Which Classes of Origin Graphs Are Generated by Transducers?*

Mikołaj Bojańczyk[1], Laure Daviaud[2], Bruno Guillon[3], and Vincent Penelle[4]

1 University of Warsaw, Warsaw, Poland
`mbojanczyk@mimuw.edu.pl`
2 University of Warsaw, Warsaw, Poland
`ldaviaud@mimuw.edu.pl`
3 University of Warsaw, Warsaw, Poland
`guillonb@mimuw.edu.pl`
4 University of Warsaw, Warsaw, Poland
`penelle@mimuw.edu.pl`

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.

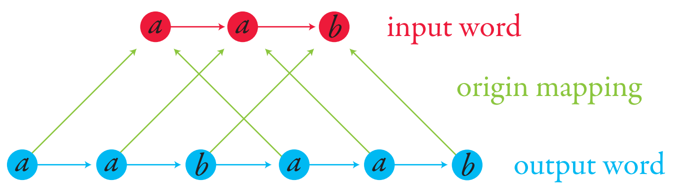
1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Streaming String Transducers, Origin Semantics, String-to-String Transductions, MSO Definability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.114

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 $w \mapsto ww$ 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:



* 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 No 683080).

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 [5], where it was shown that existing models of transducers, such as two-way transducers (also called two-way automata with outputs, e.g. [11]), 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 [5]. We aim 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 nondeterministic 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 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 nondeterministic streaming string transducer if and only if it has three properties: (1) it is MSO-definable as a set of coloured graphs; (2) it has bounded degree; and (3) 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 [5]. We believe however that origin graphs are a more intuitive and visual notion than the factorised words used in [5]. 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. Hence, this paper can be seen as a natural complement to the results form [5], or possibly a clearer picture. Furthermore, we study more general models than [5], in particular we allow nondeterminism and ε-transitions.

Important related work is the paper [10], which proposes to use logic to describe properties of origin graphs (they use the name *productions*). In [10], the authors ask 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 transducer. In [10], the logic used to express properties of origin graphs is a strict fragment of MSO, called $\mathcal{L}_T$, a type of two-variable logic. Therefore, our Theorem 6 is stronger than the model-checking result mentioned in [10, Section VI]. The reason why [10] 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 satisfiability when evaluated on the class of all origin graphs, contrary to MSO.

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

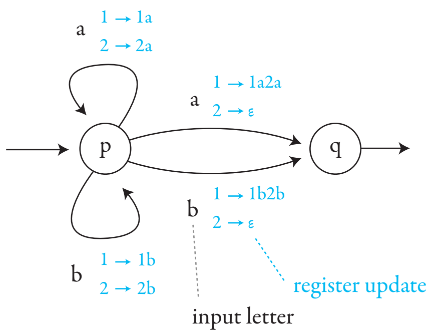
The main topic of this paper is the class of string-to-string transductions recognised by streaming string transducers [1]. Another equivalent presentation of this class is MSO string-to-string transductions, or a suitably defined nondeterministic version of two-way automata with output. We will mainly 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 nondeterminism, 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 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 nondeterministic 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.

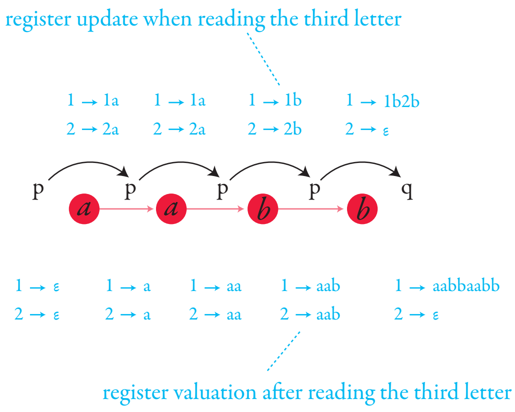


1 Our syntax for streaming string transducers is different than the one used in [1], but it is routine to show that the expressive power is the same, as long as we allow nondeterminism.

The automaton uses nondeterminism to guess the last position so that the two registers are concatenated, however every nonempty 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.

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:

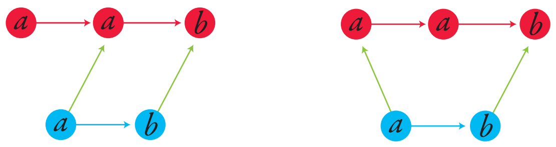
- the universe is the disjoint union of positions in w and positions in v;
- there are two binary predicates for the successor relations in w and v;
- there is a binary predicate, called the *origin mapping*, which is a total function from output positions to input positions;
- for each $a \in \Sigma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

[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].

