# Regular Resynchronizability of Origin Transducers Is Undecidable

**Denis Kuperberg**
CNRS, LIP, ENS Lyon, France
denis.kuperberg@ens-lyon.fr

**Jan Martens**
Eindhoven University of Technology, The Netherlands
j.j.m.martens@tue.nl

## Abstract

We study the relation of containment up to unknown regular resynchronization between two-way non-deterministic transducers. We show that it constitutes a preorder, and that the corresponding equivalence relation is properly intermediate between origin equivalence and classical equivalence. We give a syntactical characterization for containment of two transducers up to resynchronization, and use it to show that this containment relation is undecidable already for one-way non-deterministic transducers, and for simple classes of resynchronizations. This answers the open problem stated in recent works, asking whether this relation is decidable for two-way non-deterministic transducers.

**2012 ACM Subject Classification** Theory of computation → Transducers

**Keywords and phrases** transducers, origin, resynchronisation, MSO, one-way, two-way, undecidability

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2020.58

**Related Version** `https://arxiv.org/abs/2002.07558`.

## 1 Introduction

The study of transductions, that is functions and relations from words to words, is a fundamental field of theoretical computer science. Many models of transducers have been proposed, and robust notions such as *regular transductions* emerged [7, 1]. However, many natural problems on transductions are undecidable, for instance equivalence of one-way non-deterministic transducers [9, 10].

In order to circumvent this, and to obtain a better-behaved model, Bojańczyk introduced transducers with *origin information* [2], where the semantics takes into account not only the input/output pair of words, but also the way the output is produced from the input. It is shown in [2] that translations between different models of transducers usually preserve the origin semantics, more problems become decidable, such as the equivalence between two transducers, and the model of transduction with origins is more amenable to an algebraic approach.

The fact that two transducers are origin-equivalent if they produce their output in exactly the same way can seem too strict, and prompted the idea of *resynchronization*. The idea, introduced in [8], where the main focus was the *sequential uniformization* problem, and developed in [5, 4], is to allow a distortion of the origins in a controlled way, in order to recognize that two transducers have a similar behaviour.

It is shown in [5], that containment of 2-way transducers up to a fixed resynchronization is in PSpace, so no more difficult than classical containment of non-deterministic one-way automata. This covers in particular the case where the resynchronization is trivial, in which case the problem boils down to testing strict origin equivalence.


45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020).
Editors: Javier Esparza and Daniel Král'; Article No. 58; pp. 58:1–58:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In [4], the *resynchronizer synthesis* problem was studied. The goal is now to decide whether there exists a resynchronizer $R$ such that containment or equivalence holds up to $R$. Some results are obtained for two notions of resynchronizers. The first notion, introduced in [8] is called *rational resynchronizers*, it is specialized for 1-way transducers, and uses an interleaving of input and output letters. The second notion is called *(bounded) regular resynchronizers*, it is the focus of [5] and is defined for two-way transducers.

For rational resynchronizers, a complete picture is obtained in [4]: the synthesis problem is decidable for $k$-valued transducers, but undecidable in general. For regular resynchronizers, it is shown in [4] that the synthesis problem is decidable for unambiguous two-way transducers, i.e. transducers that have at most one accepting run on each input word. The ambiguous case is left open. It was also shown in [4] that for one-way transducers, the notion of rational and regular resynchronizer do not match. The picture for resynchronizability from previous works is summed up in this table, where the first line describes constraints on the input pair of transducers:

| | unambiguous | functional/finite-valued | general case |
|---|---|---|---|
| Fixed regular resync. (2-way) | PSPACE | PSPACE-c | PSPACE-c. |
| Unknown rational resync. (1-way) | decidable | decidable | undecidable |
| Unknown regular resync. (2-way) | decidable | ? | ? |

In this work, we tackle the general case (last question mark), and show a stronger result: the synthesis of regular resynchronizers is already undecidable for one-way transducers.

To do so, we introduce the notion of limited traversal, which characterizes whether two transducers verify a containment relation up to some unknown resynchronization. Outside of this undecidability proof, this notion can be used to show that some natural transducers, equivalent in the classical sense, cannot be resynchronized. As a by-product, we also obtain that the resynchronizer synthesis problem is undecidable even if we restrict regular resynchronizers to any natural subclass containing the simple "shifting" resynchronizations, allowing origins to change by at most $k$ positions for a fixed bound $k$. Our proof can also be lifted to show a different statement, emphasizing the difference between rational and regular resynchronization: even in presence of regular resynchronization, synthesis of a rational resynchronizer is undecidable. Due to space constraints, some auxiliary material is given in the full version of the paper, available at `https://arxiv.org/abs/2002.07558`.

## Notations

If $i, j \in \mathbb{N}$, we denote $[i, j]$ the set $\{i, i+1, \ldots, j\}$. We will note $\mathbb{B} := \{0, 1\}$ the set of booleans. If $X$ is a set, we denote $X^* := \bigcup_{i \in \mathbb{N}} X^i$ the set of words on alphabet $X$. The empty word is denoted $\varepsilon$. We will denote $u \sqsubseteq v$ if $u$ is a prefix of $v$. We will denote $\Sigma$ and $\Gamma$ for arbitrary finite alphabets throughout the paper. If $u \in \Sigma^*$, we will denote $|u|$ its length and $dom(u) = \{1, 2, \ldots, |u|\}$ its set of positions.

# 2 Transductions

## 2.1 One-way Non-deterministic Transducers

A *one-way non-deterministic transducer* (1NT) is a tuple $T = \langle Q, \Sigma, \Gamma, \Delta, I, F \rangle$, where $Q$ is a finite set of states, $\Sigma$ is a finite input alphabet, $\Gamma$ is a finite output alphabet, $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$ is the transition relation, $I$ is the set of initial states, and $F$ the set of final states. A transition $(p, a, v, q)$ of $\Delta$ will be denoted as $p \xrightarrow{a|v} q$. A *run* of $T$ on an input word $u \in \Sigma^*$ is a sequence of transitions $p_0 \xrightarrow{a_1|v_1} p_1 \xrightarrow{a_2|v_2} \dots \xrightarrow{a_n|v_n} p_n$, such that $u = a_1 a_2 \dots a_n$, $p_0 \in I$ and $p_n \in F$. The *output* of this run is the word $v = v_1 \dots v_n$. The *relation* computed by $T$ is $[\![T]\!] = \{(u, v) \mid \text{there exists a run of } T \text{ on } u \text{ with output } v\} \subseteq \Sigma^* \times \Gamma^*$. To avoid unnecessary special cases, we will always assume throughout the paper that the input word $u$ is not empty. Two transducers $T_1, T_2$ are *classically equivalent* if $[\![T_1]\!] = [\![T_2]\!]$. It is known from [9] that classical equivalence of 1NTs is undecidable.

