition of all the operations. In other words, $\phi_{\mathrm{Orig}}(i, j)$ holds if $i < j$, $w_i = \mathtt{input}(a)$ for some $a \in \Sigma$, $w_j \neq \mathtt{input}(a')$ for all $a' \in \Sigma$ and there is no $i < \ell < j$ such that $w_\ell = \mathtt{input}(a')$ for some $a'' \in \Sigma$.

- ϕ_{Out} defines the edges of the output word. Thus, $\phi_{\mathrm{Out}}(i, j)$ holds if i and j are labelled by letters of the output word (as seen earlier) and we are in one of the two following cases:
  - $j < i$, $w_i = \mathtt{L\text{-}append}_m(b)$, or $w_i = \mathtt{concat}_{n,m}(b)$ for some $b \in \Gamma$, $m, n \in \{1, \ldots, k\}$ and
  $$w_j \in \{\mathtt{create}_m(b'), \mathtt{L\text{-}append}_m(b') \mid b' \in \Gamma\},$$
  and there is no $j < \ell < i$ such that
  $$w_\ell \in \{\mathtt{create}_m(b'), \mathtt{L\text{-}append}_m(b'), \mathtt{concat}_{n,m}(b') \mid b' \in \Gamma, 1 \leq n \leq k\}.$$
  Informally, this means that j adds a letter at the left of a block, i adds a letter at the left of the same block, and no vertex in between does the same.
  - $i < j$, $w_j = \mathtt{R\text{-}append}_m(b)$ on $w_j = \mathtt{concat}_{m,n'}(b)$ for some $b \in \Gamma$, $m, n' \in \{1, \ldots, k\}$ and there exists $n \in \{1, \ldots, k\}$ such that
  $$w_i \in \{\mathtt{create}_n(b'), \mathtt{R\text{-}append}_n(b') \mid b' \in \Gamma\}$$
  and there exists a set of positions $i = i_0 < i_1 < i_2 < \ldots < i_s < i_{s+1} = j$ such that for all $1 \leq z < s$, $w_{i_z} = \mathtt{concat}_{m_z, m'_z}(b')$ for some $b' \in \Gamma$, $m_z, m'_z \in \{1, \ldots, k\}$ such that $m_z = m'_{z+1}$, $m'_1 = n$ (denoted m_0 in what follows), $m_s = m$ and for $0 \leq z \leq s$, there is no $i_z < \ell < i_{z+1}$ such that w_ℓ belongs to:
  $$\{\mathtt{create}_{m_z}(b'), \mathtt{R\text{-}append}_{m_z}(b'), \mathtt{concat}_{m_z,m'}(b'), \mathtt{concat}_{m',m_z}(b') \mid b' \in \Gamma, 1 \leq m' \leq k\}.$$
  Informally, this means that i adds a letter at the right of a block, j adds a letter at the right of the same block, and no vertex in between does the same. As the numbering of a block can change if it is appended to another block at its left, the formula needs to keep track of that.

All of these conditions are expressible in the logic MSO over Ω_k^*.

Proposition 30. *Given a positive integer k, for all words w in Ω_k^*, $\mathcal{I}_k(w) = \alpha_k(w)$.*

Proof. The proof is made by induction on the length of w as a direct consequence of the definition of $\mathcal{I}_k(w)$. □

With the two last sections, we are now able to prove the following Lemma.

Lemma 31. *Given a positive integer k, for any* MSO*-definable k-crossing origin transduction $\mathcal{G}$, there is a regular language $L_\mathcal{G}$ over Ω_k such that α_k defines a surjective mapping from $L_\mathcal{G}$ into $\mathcal{G}$.*

Proof. As $\mathcal{G}$ is k-crossing, Lemma 27 shows that for any $g \in \mathcal{G}$, there is a $w \in \Omega_k^*$ such that $\alpha_k(w) = g$. Theorem 29 gives us the existence of a regular language $L_\mathcal{G}$ as the inverse image of $\mathcal{G}$ by the interpretation $\mathcal{I}_k$. By Proposition 30, we get α_k and $\mathcal{I}_k$ coincide over Ω_k^*, and thus that $\alpha_k(L_\mathcal{G}) = \mathcal{G}$, which concludes the proof. □

B.2.3 εNSST and k-blog

The purpose now is, given a MSO-definable k-crossing set of origin graphs $\mathcal{G}$, to construct a εNSST $\mathcal{A}$ with k registers such that $\mathcal{G}$ is recognised by $\mathcal{A}$. We thus use k-BLOGs as a way to describe the configurations of a εNSST .

