Some Remarks on Deciding Equivalence for Graph-To-Graph Transducers

Mikołaj Bojańczyk
Institute of Informatics, University of Warsaw, Poland
bojan@mimuw.edu.pl

Janusz Schmude
Institute of Informatics, University of Warsaw, Poland
jschmude@mimuw.edu.pl

Abstract

We study the following decision problem: given two MSO transductions that input and output graphs of bounded treewidth, decide if they are equivalent, i.e. isomorphic inputs give isomorphic outputs. We do not know how to decide it, but we propose an approach that uses automata manipulating elements of a ring extended with division. The approach works for a variant of the problem, where isomorphism on output graphs is replaced by a relaxation of isomorphism.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases equivalence for mso transductions, bounded treewidth, Hilbert basis theorem

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.19

Funding Supported by ERC Consolidator Grant LIPA, agreement no. 683080.

1 Introduction

Monadic second-order transductions (MSO transductions) are transformations which input structures and output structures such as words, trees or graphs. The idea is that the output structure is defined in terms of the input structure by using MSO logic. MSO transductions were originally introduced to describe graph-to-graph transformations, see [4, p. 43], [15, Definition 6], [10, Defintion 2.2], and it is graph transformations that are the principal topic of the current paper. Nevertheless, MSO transductions have also recently seen a lot interest for simpler structures such as strings or trees. In [16, Theorem 13], Engelfriet and Hoogeboom proved a seminal result which says that functional MSO transductions define exactly the same string-to-string functions as two-way deterministic transducers. This result made a connection between MSO transductions and the well-established literature on transducers.

In the transducer literature, a central problem is equivalence: given two transducers, decide if they represent that same function. The equivalence problem is known to be decidable for two-way deterministic transducers [17, Theorem 1]. Putting this together with the theorem of Engelfriet and Hoogeboom, we see that equivalence is also decidable for string-to-string MSO transductions. Similar decidability results for the equivalence problem have been obtained for tree-to-tree and tree-to-string transductions. The tree-to-string case is at least as general as the tree-to-tree case, because an output tree can be injectively represented as a string using pre-order traversal, and injectivity of this encoding guarantees that the representation does not affect equivalence of transductions. In [23, Corollary 8.2], Seidl, Maneth and Kemper show that equivalence is decidable for tree-to-string streaming transducers; the latter are shown by Alur and D'Antoni to be equivalent to MSO transductions in [1, Theorem 4.6]. Putting these results together, we see that equivalence is decidable for tree-to-tree (and tree-to-string) MSO transductions.


© Mikołaj Bojańczyk and Janusz Schmude;
licensed under Creative Commons License CC-BY

The equivalence problem often becomes harder and more interesting when the output structures have more (but not too many) automorphisms. Here is an example of this phenomenon: compare binary trees that have a sibling order with binary trees that do not have a sibling order. A binary tree with sibling order can be encoded (via an MSO transduction) in a tree without sibling order, by using labels (or the tree structure if labels are not allowed) to differentiate between left and right children. The other direction does not work: there is no MSO transduction that injectively maps unordered trees to ordered trees. For these reasons, the equivalence problem for tree-to-tree transductions becomes harder when considering trees without a sibling order. Nevertheless, the latter problem is still decidable [6, Section 6], which is shown by encoding trees without a sibling order as polynomials in $\mathbb{Z}[x]$, and then applying the "Hilbert method", which is an algorithm using Hilbert's Basis Theorem that is described in [7] and based on [23].

Since the equivalence problem gets more interesting for structures with more automorphism, we propose in this paper to study the equivalence problem for transductions that operate on (finite) graphs, where automorphism are as general as they can be. Since we use MSO as our transduction formalism, and MSO is undecidable on graphs of unbounded treewidth [22, Theorem 8], we consider MSO transductions for graphs of bounded treewidth.

We do not know if the equivalence problem is decidable for MSO graph-to-graph transductions of bounded treewidth. We propose an approach to the problem – which uses the Hilbert method of encoding combinatorial objects (graphs, trees, strings, etc.) as algebraic objects (numbers, polynomials, rational functions, etc.) In our specific case, we get some limited success by representing graphs as rational power series. This, together with a recent result of Grohe, Dell and Rattan [13, Theorem 2], which describes a certain relaxation of isomorphism in terms of counting walks, gives us our main result, Theorem 13, which says that equivalence is decidable for graph-to-graph transductions of bounded treewidth, where output graphs are equivalent modulo the relaxation of isomorphism from [13].

We only scratch the surface of the equivalence problem for graph-to-graph transductions. We view this paper mainly as invitation to study the problem, together with some basic infrastructure for its potential solution, such as register transducers and encodings that use algebra. It remains to be seen how helpful this infrastructure will be. We also highlight some special cases of the problem, which seem approachable, but are nevertheless still open.

2 Graphs, treewidth, and logic on graphs

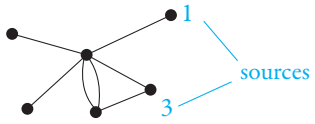
The graphs in this paper are finite and undirected, with parallel edges, and possible self-loops. Formally, a *graph* consists of: a set of *vertices*, a set of *edges* and a function that associates to each edge a set of at most two endpoints. We call distinct edges *parallel* if their sets of endpoints are equal.

2.1 Treewidth

As mentioned in the introduction, we work with graphs of bounded treewidth to avoid undecidability for MSO transductions. For treewidth, we use Courcelle's approach via universal algebra [3]. (Later, we also use algebra in the classical sense: rings, fields, polynomials, etc.)

