Logics for Word Transductions with Synthesis

Luc Dartois
Université Libre de Bruxelles
Belgium
luc.dartois@ulb.ac.be

Emmanuel Filiot
Université Libre de Bruxelles
Belgium
efiliot@ulb.ac.be

Nathan Lhote
Université Libre de Bruxelles
Belgium
LaBRI, Université de Bordeaux
France
nlhote@labri.fr

Abstract

We introduce a logic, called $\mathcal{L}_T$, to express properties of transductions, *i.e.* binary relations from input to output (finite) words. In $\mathcal{L}_T$, the input/output dependencies are modelled via an *origin function* which associates to any position of the output word, the input position from which it originates. $\mathcal{L}_T$ is well-suited to express relations (which are not necessarily functional), and can express all regular functional transductions, *i.e.* transductions definable for instance by deterministic two-way transducers.

Despite its high expressive power, $\mathcal{L}_T$ has decidable satisfiability and equivalence problems, with tight non-elementary and elementary complexities, depending on specific representation of $\mathcal{L}_T$-formulas. Our main contribution is a synthesis result: from any transduction R defined in $\mathcal{L}_T$, it is possible to synthesise a *regular* functional transduction f such that for all input words u in the domain of R, f is defined and $(u, f(u)) \in R$. As a consequence, we obtain that any functional transduction is regular iff it is $\mathcal{L}_T$-definable.

We also investigate the algorithmic and expressiveness properties of several extensions of $\mathcal{L}_T$, and explicit a correspondence between transductions and data words. As a side-result, we obtain a new decidable logic for data words.

*.KeywordsTransductions, Origin, Logic, Synthesis, Data words

Acknowledgments

This work was supported by the French ANR project *ExStream* (ANR-13-JS02-0010), the Belgian FNRS CDR project *Flare* (J013116) and the ARC project *Transform* (Fédération Wallonie Bruxelles).

We are also grateful to Jean-Fran cois Raskin from fruitful discussions on this work.

1 Introduction

The theory of regular languages of finite and infinite words is rich and robust, founded on the equivalence of a descriptive model (monadic second-order logic, MSO) and a computational one (finite automata), due the works of Büchi, Elgot, McNaughton and Trahtenbrot [33]. Since then, many logics have been designed and studied to describe languages (see for instance [13, 32]), among which temporal logics, with notable applications in model-checking [34].

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209181

In this paper, we consider transductions, *i.e.* binary relations relating input to output words. *E.g.* the transduction τ_{shuffle} associates with a word all its permutations – $(ab, ab), (ab, ba) \in \tau_{\text{shuffle}}$. Operational models, namely extensions of automata with outputs, called *transducers*, have been studied for computing transductions. This includes finite transducers, *i.e.* finite automata with outputs, which have been studied since the 60s [23, 29] and two-way transducers (two-way automata with a one-way output tape). When restricted to transducers defining functions (called functional transducers), the latter model has recently received a lot of attention due to its appealing algorithmic properties, its expressive power and its many equivalent models: deterministic two-way transducers [12], reversible two-way transducers [11], deterministic (one-way) transducers with registers [2] (also known as streaming string transducers), regular combinator expressions [4] and Courcelle's MSO-transducers [15] (MSOT), a model we will come back to in the related work section. Because of these many characterisations, the class defined by these models has been coined *regular transductions*, or regular functions.

However, much less is known about logics to describe transductions (see for instance [18] for a brief overview). Recently, Bojańczyk, Daviaud, Guillon and Penelle have considered an expressive logic, namely MSO over *origin graphs* (o-graphs) [6]. Such graphs encode pairs of words together with an *origin mapping*, relating any output position to an input position, as depicted in Fig. 1. Intuitively, if one thinks of an operational model for transductions, the origin of an output position is the input position from which it has been produced. As noticed in [5], most known trans-

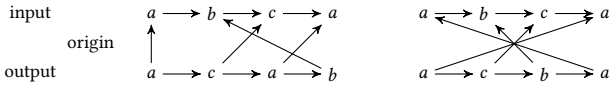


Figure 1. Possible o-graphs for τ_{shuffle}

ducer models not only define transductions, but *origin transductions* (o-transductions), *i.e.* sets of o-graphs, and can thus be naturally interpreted in both *origin-free semantics* (*i.e.* usual semantics) or the richer *origin semantics*. We denote by MSO_o monadic second-order logic over o-graphs. Precisely, it is MSO equipped with monadic predicates $\sigma(x)$ for position labels, a linear order $\leq_{\texttt{in}}$ (resp. $\leq_{\texttt{out}}$) over input (resp. output) positions, and an origin function $\texttt{o}$. We denote by $[\![\phi]\!]_o$ the origin-transduction defined by ϕ, *i.e.* the set of o-graphs satisfying ϕ, and by $[\![\phi]\!]$ the transduction defined by ϕ (obtained by projecting away the origin mapping of $[\![\phi]\!]_o$). While [6] was mostly concerned with characterising classes of o-graphs generated by particular classes of transducers, the authors have shown another interesting result, namely the decidability of model-checking regular functions with origin against MSO_o properties: it is decidable, given an MSO_o sentence ϕ and a deterministic two-way transducer T, whether all o-graphs of T satisfy ϕ.

Satisfiability and synthesis. Important and natural verification-oriented questions are not considered in [6]. The first is the satisfiability problem for $\mathrm{MSO}_{\mathsf{o}}$: given a sentence ϕ, is it satisfied by some o-graph? While being one of the most fundamental problem in logic, its decidability would also entail the decidability of the equivalence problem, a fundamental problem in transducer theory: given two sentences ϕ_1, ϕ_2 of $\mathrm{MSO}_{\mathsf{o}}$, does $[\![\phi_1]\!]_o = [\![\phi_2]\!]_o$ hold? The second problem is the regular synthesis problem: given an $\mathrm{MSO}_{\mathsf{o}}$-sentence ϕ, does there exist a deterministic two-way transducer T such that (1) T has the same domain as $[\![\phi]\!]$ (the set of words which have some image by $[\![\phi]\!]$) and (2) for all u in the domain of T, its image $T(u)$ satisfies $(u, T(u)) \in [\![\phi]\!]$. Note that without requirement (1), any transducer T with empty domain would satisfy (2). So, instead of designing a transducer and then verifying a posteriori that it satisfies some $\mathrm{MSO}_{\mathsf{o}}$ properties, the goal is to check whether some transducer can be automatically generated from these properties (and to synthesise it), making it correct *by construction*. Unsurprisingly, we show that both these problems are *undecidable* for $\mathrm{MSO}_{\mathsf{o}}$.

Contribution: The fragment $\mathcal{L}_T$. We define a fragment of $\mathrm{MSO}_{\mathsf{o}}$ called $\mathcal{L}_T$ for which, amongst other interesting properties, the two problems mentioned before are decidable. Before stating our precise results on $\mathcal{L}_T$, let us intuitively define it and provide examples. $\mathcal{L}_T$ is the two-variable fragment[1] of first-order logic – FO^2. The predicates in its signature are the output labels, the linear order $\leq_{\mathsf{out}}$ for the output positions, the origin function o, and *any* binary MSO predicate restricted to input positions, using input label predicates and the input order $\leq_{\mathsf{in}}$. We write it $\mathcal{L}_T := \mathrm{FO}^2[\Gamma, \leq_{\mathsf{out}}, \mathsf{o}, \mathrm{MSO}_{\mathsf{bin}}[\leq_{\mathsf{in}}, \Sigma]]$ where Γ is the output alphabet and Σ the input alphabet.

As an example, let us define the transduction τ_{shuffle} in $\mathcal{L}_T$. We express that (1) o preserves the labelling: $\forall^{\mathsf{out}} x \bigwedge_{\sigma \in \Gamma} \sigma(x) \rightarrow \{\sigma(\mathsf{o}(x))\}$, and (2) o is bijective, *i.e.* injective: $\forall^{\mathsf{out}} x, y\ \{\mathsf{o}(x) = \mathsf{o}(y)\} \rightarrow x = y$ and surjective: $\forall^{\mathsf{in}} x \exists^{\mathsf{out}} y\ \{x = \mathsf{o}(y)\}$. The notation $\forall^{\mathsf{out}}$ is a macro which restricts quantification over output positions, and we use brackets $\{,\}$ to distinguish the binary MSO predicates. Extending this, suppose we have some alphabetic linear order $\preceq$ over Σ and we want to sort the input labels by increasing order. This can be done by adding the requirement $\forall^{\mathsf{out}} x, y \bigwedge_{\sigma \prec \sigma'} \sigma(x) \wedge \sigma'(y) \rightarrow x \leq_{\mathsf{out}} y$. This simply defined transduction can be realised by a two-way transducer, which would make one pass per symbol σ (in increasing order), during which it copies the σ-symbols on the output tape and not the others.

Results. We show the following results on $\mathcal{L}_T$:

- it is *expressive*: any regular functional transduction is definable in $\mathcal{L}_T$. Beyond functions, $\mathcal{L}_T$ is incomparable with non-deterministic two-way transducers and non-deterministic streaming string transducers (it can express τ_{shuffle} which is definable in none of these models).
- it *characterises* the regular functional transductions: a functional transduction is regular iff it is $\mathcal{L}_T$-definable. Moreover, given an $\mathcal{L}_T$-formula, it is decidable whether it defines a functional transduction.
- the satisfiability problem is decidable (in non-elementary time, which is unavoidable because of the binary MSO predicates), and ExpSpace-c if the binary MSO predicates are given by automata. Since $\mathcal{L}_T$ is closed under negation, we obtain as a consequence the decidability of the equivalence problem for $\mathcal{L}_T$-definable o-transductions.
- it admits regular synthesis: from any $\mathcal{L}_T$-sentence ϕ, one can *always* synthesise a deterministic two-way transducer which has the same domain as $[\![\phi]\!]$ and whose o-graphs all satisfy ϕ.