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 nondeterministic, 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 nondeterministic 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 nondeterministic MSO transduction, which can be done while preserving origin semantics [5][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 [5] 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 nondeterministic 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 nondeterministic transducer (such decidability, and even a polynomial time algorithm, although with inputs represented in a different way, was already given in [5]).

▶ **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.*

[3] In [5] the conversion is done in the deterministic case, but it can be easily extended to the nondeterministic case.

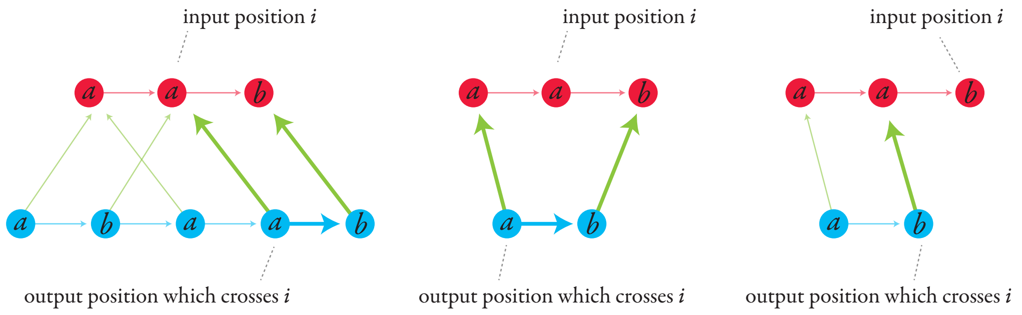
Note that the converse of the above proposition is false, even assuming that there is a bound on the number output positions which can originate in the same input position, see Examples 13 and 15. The issue is that it is more difficult to produce an origin graph (using the origin semantics of an MSO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

4 Which sets of origin graphs are recognised by SSTs?

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 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:



▶ **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 $\mathcal{G}$.*

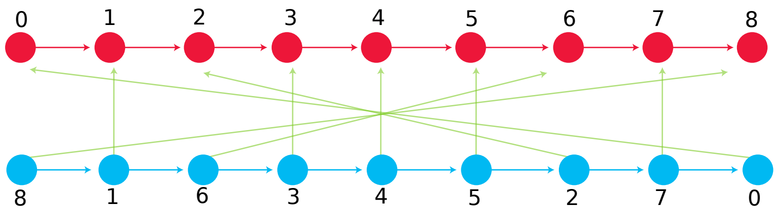
The theorem allows one to decide if a given SST can be implemented with fewer registers preserving origin semantics. This does not help with the resource minimisation problem from [4], because in [4] origin information can be be changed in the minimisation process.

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 $0, \dots, 2n$ 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):

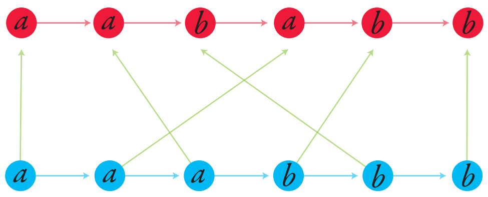
1. the origin of the first output position is the last input position;
2. the origin of the second output position is the second input position;
3. if $j > 1$ is an odd output position with origin i, then the origin of $j - 2$ is $i - 2$;
4. if $j > 1$ is an even output position with origin i, then the 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 [6] 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 [5]. 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

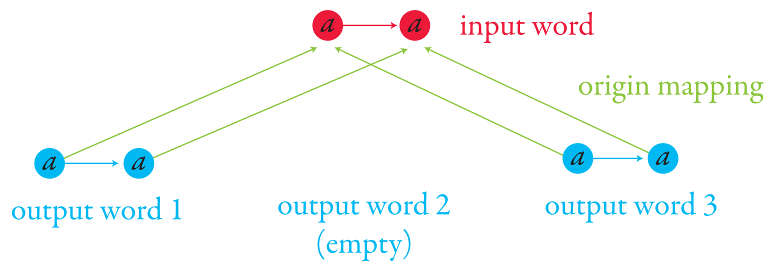
To prove Theorems 10 and 16, we first characterise the origin graphs which can be generated by streaming string transducers with ε-transitions. A streaming string transducer with ε-transitions is the generalisation of the model described in Definition 2, where ε-transitions are allowed in the underlying 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 nondeterministic 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