Algebras and terms operations We begin by introducing some terminology from universal algebra. By an *algebra*, we mean an underlying set equipped with some operations, e.g. the ring of integers $(\mathbb{Z}, +, \times, 0, 1)$ or the free monoid $(\Sigma^*, \cdot, \varepsilon)$. We write $\mathbf{A}, \mathbf{B}, \ldots$ for algebras. By abuse of notation, we write $a \in \mathbf{A}$ if a belongs to the underlying set in the algebra $\mathbf{A}$.

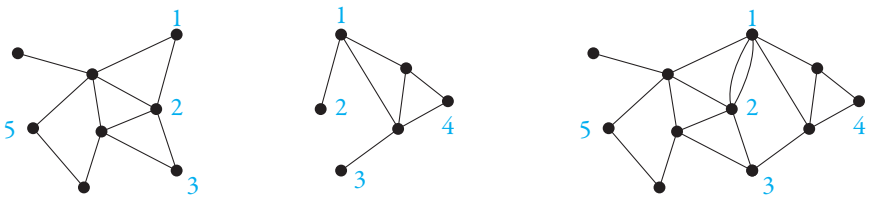
We use the name *basic operations* for the operations that are given in the algebra. This is to distinguish them from the more general *term operations*, which are defined below. For a finite set of variables X and an algebra $\mathbf{A}$, a function $f : \mathbf{A}^X \to \mathbf{A}$, is called a *term operation* if it is induced by a term constructed using the basic operations of the algebra and variables from X. If the set X is $\{0, \dots, k-1\}$ then we talk about term operations of type $\mathbf{A}^k \to \mathbf{A}$. For example, the polynomial $3x^2 - 2x + 1$ can be viewed as a term in the algebra $(\mathbb{R}, +, \times, -1, 0, 1)$, and it induces a term operation of type $\mathbb{R} \to \mathbb{R}$. Polynomials with non-integer coefficients such as 0.5 are not term operations in the algebra from the above example; for this the coefficients would need to be added as constants to the algebra. We will also allow term operations that are vectorial: a function $f : \mathbf{A}^X \to \mathbf{A}^Y$ is a term operation if for every $y \in Y$, the composition of f with the projection to coordinate y is a term operation.



The algebra of k-sourced graphs To define treewidth, we consider graphs with distinguished vertices and equip them with an algebraic structure. For $k \in \{1, 2, \dots\}$, define a *k-sourced graph* to be a graph together with a k-tuple of pairwise distinct vertices called *sources*. We allow sources to be undefined, for example the 3-sourced graph in the picture to the right has source 2 undefined. We view k-sourced graphs as an algebra, in the following sense.

▶ **Definition 1** (Algebra of k-sourced graphs). *Let $k \in \{1, 2, \dots\}$. The* algebra of k-sourced graphs *has as its underlying set the k-sourced graphs, with the following basic operations:*

- ***Constants.*** *There is a constant for every k-sourced graph where all vertices are sources and there is at most one edge.*
- ***Forget.*** *For every $i \in \{1, \dots, k\}$ there is a unary operation, which inputs a k-sourced graph and outputs the same k-sourced graph, except that the i-th source is undefined.*
- ***Fuse.*** *There is one binary operation, called* fusion*, which inputs two k-sourced graphs, and connects them along same-numbered sources, as explained in the following picture (note that fusion of graphs with no parallel edges may contain parallel edges):*



two 5-sourced graphs their fusion

Terms in this algebra correspond to tree decompositions in the original sense of Robertson and Seymour [14, Section 12.3]. More precisely, a graph has treewidth $\leq k$ if and only if it can be produced using the operations of the algebra of $(k+1)$-sourced graphs [3, Proposition 4.1]. This justifies the following definition.

▶ **Definition 2** (Treewidth). *For $k \in \{1, 2, \dots\}$, we say that a graph has treewidth $\leq k$ if it is the value of some term in the algebra of $(k+1)$-sourced graphs.*

2.2 Monadic second-order logic

We use logic, especially monadic second-order logic, to define properties and transformations of graphs.

Transductions A *vocabulary* is defined to be a set of relation names, each one with an associated arity. A *model over vocabulary* τ consists of a set, called the *universe* of the model, together with an *interpretation*, that maps the relation names from the vocabulary τ to relations over the universe of same arity. To define properties of such models, we use mainly monadic second-order logic MSO, which is the extension of first-order logic that allows quantification over sets of elements (but not sets of pairs, or triples, etc.). We assume that the reader is familiar with MSO, for an introduction see [25, Section 2].

The central topic of this paper is MSO transductions, which are graph transformations definable in MSO (when talking about MSO on graphs, we use the MSO_2 representation of graphs as models that will be described below). The transductions can be nondeterministic, in which case they represent binary relations and not functions. A *transduction*, with input vocabulary τ and output vocabulary σ is defined to be a set of pairs

$$(\underbrace{\text{model over } \tau}_{\text{input model}}, \underbrace{\text{model over } \sigma}_{\text{output model}}),$$

which is isomorphism invariant, in the sense that membership in the transduction is not affected by changing a model – on either the input or output coordinate – to an isomorphic one. The central topic of this paper is transductions that can be defined using MSO, as described in the following definition, whose phrasing is based on [8, p. 9–10].