Finally, we provide two strictly more expressive extensions of $\mathcal{L}_T$, shown to admit regular synthesis, and hence decidable satisfiability problem. The first one $\exists\mathcal{L}_T$ extends any $\mathcal{L}_T$-formula with a block of existential monadic second-order quantifiers and it captures all transductions defined by non-deterministic MSO-transducers or equivalently non-deterministic streaming string transducers [3]. Then, we introduce $\exists\mathcal{L}_T^{\mathrm{so}}$ which extends $\exists\mathcal{L}_T$ with unary predicates $L(x)$ called *single-origin predicates*, where L is a regular language, which holds in an input position x if the word formed by the positions having origin x belongs to L. For instance one could express that any input position labelled by a has to produce a word in $(bc)^*$, which cannot be done with a FO^2 formula. This extension allows us to additionally capture any rational relation, *i.e.* the transductions defined by (nondeterministic) one-way transducers [23].

Our main and most technical result is regular synthesis. Indeed, it entails satisfiability (test domain emptiness of the constructed transducer), and, since no automata/transducer model is known to be equivalent to $\mathrm{MSO}_{\mathsf{o}}$ nor $\mathcal{L}_T$, we could not directly rely on automata-based techniques. The techniques of [6] for model-checking do not apply either because the target model is not given when considering satisfiability and synthesis. Instead, we introduce a sound and complete bounded abstraction of the o-graphs satisfying a given $\mathcal{L}_T$-formula. This abstraction was inspired by techniques used in data word logics [30], although we could not directly reuse known results, since they were only concerned with the satisfiability problem. Nonetheless, we exhibit a tight connection between o-graphs and data words.

A consequence on data words. As a side contribution, we explicit a bijection between non-erasing origin graphs (the origin mapping is surjective) and words over an infinite alphabet of totally ordered symbols, called data words. Briefly, the origin becomes the data and conversely the data becomes the origin. We show that this bijection carries over to the logical level, and we obtain a *new* decidable logic for data words, which strictly extends the logic $\mathrm{FO}^2[\leq, \preceq, S_{\preceq}]$ (linear position order and linear order and successor over data), known to be decidable from [30], with any binary $\mathrm{MSO}[\preceq]$ predicate talking only about the data.

Related Work. First, let us mention some known logical way of defining transductions. Synchronised (binary) relations, also known as automatic relations, are relations defined by automata running over word convolutions [29]. A convolution $u \otimes v$ is obtained by overlapping two words u, v and by using a padding symbol $\perp$ if they do not have the same length. E.g. $aba \otimes bc = (a,b)(b,c)(a,\perp)$. By taking MSO over word convolutions, one obtains a logic to define transductions. It is however quite weak in expressive power, as it cannot even express all functional transductions definable by one-way input-deterministic finite transducers.

Courcelle has introduced MSO-transducers to define graph transductions [9] and which, casted to words, gives a logic-based formalism to define word transductions. Roughly, the predicates of

[1] Only two variable names can be used (and reused) in a formula, see *e.g.* [32]

the output word are defined by several MSO-formulas with free variables, interpreted over a bounded number of copies of the input structure. Additionally, several free parameters can be used to add a form of non-determinism. Functional MSO-transducers correspond exactly to functional regular transduction [15]. However, they have a relatively limited expressive power when it comes to relations, because, unlike $\mathcal{L}_T$, the number of images of a word is always finite. For instance, the universal transduction $\Sigma^* \times \Sigma^*$ is not definable in this formalism, while it is simply definable by the $\mathcal{L}_T$-formula $\top$, nor is τ_{shuffle} (this can be shown using cardinality arguments).

Finally, there is a number of recent works on reactive synthesis [22], since the seminal problem raised by Church [1], and studied by Pnueli and Rosner for LTL specifications [28]. In these works however, the specification is always a synchronised relation and the target implementation is a Mealy machine (an input-deterministic finite transducer alternatively reading and producing exactly one symbol at a time). While $\mathcal{L}_T$ does not make any synchronicity assumption, the target implementations in this paper are deterministic two-way transducer which are, computationally speaking, more powerful. We leave open the question of whether the following synthesis problem is decidable: given an $\mathcal{L}_T$-formula ϕ, is there a (one-way) input-deterministic (also known as sequential) transducer realising ϕ?

Transducer synthesis is also equivalently known as *uniformisation* in transducer theory [29]. This problem has been studied in the origin-free semantics for the class of rational relations. It is known that from any rational relation one can synthesise a rational function [14], and that checking whether it is realisable by a sequential function is undecidable [8, 17]. The former result is a consequence of our results on the extension $\mathcal{L}_T^{\text{so}}$: we show that any rational relation defined by a one-way transducer is $\mathcal{L}_T^{\text{so}}$-definable (while preserving the origin mappings) and moreover, any transduction defined in $\mathcal{L}_T^{\text{so}}$ is realisable by a regular function . Hence, from rational relation given as a one-way transducer T we obtain an order-preserving and functional regular o-transduction that realises the relation defined by T. Such o-transductions are easily seen to be equivalent to rational functions [5, 16]. Finally, we mention that transducer synthesis has also been recently studied in the context of trees, where the specification is a tree automatic relation [24].

Due to the lack of space, some proofs are omitted or only sketched in the body of the paper. The full proofs are given in the appendix.

2 Logics with origin for transductions

Words and transductions. We denote by Σ^* the set of finite words over some alphabet Σ, and by ϵ the empty word. The length of a word $u \in \Sigma^*$ is denoted by $|u|$, in particular $|\epsilon| = 0$. The set of positions of u is $\text{dom}(u) = \{1, \dots, |u|\}$, an element $i \in \text{dom}(u)$ denoting the ith position of u, whose symbol is denoted $u(i) \in \Sigma$.

Let Σ and Γ be two alphabets, without loss of generality assumed to be disjoint. A *transduction* is a subset of $\Sigma^+ \times \Gamma^*$ of pairs (u, v), where u is called the input word and v the output word. An *origin mapping* from a word $v \in \Gamma^*$ to a word $u \in \Sigma^+$ is a mapping $o : \text{dom}(v) \to \text{dom}(u)$. Intuitively, it means that position i was produced when processing position $o(i)$ in the input word u. We exclude the empty input word from the definition of transductions, because we require every output position to have some origin. This does not weaken the modelling power of the logics we consider, up to putting some starting marker for instance. Following the terminology of [6], an *origin-graph* (o-graph for short) is a pair $(u, (v, o))$ such that $(u, v) \in \Sigma^+ \times \Gamma^*$ and o is an origin mapping from v to u. We denote by $\mathcal{OG}(\Sigma, \Gamma)$ the set of o-graphs from Σ to Γ. A *transduction with origin* (or just o-transduction) τ from Σ to Γ is a set of o-graphs. We say that τ is functional (or is a function) if for all u, there is at most one pair (v, o) such that $(u, (v, o)) \in \tau$, and rather denote it by f instead of τ. The *domain* of an o-transduction τ is the set $\text{dom}(\tau) = \{u \mid \exists (u, (v, o)) \in \tau\}$. Finally, the *origin-free projection* of τ is the transduction $\{(u, v) \mid \exists (u, (v, o)) \in \tau\}$. Many results of this paper hold with or without origins. We always state them in their strongest version, usually without origin.

Regular functional transductions. Regular functional transductions (or regular functions) have many characterisations, as mentioned in the introduction. We will briefly define them as the transductions definable by deterministic two-way transducers, which are pairs (A, ρ) such that A is a deterministic two-way automaton with set of transitions Δ, and ρ is a morphism of type $\Delta^* \to \Gamma^*$. The transduction defined by (A, ρ) has domain $L(A)$ (the language recognised by A) and for all words u in its domain, the output of u is the word $\rho(r)$, where r is the accepting sequence of transitions of A on u. Such transducers (as well as other known equivalent models) can be naturally equipped with an origin semantics [5] and we say that a functional *o*-transduction is regular if it is equal to the set of *o*-graphs of some deterministic two-way transducer.

FO and MSO logics for transductions. We consider FO and MSO over particular signatures. Without defining their syntax formally (we refer the reader *e.g.* to [32]), recall that MSO over a set of predicates S allows for first-order quantification $\exists x$ over elements, second-order quantification $\exists X$ over element sets, membership predicates $x \in X$, predicates of S and all Boolean operators. We use the notation $\text{MSO}[S]$ (or $\text{FO}[S]$) to emphasise that formulas are built over a particular signature S. As usual, $\phi(x_1, \dots, x_n)$ denotes a formula with n free first-order variables, and we call *sentence* a formula without free variables. Finally, $\models$ denotes the satisfiability relation.

