Tree-structured Graphs and Monadic Second-order Logic

Speaker: Bruno Courcelle

October 4-7, 2007
Fall School "Algorithmic Graph Structure Theory", Schloss Blankensee

Scribe: Felix Breuer, Holger Dell, Bastian Laubner, Ruth Urner

Contents

1 Introduction

This scribe tries to give a brief and comprehensible overview on tree-structured graphs, monadic second-order logics (MSO) over graphs, and their close relationship. Our definitions and examples mostly comply with those in [8], and further details can be found there.

Tree-structured graphs are graphs that can be described by tree-like structures. One way to give such descriptions is by a decomposition of the graph into parts that are connected in a tree-like fashion (e.g., the tree-decomposition). Another way uses formal languages: Expressions in the formal language define graphs and the tree-structure of a graph is simply the syntax tree of the expression defining it.

In a formal language (or algebra), we can define atomic objects and the operations we want to use to construct larger objects. In the case of graphs, the atoms are typically single edges or vertices, and the operations connect the atoms to form larger graphs. In order to glue graphs together at the desired places, it is necessary to use *labelled* graphs, where every vertex can have a label.

What is astonishing about the formal language approach is: There is a formal language HR in which the syntax tree of every expression is *equivalent* to a tree-decomposition of the graph that is defined by that expression. Furthermore, the number of labels used in the expression is equal to the width of the corresponding tree-decomposition.

Changing the operations and atoms used in HR slightly, we can define a formal language VR, which then gives rise to the notion of *clique-width* of a graph: It is the minimum number of labels used in all expressions which define that graph.

Monadic Second-Order Logic is an extension of first-order logic. In the context of graph theory, first-order logic (FO) is the language of logical formulas in which we are allowed to quantify over vertices of the graph. In second-order logic, we are allowed to quantify over relations on the set of vertices, i.e., we are able to formulate that there exists a k-ary relation $R(x_1, \ldots, x_k)$ with a certain property. The term monadic now means that we have to restrict ourselves to unary relations, that is, we can quantify only over sets of vertices. By MSO_2, we denote monadic second-order logic with the additional possibility to quantify over edges and sets of edges (see section 5 for a precise definition). A *sentence* is a formula without free variables.

The relationship between monadic second-order logics and tree-structured graph classes is diverse. We identified three main topics (see Fig. 1).

(i) *Verification* of $\mathrm{MSO}/\mathrm{MSO}_2$-sentences on *single* graphs is fixed-parameter tractable with respect to the clique-width/tree-width.

(ii) Testing *validity* of $\mathrm{MSO}/\mathrm{MSO}_2$-sentences on *all* graphs of a class is decidable if this class is the solution to an equation system over VR/HR.

(iii) The former statement is *optimal*, in the sense that every class, for which

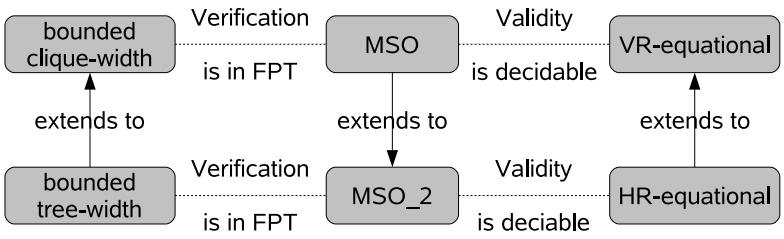


Figure 1: Overview of results (i) on the left and (ii) on the right. The vertical arrows indicate extensions of properties: Every MSO-formula is a MSO_2-formula, bounded tree-width implies bounded clique-width, and HR-equational implies VR-equational [1].

testing validity of MSO_2-formulas is decidable, is included in an HR-equational set.

Note that every equation system over HR has a fixed number of labels, therefore the solution to the equation system contains only graphs of at most this tree-width. On the other hand, most "natural" graph classes of bounded tree-width are equational, which is, however, *not* true in general. Note that (iii) is not proven for MSO and VR, but it is conjectured by Seese.

In order to establish (i), it is crucial that every MSO/MSO_2-formula can be translated into an tree automaton over VR/HR-expressions which decides for input graphs (of bounded tree-width), whether or not they satisfy the formula. Constructing that tree automaton for the input formula, transforming the input graph into an VR/HR-expression, and running the automaton on that expression can all be done in fixed-parameter tractable time. Note that the construction step is only needed once for every input formula, which may save some preprocessing time in practise.

For (ii), we need the property that intersecting the equational set with the class of all graphs that satisfy the input formula gives an equational set again. Since deciding emptiness of equational sets is decidable, we can establish (ii).

Statement (iii) needs the notion of MSO_2-transductions, which are MSO_2-expressible mapping from graphs to graphs. HR-equational sets are closed under these transductions and classes of bounded tree-width are characterized by the image of trees under these transductions.

In Section 2, we explain equational sets using the example of context-free grammars over words. In Section 3, we define the notion of recognizable sets in terms of tree automata over algebraic expressions. In Section 4, we introduce the two graph algebras HR and VR and the notion of clique-width, and we relate that notion to the tree-width. Section 5 shows us that we can construct a tree-automaton from an MSO/MSO_2-formula, and it shows us how to solve problem (ii) from that. In Section 6, we sketch the FPT-algorithm that solves problem (i). Section 7 introduces MSO-transductions, which are then used to

prove statement (iii) in Section 8.