***k*-blog origin semantics.** Consider a εNSST with k registers $\{r_1, \ldots, r_k\}$ and a run ρ with input word u ending in q with register valuation ν. The *origin-block application associated with* ρ, denoted by OriginB_ρ, is an application from $\cup_{1 \leq m \leq k} \{m\} \times \{0, \ldots, |\nu(r_m)| - 1\}$ to $\{0, \ldots, |u| - 1\}$, defined as follows: given a register r_m, a position j of the word $\nu(r_m)$, labelled by a letter a, is associated with the position in u where the input head was when a was created.

Definition 32. *Given a* εNSST *with registers* $\{r_1, \ldots, r_k\}$ *and a run* ρ *with input word* u *ending in* q *with register valuation* ν*, the* k-BLOG origin semantic *of* ρ *is the* k-BLOG *with input word* u*, blocks* $\nu(r_1)$*,* $\nu(r_2), \ldots, \nu(r_k)$ *and origin function* OriginB_ρ*.*

Construction of a εNSST with a given origin semantics. Let k be a positive integer. Let $\mathcal{G}$ be an MSO-definable, k-crossing set of origin graphs. The aim of this section is to prove the following proposition:

Proposition 33. *There is a* εNSST $\mathcal{A}_\mathcal{G}$ *with* k *registers that recognises* $\mathcal{G}$*.*

First, by Lemma 31, there exists a regular language $L_\mathcal{G}$ over Ω_k such that $\mathcal{I}_k$ defines a surjective mapping from $L_\mathcal{G}$ to $\mathcal{G}$. Then, there exists a complete and deterministic finite automaton $\mathcal{B}_\mathcal{G}$ over Ω_k recognising $L_\mathcal{G}$.

We can now define $\mathcal{A}_\mathcal{G}$ with underlying automaton $\mathcal{B}'_\mathcal{G}$, obtained from $\mathcal{B}_\mathcal{G}$ by replacing letters $\texttt{input}(a)$ with a and other letters with ε, and a fresh final state f which is the only one. Moreover, we add an ε-transition from any state final in $\mathcal{B}_\mathcal{G}$ to f.

We take the set of registers $\mathcal{R} = \{r_1, \cdots, r_k\}$ with r_1 the output register. Given a transition of $\mathcal{B}_\mathcal{G}$ labelled by $\tau \in \Omega_k$, the register update associated with the corresponding transition in $\mathcal{B}'_\mathcal{G}$ is:

$$\begin{cases} \varepsilon & \text{if} \quad \tau = \texttt{input}(a) \\ r_m \mapsto b & \text{if} \quad \tau = \texttt{create}_m(b) \\ r_m \mapsto b r_m & \text{if} \quad \tau = \texttt{L-append}_m(b) \\ r_m \mapsto r_m b & \text{if} \quad \tau = \texttt{R-append}_m(b) \\ r_m \mapsto r_m b r_n, r_n \mapsto \varepsilon & \text{if} \quad \tau = \texttt{concat}_{m,n}(b) \end{cases}$$

for some $a \in \Sigma$, $b \in \Gamma$ and $i, j \in \{1, \ldots, k\}$. The transitions ending in f have $r_1 \mapsto r_1 \cdots r_k$, and $r_i \mapsto \varepsilon$ for $i \neq 1$ as register updates.

Since $\mathcal{B}_\mathcal{G}$ is complete and deterministic, there is a bijection h from $L_\mathcal{G}$ to the successful runs of $\mathcal{A}_\mathcal{G}$. By construction, as such a run ends by an update appending all the registers in the output register, we have that for every $w \in L_\mathcal{G}$, $\alpha_k(w)$ is the origin graph associated with $h(w)$.