Origin-graphs $(u, (v, o))$ of $\mathcal{OG}(\Sigma, \Gamma)$ are seen as *structures* with domain $\text{dom}(u) \uplus \text{dom}(v)$ over the signature $\mathcal{S}_{\Sigma,\Gamma}$ composed of unary predicates $\delta(x)$, for all $\delta \in \Sigma \cup \Gamma$, holding true on all positions labelled δ, $\leq_{\texttt{in}}$ a linear-order on the positions of u, $\leq_{\texttt{out}}$ a linear-order on the positions of v, and $\texttt{o}$ a unary function for the origin, which is naturally interpreted by o over $\text{dom}(v)$, and as the identity function[2] over $\text{dom}(u)$. We also use the predicates $=$, $<_{\texttt{in}}$ and $<_{\texttt{out}}$, which are all definable in the logics we consider. We denote by $\text{MSO}_{\texttt{o}}$ the logic $\text{MSO}[\mathcal{S}_{\Sigma,\Gamma}]$. Any $\text{MSO}_{\texttt{o}}$ sentence ϕ defines an o-transduction $[\![\phi]\!]_o = \{(u, (v, o)) \in \mathcal{OG}(\Sigma, \Gamma) \mid (u, (v, o)) \models \phi\}$ and its origin-free counterpart $[\![\phi]\!]$. An o-transduction (resp. transduction) τ is $\text{MSO}_{\texttt{o}}$-definable if $\tau = [\![\phi]\!]_o$ (resp. $\tau = [\![\phi]\!]$) for some sentence $\phi \in \text{MSO}_{\texttt{o}}$.

Example 1. First, we define the transduction τ_{cfl} mapping $a^n b^n$ to $(ab)^n$, as the origin-free projection of the set of o-graphs defined by some $\text{MSO}_{\texttt{o}}$-sentence ϕ_{cfl}, which expresses that (1) the domain is in $a^* b^*$, (2) the codomain in $(ab)^*$ (both properties are regular and, hence, respectively $\text{MSO}[\Sigma, \leq_{\texttt{in}}]$- and $\text{MSO}[\Gamma, \leq_{\texttt{out}}]$-definable), and (3) the origin-mapping is bijective and label-preserving (see introduction).

[2] As functional symbols must be interpreted by total functions, we need to interpret $\texttt{o}$ over $\text{dom}(u)$ as well.

Satisfiability and synthesis problems. The satisfiability problem asks, given an $\mathrm{MSO_o}$-sentence ϕ, whether it is satisfied by some o-graph, *i.e.* whether $[\![\phi]\!]_o \neq \emptyset$ (or equivalently $[\![\phi]\!] \neq \emptyset$) holds. By encoding the Post Correspondence Problem, we show that $\mathrm{MSO_o}$ has undecidable satisfiability problem, even if restricted to the two-variable fragment of FO with the $S_{\mathtt{out}}$ predicate, denoting the successor relation over output positions:

Proposition 2. *Over o-graphs, the logic* $\mathrm{FO}^2[\Sigma, \Gamma, \leq_{\mathtt{in}}, \leq_{\mathtt{out}}, S_{\mathtt{out}}, \mathtt{o}]$ *has undecidable satisfiability problem.*

Given a transduction τ and a functional transduction f, we say that f *realises* τ if $\mathrm{dom}(f) = \mathrm{dom}(\tau)$, and for all input u, $(u, f(u)) \in \tau$. The *regular synthesis problem* asks whether given an o-transduction τ, there exists a regular functional o-transduction f which realises it. As claimed in the introduction, this problem is undecidable when τ is defined in $\mathrm{MSO_o}$.

Proposition 3. *The regular synthesis problem is undecidable for* $\mathrm{MSO_o}$*-definable transductions.*

Sketch. We reduce the $\mathrm{MSO_o}$ satisfiability problem. First, consider the $\mathrm{MSO_o}$-sentence ϕ_{cfl} of Ex. 1 defining a transduction with non-regular domain. Then, given an $\mathrm{MSO_o}$-formula ψ of which one wants to check satisfiability, we define in $\mathrm{MSO_o}$, using ψ and ϕ_{cfl}, the transduction τ mapping any word $u_1 \# u_2$ to $v_1 \# v_2$ such that $(u_1, v_1) \in [\![\psi]\!]$ and $(u_2, v_2) \in [\![\phi_{\mathrm{cfl}}]\!]$. Then, $\mathrm{dom}(\tau)$ is non-regular iff it is nonempty. Since regular functions have regular domains, τ is realisable by a regular function iff $\mathrm{dom}(\tau) = \emptyset$ iff $[\![\psi]\!] = \emptyset$ iff $[\![\psi]\!]_o = \emptyset$. □

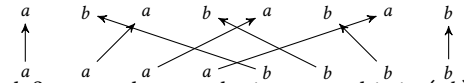
The logic $\mathcal{L}_T$ for transductions. Informally, the logic $\mathcal{L}_T$ extends the two-variable logic $\mathrm{FO}^2[\Sigma, \Gamma, \leq_{\mathtt{in}}, \leq_{\mathtt{out}}, \mathtt{o}]$ with any binary predicate definable in $\mathrm{MSO}[\leq_{\mathtt{in}}, \Sigma]$, *i.e.* any binary MSO predicate that is only allowed to talk about the input positions, in order to capture regular input properties. Formally, we denote by $\mathrm{MSO_{bin}}[\leq_{\mathtt{in}}, \Sigma]$ the set of n-ary predicates, $n \in \{0, 1, 2\}$, denoted by $\{\phi\}$, where ϕ is an $\mathrm{MSO}[\leq_{\mathtt{in}}, \Sigma]$-formula with at most n free first-order variables. Over a word u, such a formula ϕ defines an n-ary relation $R_{\phi,u}$ on its position, and over an o-graph $(u, (v, o))$, we interpret $\{\phi\}$ by $R_{\phi,u}$. The logic $\mathcal{L}_T$ is the two-variable fragment of first-order logic over the output symbol predicates, the linear-order $\leq_{\mathtt{out}}$, and all predicates in $\mathrm{MSO_{bin}}[\leq_{\mathtt{in}}, \Sigma]$, *i.e.* $\mathcal{L}_T := \mathrm{FO}^2[\Gamma, \leq_{\mathtt{out}}, \mathtt{o}, \mathrm{MSO_{bin}}[\leq_{\mathtt{in}}, \Sigma]]$. Modulo removing the brackets $\{, \}$, it is a proper fragment of $\mathrm{MSO_o}$.

Examples of $\mathcal{L}_T$-transductions. The true formula $\top$ is satisfied by any o-graph. Hence $[\![\top]\!] = \Sigma^+ \times \Gamma^*$. Let us now define several macros that will be useful throughout the paper. The formula $\mathtt{in}(x) \equiv x \leq_{\mathtt{in}} x$ (resp. $\mathtt{out}(x) \equiv x \leq_{\mathtt{out}} x$) holds true if x belongs to the input word (resp. output word). Now for $\alpha \in \{\mathtt{in}, \mathtt{out}\}$, we define the guarded quantifiers $\exists^\alpha x\ \phi$ and $\forall^\alpha x\ \phi$ as shortcuts for $\exists x\ \alpha(x) \wedge \phi$ and $\forall x\ \alpha(x) \rightarrow \phi$ (note that $\neg \exists^\alpha x\ \phi$ is equivalent to $\forall^\alpha x\ \neg\phi$).

Preservation of the input/output orders is expressed by the $\mathcal{L}_T$-formula $\forall^{\mathtt{out}} x, y\ (x \leq_{\mathtt{out}} y) \rightarrow \{x' \leq_{\mathtt{in}} y'\}(\mathtt{o}(x), \mathtt{o}(y))$. Note that we could equivalently replace x' and y' by any variable (even x and y), without changing the semantics: the formula $x' \leq_{\mathtt{in}} y'$ defines a binary relation on the input word, which is used as an interpretation of the predicate $\{x' \leq_{\mathtt{in}} y'\}$ in o-graphs. To ease the notations, any predicate $\{\phi\}(t_1, t_2)$ where ϕ has two free variables x_1 and x_2 may be sometimes written $\{\phi[x_1/t_1, x_2/t_2]\}$, *i.e.* ϕ in which t_i has been substituted for x_i. We keep the brackets { and } to emphasise the fact that it is a binary MSO formula which speaks about the input word. Hence, the previous formula may also be written $\phi_{\mathrm{pres}} \equiv \forall^{\mathtt{out}} x, y\ (x \leq_{\mathtt{out}} y) \rightarrow \{\mathtt{o}(x) \leq_{\mathtt{in}} \mathtt{o}(y)\}$.

The fact that o is a bijective mapping is expressible by some $\mathcal{L}_T$-formula ϕ_{bij}, as seen in the introduction. Then, the shuffle transduction τ_{shuffle} is defined by $\phi_{\mathrm{shuffle}} \equiv \phi_{\mathrm{bij}} \wedge \forall^{\mathtt{out}} x \bigwedge_{\sigma \in \Gamma} \sigma(\mathtt{o}(x)) \rightarrow \sigma(x)$. If the origin mapping is also required to be order-preserving, we get a formula defining identity: $\phi_{\mathrm{id}} \equiv \phi_{\mathrm{shuffle}} \wedge \phi_{\mathrm{pres}}$.

Let us now consider the transduction $\tau : (ab)^n \mapsto a^n b^n$. By taking any bijective and label-preserving origin mapping, *e.g.* as follows:



one can define τ, as long as the input word is in $(ab)^*$, which is regular, hence definable by some $\mathrm{MSO}[\leq_{\mathtt{in}}, \Sigma]$-formula $\phi_{(ab)^*}$. Then, τ is defined by: $\{\phi_{(ab)^*}\} \wedge \phi_{\mathrm{bij}} \wedge \bigwedge_{\alpha \in \{a,b\}} \forall^{\mathtt{out}} x\ \alpha(x) \rightarrow \{\alpha(\mathtt{o}(x))\} \wedge \forall^{\mathtt{out}} x, y\ a(x) \wedge b(y) \rightarrow x \leq_{\mathtt{out}} y$. More generally, one could associate with any word $(ab)^n$ the set of all well-parenthesised words of length n over Γ.