▶ **Definition 3** (MSO transduction). *An MSO transduction is any transduction that can be obtained by composing a finite number of transductions of the following kinds. Kind 1 is a partial function, kinds 2, 3, 4 are functions, and kind 5 is a relation.*

1. ***Filtering.*** *For every MSO sentence φ over the input vocabulary there is a transduction that filters out models where φ is violated. Formally, the transduction is the partial identity – with input and output vocabularies being equal – whose domain consists of the models that satisfy the sentence.*
2. ***Universe restriction.*** *For every MSO formula $\varphi(x)$ over the input vocabulary with one free first-order variable, there is a transduction, which restricts the universe of the input model to those elements that satisfy φ. The input and output vocabularies are the same, the interpretation of each relation in the output model is defined as the restriction of its interpretation in the input model to tuples of elements that remain in the universe.*
3. **Mso *interpretation.*** *This kind of transduction changes the vocabulary of the model while keeping the universe intact. To specify an MSO interpretation, for for every relation name R of the output vocabulary, one gives an MSO formula $\varphi_R(x_1, \ldots, x_k)$ over the input vocabulary which has as many free first-order variables as the arity of R. The output model is obtained from the input model by keeping the same universe, and interpreting each relation R of the output vocabulary as the set of tuples satisfying φ_R.*
4. ***Copying.*** *For $k \in \{1, 2, \ldots\}$, define k-copying to be the transduction which inputs a model and outputs a model consisting of k disjoint copies of the input, with an additional relation that ties the copies together. Formally, the output universe consists of k copies of the input universe. The output vocabulary is the input vocabulary enriched with a binary predicate* copy *which selects pairs which originate from the same element, and unary predicates* $\mathsf{layer}_1, \mathsf{layer}_2, \ldots, \mathsf{layer}_k$ *which select elements belonging to the first, second, etc.*

copies of the universe. In the output model, a relation name R of the input vocabulary is interpreted as the set of all those tuples over the output model, which are in the same layer, and where the original elements of the copies were in relation R in the input model.

5. ***Colouring.*** *Colouring adds a new unary predicate to the input model, and interprets it nondeterministically. Formally, the universe as well as the interpretations of all relation names of the input vocabulary stay intact, but the output vocabulary has one more unary predicate. For every possible interpretation of this unary predicate, there is a different output with this interpretation implemented.*

Logic and transductions on graphs To define properties and transductions for graphs in MSO, we represent graphs as models. There are at least two ways to do this, called MSO_1 and MSO_2 following Courcelle [11, Definition 1.7]. In this paper, we use the more expressive MSO_2 model: the universe is the disjoint union of vertices and edges, and there is a binary *incidence relation* that consists of pairs (v, e) such that vertex v participates in edge e. We present our results for monadic second-order logic without modulo counting, but all of our results can be easily extended to include modulo counting.

Define a *graph-to-graph MSO transduction* to be an MSO transduction, where the input and output vocabularies are the same, namely the vocabulary of graphs (one binary relation for incidence), and which only contains pairs of graphs (more formally, pairs of MSO_2 models of graphs). By abuse of notation, when $\mathcal{T}$ is a graph-to-graph MSO transduction, and G, H are graphs, we write $(G, H) \in \mathcal{T}$ instead of saying that $\mathcal{T}$ contains the pair (MSO_2 model of G, MSO_2 model of H). If $\mathcal{T}$ is functional, which means that for every input graph there is at most one output graph up to isomorphism, then we write $\mathcal{T}(G)$ for the (isomorphism type of the) resulting graph (which may be undefined, because $\mathcal{T}$ might be a partial function).

▶ **Example 4.** Consider the partial function which maps a clique of size n^2 to an $n \times n$ grid, and is undefined for other inputs. This is an MSO transduction: (1) nondeterministically colour the edges to select those that participate in the grid; (2) remove the other edges; (3) check that the result is indeed a grid. For this example it is important that we use the MSO_2 representation, which enables colouring edges. Also note different sets of edges might be removed in steps (2), but the result is always the same up to isomorphism.

The equivalence problem The topic of this paper is the following decision problem for functional MSO graph-to-graph transductions.

Name: equivalence for functional graph-to-graph MSO transductions of bounded treewidth
Instance: $\ell, k \in \{1, 2, \ldots\}$ and two functional graph-to-graph MSO transductions $\mathcal{T}_1, \mathcal{T}_2$.
Question: is it the case that for every input G of treewidth at most ℓ, the two outputs $\mathcal{T}_1(G), \mathcal{T}_2(G)$ are isomorphic and have treewidth at most k?

If the transductions in the instance are not functional, then there are no requirements on the answer to the algorithm. This motivates the following problem:

Name: functionality for graph-to-graph MSO transductions of bounded treewidth.
Instance: $\ell, k \in \{1, 2, \ldots\}$ and a graph-to-graph MSO transduction, not necessarily functional.
Question: is it the case that for every input of treewidth at most ℓ, there is at most one output, and this output – if defined – has treewidth at most k?

We do not know how to solve any of the above problems, but at least we know that they are equi-decidable, i.e. if one is decidable then the other is decidable.