Proof of Theorem 18. Let $g \in \mathcal{G}$, there is a word $w \in L_\mathcal{G}$ such that $g = \alpha_k(w)$. Moreover by construction, $\alpha_k(w)$ is the origin graph associated with the successful run $h(w)$. Thus $\mathcal{G}$ is included in the origin semantics of $\mathcal{A}_\mathcal{G}$.

Conversely, let ρ be a successful run of $\mathcal{A}_\mathcal{G}$ with associated origin graph g. There is a word $w \in L_\mathcal{G}$ such that $\rho = h(w)$. Then, since $w \in L_\mathcal{G}$, $\alpha_k(w)$ is defined and belongs to $\mathcal{G}$. Moreover $\alpha_k(w) = g$. Then $g \in \mathcal{G}$.

C Proofs of Theorem 10 and 16

This section shows how Theorems 10 and 16 may be obtained from Theorem 18.

C.1 Proof of the only-if directions of Theorems 10 and 16

Proposition 34. *The origin transduction recognised by a streaming string transducer has bounded origin and is* MSO*-definable. It is furthermore functional if the streaming string transducer is unambiguous.*

Proof. Suppose that $\mathcal{A}$ is a streaming string transducer. Let ℓ denote the largest number of create operations appearing in a transition. Then, for all runs and all positions i of the input word, at most ℓ positions of the output word can originate in i and so the origin string-to-string transduction recognised by $\mathcal{A}$ has bounded origin. Furthermore, we have already shown in Proposition 9 that it is MSO-definable

Finally, if $\mathcal{A}$ is unambiguous, for every input word $u \in \Sigma^*$, there is at most one successful run ρ having u as input word. Thus, there is only one graph in the origin semantics of $\mathcal{A}$– the one associated with ρ – having u as input word. Thus it is functional. □

C.2 Proof of the if direction of Theorem 10

The following proposition together with Theorem 18 concludes the proof of Theorem 10.

Proposition 35. *Let $\mathcal{G}$ be an origin transduction recognised by a εNSST . If $\mathcal{G}$ has bounded origin, then $\mathcal{G}$ is recognised by a streaming string transducer (without ε-transitions).*

Proof. We start with a streaming string transducer with ε-transitions $\mathcal{A}$ recognising $\mathcal{G}$ which is supposed m-origin bounded. Without loss of generality, we can suppose that in all successful runs of $\mathcal{A}$, a letter created on a register is used in the final output word, i.e., no created letter is erased or left in some non-output register at the end of the computation. Indeed, this property can be obtained by maintaining the information of which register is empty along the run and guessing which register will be used at the end of the computation, when some letter is created in an empty one. Moreover, using a counter of size m, which is resetted every time a non-ε-transition is performed, we may ensure that no more than m letter originate in a position, in any block origin graph which describes the origin valuation of the registers after some partial computation that started from the empty valuation.

Therefore, there are only finitely many register updates (a particular function $\mathcal{R} \to (\mathcal{R} \cup \Gamma)^*$ as defined in Section 2) associated with ε-runs (i.e., partial runs performing ε-transitions only). Indeed, the updates are copyless and at most m letters are appended on some registers. Thus, we can find a bound N such that, every pair (u, v) is realised through a successful run which contains no ε-run of length greater than N. We can furthermore syntactically bound these ε-runs, by using a counter of size N that is reseted every time a non-ε-transition is performed.

We conclude by observing that this simplified device admits only finitely many ε-runs whose associated register updates can therefore be shrinked and appended to the register update associated with the last non-ε-transition performed. Doing so, we obtain a non-deterministic streaming string transducer recognising $\mathcal{G}$. □

As a direct consequence of Proposition 9, we also obtain the MSO definability of the origin transductions considered in the previous proposition.

Corollary 36. *If the origin semantics of a streaming string transducer with ε-transitions has bounded origin, then it is* MSO*-definable.*

C.3 Proof of the if direction of Theorem 16

Concerning the functional case, the end of the proof of Theorem 16 comes from the following proposition.