## 2.2 Two-way Transducers

In 1NTs, transitions can either leave the reading head on the same input letter, or move it one step to the right. If the possibility of moving to the left is added, we obtain the model of *two-way non-deterministic transducer* (2NT). The transition relation is now of the form $\Delta \subseteq Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \Gamma^* \times \{\mathsf{left}, \mathsf{right}\} \times Q$, where the symbol $\vdash$ (resp. $\dashv$) marks the beginning (resp. end) of the input word. When reading this symbol, we forbid the production of a non-empty output, and the only allowed direction for transitions is $\mathsf{right}$ (resp. $\mathsf{left}$). The semantics $[\![T]\!] \subseteq \Sigma^* \times \Gamma^*$ of a 2NT is defined in a natural way: the output of a run $p_0 \xrightarrow{a_1|v_1,d_1} p_1 \xrightarrow{a_2|v_2,d_2} \dots \xrightarrow{a_n|v_n,d_n} p_n$ is $v_1 v_2 \dots v_n$. See [5] for a formal definition. Notice that $\varepsilon$-transitions are not necessary anymore, since a transition $p \xrightarrow{\varepsilon|v} q$ can be simulated by two transitions going right then left (or left then right if the symbol $\dashv$ is reached).

If the transition relation is deterministic, i.e. if for all $(p, a) \in Q \times (\Sigma \cup \{\vdash, \dashv\})$ there exists at most one $(v, d, q) \in \Gamma^* \times \{\mathsf{left}, \mathsf{right}\} \times Q$ such that $p \xrightarrow{a|v,d} q$ is a transition in $\Delta$, we say that the transducer is a *two-way deterministic transducer* (2DT).

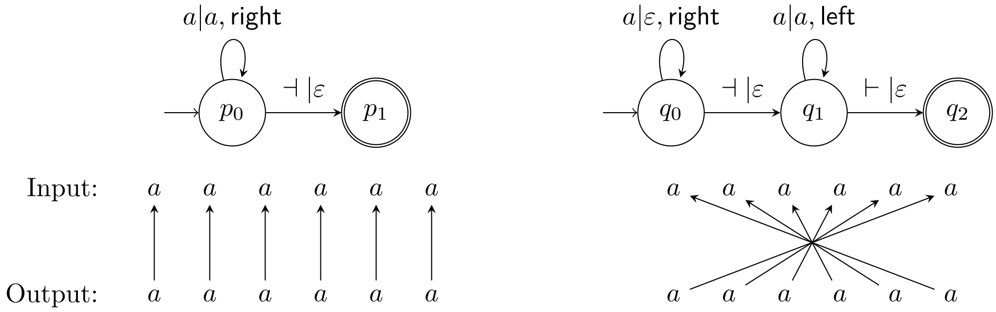
Notice that the relation defined by a 2DT $T$ is necessarily a (partial) function: for all $u \in \Sigma^*$ there is at most one $v \in \Gamma^*$ such that $(u, v) \in [\![T]\!]$. The class of functions definable by 2DTs is called *regular string-to-string functions*. It has equivalent characterizations, such as MSO transductions [7] and streaming transducers [1].

## 2.3 Origin information

The origin semantics was introduced in [2] as an enrichment of the classical semantics for string-to-string transductions. The principle is that the contribution of a run of $T$ to the semantics of $T$ is not only the input/output pair $(u, v)$, but an *origin graph* describing how $v$ is produced from $u$ during this run.

Formally, an origin graph is a triple $(u, v, \mathrm{orig})$ where $u \in \Sigma^*$, $v \in \Gamma^*$, and $\mathrm{orig} : dom(v) \to dom(u)$ associates to each position in $v$ a position in $u$: its *origin*. An origin graph is associated to a run of a transducer $T$ in a natural way, by mapping to each position $y$ in $v$ the position $\mathrm{orig}(y)$ of the reading head in $u$ when writing to this position $y$. If an output is produced by an $\varepsilon$-transition after the whole word has been processed in a 1NT, we take the last input letter as origin. The *origin semantics* $[\![T]\!]_o$ of $T$ is the set of origin graphs associated with runs of $T$.

▶ **Example 1.** The two following 2DTs $T_{id}$ and $T_{rev}$ are classically equivalent and compute the identity relation $\{(a^n, a^n) \mid n \in \mathbb{N}\}$, but their origin semantics differ, as witnessed by their unique origin graphs for input $a^6$ given below.




Two transducers are said *origin equivalent* if they have the same origin semantics. It is shown in [2] that origin equivalence is decidable for regular transductions, and in [5] that origin equivalence is PSPACE-complete for 2NTs. See full version for an example of two one-way transducers both computing the full relation $\Sigma^* \times \Gamma^*$, but not origin equivalent.

## 3 MSO Resynchronizers

While origin semantics gives a satisfying framework to recover decidability of transducer equivalence, it can be argued that this semantics is too rigid, as origin equivalence require that the output is produced in exactly the same way in both transducers.

In order to relax this constraint, the intermediate notion of *resynchronization* has been introduced [8, 5]. The idea is to let origins differ in a controlled way, while preserving the input/output pair. Several notions of resynchronizations have been considered [8, 5, 4], we will focus in this work on MSO resynchronizers, also called regular resynchronizers.