▶ **Fact 5.** *The equivalence and functionality problems described above are equi-decidable.*

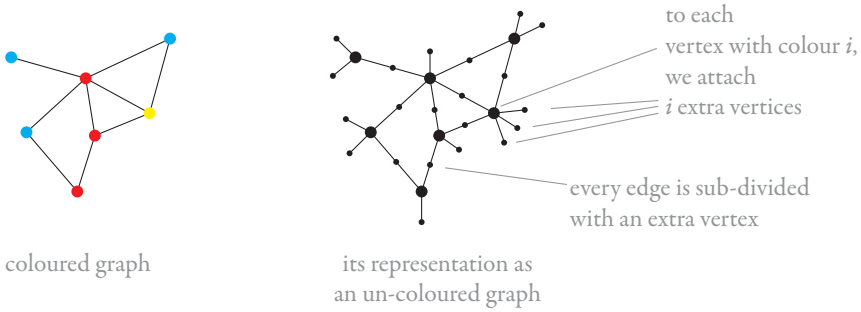
Proof. In both problems we require that: (*) if the output has treewidth at most ℓ, then every output has treewidth at most k. This is a decidable property, because the graphs of treewidth strictly bigger than k are definable in MSO (as a complement of a minor closed class), MSO definable languages are closed under inverse images of MSO transductions [12, Backwards Translation Theorem], and satisfiability of MSO is decidable on graphs of treewidth at most ℓ [12, Theorem 5.80].

We can therefore assume that the transductions at hand satisfy (*). Equivalence reduces to functionality, because two transductions are equivalent if and only if they have the same domains when restricted to treewidth at most ℓ, and their union is functional. The domain equivalence can be checked in the same way as (*).

Consider now the converse reduction: from functionality to equivalence. Suppose that $\mathcal{T}$ is an MSO transduction, and we want to check if $\mathcal{T}$ is functional for inputs of treewidth at most ℓ and outputs of treewidth at most k. Because colourings can be pushed to the head of an MSO transduction, see [12, Section 1.7], we can decompose $\mathcal{T}$ as $\mathcal{T} = \mathcal{F} \circ \mathcal{S}$, where:

- $\mathcal{C}$ is a colouring, i.e. a composition of transductions as in item 5 of Definition 3.
- $\mathcal{F}$ is a composition of transductions from items 1-4 of Definition 3.

In particular, since items 1-4 of Definition 3 describe partial functions, $\mathcal{F}$ is a partial function. There is some finite set of colours C (if $\mathcal{F}$ is a composition of n copying transductions, then $C = 2^n$) such that the transduction $\mathcal{C}$ inputs a graph, and nondeterministically outputs a C-coloured graph (i.e. a graph with vertices labelled by colours from C). For $i \in \{1, 2\}$, define $\mathcal{F}_i$ to be the MSO transduction which inputs a $(C \times C)$-coloured graph, projects the colours to the i-th coordinate, and then applies $\mathcal{F}$. The transduction $\mathcal{T}$ is functional if and only if the transductions $\mathcal{F}_1$ and $\mathcal{F}_2$ are equivalent (and the same statement is true if we restrict inputs and outputs to have given treewidth). Therefore, the functionality problem reduces to the equivalence problem for coloured graphs. Finally, colours can be simulated by the graph structure without affecting treewidth, as explained in the following picture:



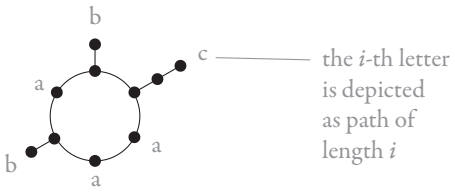
Let $\mathcal{T}$ be the MSO transduction from coloured graphs to uncoloured graphs, and let $\mathcal{T}^{-1}$ be its one-sided inverse, which is a partial function from uncoloured graphs to coloured graphs. Both $\mathcal{T}$ and $\mathcal{T}^{-1}$ do not change treewidth. Two functional MSO transductions $\mathcal{F}_1$ and $\mathcal{F}_2$ from coloured graphs to coloured graphs are equivalent if and only if the functional graph-to-graph MSO transductions $\mathcal{T} \circ \mathcal{F}_1 \circ \mathcal{T}^{-1}$ and $\mathcal{T} \circ \mathcal{F}_2 \circ \mathcal{T}^{-1}$ are equivalent. Therefore, the functionality and equivalence problems for coloured graphs reduce to the same problems for un-coloured graphs. ◀

The rest of this paper is devoted to a discussion of the above two problems, including a solution for a variant where output graphs are identified up to a relaxation of isomorphism. We begin with two examples which illustrate the fact that equivalence problem for graph-to-graph transductions is unsolved even for very restricted instances.

Example: treewidth 1. Consider the case where both input and output graphs have treewidth 1. (As we will see in Lemma 10, it is the output treewidth that is important, since the input treewidth can always be reduced to 1 without making the problem any easier.) This is a very special case of the equivalence problem, but even this special case we do not know how to solve.

Let us compare this with a very similar problem that is already known to be decidable. There are three kinds of trees that can be considered, listed in order of increasing rigidity: (a) without a distinguished root and without sibling order, (b) with a distinguished root and without sibling order, (c) with a distinguished root and with sibling order. The (a) variant is the equivalence problem for graphs of treewidth 1. As discussed in the introduction, the equivalence problem for MSO transductions is harder in variant (b) than in variant (c), and for similar reasons (a) is harder than (b). Variant (b) is known to be decidable, but variant (a) remains open. One idea would be to reduce variant (a) to variant (b) by transforming an un-rooted tree into the rooted forest of all possible ways of choosing a root; this transformation has quadratic size outputs, and therefore it cannot be an MSO transduction.