Proposition 37. *Every streaming string transducer with k registers recognising a functional origin transduction is effectively equivalent to an unambiguous streaming string transducer with k registers recognising the same origin transduction.*

Proof. Our proof is based on the classic construction which transforms an automaton into an equivalent unambiguous one. Since not only acceptance but also the sequence of transitions should be simulated in order to be able to extend the construction for transducers in which register updates label the transitions (Definition 2), we adapt the classical powerset construction to our purpose. The simulation works in such a way that each successful run of the simulating automaton, corresponds, by projection, to a (unique) successful run of the simulated one. This recovered successful run is the smallest one, for some lexicographical order on runs obtained from an arbitrary fixed ordering on states.

Let $\mathcal{B}$ be an automaton over some alphabet Σ, with set of states Q and set of transitions Δ, and let $<$ be a total order on Q. Our simulating automaton $\mathcal{B}'$ uses the extended set of states $\{(q, X) \mid q \in \mathrm{Q},\ X \subseteq \mathrm{Q} \setminus \{q\}\}$. Intuitively, the first component will recover the successive states of a

simulated run, while the second component will store the set of states that can be reached through a smaller run. Formally, the set of transition Δ' of $\mathcal{B}'$ is defined as follows. For each $a \in \Sigma$, each states q and q' in Q and each subsets $X \subseteq \mathrm{Q} \setminus \{q\}$ and $X' \subseteq \mathrm{Q} \setminus \{q'\}$, $((q, X), a, (q', X'))$ is a transition in Δ' if and only if

- (q, a, q') belongs to Δ;
- for no $q_- \in X$, (q_-, a, q') belongs to Δ;
- $X' = \{q_+ \mid (q_-, a, q_+) \in \Delta \text{ for some } q_- \in X\} \cup \{q_+ \mid q_+ < q',\ (q, a, q_+) \in \Delta\}$.

Finally, a state (q, X) is final in $\mathcal{B}'$ if and only if, q is final and no state in X is final in $\mathcal{B}$. Observe that X' is deterministically defined from q, X and q', that is, the only non-determinism is used in the choice of q' only. By construction, $\mathcal{B}'$ is unambiguous and equivalent to $\mathcal{B}$.

More precisely, define the projection of a transition $\tau = ((q, X), a, (q', X')) \in \Delta'$ into Δ as the transition $\pi(\tau) = (q, a, q')$. This projection extends to runs (i.e., sequences of transitions). Given a successful run of $\mathcal{B}'$, its projection is a successful run of $\mathcal{B}$, and hence its label is a word of the recognised language. Conversely, the smallest run of $\mathcal{B}$ over some accepted input word is the projection of the unique successful run of $\mathcal{B}'$ over this input.

Finally, we extend the construction to transducers by defining the label of a transition

$$((q, X), a, (q', X')) \in \Delta'$$

as being equal to those of the transition $(q, a, q') \in \Delta$. The so obtained transducer is clearly an unambiguous equivalent transducer, which furthermore preserves the number of register names and the origin informations. □

In order to complete the proof of Theorem 16, we need to prove that, in the functional case, the condition on bounded origin becomes superfluous. We first prove an intermediate result.

Lemma 38. *If an origin graph is k-crossing, then for every input position i, there are at most $2k$ maximal factors of the output whose positions all originate in input position i. The bound is tight.*

Proof. For an integer k, a simple example of k-crossing origin graph that have $2k$ such factors for some input position i, is the origin graph given by the triple $\big(abc, (cbaabc)^k, f\big)$ where f is the unique origin mapping which is letter-preserving. Indeed, each input position is crossed by at most k output positions, and the b's of the output word form $2k$ maximal factors that all originate in the second input position.

Now consider a k-crossing origin graph g with output word v, and let i be an input position of g. We denote by n the number of maximal factors of v whose positions all originate in input position i. We want to prove $n \leq 2k$.

If $i = 0$ (i.e., i is the leftmost input position), then the result follows from the boundedness of the crossing of g in i, indeed $n \leq k$.

From now on, we suppose $i > 0$. Suppose that each output position is coloured according to whether it originates left of i (colour 1), in i (colour 2) or right of i (colour 3). This defines a factorisation of v, where each factor is monochrome and maximal with respect to this property. The number of 2-coloured factors is exactly n. Since every two successive 2-coloured factor are separated by at least one factor of different colour, we can bound n by bounding the number of 1- and 3-coloured factors.