### 3.1 Regular languages and MSO

We recall here how Monadic Second-Order logic (MSO) can be used to define languages. This framework will be then used to represent resynchronizers. Formulas of MSO are defined by the following grammar, where $a$ ranges over the alphabet $\Sigma$:

$$\varphi, \psi := a(x) \mid x \leq y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \psi \mid \neg\varphi$$

Such formulas are evaluated on structures induced by finite words: the universe is the set of positions of the word, $a(x)$ means that position $x$ is labelled by letter $a$, and $x \leq y$ means that position $x$ occurs before position $y$. Lowercase notation is used for first-order variables, ranging over positions of the word, and uppercase notation is used for second-order variables, ranging over sets of positions. Other classical operators such as $\wedge, \Rightarrow, \forall, =, +1, +2, \mathit{first}, \mathit{last}, \dots$ can be defined from this syntax and will be used freely. Let $\top$ be a tautology, defined for instance as $\exists x.a(x) \vee \neg(\exists x.a(x))$.

If $\varphi$ is an MSO formula and $u \in \Sigma^*$, we will note $u \models \varphi$ if $u$ is a model of $\varphi$, with classical MSO semantics. The language $L(\varphi)$ defined by a closed formula $\varphi$ is $\{u \in \Sigma^* \mid u \models \varphi\}$.

If $\varphi$ contains free variables $X_1, \dots, X_n, x_1, \dots, x_k$, we can still define the language of $\varphi$, using an extended alphabet $\Sigma \times \mathbb{B}^{n+k}$. Extra boolean components at each position are used to convey the values of free variables at this position: it is 1 if the value of the second-order variable contains the position (resp. if the value of the first-order variable matches the position) and 0 otherwise. The language of $\varphi$ is in this case a subset of $(\Sigma \times \mathbb{B}^{n+k})^*$, i.e. a set of words on $\Sigma$ enriched with valuations for the free variables. If $I_1, \dots, I_n, i_1, \dots, i_k$ is an instantiation for the free variables of $\varphi$ in a word $u$, we will also write $(u, I_1, \dots, I_n, i_1, \dots, i_k) \models \varphi$ to signify that $u$ with this instantiation of the free variables satisfies $\varphi$.

For instance if $\varphi = \exists x.(x \in X \wedge a(x))$ uses a free second-order variable $X$, then the word $u = (a, 0), (b, 1), (a, 1) \in (\Sigma \times \mathbb{B})^*$ is a model of $\varphi$, denoted also $(aba, \{2, 3\}) \models \varphi$, but the word $(a, 0), (b, 1), (a, 0)$ is not.

A language $L \subseteq (\Sigma \times \mathbb{B}^n)^*$ is *regular* if and only if there is a formula $\varphi$ of MSO with $n$ free variables recognizing $L$. This is equivalent to $L$ being recognizable by a deterministic finite automaton (DFA) on alphabet $\Sigma \times \mathbb{B}^n$ [6].

## 3.2 MSO Resynchronizers

The principle behind MSO resynchronizers as defined in [5] is to describe in a regular way, with MSO formulas, how the origins can be redirected. This will induce a relation between sets of origin graphs: containment up to resynchronization.

The MSO formulas will be allowed to use a finite set of *parameters*: extra information labelling the input word. This is reminiscent of the model of non-deterministic two-way transducers with *common guess* [3], where the guessing of extra parameters can be done in a consistent way through different visits of the same position in the input word.

### 3.2.1 Definition

We now define a subclass of regular resynchronizers from [5, 4]. We will see that for our purpose of resynchronizer synthesis, this subclass is equivalent to the full class of resynchronizers from [5, 4]. Intuitively, the full definition from [5, 4] allows to further restrict the semantics of a resynchronizer, which is not useful if we are just interested in the existence of a resynchronization between two transducers. This is further explained in Section 4.1 and full version of the paper.

Given an origin graph $\sigma = (u, v, \text{orig})$, an *input parameter* is a subset of the input positions, encoded by a word on $\mathbb{B}$. Thus, a valuation for $m$ input parameters is given by a tuple $\bar{I} = (I_1, \dots, I_m)$ where for each $i \in [1, m], I_i \in \mathbb{B}^{|u|}$.

The main differences between the following simplified definition and the one from [5, 4] is that we ignored output parameters (an extra labelling of the output word), and also removed extra formulas constraining the behaviour of the resynchronization with respect to both input and output parameters.

▶ **Definition 2.** *An MSO (or* regular*) resynchronizer $R$ with $m$ input parameters is an MSO formula $\gamma$ with $m + 2$ free variables $\gamma(\bar{I}, x, y)$, evaluated over the input word $u$.*

Intuitively, $\gamma(\bar{I}, x, y)$ indicates that the origin $x$ of an output position can be redirected to a new origin $y$, as made precise in Definition 3. Although $R$ and $\gamma$ are actually the same object here, we will keep the two notations to maintain coherence with [5], using $R$ for the abstract resynchronizer and $\gamma$ for the MSO formula, which is only one of the components of $R$ in [5]. We now describe formally the semantics of an MSO resynchronizer.

▶ **Definition 3.** *[5] An MSO resynchronizer $R$ induces a relation $[\![R]\!]$ on origin graphs in the following way. If $\sigma = (u, v, \text{orig})$ and $\sigma' = (u', v', \text{orig}')$ are two origin graphs, we have $(\sigma, \sigma') \in [\![R]\!]$ if and only if $u = u', v = v'$, and there exists input parameters $\bar{I} \in (\mathbb{B}^{|u|})^m$, such that for every output position $z \in dom(v)$, we have $(u, \bar{I}, \text{orig}(z), \text{orig}'(z)) \models \gamma$.*

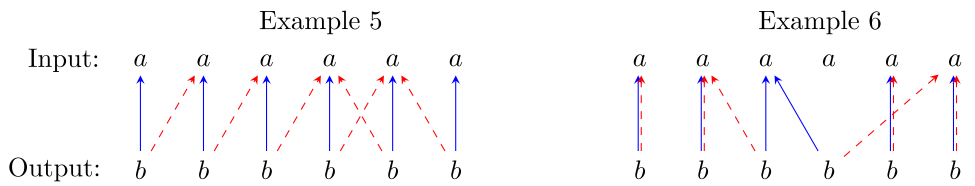
### 3.2.2 Examples

Plain blue arrows will represent the "old" origins in $\sigma$, and red dotted arrows the "new" origins in $\sigma'$.

▶ **Example 4.** [5] The resynchronizer without parameters $R_{univ}$, using only a tautology formula $\gamma = \top$, is called the universal resynchronizer, and resynchronizes any two origin graphs that share the same input and output.