Example: strings modulo cyclic shift. Here is another special case of our problem, which seems entertaining and is also open. Define *cyclic equivalence* to be the equivalence relation $\sim$ on strings which identifies two strings modulo cyclic shifts (for example $abcd \sim cdab$). Consider the following problem: given two functional string-to-string MSO transductions [16,

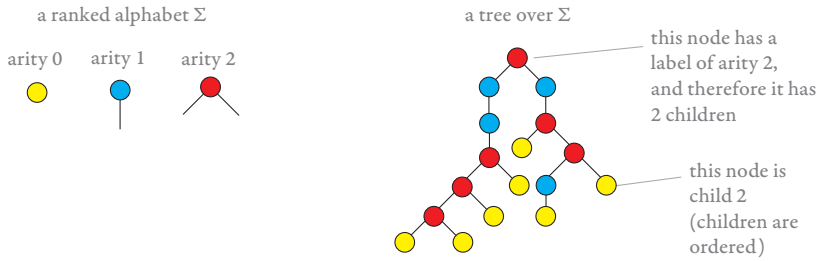


Definition 1], decide if for every input string, the two output strings are equal modulo cyclic equivalence. It is a special case of the graph-to-graph equivalence problem, since strings modulo cyclic equivalence can be represented as graphs of treewidth at most 2, as explained in the picture. (The cycle is directed, which can be encoded using gadgets in undirected graphs.)

3 Register transducers

To approach the equivalence problem for graph-to-graph transductions, we use register transducers that input trees. The idea is that graphs are given by tree decompositions. This leads to yet another notion of trees, as compared to the three notions described in the example at the end of the previous section, namely trees over a ranked alphabet (which are roughly the same thing as terms without variables).

In this paper, when transducers input trees, then these trees are ranked trees over a ranked alphabet (every letter has an arity in $\{0, 1, \dots\}$), as described in the following picture.



We use standard tree terminology, such as descendant, ancestor or parent. From now on, when talking about trees, we mean the ranked trees described above, and not any of the other trees discussed in the previous section.

We now describe the transducer model that will be used in the paper. The idea is that the transducer processes an input tree bottom-up, and uses registers to stores elements of some output algebra. This idea has its roots in the synthesized attributes of attribute grammars of Knuth [18], and can also be found in the cost register automata of Alur et al. [2, p. 15]. Our definition is based on terminology of universal algebra that was discussed in Section 2.1. For a short definition, we omit states from the model. They would make some constructions easier, but they do not add expressive power, because they can be simulated using extra registers.

▶ **Definition 6** (Register transducer). *The syntax of a* (implicitly, nondeterministic) register transducer *consists of:*

1. *an output algebra* $\mathbf{A}$*, which is an algebra in the sense described in Section 2.1;*
2. *an input alphabet* Σ*, which is a finite ranked set;*
3. *a finite set* R *of register names, with a distinguished output register* $r \in R$*;*
4. *an initial register valuation, of type* $\mathbf{A}^R$*;*
5. *for every letter* $a \in \Sigma$ *of the input alphabet, of arity* $n \in \{0, 1, \ldots\}$*, a finite set of associated* register updates*, which are term operations of type* $\mathbf{A}^{\{1,\ldots,n\} \times R} \to \mathbf{A}^R$*.*

If for each input letter there is exactly one associated register update, then the register transducer is called deterministic.

A register transducer with output algebra $\mathbf{A}$ is also called a register transducer over $\mathbf{A}$. A register transducer computes a binary relation, which consists of pairs (tree over the input alphabet, element of the output algebra). Pairs in this relation are obtained by executing the register updates on the input tree in a bottom-up way, and then returning the value of the first register. Here is a more detailed explanation. For an input tree t over the input alphabet Σ, define the *reachable register valuation* on t to be the subset of $\mathbf{A}^R$ that is defined as follows by induction on the size of t. Consider a tree which has root label a and child subtrees $t_1, \ldots, t_n$. (The induction base of the construction is when a has arity $n = 0$ and there are no child subtrees, in this case only reachable valuation is initial valuation.) Let $v_1, \ldots, v_n \in \mathbf{A}^R$ be register valuations that are reachable, respectively, for the child subtrees, as defined by induction assumption. Combine these register valuations into a register valuation from $\mathbf{A}^{\{1,\ldots,n\} \times R}$, and apply to this tuple any register update corresponding to a. The resulting register valuation is reachable from for the tree t. The binary relation computed by a register transducer consists of pairs (t, value of output register in some register valuation reachable in t). If the register transducer is deterministic, then the binary relation is a function.

▶ **Example 7.** Consider nondeterministic register transducers over the Boolean algebra

$$\mathbf{A} = (\{0, 1\}, \vee, \wedge, \neg, 0, 1).$$

A valuation of the registers can be seen as a state from a finite set of bit vectors. Since every operation on bit vectors can be described by a term of Boolean algebra, it follows that register transducers with this output algebra are the same thing as nondeterministic bottom-up tree automata, which define exactly the regular tree languages [24, Section 2].

For register transducers over a fixed output algebra, we will study the functionality and equivalence problems. The functionality problem is defined as follows.
Name: functionality for nondeterministic register transducers over algebra $\mathbf{A}$.
Instance: a nondeterministic register transducer over algebra $\mathbf{A}$.
Question: is the transducer functional, i.e. for every input there is exactly one output?
The equivalence problem is defined similarly, except that the input is two deterministic register transducers, and the question is if they produce equal outputs for all inputs.