Remark 4. According to the previous examples, one can express in $\mathcal{L}_T$ the transduction τ_1 defined as the shuffle over the language a^*b^*, and also $\tau_2 : (ab)^n \mapsto a^n b^n$. Hence the composition $\tau_2 \circ \tau_1 : a^n b^n \mapsto a^n b^n$ has a non-regular domain. However, as we will see in Section 3, the domain of an $\mathcal{L}_T$-transduction is always regular, which means that $\mathcal{L}_T$-transductions are not closed under composition.

3 Expressiveness, satisfiability and synthesis

3.1 Expressiveness of $\mathcal{L}_T$

Our first result is that $\mathcal{L}_T$ can express all regular functions. To show this result, we use their characterisation as deterministic MSO-transducers [15]. We briefly recall that an MSO-transducer is defined by some $\mathrm{MSO}[\leq_{\mathtt{in}}, \Sigma]$-formulas interpreted over the input word structure (with linear order denoted here by $\leq_{\mathtt{in}}$), which specify the predicates of the output word structure, the domain of which are copies of the input nodes. More precisely, a constant k specifies the number of copies of the input word structure, $\mathrm{MSO}[\leq_{\mathtt{in}}]$-formulas $\phi^c_{pos}(x)$ specify whether the cth copy of node x is kept in the output structure, monadic formulas $\phi^c_\gamma(x)$ for each copy $c \in \{1, \ldots, k\}$ and $\gamma \in \Gamma$, specify whether the cth copy of input node x is labelled γ in the output structure, and ordering formulas $\phi^{c,d}_{\leq_{\mathtt{out}}}(x, y)$, say if the cth copy of x is before the dth copy of y in the output.

Theorem 5. *Any regular function is* $\mathcal{L}_T$*-definable.*

Sketch of proof. Let f be a regular function. Since it is regular, there exists an MSO-transducer defining it. We convert it into an $\mathcal{L}_T$-formula. First, it is not difficult to define an $\mathrm{MSO}[\leq_{\mathtt{in}}, \Sigma]$-formula $\phi_{c_1,\ldots,c_l,v}(x)$, $c_1, \ldots, c_l \in \{1, \ldots, k\}$ and $v \in \Gamma^*$, which holds true if and only if in the output structure generated by the MSO-transducer, the copies of x that are used are exactly $c_1, \ldots, c_l$, they occur in this order in the output structure, and they are respectively labelled $v(1), \ldots, v(l)$. In other words, input position x generates the subword v in the output structure. Then, we define $\mathcal{L}_T$-formulas $C_i(x)$, for all $i \in \{1, \ldots, k\}$ and x an output node (in the o-graph),

which hold, respectively, iff x is the ith node (in the output order) whose origin is $\mathsf{o}(x)$. This can be done using only two variables: $C_1(x) \equiv \mathsf{out}(x) \wedge \forall^{\mathsf{out}} y,\ y <_{\mathsf{out}} x \rightarrow \{\mathsf{o}(x) \neq \mathsf{o}(y)\}$ and for $i \geq 1, C_{i+1}(x) \equiv$

$$\exists^{\mathsf{out}} y\ (y <_{\mathsf{out}} x \wedge \{\mathsf{o}(x)=\mathsf{o}(y)\} \wedge C_i(y)) \wedge \big(\forall y\ (y < x \wedge \{\mathsf{o}(x)=\mathsf{o}(y)\}) \rightarrow \neg(\exists^{\mathsf{out}} x\ (x <_{\mathsf{out}} y \wedge \{\mathsf{o}(x)=\mathsf{o}(y)\} \wedge C_i(x)))\big)$$

Finally, we construct the final $\mathcal{L}_T$-formula (omitting some minor details) as a conjunction, for all $m, l \leq k$, all copies $c_1, \ldots, c_l$ and $d_1, \ldots, d_m$, all words $v \in \Gamma^l$ and $w \in \Gamma^m$, all $i \leq l$ and $j \leq m$, of the formulas:

$$\forall^{\mathsf{out}} x, y\ \big(\{\phi_{\leq}^{c_i,d_j}(\mathsf{o}(x),\mathsf{o}(y)) \wedge \phi_{c_1,\ldots,c_l,v}(\mathsf{o}(x)) \wedge \phi_{d_1,\ldots,d_l,w}(\mathsf{o}(y))\} \wedge C_i(x) \wedge C_j(y)\big) \rightarrow (x \leq_{\mathsf{out}} y \wedge v(i)(x) \wedge w(j)(y))$$

□

MSO-transducers have been extended with nondeterminism (NMSO-transducers or just NMSOT) to express non-functional transductions, by using a set of monadic second-order parameters $X_1, \ldots, X_n$ [15]. Each formula of an NMSO-transduction can use $X_1, \ldots, X_n$ as free variables. Once an interpretation for these variables as sets of positions has been fixed, the transduction becomes functional. Therefore, the maximal number of output words for the same input word is bounded by the number of interpretations for $X_1, \ldots, X_n$. NMSO-transducers are linear-size increase (the length of any output word is linearly bounded by the length of the input word), hence the universal transduction $\Sigma^+ \times \Gamma^*$ is not definable in NMSO, while it is $\mathcal{L}_T$-definable by $\top$. The shuffle transduction is not definable in NMSOT as well (this can be shown by cardinality arguments). Conversely, it turns out that a transduction like (u, vv) where v is a subword of u of even length is not $\mathcal{L}_T$-definable whereas is it in NMSOT.

Rational relations are transductions defined by (non-deterministic) finite transducers (finite automata over the product monoid $\Sigma^* \times \Gamma^*$), denoted 1NFT [23]. This class is incomparable with $\mathcal{L}_T$: the shuffle is not a rational relation, while the relation $\{a\} \times L$, where L is a non-FO^2-definable regular language is not $\mathcal{L}_T$-definable. Indeed, when all inputs are restricted to the word a, the expressive power of $\mathcal{L}_T$ is then restricted to FO$^2[\leq_{\mathsf{out}}, \Gamma]$ over the output.

Non-deterministic two-way transducers (2NFT), are incomparable to NMSO [15], and also to $\mathcal{L}_T$, since they extend 1NFT and cannot define the shuffle transduction. Fig. 2 depicts these comparisons, summarised by the following proposition:

Proposition 6. *The classes of $\mathcal{L}_T$, 2NFT (resp. 1NFT), and NMSOT-definable transductions are pairwise incomparable.*

3.2 Satisfiability and equivalence problems

Our first main contribution is the following result, whose proof is sketched in Section 4. Here and throughout the paper, by effectively we mean that the proof effectively constructs a finite object.

Proposition 7. *The input domain of any $\mathcal{L}_T$-transduction is (effectively) regular.*

Theorem 8. *Over o-graphs, the logic $\mathcal{L}_T$ has decidable satisfiability problem.*

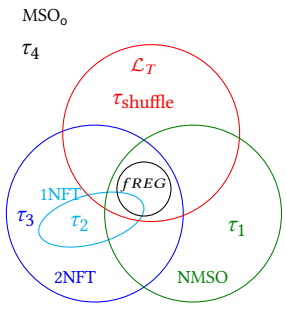


Figure 2. Expressiveness of $\mathcal{L}_T$, compared to MSO$_o$, non-deterministic MSO transductions, non-deterministic one-way and two-way transducers and regular functions. Here, $\tau_1 = \{(u, vv) \mid v \text{ is a subword of } u \text{ of even length}\}$, $\tau_2 = \{a\} \times (ab)^*$, $\tau_3 = \{(u, u^n) \mid n \geq 0\}$ and $\tau_4 = \{a^n b^n, (ab)^n \mid n > 0\}$.

This latter theorem is a consequence of Thm 9. We point out that it holds also for origin-free transductions, because given an $\mathcal{L}_T$-formula ϕ, $[\![\phi]\!] = \varnothing$ iff $[\![\phi]\!]_o = \varnothing$. The *equivalence problem* asks, given two formulas ϕ_1, ϕ_2, whether $[\![\phi_1]\!]_o = [\![\phi_2]\!]_o$, *i.e.* whether $\phi_1 \leftrightarrow \phi_2$ is universally true. As a consequence of Thm. 8 and closure under negation of $\mathcal{L}_T$ we have the decidability of the equivalence problem for $\mathcal{L}_T$.

With respect to satisfiability, $\mathcal{L}_T$ seems to lie at the decidability frontier. Adding just the successor relation over outputs already leads to undecidability, by Prop. 2.

3.3 Regular synthesis of $\mathcal{L}_T$ and consequences

Our main result is the regular synthesis of $\mathcal{L}_T$-transductions.

Theorem 9 (Regular synthesis of $\mathcal{L}_T$)**.** *Let ϕ be an $\mathcal{L}_T$ formula. The transduction defined by ϕ is (effectively) realisable by a regular function.*

In other words, from any specification ϕ written in $\mathcal{L}_T$, one can synthesise a functional transduction f, in the proof represented by an MSO-transducer T, such that $\mathrm{dom}(f) = \mathrm{dom}([\![\phi]\!])$ and $f = [\![T]\!] \subseteq [\![\phi]\!]$. Moreover, it turns out that the constructed transducer T defines a functional o-transduction $[\![T]\!]_o$ such that $[\![T]\!]_o \subseteq [\![\phi]\!]_o$. In other words, T does not change the origins specified in ϕ. Since we rely on MSO-to-automata translation in the construction, the size of the constructed MSO-transducer is non-elementary in the size of ϕ. One of the main consequences of the synthesis and expressiveness results is a new characterisation of the class of regular functions.