The rightmost position of a 1-coloured factor crosses $i - 1$, so there are at most k such factors. Similarly, the predecessor of the leftmost position of a 3-coloured factor, whenever it exists, crosses i, and therefore there are at most $k + 1$ such factors (where $k + 1$ is reached only if the first factor of v is 3-coloured). We proceed by cases:

Case 1 If the first and the last factor of v are 2-coloured, then the rightmost position of the last factor crosses i. Thus, the bound on the number of 3-coloured factors becomes $k - 1$. Hence, the number of 2-coloured factors is bounded by $k + (k - 1) + 1 = 2k$.

Case 2 Otherwise, if the last factor of v is 2-coloured, then its rightmost position crosses i. Thus there are at most k 3-coloured factors. Hence, the number of 2-coloured factors is bounded by $k + k = 2k$.

Case 3 Otherwise, the number of 2-coloured factors is bounded by $k + (k + 1) - 1 = 2k$.

In all the cases, we have $n \leq 2k$. □

Now, we prove the origin-boundedness of MSO-definable k-crossing functional origin transductions.

Proposition 39. *If a functional origin transduction is* MSO*-definable and has bounded crossing, then it has bounded origin.*

Proof. Let $\mathcal{G}$ be a class of origin graphs MSO-definable, k-crossing and functional. There is a MSO-formula ϕ such that $g \in \mathcal{G}$ if and only if ϕ holds on g. Thus, there is a bound ℓ (depending only on ϕ) such that, for all origin graphs g with input word u, output word vwz such that $|w| \geq \ell$ and all the positions in w originate in the same position of u, there exists an origin graph g' with input word u, output word $vw'z$ with $|w'| < |w|$ such that ϕ holds on g' if and only if ϕ holds on g.

Suppose now that $\mathcal{G}$ doesn't have bounded origin. Then there exist $g \in \mathcal{G}$ with input word u, output word v and a position i of u with at least $2k\ell$ pre-images by the origin mapping. Since the crossing in g is at most k, there are at most $2k$ maximal factors of v with all positions originating in i, by Lemma 38. Thus, at least one of them has size at least ℓ. So there exist a graph g' with input word u, output word $v' \neq v$ such that ϕ holds on g' and therefore $g' \in \mathcal{G}$. This contradicts the functionality of $\mathcal{G}$. □

D Bounded path width origin transductions

The goal of this section is to prove Proposition 14. First, we briefly define path width of graphs. For a complete description of path width, we refer to the literature, e.g. [10]. Then we prove the result, which is basically a corollary of Lemma 27.

The path width of a graph measures how far from a simple path is the graph. A small path width means that the graph is close to being a path, e.g. a cycle, while a large path width means that the graph is far from being a path, e.g. cliques. The measure is made thanks to the notion of path decomposition.

Definition 40 (Path decomposition and path width)**.** *A* path decomposition *of a graph (V, E) is a sequence $\mathcal{P} = X_1, X_2, \dots X_r$ of subsets of V, called* bags*, such that the following holds:*

- $\bigcup_{i=1}^{r} X_i = V$*, i.e., every vertex of the graph is in at least one bag of $\mathcal{P}$;*
- *for every $(v, v') \in E$, there exists $i \in \{1, \dots, r\}$ such that the bag X_i contains both the vertices v and v';*
- *for every $v \in V$, if $v \in X_i \cap X_j$ for some $1 \leq i \leq j \leq r$, then $v \in X_k$ for each k such that $i \leq k \leq j$.*

The width *of a path decomposition is the cardinality of the largest bag minus* 1*. The* path width *of the graph is the minimum width, over all its path decompositions.*

We now prove Proposition 14. We use tools developed in Appendix B.2.1, namely BLOGs, folding and sided interfaces of BLOGs. Recall that a connection between k-crossing bounded graphs and words (i.e., paths) is already given by Lemma 27. We will use this results in order to find a path decomposition of any k-crossing graph, which has width $2k + 1$.

Proposition 14. *Every bounded crossing origin transduction has bounded path width.*

Proof. Let g be a k-crossing origin graph. By Lemma 27, there exists a word $w = \sigma_1 \cdots \sigma_\ell$ over the alphabet Ω_k such that $g = \alpha_k(w)$. We fix such a w. Each prefix $\sigma_1 \cdots \sigma_i$ of w defines a k-BLOG, denoted g_i, which corresponds to its folding, i.e., $g_i = \alpha_k(\sigma_1 \cdots \sigma_i)$. In particular, $g_{|w|}$ is equal to g and for each $0 < i \leq \ell$, $g_i = g_{i-1} \cdot \sigma_i$ with the convention that g_0 is the empty k-BLOG.