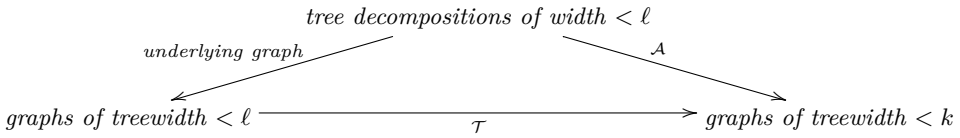
▶ **Fact 8.** *For every output algebra, functionality and equivalence are equi-decidable.*

Proof. Same reasoning as in Fact 5, with the following differences: (1) the domain of a register transducer is always a regular tree language, which can be computed effectively; and (2) there is no need to encode colours in the input graph, since the model of register transducers allows for coloured inputs. ◀

▶ **Example 9.** In [23], Maneth, Seidl and Kemper study the equivalence problem for deterministic register transducers where the output algebra is Σ^*, equipped with concatenation and constants for all letters and the empty string. Using Hilbert's Basis Theorem, checking equivalence of tree-to-string transducers is shown decidable [23, Corollary 8.2]. We will use the same method in the proof of Theorem 13.

Connection with graph-to-graph transductions We now show how register transducers can be used to compute functional graph-to-graph MSO transductions of bounded treewidth. We are mainly interested in the case when the input trees are tree decompositions. In the context of this paper, a tree decomposition of width $< k$ is defined to be a tree, where the input alphabet is the set of operations in the algebra of k-sourced graphs.

▶ **Lemma 10.** *Let $\ell, k \in \{1, 2, \ldots\}$, and let $\mathcal{T}$ be an MSO transduction where all output graphs have treewidth at most k. One can compute a nondeterministic register transducer $\mathcal{A}$ over the algebra of $2k$-sourced graphs, which makes the following diagram commute (arrows in the diagram are binary relations):*



The proof of the above lemma uses ideas from [9, 5, 1], and is relegated to the appendix.

▶ **Corollary 11.** *For every $k \in \{1, 2, \ldots\}$ there is some $\ell \in \{1, 2, \ldots\}$ such that equivalence of functional graph-to-graph MSO transductions of treewidth at most k reduces to the equivalence problem for deterministic register transducers over ℓ-sourced graphs.*

4 Equivalence modulo a certain equivalence relation

In this section, we present our main result, which says that equivalence is decidable for graph-to-graph transductions of bounded treewidth, modulo a certain equivalence on the output graphs.

Counting homomorphisms The equivalence on output graphs is based on a paper by Grohe, Dell and Rattan [13], which gives examples of how various relaxations of isomorphism on graphs, notably Weisfeiler-Leman equivalence, can be characterised in terms of counting homomorphisms. Define a *homomorphism* from a graph G to a graph H to be function h which maps vertices of G to vertices of H and edges of G to edges of H, which is consistent with the incidence relation in the natural way. We write $\mathrm{Hom}(G, H)$ for the set of homomorphisms from G to H. A seminal result of Lovász [19, p. 326] says that two graphs are isomorphic if and only if they admit the same number of homomorphisms from every graph:

$$\underbrace{G \simeq H}_{\text{isomorphism}} \qquad \text{iff} \qquad |\mathrm{Hom}(F, G)| = |\mathrm{Hom}(F, H)| \text{ for every graph } F.$$

In [13], Grohe, Dell and Rattan show that if one restricts the class of graphs from which F is taken, then one gets various well-studied relaxations of isomorphism. For example, if we only count homomorphisms from graphs F of treewidth at most k, then the arising equivalence relation on graphs is Weisfeiler-Leman equivalence of dimension k. The result of [13] that we use in this paper, stated below[1], counts homomorphisms from paths.

▶ **Theorem 12.** *([13, Theorem 2]) For all graphs G_1, G_2, the following are equivalent:*

1. *for every $i \in \{0, 1, \ldots\}$, if P_i is the path of length n then*

$$|\mathrm{Hom}(P_i, G_1)| = |\mathrm{Hom}(P_i, G_2)|.$$

2. *G_1 and G_2 have the same number of vertices, say n, and there is an $n \times n$ square matrix X with real coefficients such that:*
   a. *each row in X sums up to 1*
   b. *each column in X sums up to 1*
   c. *$A_1 X = X A_2$, where A_i is the adjacency matrix of graph G_i.*

Note that a homomorphism from a path of length n is the same thing as a walk of length n i.e. a sequence of $n + 1$ vertices (possibly with repetitions) interleaved with n connecting edges. When counting walks, we distinguish walks that use different parallel edges. We call a walk *empty* if it contains no edges. In other words, condition 1 above says that for every n, the two graphs have the same number of walks of length n. In particular, if we take $n = 0$, it implies that the graphs have the same number of vertices.

Condition 2 in the above theorem can be seen as a relaxation of isomorphism in the following sense. If in condition 2 we add the requirement that all coefficients in X are non-negative (real or rational, does not make a difference), then we get *fractional isomorphism*, which is characterised by homomorphisms from trees [13, Theorem 1]. If we add the requirement that all coefficients in X are non-negative integers, then we get isomorphism.

Decidable equivalence, modulo counting homomorphisms from paths In the following theorem, we show that equivalence for graph-to-graph MSO transductions of bounded treewidth is decidable, when output graphs are identified modulo the equivalence relation from Theorem 12.

▶ **Theorem 13.** *Let $\sim$ be the equivalence relation on graphs described in Theorem 12. The following problem is decidable:*

[1] The paper [13] uses graphs without parallel edges, but the authors remark in the Preliminary Section that the results work with parallel edges as well.

Name: *$\sim$-equivalence for graph-to-graph MSO transductions of bounded treewidth.*
Instance: *$\ell, k \in \{1, 2, \ldots\}$ and two functional graph-to-graph MSO transductions $\mathcal{T}_1$ and $\mathcal{T}_2$.*
Question: *is it the case that for every input G of treewidth at most ℓ, the two outputs $\mathcal{T}_1(G), \mathcal{T}_2(G)$ are equivalent under $\sim$ and have treewidth at most k?*

For the same reasons as in Fact 5, the above problem is equi-decidable with the $\sim$-functionality problem, where one asks if a nondeterministic transduction can produce two $\sim$-inequivalent outputs on some input, under assumption of bounded treewidth.