Theorem 10 (New characterisation of regular functions)**.** *Let $f : \Sigma^* \rightarrow \Gamma^*$. Then, f is regular iff f is $\mathcal{L}_T$-definable.*

Proof. By Thm. 5, f regular implies f is $\mathcal{L}_T$-definable, which implies by Thm. 9 that f is regular. □

A consequence of synthesis is the following positive result on functionality:

Corollary 11 (Functionality)**.** *Given an $\mathcal{L}_T$-sentence ϕ, it is decidable whether the o-transduction $[\![\phi]\!]_o$ is functional.*

Proof. To test whether $[\![\phi]\!]_o$ is functional, first realise it by a regular function (Thm. 9), defined *e.g.* by a deterministic two-way

transducer T, and then test whether $[\![\phi]\!]_o \subseteq [\![T]\!]_o$. The latter is decidable since T can be converted (while preserving origins) into an equivalent $\mathcal{L}_T$-formula ψ (Thm. 5) and test that $\phi \to \psi$ is satisfiable (Thm.8). □

4 Domain regularity and synthesis: sketch of proofs

In this section, we sketch the proofs of Prop. 7 (domain regularity of $\mathcal{L}_T$-transductions) and Thm. 9 (regular synthesis). These two results are based on common tools which we now describe. We let ϕ be an $\mathcal{L}_T$-sentence over input and output alphabets Σ, Γ respectively. We assume that $\mathcal{L}_T$ defines a non-erasing o-transduction, *i.e.* an o-transduction which uses every input position at least once (the origin mapping is surjective). This can be done without loss of generality, *i.e.* one can transform in polynomial time any $\mathcal{L}_T$-sentence into a non-erasing one (by adding dummy output positions the origins of which are the erased input positions), while preserving the domain and set of regular functions realising it (modulo the previous encoding).

Scott normal form. The $\mathcal{L}_T$ formula ϕ is then transformed into a Scott normal form (SNF), a standard transformation when dealing with two-variable logics (see for instance [19]). By enriching the alphabet, the transformation allows to restrain ourself to the easier setting of formulas of quantifier-depth two. Precisely, we obtain a formula of the form:

$$\forall^{\mathtt{out}} x, y\ \psi(x,y) \wedge \bigwedge_{i=1}^{m} \forall^{\mathtt{out}} x \exists^{\mathtt{out}} y\ \psi_i(x,y)$$

where the formulas ψ and ψ_i, $i = 1, \ldots, m$, are quantifier free, but over an extended output alphabet $\Gamma \times \Gamma'$ (where Γ' may be exponential in ϕ). These subformulas can also still contain binary MSO predicates over the input, which are not restricted in any way. Up to projection over Γ, the SNF formula accepts the same models as ϕ, and hence we now just assume that ϕ is a formula of the above form over an input alphabet Σ and output alphabet Γ. In the full proof (Appendix), the SNF is further equivalently transformed into what we call a system of universal and existential constraints (in the vein of [30]), which are easier to manipulate in the proofs than the formulas ψ and ψ_i, but are not necessary at a conceptual level, so we do not include them in the sketch.

The profile abstraction. We define an abstraction which maps any o-graph $(u, (v, o))$ to a sequence of $|u|$ tuples $\lambda_1 \ldots \lambda_{|u|}$ called *profiles*, one for each input position. A profile contains bounded information (bounded in the size of ϕ) about the binary input MSO predicates, the input symbol and some output positions. To explain this abstraction, we first informally define what we call the *full graph* of an o-graph $(u, (v, o))$. Intuitively, the full graph contains a node for each pair $(p, p') \in \mathrm{dom}(u) \times \mathrm{dom}(v)$, labelled by some information called *clause* about the "effect" of position p' at position p. To understand it, it is convenient to see the full graph as a two-dimensional structure with the input position as x-axis (ordered by $\leq_{\mathtt{in}}$) and the output position as the y-axis (ordered by $\leq_{\mathtt{out}}$). Figure 3 shows such a representation. *E.g.* the top-left figure represents the full graph of an o-graph which translates $\sigma_1 \ldots \sigma_5$ into $(\beta\gamma)^3$ (for instance, the origin of the last output position, labelled γ, is the third input position, labelled σ_3), plus some additional information which we now detail.

Each row contains a single node labelled in Γ, corresponding to an output position, and placed according to its origin. Let (p, p') (output position p' with origin p) be such a node, labelled by some $\gamma \in \Gamma$. This node generates an horizontal trace around it, whose elements are of the form $\gamma \overleftarrow{R}$ or $\gamma \overrightarrow{R}$. The arrows indicate in which direction the γ-labelled node is. The elements R, say at coordinates (s, p'), are MSO$[\Sigma, \leq_{\mathtt{in}}]$-types (of bounded quantifier rank) talking about the input word u with the positions s and p marked. In the proof, we represent these MSO-types as state information of node selecting automata (or query automata, see *e.g.* [27]). The idea behind this information is that, by looking independently at each column of the full graph of an o-graph, it is possible to decide whether this o-graph satisfies ϕ. Suppose for instance we want to check whether the o-graph satisfies a formula of the form $\forall^{\mathtt{out}} x \exists^{\mathtt{out}} y \cdot \gamma(x) \to \gamma'(y) \wedge y <_{\mathtt{out}} x \wedge \{\xi\}(\mathtt{o}(y), \mathtt{o}(x))$. Then, for every column containing a γ-labelled node, say at coordinate (p, p'), one has to check that there exists a node in the same column, say at position (p, p''), labelled by some $(\gamma' \overleftarrow{R})$ or some $(\gamma' \overrightarrow{R})$, such that $p'' < p'$ and R satisfies ξ. Suppose that in the SNF we also have a conjunct of the form $\forall^{\mathtt{out}} x, y \cdot (\gamma(x) \wedge \gamma'(y) \wedge \{\mathtt{o}(x) <_{\mathtt{out}} \mathtt{o}(y))\} \to \{\xi'\}(\mathtt{o}(x), \mathtt{o}(y))$, then we must additionally checks that for every column, for every γ-labelled node in this column and every $\gamma' \overrightarrow{R}$-labelled node on the same column, R satisfies ξ'. We call a column which satisfies the SNF formula ϕ a *valid* column.

A *key property* we now use is that, if on a column there exists at least three nodes with the same label, then removing all but the smallest and greatest (in output order) of these nodes does not influence the validity of the column. It is easy to see for subformulas of ϕ of the form $\forall^{\mathtt{out}} x, y\ \psi(x,y)$ (removing nodes makes such a formula "easier" to satisfy). For subformulas of the form $\forall^{\mathtt{out}} \exists^{\mathtt{out}} y\ \psi_i(x,y)$, it is due to the fact that ψ_i is quantifier-free, and therefore it is safe to keep only the extremal witnesses y for x.

This observation leads us to the notion of *abstract graph*, the subgraph of the full graph obtained by keeping only the extremal occurrences of every node with same labels. Figure 3 illustrates this abstraction, on hypothetical full graphs where label equalities have been underlined. Each column indexed by position p of this abstract graph, together with the input symbol, is what we call the *profile* of p. Note that this is a bounded object. Then, to any o-graph one can associate a sequence of profiles this way, but this association is not injective in general since we may lose information, as shown in the figure. Put differently, the abstract graph can in general be concretised in more than one full graph.

Properties of profile sequences. The key ingredient of the proof is to define properties on profile sequences s (which are nothing but words over the finite alphabet of profiles), that can be checked in a regular manner (by an automaton) so that there exists at least one o-graph g such that (1) s is the profile sequence of g and (2) $g \vDash \phi$. Property (2) is ensured by the notion of validity defined before, and by a notion of *maximality* for the MSO-types R (no information can be withheld). Property (1) is ensured by a notion of consistency between profiles. Intuitively, it asks that the information declared in one profile is consistent, in some precise way, with the information declared in the next profile. Roughly, since we use automata to represent the information R, one step consistency corresponds to one step in the runs of the automata. Maximal and consistent sequences of valid profiles are called *good* profile sequences.