Recall that an interface of side L (resp. R) and index m of a BLOG is the leftmost (resp. rightmost) vertex of its m-th block, whenever it exists. For each i, with $0 < i \leq \ell$, we define the bag:

$$X_i = \left\{ v \;\middle|\; \begin{array}{l} v \text{ is the rightmost input position in } g_{i-1} \\ \text{or } v \text{ is an interface in } g_{i-1} \\ \text{or } v \text{ is the vertex created by the operation } \sigma_i \end{array} \right\}.$$

Observe that the size of each X_i is bounded by $2k + 2$, since there are at most $2k$ interfaces. The sequence of X_i's forms a path decomposition of g of width $2k + 1$. Indeed,

- each vertex is created by some σ_i and therefore belongs to the corresponding X_i (let us call this i the index of creation of the vertex);
- for each edge (v, v'):
  - if the edge is an input (resp. origin) edge, then necessarily the index i of creation of v' (resp. v) is greater than those of v (resp. v') which is the rightmost input vertex in g_{i-1}, and therefore both v and v' belong to X_i;
  - if the edge is an output edge and if the index i of creation of v' (resp. v) is greater than those of v (resp. v'), then both v and v' belong to X_i, because v (resp. v') should be an interface of side R (resp. L) in g_{i-1}.
- finally, the third condition on path decompositions is a direct consequence of the following fact: for each vertex, the property of being either an interface or the rightmost input vertex, defines an interval in the sequence $g_1, \dots, g_{\ell-1}$ of k-BLOGs. □

The path decomposition we obtain in the proof satisfies the following property: there exists a sub-sequence of distinct bags $Y_1, \dots, Y_{n-1}$, where n is the length of the input word of the origin graph, such that for each $0 < i < n$, Y_i is the only bag which contains the input edge connecting the i-th to the $(i+1)$-th input vertex. In particular these edges occur in the order given by the input word. We can prove the opposite direction, namely, if an origin graph admits such a path decomposition with bounded width, then its crossing is bounded.

E Intersections of classes of origin transductions

The following result shows that the non-determinism of two-way transducers and the one of streaming string transducers are of completely different nature. Indeed, while the former allows unboundedness of computations and therefore of image length, the latter is a simple common guess.

Proposition 41. *The class of origin transductions having bounded origin which are recognised by a non-deterministic two-way transducer with common guess is exactly the class of origin transductions recognised by a non-deterministic streaming string transducer.*

Proof. We prove the more general fact that every origin transduction admitting a finite number of images for every input word and recognised by a non-deterministic two-way transducer is also recognised by a non-deterministic streaming string transducer. In particular bounded origin transductions have that property.

Consider a non-deterministic two-way transducer $\mathcal{T}$ recognising such an origin transduction. As it admits a finite number of images for each word, there is no producing loop in a successful run, i.e., there is no sub-run producing a non-empty output which starts and ends in the same position and same state (otherwise by iterating the loop, the same input word would have an infinite number of images). The non-producing loops (i.e., loops that produce the empty word) can also be removed from the successful runs, so that for all g in the origin transduction, there is a loop-free successful run of $\mathcal{T}$ recognising g. By loop-freeness, on such a run, the reading head scans a given input position only a bounded number of times (bounded by the number of states). Thus, this transduction is recognised by a machine that first guesses a successful run (by colouring the positions by the bounded ordered sequence of states and the transitions associated taken by the run), and then performs a deterministic two-way transducer.

Moreover, the class of origin transductions recognised by a deterministic two-way transducer with common guess is equal to the class of origin transductions recognised by non-deterministic streaming string transducers [3]. □

We now prove that the MSO-definable origin semantics of streaming string transducers with ε-transitions, or equivalently, MSO-definable bounded-crossing origin transductions, are also origin semantics of two-way non-deterministic transducer with common guess.