▶ **Example 5.** [5] The resynchronizer without parameters $R_{\pm 1}$ shifts all origins by exactly 1 position left or right. This is achieved using a formula $\gamma(x, y) = (x = y + 1) \vee (y = x + 1)$.

▶ **Example 6.** The resynchronizer with one parameter defined by $\gamma = (I = \{x\}) \vee (x = y)$ allows at most one input position to be resynchronized to different origins.




## 3.3 Containment up to resynchronization

▶ **Definition 7.** *[5] For a resynchronizer $R$ and two transducers $T_1, T_2$ we note $T_1 \subseteq R(T_2)$ if for every origin graph $\sigma_1 \in [\![T_1]\!]_o$, there exists $\sigma_2 \in [\![T_2]\!]_o$ such that $(\sigma_2, \sigma_1) \in [\![R]\!]$.*

In other words this means that $[\![T_1]\!]_o$ is contained in the resynchronization expansion of $[\![T_2]\!]_o$. Examples can be found in full version.

For a fixed resynchronizer $R$ and a 2NT $T$, it might not be the case that $T \subseteq R(T)$, as witnessed by the resynchronizer $R_{\pm 1}$ from Example 5. Moreover, if $T_1 \subseteq R(T_2)$ and $T_2 \subseteq R(T_3)$ it might not be the case that $T_1 \subseteq R(T_3)$, again this is examplified by $R_{\pm 1}$. This means that the containment relation up to a fixed resynchronizer $R$ is neither reflexive nor transitive in general.

## 3.4 Bounded resynchronizers

Note that the universal resynchronizer $R_{univ}$ from Example 4 relates any two graphs that share the same input and output. This causes the containment relation up to $R_{univ}$ to boil down to classical containment, ignoring the origin information. I.e. we have $T_1 \subseteq R_{univ}(T_2)$ if and only if $[\![T_1]\!] \subseteq [\![T_2]\!]$. This inclusion relation is undecidable, even in the case of one-way non-deterministic transducers [9]. Thus containment up to a fixed resynchronizer is undecidable in general, if no extra constraint is put on resynchronizers. That is why the natural *boundedness* restriction is introduced on MSO resynchronizers in [5].

▶ **Definition 8.** *[5] (Boundedness) A regular resynchronizer $R$ has bound $k$ if for all inputs $u$, input parameters $\bar{I}$, and target position $y \in dom(u)$, there are at most $k$ distinct positions $x_1, \ldots x_k \in dom(u)$ such that $(u, \bar{I}, x_i, y) \models \gamma$ for all $i \in [1, k]$. A regular resynchronizer is bounded if it has bound $k$ for some $k \in \mathbb{N}$.*

All examples of resynchronizations given in this paper are bounded, except for $R_{univ}$. In the full version, we give examples of bounded resynchronizations that displace the origin by a distance that is not bounded.

Boundedness is a decidable property of MSO resynchronizers [5, Prop. 15]. As stated in the next theorem, boundedness guarantees that the containment problem up to a fixed resynchronizer becomes decidable. Moreover, for any fixed bounded MSO resynchronizer, the complexity of this problem matches the complexity of containment with respect to strict origin semantics, or more simply the complexity of inclusion of non-deterministic automata.

▶ **Theorem 9.** *[5, Cor. 17] For a fixed bounded MSO resynchronizer $R$ and given two 2NTs $T_1, T_2$, it is decidable in PSPACE whether $T_1 \subseteq R(T_2)$.*

## 4 Resynchronizability

We will now be interested in the containment up to an unknown bounded resynchronizer. Let us define the relation $\preceq$ on 2NTs by $T_1 \preceq T_2$ if there exists a bounded resynchronizer $R$ such that $T_1 \subseteq R(T_2)$. This relation has been introduced in [4], along with the same notion with respect to rational resynchronizers.

Focusing on bounded regular resynchronizers, the following result is obtained in [4]:

▶ **Theorem 10.** *[4] The relation $\preceq$ is decidable on unambiguous 2NTs.*

The problem is left open in [4] for general 2NTs, and this is the purpose of the present work. Now that the necessary notions have been presented, we move to our contributions.

### 4.1 Containment relation

Let us start by expliciting a few properties of $\preceq$. First, let us emphasize that our simplified definition of MSO resynchronizer is justified by the fact that this definition yields the same relation $\preceq$ as the one from [5, 4]. This is explicited in the full version.

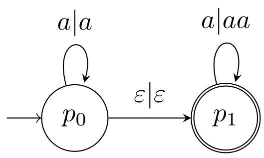
This simplified definition allows us to show basic properties of the $\preceq$ relation, see full version for a detailed proof:

▶ **Lemma 11.** *The relation $\preceq$ is reflexive and transitive.*

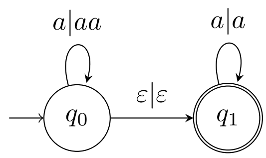
Since $\preceq$ is a pre-order, it induces an equivalence relation $\sim$ on 2NTs, defined by $\sim = \preceq \cap \succeq$. Notice that this equivalence relation is intermediate between classical equivalence and origin equivalence, but it is not immediately clear that it does not coincide with classical equivalence.

The following claim presents two pairs of transducers (one pair of 2DTs and one pair of 1NTs) equivalent for the classical semantics, but not $\sim$-equivalent.

▷ Claim 12.
- The 2NTs $T_{id}$ and $T_{rev}$ from Example 1 are not $\sim$-equivalent.
- The two following 1NTs $T_{one-two}, T_{two-one}$ have the same classical semantics $\{(a^n, a^m) \mid n \leq m \leq 2n\}$, but are not $\sim$-equivalent.




Transducer $T_{one-two}$




Transducer $T_{two-one}$

A variant of the pair $T_{one-two}, T_{two-one}$ is presented in [4, Example 5], where it is claimed without proof that no bounded regular resynchronizer exists. A proof of Claim 12 will be obtained as a by-product of Theorem 17 and explicited in Corollary 19.