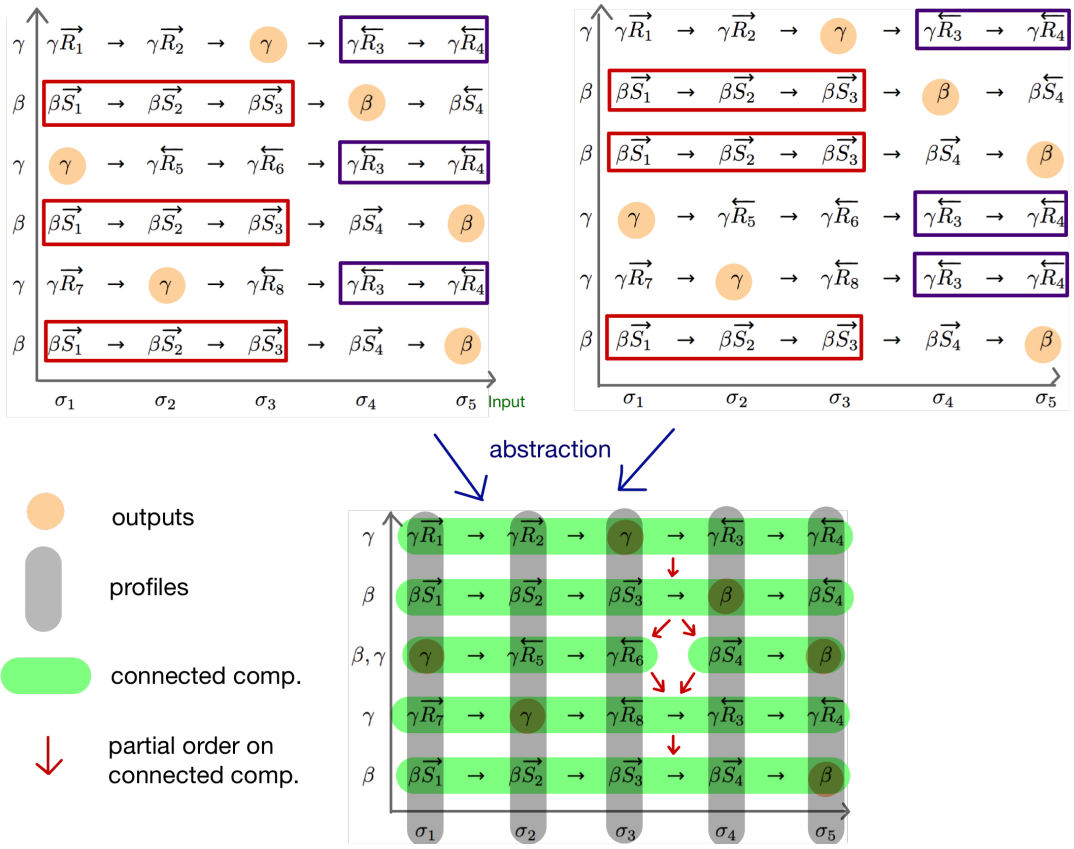


Figure 3. The profile abstraction and the graph of clauses

We then prove a completeness result: the profile sequence of any model of ϕ is good. We also prove a soundness result: any good profile sequence is the profile sequence of at least one model of ϕ. As a matter of fact, we prove a slightly stronger result which allows one to recover not just one but potentially several models of ϕ. As illustrated on the figure, every connected component of the abstract graph corresponds to exactly one node labelled in Γ. The notion of consistency ensures this property as well, and, as a matter of fact, the output positions of the models we reconstruct out of good profile sequences are in bijection with these connected components (CC). We can even order them partially, as illustrated on the figure, by overlapping: a CC is the successor of another one if they overlap horizontally, and the former is above the latter (again, our definition of consistency ensures that there is no "crossing" in abstract graphs, hence this relation can indeed be shown to be a partial order). Hence, a good profile sequence defines an abstract graph which gives us: the input position with their labels, the output positions with their labels and origins, and some partial order between these output positions. What's missing is a linear order on these output positions, but we prove that *any* linearisation of this partial order actually defines an o-graph which satisfies ϕ. Coming back to the example, the two possibly linearisations give the output words $\beta\gamma\beta\beta\gamma$ and $\beta\gamma\gamma\beta\gamma$.

Back to the theorems. To show domain regularity (Prop. 7), we observe that the domain is the projection on input alphabet Σ of the set of good profile sequences, which turns out to be regular (the whole point of defining the notions of validity, maximality and consistency is that they can be checked by an automaton). Since regular languages are closed under projection, we get the result.

Showing regular synthesis (Thm. 9) is a bit more technical. The main idea is to show that the mapping which takes as input a word u over Σ, and which outputs all the abstract graphs of o-graphs which satisfy ϕ and have u as input, is definable by a non-deterministic MSO word-to-DAG transduction T_1. It is possible since the notions of consistency, maximality and validity are all MSO-definable, and an abstract graph is always a DAG. Then, we use a result of Courcelle which states that there exists a deterministic MSO DAG-to-word transduction R_2 which, given a DAG, produces a topological sort of it [10]. The DAG additionally needs to be *locally ordered* (the successors of a node are linearly ordered), but we can ensure this property in our construction. Then, we use closure under composition of NMSOT to show that $R_2 \circ R_1$ is definable by some word-to-word NMSOT, which can be easily realised by a (deterministic) MSOT, concluding the proof.

Comparison with [30]. We would like to point out that this proof was inspired by a decidability proof for the logic $\mathrm{FO}^2[\leq, \preceq, S_\preceq]$ over data words (a linear order over positions and a linear order and

successor over data). We somehow had to cast it to transductions, and extend it with binary MSO predicates. Moreover, further manipulations and notions were needed to extract the synthesis result. In particular, the ideas of using Scott normal form, to see o-graphs as two-dimensional structures, and the abstraction, were directly inspired from [30].

5 A decidable logic for typed data words

We make here a bijective connection between o-transductions and what we call typed data words, which slightly generalise data words, and introduce a new decidable logic $\mathcal{L}_D$ for typed data words, whose decidability stems from the equivalence with $\mathcal{L}_T$.

Typed data words. We consider *typed data words* over an ordered data domain, such that each datum also carries a label (type) from a finite alphabet. Formally, a typed data word of length n and *data size* m over two disjoint alphabets Γ and Σ is a word over the alphabet $\Gamma \times \mathbb{N} \times \Sigma$, $w = (\gamma_1, d_1, \sigma_1)\cdots(\gamma_n, d_n, \sigma_n)$ verifying the following properties: d_i is called the *datum of position* i, we have that $\{d_1, \ldots, d_n\} = \{1, \ldots, m\}$ [3] and we also have for any positions i, j that $d_i = d_j \implies \sigma_i = \sigma_j$, hence σ_i is called the *type of datum* d_i.

We denote by $\mathcal{TDW}(\Sigma, \Gamma)$ the set of typed data words over alphabets Σ, Γ of *any* length n and *any* data size m.

The data of a typed data word w induce a total preorder $\preceq$ over the positions of w defined by $i \preceq j$ if $d_i \leq d_j$. This preorder induces itself an equivalence relation $\sim$ defined by $i \sim j$ iff $i \preceq j$ and $j \preceq i$, which means that the positions i and j carry the same datum. Hence, a typed data word will equivalently be seen as a structure with letter predicates $\gamma \in \Gamma$, $\sigma \in \Sigma$, the linear order over positions and the total preorder $\preceq$ previously defined.

The logic $\mathcal{L}_D$ for typed data words. It is known from [7] that the logic MSO over untyped data words (*i.e.* $|\Sigma| = 1$) is undecidable (even the first-order fragment). We consider here a decidable fragment, over typed data words, called $\mathcal{L}_D$. A formula of $\mathcal{L}_D$ can be seen as an FO^2 formula using the linear order of the positions and some additional binary data predicates. The logic $\mathcal{L}_D$ is indeed built on top of MSO n-ary predicates, for $n \leq 2$, which are allowed to speak only about the data. Precisely, we define $\mathrm{MSO}_{\mathrm{bin}}[\Sigma, \preceq]$ to be the set of n-ary predicates written $\{\phi\}$, for $n \leq 2$, where ϕ is an MSO-formula with n-free first-order variables, over the unary predicates $\sigma(x)$ and the preorder $\preceq$, with the following semantic restriction[4]: second-order variables are interpreted by $\sim$-closed sets of positions. Over typed data words, predicates $\{\phi\}$ are interpreted by relations on positions defined by formulas ϕ.

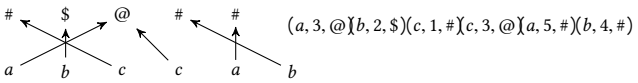
Due to the semantic restriction, formulas in $\mathrm{MSO}_{\mathrm{bin}}[\Sigma, \preceq]$ cannot distinguish positions with the same data and therefore, they can be thought of as formulas which quantify over data and sets of data. As an example, the formula $\forall y\ x \preceq y$ expresses that the datum of position x is the smallest, and it holds true for any x' with the same datum. Then, the logic $\mathcal{L}_D$ is defined as $\mathcal{L}_D := \mathrm{FO}^2[\Gamma, \leq, \mathrm{MSO}_{\mathrm{bin}}[\Sigma, \preceq]]$.

[3] We make this assumption without loss of generality, because the logic we define will only be able to compare the order of data, and so cannot distinguish typed data words up to renaming of data, as long as the order is preserved. *E.g.* $(a, b, 1)(c, d, 3)(e, f, 2)$ and $(a, b, 2)(c, d, 5)(e, f, 4)$ will be indistinguishable by the logic.

[4] Note that the semantic restriction could also be enforced in the logic by guarding quantifiers $\exists X \psi$ with $\exists X[\forall x \forall y\ x \in X \wedge y \sim x) \to y \in X] \to \psi$.

Example 12. First, let us mention that $\mathrm{MSO}_{\mathrm{bin}}[\Sigma, \preceq]$ predicates can express any regular properties about the data, in the following sense. Given a typed data word w, the total preorder $\preceq$ over positions of w can be seen as a total order $\preceq_\sim$ over the equivalence classes of $\mathrm{dom}(w)/_\sim$, by $[i]_\sim \preceq_\sim [j]_\sim$ if $i \preceq j$. Then, any typed data word induces a word $\sigma_1 \ldots \sigma_n \in \Sigma^*$ such that σ_i is the type of the elements of the ith equivalence class of $\preceq_\sim$. Any regular property of these induced words over Σ transfers into a regular property about the data of typed data words (it suffices to replace in the MSO-formula on Σ-words expressing the property, the linear order by $\preceq$ and the equality by $\sim$). Examples of properties are: n is even, which transfers into "there is an even number of pieces of data", or $\sigma_1 \ldots \sigma_n$ contains an even number of $\sigma \in \Sigma$, for some σ, meaning "there is an even number of pieces of data of type σ".