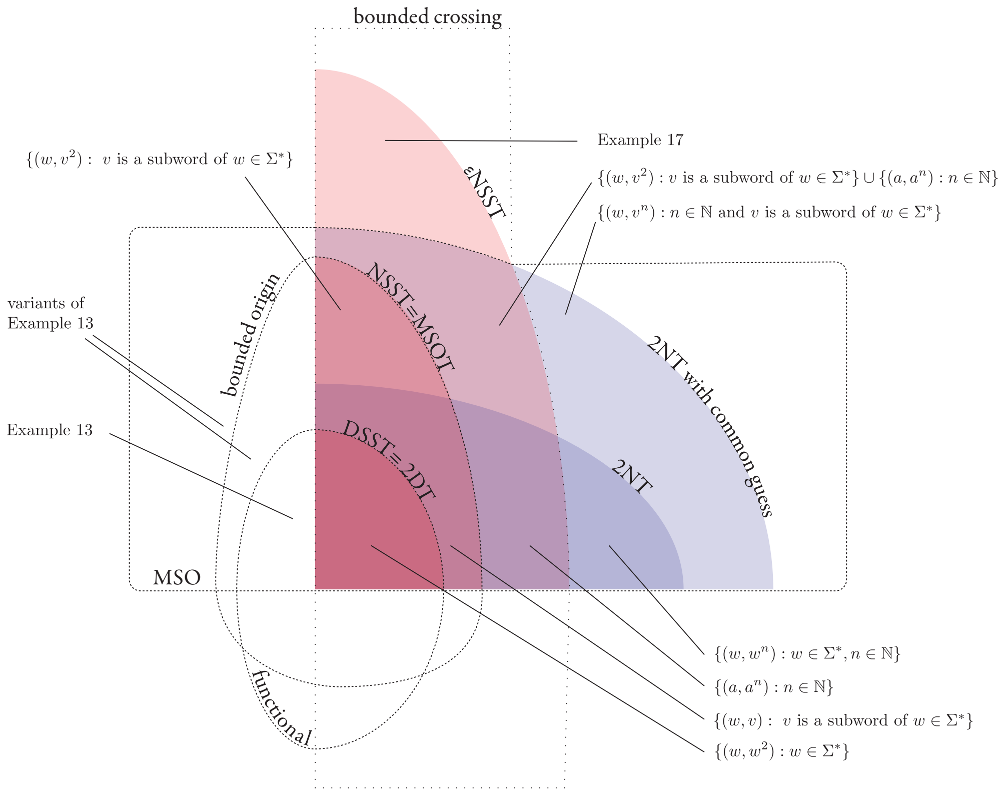
Our main contribution is a characterisation of the origin semantics of streaming string transducers (deterministic; nondeterministic; with ε-transitions providing MSO-definability), using properties of the origin graphs such as functionality, origin boundedness, crossing boundedness and MSO-definability. 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. nondeterministic, nondeterministic 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 nondeterministic 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 [11, 1] (resp. nondeterministic streaming string transducers [3]), even when considering the origin semantics [5].

Finally, we denote by MSOT the family of origin transductions recognised by a nondeterministic MSO-transduction. It is equal to NSST [3, 11]. The transductions recognised by deterministic MSO-transductions are the same as for DSST [1], this remains true with origin semantics [5].

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


bounded crossing
Example 17
$\{(w, v^2) :\ v \text{ is a subword of } w \in \Sigma^*\}$
$\{(w, v^2) : v \text{ is a subword of } w \in \Sigma^*\} \cup \{(a, a^n) : n \in \mathbb{N}\}$
$\{(w, v^n) : n \in \mathbb{N} \text{ and } v \text{ is a subword of } w \in \Sigma^*\}$
εNSST
NSST=MSOT
bounded origin
variants of Example 13
Example 13
DSST=2DT
2NT
2NT with common guess
MSO
functional
$\{(w, w^n) : w \in \Sigma^*, n \in \mathbb{N}\}$
$\{(a, a^n) : n \in \mathbb{N}\}$
$\{(w, v) :\ v \text{ is a subword of } w \in \Sigma^*\}$
$\{(w, w^2) : w \in \Sigma^*\}$

References

1 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, pages 1–12, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.1.

2 Rajeev Alur and Loris D'Antoni. Streaming tree transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 42–53. Springer, 2012. URL: http://link.springer.com/chapter/10.1007/978-3-642-31585-5_8.

3 Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 1–20. Springer, 2011. URL: `http://link.springer.com/10.1007%2F978-3-642-22012-8_1`.

4 Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. Minimizing resources of sweeping and streaming string transducers. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

5 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. `doi:10.1007/978-3-662-43951-7_3`.

6 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. `doi:10.1145/2933575.2934508`.

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. `doi:10.1002/malq.19600060105`.

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. `doi:10.1016/0304-3975(91)90387-H`.

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

10 Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Decidable logics for transductions and data words. *CoRR*, abs/1701.03670, 2017. URL: `http://arxiv.org/abs/1701.03670`.

11 Joost Engelfriet and Hendrik Jan Hoogeboom. Mso definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001. `doi:10.1145/371316.371512`.