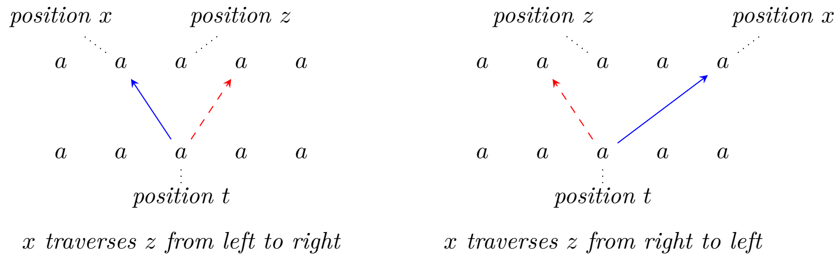
### 4.2 Limited traversal

The goal of this section is to exhibit a pattern characterizing families of origin graphs that cannot be resynchronized with a bounded MSO resynchronizer.

▶ **Definition 13.** *Let $\sigma = (u, v, \text{orig})$ and $\sigma' = (u, v, \text{orig}')$ be two origin graphs with same input/output words. Given two input positions $x, z \in dom(u)$, we say $x$* traverses *$z$ if there exists an output position $t \in dom(v)$ with $orig(t) = x$ and either:*

- *$x \leq z$ and $orig'(t) > z$ (left to right traversal);*
- *$x \geq z$ and $orig'(t) < z$ (right to left traversal).*

*Intuitively, $x$ traverses $z$ if $x$ is resynchronized to some $y \neq z$, and $z$ is between the two positions $x, y$.*




Let $k \in \mathbb{N}$, a pair of origin graphs $(\sigma, \sigma')$ on input/output words $(u, v)$ is said to have *$k$-traversal* if for every $z \in dom(u)$, there are at most $k$ distinct positions of $dom(u)$ that traverse $z$. A resynchronizer $R$ is said to have $k$-traversal if every pair of origin graphs $(\sigma, \sigma') \in [\![R]\!]$ has $k$-traversal. A resynchronizer $R$ has *limited traversal* if there exists $k \in \mathbb{N}$ such that $R$ has $k$-traversal.

For any $k \in \mathbb{N}$ we want to construct a bounded resynchronizer $R_k$ that relates any pair of origin graphs that have $k$-traversal. We will use $2k$ input parameters: $Right_i$ and $Left_i$ for $i \in [0, k-1]$. Each parameter $Right_i$ (resp. $Left_i$) corresponds to a guessed set of input positions that may be redirected to the right (resp. left), but without traversing a position of the same set. For instance it is not possible for a position of $R_3$ to traverse another position of $R_3$ from left to right. Similarly, a position of $L_2$ cannot traverse another position of $L_2$ from right to left. We do not a priori require any of these sets to be disjoint from each other. We construct $\gamma(x, y) = (x = y) \vee R_{trav} \vee L_{trav}$ to ensure this fact, where

$$R_{trav} = \bigvee_{1 \leq i \leq k} \big(x \in Right_i \wedge x < y \wedge (\forall z \in [x+1, y].z \notin Right_i)\big)$$

verifies that positions labelled by the same $Right_i$ do not traverse each other, and $L_{trav}$ does the same for the $Left_i$ labels. This achieves the description of the resynchronizer $R_k$, which will be proved correct in Lemmas 14 and 15.

▶ **Lemma 14.** *The resynchronizer $R_k$ is bounded.*

**Proof.** For each potential target position $y$, if two sources $x$ were labelled with the same input parameter, either one would traverse the other, or one would be at the left of $y$, which would contradict the definition of the formula. This means that if $\gamma(x, y)$ is valid then either $x = y$ or one of the parameters is used to indicate a single $x$ as source. There are only $2k$ parameters so for every input position $y$ there are at most $2k + 1$ distinct positions $x$ such that $\gamma(x, y)$ is valid. ◀

▶ **Lemma 15.** *If a pair of origin graphs $(\sigma, \sigma')$ has $k$-traversal, then $(\sigma, \sigma') \in [\![R_k]\!]$.*

**Proof sketch.** We describe an algorithm performing a left to right pass of the input word, and assigning labels $Right_0, Right_1, \dots, Right_{k-1}$ to positions that are resynchronized to the right. We always assign to a position the minimal index currently available, in order to avoid the right traversal of any position by another position with the same label. We then show that under the hypothesis of $k$-traversal, this algorithm succeeds in finding an assignment of labels witnessing $(\sigma, \sigma') \in [\![R_k]\!]$. The same algorithm is then run in the other direction (right to left), to assign labels $Left_i$. See the full version for the detailed construction. ◀

▶ **Lemma 16.** *An MSO resynchronizer $R$ has limited traversal if and only if it is bounded.*

**Proof.** Let $m$ be the number of input parameters used in $R$.

($\Rightarrow$) Assume $R$ is not bounded, and let $k \in \mathbb{N}$, we want to build a pair $(\sigma, \sigma') \in [\![R]\!]$ exhibiting $k$-traversal. Since $R$ is not bounded, there exists a word $u \in \Sigma^*$, with input parameters $\bar{I}$, a position $y$, and a set $X$ of $2k+1$ distinct positions such that for all $x \in X$, we have $(u, \bar{I}, x, y) \models \gamma$. Without loss of generality, we can assume that there are $k$ distinct positions $x_1, \dots x_k$ in $X$ that are strictly to the left of $y$. Let $a \in \Gamma$ be an arbitrary output letter and $v = a^k$. We define the origin graphs $\sigma, \sigma'$ on $(u, v)$ by setting for each $i \in [1, k]$ the origin of the $i^{th}$ letter of $v$ to $x_i$ in $\sigma$ and to $y$ in $\sigma'$. As witnessed by parameters $\bar{I}$, we have $(\sigma, \sigma') \in [\![R]\!]$. Moreover, the input position $y-1$ is traversed from left to right by $k$ different sources. Since $k$ is arbitrarily chosen, $R$ does not have limited traversal.