From transductions to data words and back. There is a straightforward encoding t2d of non-erasing o-graphs into typed data words, and conversely. A non-erasing o-graph $(u, (v, o))$, with $v = v_1 \ldots v_n$ and $u = u_1 \ldots u_m$ is encoded as the typed data word $\mathrm{t2d}((u, (v, o))) = (v_1, o(1), u_{o(1)}) \ldots (v_n, o(n), u_{o(n)})$. Given a typed data words $w = (\gamma_1, d_1, \sigma_1) \ldots (\gamma_n, d_n, \sigma_n)$, we set $\mathrm{t2d}^{-1}(w)$ the non-erasing o-graph $\mathrm{t2d}^{-1}(w) = (u, (v, o))$ such that $v = \gamma_1 \ldots \gamma_n$, $o(i) = d_i$, and if we write $d_{i_j} = j$ then $u = \sigma_{i_1} \cdots \sigma_{i_m}$ where $m = \max_i d_i$. We give here an example of this transformation:



Theorem 13. *Non-erasing o-graphs of $\mathcal{OG}(\Sigma, \Gamma)$ and typed data words of $\mathcal{TDW}(\Sigma, \Gamma)$ are in bijection by t2d. Moreover, a non-erasing o-transduction τ is $\mathcal{L}_T$-definable iff $\mathrm{t2d}(\tau)$ is $\mathcal{L}_D$-definable. Conversely, a language of typed data words L is $\mathcal{L}_D$-definable iff $\mathrm{t2d}^{-1}(L)$ is $\mathcal{L}_T$-definable.*

The main idea of the proof is to make a bijective syntactic transformation that mimics the encoding t2d: once inconsistent use of terms have been removed (such as *e.g.*, $o(x) \preceq_{\mathtt{out}} y$), terms $o^n(x)$ are replaced by x, predicates $\preceq_{\mathtt{in}}$ by $\preceq$ and $\preceq_{\mathtt{out}}$ by $\preceq$. Hence, this theorem and the decidability of $\mathcal{L}_T$ (Thm. 8) gives the following corollary.

Corollary 14. *Over typed data words, the logic $\mathcal{L}_D$ has a decidable satisfiability problem.*

As a remark, we also note that thanks to the correspondence between transductions and data words and some minor manipulations, we can also obtain the decidability of $\mathrm{FO}^2[\preceq_{\mathtt{in}}, \preceq_{\mathtt{out}}, S_{\mathtt{in}}, \mathtt{o}]$ (for $\mathtt{in}$ the input successor), which is a strict fragment of $\mathcal{L}_T$, over o-graphs from the decidability of $\mathrm{FO}^2[\preceq, \preceq, S_\preceq]$ over data words, proved in [30]. However, the logic $\mathrm{FO}^2[\preceq, \preceq, S_\preceq]$ is a strict fragment of $\mathcal{L}_D$.

6 Complexity of satisfiability

To achieve decidability results for $\mathcal{L}_T$, the binary MSO predicates over the input of $\mathcal{L}_T$-formulas are decomposed into MSO-types, that we handle using query automata, as explained in the sketch of proof in Section 4. A query automaton for a binary MSO formula $\psi(x, y)$ is a non-deterministic finite automaton $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ equipped with a set $SP \subseteq Q^2$ of *selecting pairs* with the following

property: for any word $u \in \Sigma^*$ and any pair of positions (i, j) of u, we have $u \vDash \psi(i, j)$ if, and only if, there exists an accepting run π of $\mathcal{A}$ and a pair $(p, q) \in SP$ such that π reads $u(i)$ in state p and $u(j)$ in state q. Due to the MSO-formulas and their translation into query automata, the complexity of the satisfiability of $\mathcal{L}_T$ is non-elementary, and this is unavoidable [31]. However, if the binary MSO-formulas are already given as query automata, we get a tight elementary complexity. Likewise, the binary MSO predicates of the data word logic $\mathcal{L}_D$ can be also represented as query automata, and we get the same complexity as $\mathcal{L}_T$.

Theorem 15. *The satisfiability problem of $\mathcal{L}_T$ and $\mathcal{L}_D$ is* ExpSpace*-complete when the binary MSO predicates are given as query automata.*

Sketch of proof. First, as the translation between $\mathcal{L}_T$ and $\mathcal{L}_D$ is linear, the complexity of both logics is equivalent. In showing decidability of the satisfiability of $\mathcal{L}_T$, we obtain that the set of "good" profile sequences is effectively regular by Prop. 7. With a careful analysis it is possible to construct a doubly exponential deterministic automaton recognising the good sequences. By checking emptiness on-the-fly instead of constructing the automaton, we get the NLogSpace emptiness of the automaton, and hence the ExpSpace complexity. Finally, since the logic $\mathrm{FO}^2[\Gamma, \preceq, S_\preceq, \leq]$ is ExpSpace-complete [30], we get ExpSpace-hardness as this logic is a syntactic fragment of $\mathcal{L}_D$. □

7 Decidable Extensions of $\mathcal{L}_T$

We present here two main extensions to $\mathcal{L}_T$ showing its robustness. The first one consists in adding a block of existential monadic second-order quantifiers in front of the formula while the second one consists in adding new predicates to the logic; both extensions preserve many properties of the logic which we describe below.

Existential $\mathcal{L}_T$. This new logic is denoted by $\exists\mathcal{L}_T$ and allows us to capture all non-deterministic MSO-transductions, but we lose the closure under negation of the logic. Formally, we consider all formulas of the form $\exists X_1 \dots \exists X_n \phi$ where ϕ is a formula of $\mathcal{L}_T$ which can additionally use predicates of the form $x \in X_i$. The variables X_i range over sets of output, and also input positions.

Proposition 16. *Any NMSO-transduction is $\exists\mathcal{L}_T$-definable.*

The synthesis result extends to $\exists\mathcal{L}_T$ using a quite common trick of considering for a formula $\exists X_1 \dots \exists X_n \phi$, the formula ϕ but over an extended alphabet.

Proposition 17. *Any $\exists\mathcal{L}_T$-transduction can be (effectively) realised by a regular function.*

One result of $\mathcal{L}_T$ does not carry over to $\exists\mathcal{L}_T$, namely the decidability of the equivalence problem. Indeed $\exists\mathcal{L}_T$ is not closed under negation and thus equivalence of formulas cannot be reduced to satisfiability. Equivalence turns out to be undecidable for $\exists\mathcal{L}_T$ and in fact the validity problem, which asks given a formula if it is satisfied by all o-graphs and which can be seen as the particular case of the equivalence with the formula $\top$, is itself undecidable for $\exists\mathcal{L}_T$.

Proposition 18. *The validity and equivalence problems for $\exists\mathcal{L}_T$ over o-graphs are undecidable.*

Single-origin predicates. One "weak" point of $\mathcal{L}_T$ is that if the input is restricted to, for instance, a single position, then the expressive power over the output is only $\mathrm{FO}^2[\leq_{\mathtt{out}}]$. For instance the transduction $\{a\} \times L$ is not definable if L not an $\mathrm{FO}^2[\leq_{\mathtt{out}}]$-definable language. A more general expression of this problem is that the class of transductions definable by one-way transducers, also known as rational transductions [23], is incomparable with the class of $\mathcal{L}_T$ (resp. $\exists\mathcal{L}_T$) transductions. The following extension, called $\mathcal{L}_T^{\mathrm{so}}$ adds new predicates, called here *single-origin predicates*, and we show that it captures all the rational transductions. These new predicates allow to test any regular property of a subword of the output word restricted to positions with a given origin position.

Given an o-graph $(u, (v, o))$ and an input position i of u, we denote by $v_{|i}$ the subword of v consisting of all the positions of v whose origin is i, and we call this word the *single-origin restriction of v to i*.

Given any regular language L (represented as an MSO formula for instance), we define a unary input predicate $L(x)$, whose semantics over an o-graph $(u, (v, o))$ is the set of input positions $i \in \mathrm{dom}(u)$ such that $v_{|i} \in L$. The logic $\mathcal{L}_T^{\mathrm{so}}$ (resp. $\exists\mathcal{L}_T^{\mathrm{so}}$) is the extension of $\mathcal{L}_T$ (resp. $\exists\mathcal{L}_T$) with the predicates $L(x)$, for any regular language L. These predicates can be used just as the other unary input predicates and using the previous notation we have $\mathcal{L}_T^{\mathrm{so}} := \mathrm{FO}^2[\Gamma, \leq_{\mathtt{out}}, \mathtt{o}, \mathrm{MSO}_{\mathrm{bin}}[\leq_{\mathtt{in}}, \Sigma \uplus \{L(x) \mid L \text{ regular}\}]]$. For instance, let L denote the language $(ab)^*$ then the formula $\forall^{\mathtt{out}} x\ a(x) \to \{\mathrm{even}(\mathtt{o}(x)) \wedge L(\mathtt{o}(x))\}$ states that the origin of each output position x labelled by a must be even and that the subword of origin $o(x)$ must be in L.

Proposition 19. *Any rational transduction is $\mathcal{L}_T^{\mathrm{so}}$ definable.*

Our synthesis result transfers to $\mathcal{L}_T^{\mathrm{so}}$ (and $\exists\mathcal{L}_T^{\mathrm{so}}$):

Proposition 20. *Any $\exists\mathcal{L}_T^{\mathrm{so}}$-transduction can be (effectively) realised by a regular function.*

Remark 21. From the regular synthesis of $\exists\mathcal{L}_T^{\mathrm{so}}$, we can deduce several results which we express in their strongest form: The input domain of any $\exists\mathcal{L}_T^{\mathrm{so}}$-transduction is effectively regular, the satisfiability problem for $\exists\mathcal{L}_T^{\mathrm{so}}$ is decidable, the equivalence problem for $\mathcal{L}_T^{\mathrm{so}}$ is decidable. Finally, given a functional transduction, it is regular if and only if it is $\exists\mathcal{L}_T^{\mathrm{so}}$-definable, and, given an $\exists\mathcal{L}_T^{\mathrm{so}}$ sentence ϕ, it is decidable whether $[\![\phi]\!]_o$ is functional.