The rest of this section is devoted to showing decidability for the $\sim$-equivalence problem. We first show how a register transducer can compute a representation of the output graph modulo $\sim$ in terms of a power series. Then we show that equivalence is decidable for register transducers which manipulate such power series. Combining this with the reduction to register transducers from Lemma 10, we get the theorem.

Walk series. An infinite sequence $a_0, a_1, \ldots \in \mathbb{N}$ can be visualised as an infinite polynomial

$$A = a_0 + a_1 x^1 + a_2 x^2 + \cdots$$

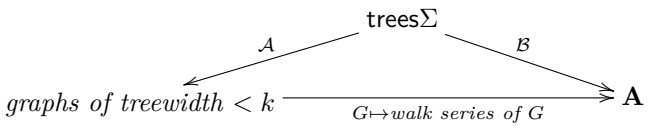
with one variable x. We use the name *power series* for such polynomials, and write $\mathbb{N}[\![x]\!]$ for the set of power series. We use the name *i-th term* for a_i; and we call a_0 the *constant term*. The power series form a semiring, with addition defined coordinatewise, and product being Cauchy product (see [20, part III] for more on rational power series). Among all power series, we are particularly interested in the rational ones: a power series is called *rational* if it can be generated from finite power series (a finite power series is one where all but finitely many terms are zero) using the semiring operations $+$ and $\times$ of power series, as well as the unary *Kleene plus* operation, which maps A to $A^+ = A + A^2 + A^3 + \cdots$. The Kleene plus is defined only if the power series is *proper* (i.e. its constant term is zero), which ensures that the infinite sum in A^+ is well defined. If A were not proper, then its Kleene plus would need to have the divergent value $a_0 + a_0^2 + a_0^3 + \cdots$ as its constant term.

For a graph G, define its *shifted walk series* to be the proper power series

$$a_0 x^1 + a_1 x^2 + \cdots \qquad \text{where } a_i \text{ is the number of walks in } G \text{ of length } i.$$

(This is a shift of *walk series* $\sum_{n=0} a_n x^n$, with a_n defined as above; we use the shifted one to avoid technicalities, like e.g. sorted algebras.) By definition, two graphs are $\sim$-equivalent if and only if they have the same shifted walk series. The first step in the proof of Theorem 13 is the following lemma, which says that the shifted walk series is always a rational power series, and can be computed by a register transducer.

▶ **Lemma 14.** *Define* **A** *to be the algebra where*
Domain: *proper rational power series in $\mathbb{N}[\![x]\!]$;*
Operations: *$+$, $\times$, Kleene plus and constant x.*
Given $k \in \{1, 2, \ldots\}$ and a nondeterministic register transducer $\mathcal{A}$ over the algebra of k-sourced graphs, one can compute a nondeterministic register transducer $\mathcal{B}$ over the algebra **A** *defined above which makes the following diagram commute (arrows are binary relations):*



Proof sketch. The main idea in the proof is the compositionality of walk series which is explained below. An *inner walk* in a k-sourced graph is defined to be a walk that does not visit sources, with the possible exception of the first and last vertex in the walk. For $i, j \in \{1, \ldots, k\}$ define $\varphi_{ij}(G)$ to be the (non-shifted) proper power series where the n-th term is the number of nonempty inner walks of length n that begin in the i-th source and end in the j-th source. In this proof sketch, we show how the series described above can be maintained compositionally while applying the operations from the algebra of k-sourced graphs. Using variations on this compositionality principle, which take into account some minor technical details, the lemma is proved in the appendix.

Consider first the fusion of two k-sourced graphs: an inner walk in the fusion is the same thing as an inner walk in either the first or second component of the fusion, and these cases are disjoint (because of parallel edges). Therefore, we have:

$$\varphi_{ij}(\text{fusion of } G \text{ and } H) = \varphi_{ij}(G) + \varphi_{ij}(H).$$

The more interesting case is the forget operation. Suppose that G is a k-sourced graph and G_ℓ is the result of forgetting source $\ell \in \{1, \ldots, k\}$. An inner walk in G_ℓ is either an inner walk in G, or it can be decomposed as a composition of: first go to source ℓ, then take a finite number of loops around source ℓ, and then leave source ℓ. Since composing walks corresponds to Cauchy product of power series, we get the following equation:

$$\varphi_{ij}(\text{forget source } \ell \text{ in } G) = \varphi_{ij}(G) + \varphi_{i\ell}(G) \times (1 + (\varphi_{\ell\ell}(G))^+) \times \varphi_{\ell j}(G). \qquad \blacktriangleleft$$

Equivalence for transducers over rational power series. Thanks to the above lemma, in order to prove Theorem 13, it remains to show that equivalence is decidable for register transducers over the algebra **A** of rational power series that is used in the above lemma. For this decidability result, we use the embedding of rational power series into rational functions. Recall that a *rational function* with variable x is a fraction $P(x)/Q(x)$ where the numerator $P(x)$ and denominator $Q(x)$ are polynomials in $\mathbb{Z}[x]$, modulo equality defined by

$$P_1(x)/Q_1(x) = P_2(x)/Q_2(x) \qquad \text{iff} \qquad P_1(x) \times Q_2(x) = P_2(x) \times Q_1(x).$$

The denominator must be nonzero. We write $\mathbb{Z}(x)$ for the rational functions; this is a field. Let A be a proper series. Then $A^+ \times (1 - A) = A$, hence A^+ can be seen as a fraction of polynomials of A; by induction, all proper rational power series are fractions of finite power series, which yields the embedding of the algebra **A** into the field $\mathbb{Z}(x)$.