($\Leftarrow$) For the other direction, assume $R$ has no limited traversal. Let $\mathcal{A}$ be a deterministic automaton recognizing $\gamma$, on alphabet $\Sigma_{\mathcal{A}} = \Sigma \times \mathbb{B}^{m+2}$, and $Q$ be the state space of $\mathcal{A}$. Let $k \in \mathbb{N}$ be arbitrary. There exists $(\sigma, \sigma') \in [\![R]\!]$ a pair of origin graphs on words $(u, v)$, and a position $z \in dom(u)$ such that, without loss of generality, $z$ is traversed by $K = k \cdot |Q|$ positions $x_1 < x_2 < \cdots < x_K$ from left to right, i.e. $x_K \leq z$. Let $\bar{I}$ be the input parameters witnessing $(\sigma, \sigma') \in [\![R]\!]$. This means that for each $i \in [1, K]$ there exists $y_i > z$ with $(u, \bar{I}, x_i, y_i) \models \gamma$. Let us split the input sequence $U = (u, \bar{I}) \in \Sigma_{\mathcal{A}}^*$ according to position $z$: $U = wr$, where the last letter of $w$ is in position $z$. For each $i \in [1, K]$, let $w_i \in \Sigma_{\mathcal{A}}^*$ be the word $w$ with two extra boolean components: the source is marked by a bit 1 in position $x_i$, and the target is left to be defined. We know that for each $i$ there exists $r_i \in \Sigma_{\mathcal{A}}^*$ extending $r$ with a target position such that $w_i r_i$ is accepted by $\mathcal{A}$. Let $q_i$ be the state reached by $\mathcal{A}$ after reading $w_i$. By choice of $K$, there exists $q \in Q$ such that $q_i = q$ for $k$ distinct values $i_1, \dots i_k$ of $i$. This means that for each $j \in [1, k]$, we have $w_{i_j} r_{i_1}$ accepted by $\mathcal{A}$, i.e. $(u, \bar{I}, x_{i_j}, y_{i_1}) \models \gamma$. This achieves the proof that $R$ is not bounded. ◀

▶ **Theorem 17.** *Let $T_1, T_2$ be 2NTs. Then $T_1 \preceq T_2$ if and only if there exists $k \in \mathbb{N}$ such that for every $\sigma' \in [\![T_1]\!]_o$, there exists $\sigma \in [\![T_2]\!]_o$ with same input/output and $(\sigma, \sigma')$ has $k$-traversal.*

**Proof.** Assume such a bound $k$ exists. By Lemma 15, for every $\sigma' \in [\![T_1]\!]_o$ there exists $\sigma \in [\![T_2]\!]_o$ such that $(\sigma, \sigma') \in [\![R_k]\!]$. This implies $T_1 \subseteq R_k(T_2)$, and by Lemma 14 this $R_k$ is bounded thus witnessing $T_1 \preceq T_2$.

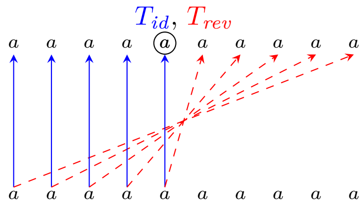
Conversely, assume that no such bound $k$ exists, but that there is a bounded resynchronizer $R$ witnessing $T_1 \preceq T_2$. By Lemma 16, $R$ has $k$-traversal for some $k \in \mathbb{N}$. By assumption, there exists $\sigma' \in [\![T_1]\!]_o$ such that for all $\sigma \in [\![T_2]\!]_o$, $(\sigma, \sigma')$ does not have $k$-traversal. However, there must exists $\sigma$ such that $(\sigma, \sigma') \in [\![R]\!]$, contradicting the fact that $R$ has $k$-traversal. ◀

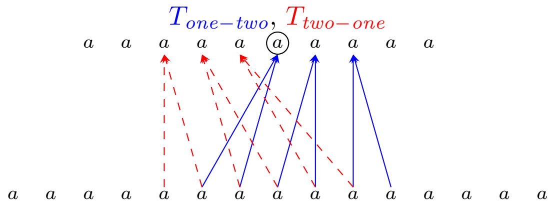
▶ Remark 18. We have shown here that the resynchronizers $R_k$ are universal: if two transducers can be resynchronized, then this is witnessed by a resynchronizer $R_k$. This gives for instance a bound on the logical complexity of the MSO formulas needed in resynchronizers: the formula for $R_k$ is a disjunction of formulas using only one $\forall$ quantifier.

Notice that unlike the existence of bounded resynchronizer, the notion of limited traversal is directly visible on pairs of origin graphs, and is therefore useful to prove that two transducers cannot be resynchronized. This is exemplified in the following corollary.

▶ **Corollary 19.** *The transducers from Claim 12 are not $\sim$-equivalent. Indeed, in both cases, for a given input/output pair $(u, v)$ in the relation, only one pair $(\sigma, \sigma')$ of origin graphs is compatible with $(u, v)$, and these pairs of graphs exhibit traversal of arbitrary size.*

Here are visualizations of the phenomenon. The first picture shows a pair of graphs with 5-traversal for $T_{id}, T_{rev}$, witnessed by the only origin graphs on words $(a^{10}, a^{10})$. The second picture does the same for the two 1NTs $T_{one-two}, T_{two-one}$, which has 3-traversal on words $(a^{10}, a^{15})$. In both cases, the input position being traversed is circled, and only origin arrows relevant to the traversal of this position are represented.




## 5 Undecidability of containment and equivalence

The aim of this section is to prove our main result:

▶ **Theorem 20.** *Given two 2NTs $T_1, T_2$, it is undecidable whether $T_1 \preceq T_2$.*

The result remains true if $T_1, T_2$ are 1NTs, with equivalence instead of containment, and if we restrict to any class of resynchronization that contains the "shift resynchronizations" : for each $k \in \mathbb{N}$, the $k$-shift resynchronization is defined by $\gamma(x, y) = (y \leq x \leq y + k)$.

We will proceed by reduction from the problem *BoundTape*, which asks given a deterministic Turing Machine $M$, whether it uses a bounded amount of its tape on empty input. For completeness, we prove in the full version that this problem is undecidable, by a simple reduction from the Halting problem. To perform the reduction from *BoundTape* to the $\preceq$ relation, we first describe a classical construction used to encode runs of a Turing machine.

### 5.1 The Domino Game

Let $M$ be a deterministic Turing Machine with alphabet $A$, states $Q$, and transition table $\delta : Q \times A \to Q \times A \times \{\mathsf{left}, \mathsf{right}\}$. Let $q_0$ (resp. $q_f$) be the initial (resp. final) state of $M$, and $B$ be the special blank symbol from the alphabet $A$, initially filling the tape.