Proposition 42. *If an origin transduction is* MSO*-definable and has bounded crossing, then it is the origin semantics of a two-way transducer with common guess.*

Proof. We fix a k-crossing MSO-definable origin transduction $\mathcal{G}$ with input alphabet Σ and output alphabet Γ. The main idea of the proof is to show that $\mathcal{G}$ is the origin semantics of the composition of a streaming string transducer (without ε-transition) and a one-way non-deterministic finite transducer. Indeed, the first model is equivalent to two-way deterministic transducers with common guess, which is a particular case of its non-deterministic counterpart, which is itself closed under composition to the right with one-way non-deterministic transducers.

For an origin graph g and an input position i, define an i-factor as being a maximal factor of the output word of g such that all its positions originate in i. Since $\mathcal{G}$ is MSO-definable, there exist finitely many disjoint (non empty) regular languages $L_1, \dots, L_\ell$, such that in each $g \in \mathcal{G}$, for each input position i of g and every i-factor v, there exists j, called *the index of the i-factor v*, such that v belongs to L_j, and furthermore, replacing v by any word from L_j (with all positions originating in i) yields an origin graph in $\mathcal{G}$.

Given $g \in \mathcal{G}$, define the *factor-shrinking* of g as being the origin graph with input alphabet Σ and output alphabet $\{1, \dots, \ell\}$ obtained from g by replacing every i-factor by a single vertex originating in i and labelled by its index. Let $\mathcal{G}'$ be the set of the factor-shrinking of all origin graphs in $\mathcal{G}$. Clearly, there exists an MSO transduction which transforms each shrinked graph into one of its pre-image by factor-shrinking[4]. Therefore, using the Backward Translation Theorem, we obtain that $\mathcal{G}'$ is MSO-definable. Furthermore, by Lemma 38, the number of i-factors in an origin graph of $\mathcal{G}$ is bounded by $2k$ and thus $\mathcal{G}'$ has bounded-origin. Finally, it is clear that factor-shrinking preserves the crossing, and therefore $\mathcal{G}'$ is k-crossing as $\mathcal{G}$. Hence, it is the origin semantics of a streaming string transducer $\mathcal{A}$, by Theorem 10.

Finally, observe that replacing each letter j of a word in $\{1, \dots, \ell\}$ by a word in L_j, is doable by a one-way non-deterministic transducer (using ε-transitions that simulate an automaton accepting L_j on the output), in such a way that the origin of a so-created word in L_j is the origin of the corresponding letter j. Therefore, $\mathcal{G}$ is the origin semantics of the composition of $\mathcal{A}$ with $\mathcal{A}'$. □

Proposition 43. *The origin semantics of a two-way non-determinisitc transducer with common guess is* MSO*-definable.*

Proof. For a given transducer, we exhibit an MSO formula over origin vocabulary, such that an origin graph satisfies the formula if and only if it is in the origin semantics of the transducer. Without loss of generality, we suppose that the transducer has no ε-transition and that each transition produces at most one letter. Furthermore, we do not consider common guess, since it is an operation which is MSO-definable by definition.

The MSO formula first guesses the states from which transitions producing the letters of the output word are taken. The output positions are coloured accordingly to these states. This colouring gives the skeleton of the successful run that we aim to recover. Indeed, the so guessed sequence of states (taken in the output order) is a sub-sequence of the sequence of states corresponding to the simulated run, since two-way transducers generate the output from left to right.

For each two consecutive positions j and j' of the output word, respectively labeled by q and q', we are able to check within MSO that there exists a run of the transducer: (1) starting from the origin of j in state q; (2) ending in the origin of j' in state q'; (3) and which has an empty associated production. Indeed, if such a run exists, then there exists one which has no loop and therefore, which visits each position a bounded number of times. It is then possible to guess the space/time diagram of such a run in MSO and check that it is valid. Finally, we similarly check within MSO that the configuration in the first (resp. last) output letters (i.e., the origin of the first (resp. last) output letter and the associated state) is reachable from an initial configuration (resp. can reach an accepting configuration) by a run with empty production. □

[4]It is indeed not possible to produce all the pre-image, since MSO transduction can make only finitely many copies of vertices. However, by choosing a smallest word in each language L_j, one can produce a pre-image.