Extended logics over data words. We define similarly the extensions $\exists\mathcal{L}_D$, $\mathcal{L}_D^{\mathrm{sd}}$ and $\exists\mathcal{L}_D^{\mathrm{sd}}$ of the logic $\mathcal{L}_D$ and we obtain the same transfer results as in Thm. 13. In terms of data, the single-origin predicates become *single datum predicates (sd)* which can specify any regular property over a subword induced by a single datum.

8 Summary and Discussion

In this paper, we have introduced an *expressive* logic to define transductions, which we believe is a great tool from both a theoretical and a more practical point of view. It allows for high-level specification of transductions, while having some good properties for synthesis. As an interesting side contribution, we obtain a new characterisation of the class of regular transductions, as the (functional) transductions definable in $\mathcal{L}_T$ (and its extensions up to $\exists\mathcal{L}_T^{\mathrm{so}}$). The expressiveness and decidability frontiers on the logic $\mathcal{L}_T$ and its extensions are summarised in Fig. 4. We obtained tight complexity results for satisfiability of $\mathcal{L}_T$ both in the case of binary input

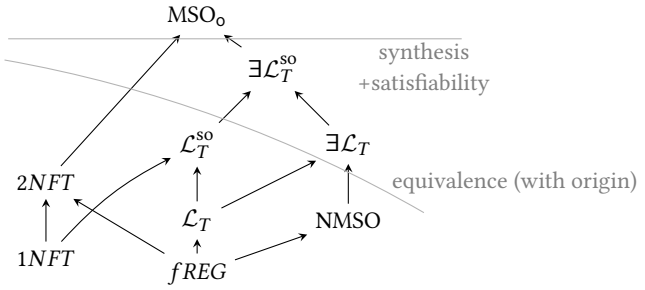


Figure 4. Summary of models for transductions and their inclusions. The lines are decidability frontiers.

predicates given by MSO-formulas (non-elementary) or automata (ExpSpace). We have also shown that slightly extending the expressiveness by adding the successor over output positions leads to undecidability.

Another question is the definition of an automata model equivalent to $\mathcal{L}_T$, or even to $\mathrm{MSO}_{\mathsf{o}}$. Automata for data words have been defined [7, 26], but none of these models capture $\mathcal{L}_D$.

The equivalence problem for $\mathcal{L}_T$ is origin-dependent. One could relax it by projecting away the origin information: given two $\mathcal{L}_T$-formulas ϕ_1, ϕ_2, are the origin-free transductions they define equal, *i.e.* $[\![\phi_1]\!] = [\![\phi_2]\!]$? This (origin-free) equivalence problem is known to be decidable for regular functions [21], and undecidable for 1NFT (and hence 2NFT) [20] as well as NMSOT [3]. It is shown by reduction from the Post Correspondence Problem and it turns out that the transductions constructed in the reduction of [20] are definable in $\mathcal{L}_T$, proving undecidability for $\mathcal{L}_T$ as well. An interesting line of research would be to consider less drastic relaxations of the equivalence problem with origin, by comparing transductions with *similar* origin, as done for instance in [17] for rational relations. Similarly, the model-checking of two-way transducers against $\mathrm{MSO}_{\mathsf{o}}$-sentences is decidable, but it is again origin-sensitive. Instead, the origin-free version of this problem is to decide whether for all the pairs of words (u, v) defined by a two-way transducer T, there exists some origin mapping o such that the o-graph (u, v, o) satisfies some formula ϕ . Once again, it is possible to show, by reducing PCP, that this relaxation yields undecidability, but it could be interesting to consider a stronger problem where the origin of T is "similar" to the origin specified in ϕ. A related problem is the satisfiability of logics where two or more origin mappings are allowed.

Another direction would be extending the logic to other structures (*e.g.* trees or infinite words), and other predicates over output positions. However, one has to be careful since the data point of view shows how close we are to undecidability (*e.g.* over data words, FO^2 with successor over data and positions is undecidable [25]).

Finally, we have established a tight connection between transductions and data words, and thus a new decidable logic for data words. The data point of view allowed us to get decidability of the transduction logic $\mathcal{L}_T$, inspired by the decidability result of [30]. Conversely, the logic $\mathcal{L}_D$ extends the known results on data words by adding MSO predicates on the ordered and labelled data. We would like to investigate if further results from the theory of transductions can be translated into interesting results in the theory of data words.

References

[1] Alonzo Church. 1962. Logic, Arithmetic and Automata. In *Int. Congr. Math.* Stockholm, 23–35.
[2] Rajeev Alur and Pavol Černý. 2010. Expressiveness of streaming string transducers. In *FSTTCS*. LIPIcs., Vol. 8:1–12. Schloss Dagstuhl.
[3] Rajeev Alur and Jyotirmoy V. Deshmukh. 2011. Nondeterministic Streaming String Transducers. In *ICALP'11*. 1–20.
[4] Rajeev Alur, Adam Freilich, and Mukund Raghothaman. 2014. Regular combinators for string transformations. In *Joint Meeting of the (CSL) conference and the (LICS) Symposium , CSL-LICS '14, Vienna, Austria,*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, 9:1–9:10.
[5] Mikolaj Bojanczyk. 2014. Transducers with Origin Information. In *ICALP'14*.
[6] Mikołaj Bojańczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle. 2017. Which Classes of Origin Graphs Are Generated by Transducers. In *ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 114:1–114:13.
[7] Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. 2006. Two-Variable Logic on Words with Data. In *LICS*. 7–16.
[8] Arnaud Carayol and Christof Löding. 2014. Uniformization in Automata Theory. In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*. London: College Publications, 153–178.
[9] Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.* 85, 1 (1990), 12–75.
[10] Bruno Courcelle. 1996. The monadic second-order logic of graphs X: linear orderings. *Theoretical Computer Science* 160 (1996), 87–143.
[11] Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. 2017. On Reversible Transducers. In *ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 113:1–113:12.
[12] Rodrigo de Souza. 2013. Uniformisation of Two-Way Transducers. In *LATA*.
[13] Volker Diekert and Paul Gastin. 2008. First-order definable languages. In *Logic and Automata: History and Perspectives (Texts in Logic and Games)*, Vol. 2. Amsterdam University Press, 261–306.
[14] Calvin C. Elgot and Jorge E. Mezei. 1965. On relations defined by generalized finite automata. *IBM Journal of Research and Development* 9 (1965), 47–68.
[15] Joost Engelfriet and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.* 2, 2 (2001), 216–254.
[16] Emmanuel Filiot. 2015. Logic-Automata Connections for Transformations. In *ICLA (LNCS)*, Vol. 8923. Springer, 30–57.
[17] Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. 2016. On Equivalence and Uniformisation Problems for Finite Transducers. In *ICALP (LIPIcs)*, Vol. 55. Schloss Dagstuhl, 125:1–125:14.
[18] Emmanuel Filiot and Pierre-Alain Reynier. 2016. Transducers, logic and algebra for functions of finite words. *SIGLOG News* 3, 3 (2016), 4–19.
[19] Erich Grädel and Martin Otto. 1999. On Logics with Two Variables. *TCS* 224, 1-2 (1999), 73–113.
[20] Timothy V. Griffiths. 1968. The Unsolvability of the Equivalence Problem for Lambda-Free Nondeterministic Generalized Machines. *JACM* 15, 3 (1968).
[21] Eitan M. Gurari. 1982. The Equivalence Problem for Deterministic Two-Way Sequential Transducers is Decidable. *SIAM JComp.* (1982).
[22] Swen Jacobs, Roderick Bloem, Romain Brenguier, Rüdiger Ehlers, Timotheus Hell, Robert Könighofer, Guillermo A. Pérez, Jean-François Raskin, Leonid Ryzhyk, Ocan Sankur, Martina Seidl, Leander Tentrup, and Adam Walker. 2017. The first reactive synthesis competition (SYNTCOMP 2014). *STTT* 19, 3 (2017), 367–390.
[23] Jean Berstel. 1979. *Transductions and Context-Free Languages*. Teubner.
[24] Christof Löding and Sarah Winter. 2014. Synthesis of Deterministic Top-down Tree Transducers from Automatic Tree Relations. In *GandALF 2014, Verona, Italy*.
[25] Amaldev Manuel, Thomas Schwentick, and Thomas Zeume. 2013. A Short Note on Two-Variable Logic with a Linear Order Successor and a Preorder Successor. *CoRR* abs/1306.3418 (2013).
[26] Amaldev Manuel and Thomas Zeume. 2013. Two-Variable Logic on 2-Dimensional Structures. In *CSL*. 484–499.
[27] Joachim Niehren, Laurent Planque, Jean-Marc Talbot, and Sophie Tison. 2005. N-ary Queries by Tree Automata. In *DBPL*. 232–246.
[28] A. Pnueli and R. Rosner. 1989. On the synthesis of a reactive module. In *ACM Symposium on Principles of Programming Languages (POPL)*. ACM.
[29] Jacques Sakarovich. 2009. *Elements of Automata Theory*. Cambridge University Press, Cambridge, England.
[30] Thomas Schwentick and Thomas Zeume. 2012. Two-Variable Logic with Two Order Relations. *LMCS* 8, 1 (2012).
[31] Larry J. Stockmeyer. 1974. *The complexity of decision problems in automata theory and logic*. Ph.D. Dissertation. Dept. of Electrical Engineering, M.I.T.
[32] Howard Straubing. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel and Berlin.
[33] Wolfgang Thomas. 1997. Languages, Automata and Logic. In *Handbook of Formal Languages*. Vol. 3, Beyond Words. Springer.
[34] Moshe Y. Vardi and Pierre Wolper. 1986. An automata-theoretic approach to automatic program verification. In *lics86*. 332–344.