Consider a register transducer over the ring $\mathbb{Z}[x]$. Such a register transducer can store a proper rational function using two registers: one for the numerator and one for the denominator. The field operations on rational functions can be simulated by term operations using this representation, including division[2]. (The field $\mathbb{Z}(x)$ with division is not an algebra in the sense used in this paper, since we require all basic operations to be total.) Therefore the equivalence problem for register transducers over the algebra **A** reduces to the equivalence problem for register transducers over $\mathbb{Z}[x]$. It remains to show that the latter problem is decidable. This follows from a more general result.

▶ **Theorem 15.** *Let $\mathbb{K}$ be a commutative ring which is computable (its elements can be enumerated so that the ring operations are computable) and has no zero divisors (if a, b are nonzero, then so is $a \times b$). Then equivalence is decidable for register transducers over $\mathbb{K}$.*

[2] The idea to use division, although simple, seems to be new and of independent interest.

This result is proved in the appendix, by adapting a proof for the special case of the ring of integers that was given in [23, Theorem 6.6]. Applying the above result to the ring $\mathbb{Z}[x]$, and then the encoding of $\mathbf{A}$ in $\mathbb{Z}[x]$, we see that equivalence is decidable for register transducers over $\mathbf{A}$. Combining this with Lemma 14, we finish the proof of Theorem 13.

Perspectives. Our approach in Theorem 13 is based on mapping a graph to an algebraic object (a polynomial, rational function, power series, etc.). The mapping should be compositional (the algebraic objects can be synthesised by a register transducer that inputs a tree decomposition) and have high distinguishing power (ideally, non-isomorphic inputs should give different algebraic objects). Finding the right combination seems to be challenging, and so far we have been unsuccessful with adapting our techniques to known algebraic invariants, either because of distinguishing power (e.g. all trees with m edges have the same Tutte polynomial x^m, while the characteristic polynomial does not a distinguish trees almost surely [21, Theorem 8]), or compositionality (e.g. counting homomorphisms from graphs of bounded treewidth has enough distinguishing power for bounded treewidth [13, Theorem 4], but the proper notion of power series seems elusive), or both.

References

1 Rajeev Alur and Loris D'Antoni. Streaming Tree Transducers. *J. ACM*, 64(5):31:1–31:55, August 2017.

2 Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *Logic in Computer Science, LICS, New Orleans, USA*, pages 13–22. IEEE Computer Society, 2013.

3 Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, January 1993.

4 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs extended abstract. In Timo Lepistö and Arto Salomaa, editors, *Automata, Languages and Programming*, pages 38–51, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

5 Roderick Bloem and Joost Engelfriet. A Comparison of Tree Transductions Defined by Monadic Second Order Logic and by Attribute Grammars. *Journal of Computer and System Sciences*, 61(1):1–50, 2000.

6 Adrien Boiret, Radoslaw Piórkowski, and Janusz Schmude. Reducing Transducer Equivalence to Register Automata Problems Solved by "Hilbert Method". In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 48:1–48:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

7 Mikołaj Bojańczyk. The Hilbert method for transducer equivalence. *SIGLOG News*, 6(1):5–17, 2019.

8 Mikołaj Bojańczyk and Michal Pilipczuk. Optimizing Tree Decompositions in MSO. In Heribert Vollmer and Brigitte Vallée, editors, *Symposium on Theoretical Aspects of Computer Science, STACS, Hannover, Germany*, volume 66 of *LIPIcs*, pages 15:1–15:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

9 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, March 1990.

10 Bruno Courcelle. The monadic second-order logic of graphs v: on closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991.

11 Bruno Courcelle. The monadic second order logic of graphs vi: on several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54(2):117–149, 1994.

12 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2012.

13 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In *International Colloquium on Automata, Languages, and Programming, ICALP, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

14 Reinhard Diestel. *Graph theory (Electronic Edition)*, volume 173 of *Graduate texts in mathematics*. Springer-Verlag, 2006.

15 Joost Engelfriet. A characterization of context-free nce graph languages by monadic second-order logic on trees. In Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Graph Grammars and Their Application to Computer Science*, pages 311–327, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

16 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO Definable String Transductions and Two-way Finite-state Transducers. *ACM Trans. Comput. Logic*, 2(2):216–254, 2001.

17 Eitan M. Gurari. The Equivalence Problem for Deterministic Two-Way Sequential Transducers is Decidable. *SIAM J. Comput.*, 11(3):448–452, 1982.

18 Donald E. Knuth. Semantics of context-free languages. *Mathematical systems theory*, 2(2):127–145, 1968.

19 László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

20 Jacques Sakarovitch. *Elements of automata theory*. Cambridge University Press, 2009.

21 Allen J Schwenk. Almost all trees are cospectral, new directions in the theory of graphs (proc. third ann arbor conf., univ. michigan, ann arbor, mich., 1971), 1973.

22 Detlef Seese. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic*, 53(2):169–195, 1991.

23 Helmut Seidl, Sebastian Maneth, and Gregor Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. *J. ACM*, 65(4), April 2018.

24 J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81, March 1968.

25 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages, Vol. 3*, pages 389–455. Springer, Berlin, 1997.