Let $\# \notin A \cup Q$ be a new separation symbol, and $\Gamma = A \cup Q \cup \{\#\}$.

We sketch here a classical idea of using *domino tiles* to simulate the run of a Turing Machine, for instance to prove undecidability of the Post Correspondence Problem [11, 12]. See full version for the detailed construction of the set of tiles.

We encode successive *configurations* of $M$ by words on $\Gamma^*$. The full run, or *computation history* of $M$ is encoded by a finite or infinite word $Hist_M \in \Gamma^* \cup \Gamma^\omega$. We use a set of *tiles* $D_M = \{(u_i, v_i) \in (\Gamma^*)^2 \mid i \in \Sigma\}$, where $\Sigma$ is a finite alphabet of tile indexes. These tiles are designed to simulate the run of $M$ in the following sense (recall that $\sqsubseteq$ stands for prefix):

▶ **Lemma 21.** *Let $\lambda = i_1 \dots i_k \in \Sigma^*$ be a sequence of tile indexes. Let $u_\lambda = u_{i_1} \dots u_{i_k}$, and $v_\lambda = q_0 \# v_{i_1} \dots v_{i_k}$. If $\lambda$ is such that $u_\lambda \sqsubseteq v_\lambda$, then we have $v_\lambda \sqsubseteq Hist_M$.*

We give here an example of how a run of $M$ is encoded, and how it is reflected on tiles:

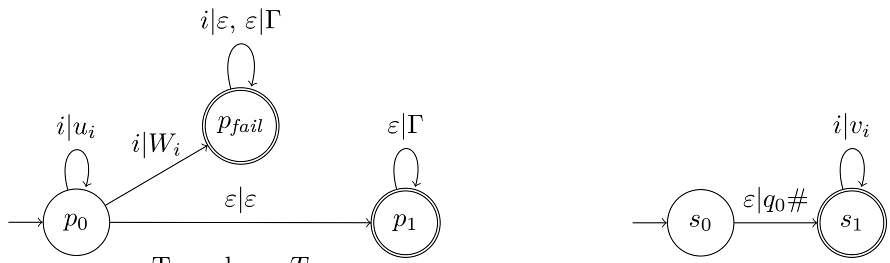
▶ **Example 22.** Consider the run of $M$ encoded by $q_0\#q_0B\#aq_1\#aq_1B\#q_2ab\# \in \Gamma^*$. This is reflected by the following sequences of tiles:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda$ : | | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $i_6$ | $i_7$ |
| $u_\lambda$ : | | $q_0\#$ | $q_0B$ | $\#$ | $a$ | $q_1\#$ | $aq_1B$ | $\#$ |
| $v_\lambda$ : | $q_0\#$ | $q_0B\#$ | $aq_1$ | $\#$ | $a$ | $q_1B\#$ | $q_2ab$ | $\#$ |

## 5.2 From tiles to transducers

We now build two 1NTs $T_{up}$ and $T_{down}$, based on the tiles of $D_M$. The input alphabet of these transducers is the set $\Sigma$ of indexes of tiles of $D_M$. The output alphabet is $\Gamma$. Roughly, on input $i$, $T_{up}$ outputs $u_i$ and $T_{down}$ outputs $v_i$. Additionally, $T_{up}$ is allowed to non-deterministically start outputting a word that is not a prefix of $u_i$, and from there output anything in $\Gamma^*$. The transducer $T_{up}$ is also allowed to output anything after the end of the input. The transducer $T_{down}$ starts by outputting $q_0\#$ at the beginning of the computation, so that on input $\lambda \in \Sigma^*$ it outputs $v_\lambda$.

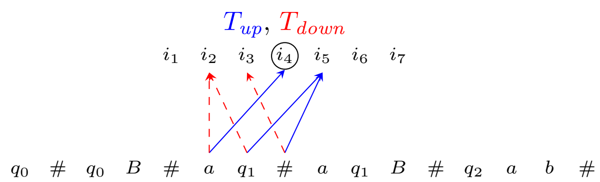
The transducers $T_{up}, T_{down}$ are pictured here, with $W_i = \{u \in \Gamma^*, |u| \leq |u_i|, u \not\sqsubseteq u_i\}$:




Transducer $T_{up}$ Transducer $T_{down}$

The main idea of this construction is that if $\lambda = i_1 \dots i_k \in \Sigma^*$ is such that $u_\lambda \sqsubseteq v_\lambda$ follow $Hist_M$ as in Example 22, then on input $\lambda$, $T_{down}$ outputs $v_\lambda$, the only matching computation of $T_{up}$ starts by outputting $u_\lambda$, and the bound on traversal will (roughly) match the size of the tape used by $M$ in this prefix of the computation. Indeed, if $T_{up}$ and $T_{down}$ output the encoding of the same configuration of size $K$ on disjoint inputs, it witnesses a traversal of size roughly $K$ ("roughly" because tiles allow up to three output letters on one input letter). The extra part of $T_{up}$ is used to guarantee that $[\![T_{down}]\!] \subseteq [\![T_{up}]\!]$ holds, even in cases when the input $\lambda$ does not correspond to a prefix of the computation of $M$.

▶ **Example 23.** Let $\lambda = i_1 i_2 \dots i_7$ be the sequence of tile indexes from Example 22. We show here a 2-traversal exhibited by $T_{up}, T_{down}$ on input $\lambda$. The traversed input position is circled, and only arrows relevant to the traversal of this position are represented.




▶ **Theorem 24.** *We have $T_{down} \preceq T_{up}$ if and only if $M \in BoundTape$.*

**Proof.** First, assume $M \in \mathit{BoundTape}$, let $K$ be the bound on the tape size used by $M$. Let $R$ be the resynchronization that shifts by at most $K + 2$ positions to the left, via $\gamma(x, y) = (y \leq x) \wedge (x \leq y + K + 2)$. We claim that $T_{down} \subseteq R(T_{up})$. It is clear that $R$ is bounded. Let $\sigma' \in [\![T_{down}]\!]_o$ be an origin graph $(\lambda, v, \text{orig}')$. Notice that by definition of $T_{down}$, we have $v = v_\lambda = q_0 \# v_{i_1} \ldots v_{i_n}$ on input $\lambda = i_1 \ldots i_n$. We now distinguish two cases:

- If $u_\lambda \sqsubseteq v_\lambda$, then by Lemma 21, we have $v_\lambda \sqsubseteq \mathit{Hist}_M$. The transducer $T_{up}$ is able to output $v_\lambda$ without going through the state $p_{fail}$, with a shift of one configuration as seen in Example 23. It only needs to pad $u_\lambda$ with the last configuration in state $p_1$. Let $\sigma$ be the origin graph for this run. Since the encoding of a configuration has size at most $K + 2$, we have $(\sigma, \sigma') \in [\![R]\!]$.
- If $u_\lambda \not\sqsubseteq v_\lambda$, let $\lambda' \sqsubseteq \lambda$ be the longest prefix such that $u_{\lambda'} \sqsubseteq v_\lambda$ . Now in order to output $v_\lambda$, the transducer $T_{up}$ has to output $u_{\lambda'}$ in $p_0$ when processing $\lambda'$. After processing $\lambda'$, the transducer $T_{up}$ is forced to move to state $p_{fail}$ in order to match the output of $T_{down}$. From this state $T_{up}$ is allowed to output anything from any positions, so in particular there exists a run where the remaining output of $v_{\lambda'}$ is produced immediately, then $T_{up}$ synchronizes with $T_{down}$ during the next configuration encoding, and finally the rest of the desired output $v_\lambda$ is produced on the same input positions as in $T_{down}$. As before, the shift when processing $\lambda$ is at most $K + 2$, and therefore this run induces an origin graph $\sigma$ with $(\sigma, \sigma') \in [\![R]\!]$.

We now assume $M \notin \mathit{BoundTape}$. We want to use Theorem 17 to conclude that $T_{down} \not\preceq T_{up}$. Let $k \in \mathbb{N}$, and $\lambda \in \Sigma^*$ such that $u_\lambda \sqsubseteq v_\lambda$ and $u_\lambda$ is a prefix of $\mathit{Hist}_M$ witnessing a configuration of size $k + 2$. Let $\sigma'$ be the only origin graph of $T_{down}$ on input $\lambda$, with output $v_\lambda$. There is only one way for $T_{up}$ to output $v_\lambda$ on input $\lambda$: it is by using a run avoiding $p_{fail}$. Let $\sigma \in [\![T_{up}]\!]_o$ be the corresponding origin graph. Since $T_{up}$ is one configuration behind, and since a configuration of size $k + 2$ is produced by at least $k$ inputs, the pair $(\sigma, \sigma')$ has a position traversed $k$ times. This is true for arbitrary $k$, so by Theorem 17, we can conclude that $T_{down} \not\preceq T_{up}$. ◀

Since *BoundTape* is undecidable, this achieves the proof of Theorem 20.

Notice that in the case where $M \in \mathit{BoundTape}$, the resynchronization does not need parameters, and can be restricted to some simple classes of resynchronizations. This is stated in the following corollary:

► **Corollary 25.** *Given $T_1, T_2$ two 1NTs, it is undecidable whether $T_1 \preceq T_2$. This result still holds when considering any restricted class of resynchronizers that contains the $k$-shift resynchronizers.*

We can also strengthen the above proof to show undecidability of equivalence up to some unknown resynchronization:

► **Theorem 26.** *Given $T_1, T_2$ two 1NTs, it is undecidable whether $T_1 \sim T_2$.*

**Proof.** It suffices to take $T'_{down} = T_{down} \cup T_{up}$ in the above proof. This way we clearly have $T_{up} \preceq T'_{down}$, and the other direction $T'_{down} \preceq T_{up}$ is equivalent to $T_{down} \preceq T_{up}$, so it reduces to *BoundTape* as well. ◀

Finally, let us mention that this proof allows us to recover and strengthen undecidability results on *rational* transducers from [4]. We recall the definition of rational transducers in the full version.

Since the shift resynchronizations are rational, and that any rational resynchronization is in particular bounded regular [4, Theorem 3], our reduction can be used in particular as an alternative proof of undecidability of rational resynchronization synthesis, shown in [4] via one-counter automata. This means we directly obtain this corollary:

▶ **Corollary 27.** *Given two 1NTs $T_1, T_2$ such that $[\![T_1]\!] \subseteq [\![T_2]\!]$, it is undecidable whether there exists a rational resynchronizer $R_{rat}$ such that $T_1 \subseteq R_{rat}(T_2)$.*

We can further strengthen the result via the following theorem:

▶ **Theorem 28.** *Given two 1NTs $T_1, T_2$ and a regular resynchronizer $R_{reg}$ such that $T_1 \subseteq R_{reg}(T_2)$, it is undecidable whether there exists a rational resynchronizer $R_{rat}$ such that $T_1 \subseteq R_{rat}(T_2)$.*

Due to space constraints, the proof is presented in the full version.

## 6 Conclusion

In this work we investigated the containment relation on transducers up to unknown regular resynchronization. We showed that this relation forms a pre-order, strictly between classical containment and containment with respect to origin semantics. We introduced a syntactical condition called limited traversal, characterizing resynchronizable transducers pairs. Using this tool we proved that the resynchronizer synthesis is undecidable already in the case of 1NTs, while the problem was left open for 2NTs in [4].

We leave open the decidability of the resynchronizability relation on *functional* transducers. Since our construction highly uses non-functionality, it seems a different approach is needed.

## References


1 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–12, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

2 Mikołaj Bojańczyk. Transducers with origin information. In *International Colloquium on Automata, Languages, and Programming*, pages 26–37. Springer, 2014.

3 Mikolaj Bojańczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle. Which classes of origin graphs are generated by transducers. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 114:1–114:13, 2017.

4 Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. On Synthesis of Resynchronizers for Transducers. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

5 Sougata Bose, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. Origin-equivalence of two-way word transducers is in PSPACE. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

6 J Richard Buchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, page 6:66–92, 1960.

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

**8** Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

**9** T. V. Griffiths. The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *J. ACM*, 15(3):409–413, July 1968.

**10** Oscar H Ibarra. The unsolvability of the equivalence problem for $\varepsilon$-free NGSM's with unary input (output) alphabet and applications. *SIAM Journal on Computing*, 7(4):524–532, 1978.

**11** Emil L. Post. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52(4):264–268, April 1946.

**12** Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.