Preliminaries

Tree-width. Let us briefly recall the well-known notion of tree-decompositions and of the tree-width of a graph $G = (V, E)$. A **tree-decomposition** of the graph G is a pair $(T, \mathcal{W})$ where $T = (V(T), E(T))$ is a tree and $\mathcal{W} = \{W_t \mid t \in V(T)\}$ is a collection of subsets of the vertexset of G satisfying

- $V = \bigcup_{t \in V(T)} W_t$,
- for every edge $uv \in E(G)$, there is a $t \in V(T)$ with $u, v \in W_t$, and
- for every vertex $u \in V(G)$, the set $\{t \in V(t) \mid u \in W_t\}$ induces a subtree of T.

The sets W_t are also called *boxes*. The width of the tree-decomposition $(T, \mathcal{W})$ is $\max\{|W_t| - 1 \mid t \in V(T)\}$ and the **tree-width** $\mathrm{twd}(G)$ of the graph is the minimum width over all tree-decompositions of G.

It is easy to see that trees have tree-width 1 (one can simply create one box per edge) and that cliques of size n have tree-width $n - 1$. Furthermore, it is known that the $n \times n$-grid has tree-width n and that outerplanar graphs (these are planar graphs with all vertices on the outer face) have tree-width 2.

Fixed-Parameter Tractability. We also briefly introduce the notion of fixed-parameter tractability, FPT for short. Considering a problem, we assume that the input is equipped with two integer valued functions, a size function $d \mapsto |d|$ and a parameter function $d \mapsto p(d)$, such that $0 < p(d) \leq |d|$. We call the problem **fixed-parameter tractable with respect to the parameter** p if it can be solved in time $\mathcal{O}\big(f(p(d)) \cdot |d|^k\big)$, where the positive integer k and the function f do not depend on the input d.

Algebra. An **F-algebra** is a set M equipped with total functions that map tuples from M^ρ to M, for arbitrary but fixed arities ρ. For simplicity, we may think of the *signature* F as the set of these total functions, although it is actually the syntax only.

An **expression** (or *term*) over F is of the form $f(t_1, \ldots, t_\rho)$ where t_i are themselves expressions over F and f is a function symbol of arity ρ. When $\rho = 0$, we speak of an *atom*. Throughout this scribe, we assume that that every element of M can be defined by an expression over F (that is, the expression *evaluates* to that element).

2 Equational (Context-Free) Sets

The goal of this section is to translate techniques that cope with languages over words to the more general case of classes of graphs.

Context-free grammars over words are usually looked at as a recursive rewriting procedure:

$$S \quad \rightarrow \quad aSa \quad | \quad b.$$

To construct a word in the language, we start with the initial non-terminal symbol S and recursively replace non-terminal symbols by words, using the rules of the grammar. As soon as the word contains only terminal symbols, we constructed a word from the language:

$$S \quad \rightarrow \quad aSa \quad \rightarrow \quad aaSaa \quad \rightarrow \quad aabaa.$$

In a second view on context-free languages, the rules are replaced by equation systems, where the non-terminal symbols are the indeterminates, viewed as sets of words.

$$\mathcal{S} \quad = \quad \{a\} \cdot \mathcal{S} \cdot \{a\} \quad \cup \quad \{b\}\,.$$

In the case of context-free languages over words, we get algebraic equation systems with the binary operator concatenation $\cdot$, and with atoms the symbols a, b, and the empty word ϵ. The **language** of the equation system is just the value of the initial indeterminate $\mathcal{S}$ in the (unique) least solution of the equation system. In the example above, the set of all words $\mathcal{S} = \{a, b\}^*$ is a solution of the equation system, but it is not the *least* solution.

The equation system view on context-free languages can be used to define context-free languages over arbitrary objects, with respect to arbitrary operations on these objects. A set is called **equational** with respect to some algebra, if it is the language of some equation system over that algebra.

In the following section, we introduce a generalization of the recognizability of languages in terms of automata, but usable in arbitrary F-algebras.

3 Recognizable Sets

In the context of formal language theory over words, a set is called recognizable if there is a *finite-state automaton* that decides whether or not a given input word is in the set. In the more general case of languages over arbitrary algebras, we have a "tree-automaton" instead of the finite-state machine.

So let us assume that we have an algebra M over a signature F, that is, F contains the operations on the elements of M. In the case of words, M is the set of all words and F contains the binary operation concatenation and as constant operations the symbols and the empty word.

An **F-automaton $\mathcal{A}$** is a tuple $(Q, Q_{\text{acc}}, \delta)$ where Q is the finite set of states, $Q_{\text{acc}} \subset Q$ is the set of accepting states, and $\delta : F \times Q^* \rightarrow Q$ is the transition function. The automaton is supposed to *accept* or *reject* any given expression over F. The set of all accepted expression is then called the **language of $\mathcal{A}$**.

In order to specify the acceptance condition, we define the *output state* $h(t)$ of the automaton on input $t = f(x_1, \ldots, x_k)$ inductively as

$$h(f(x_1, \ldots, x_k)) \quad = \quad \delta(f, h(x_1), \ldots, h(x_k)).$$

Note that, if t is an atom of the algebra, then $k = 0$ and the transition function determines the state $\delta(t)$ for that symbol. This is why the recursion terminates. We say that the automaton $\mathcal{A}$ accepts t if $h(t)$ is an accepting state, that is, if $h(t)$ is an element of Q_{acc}.

The language of L is exactly $h^{-1}(Q_{\text{acc}})$, that is, the set of all expressions that lead the automaton into an accepting state. We say that a set $L \subset M$ is **F-recognizable** if it is the language of some F-automaton. Note that, in the former step, we first need to identify terms over F with elements of M: Before running the automaton, we need to find an *arbitrary* expression over F that evaluates to the element of M.

Later on, for the fixed-parameter tractable decision procedure, we will need the fact that, given a description of the automaton and a term t, it takes time $\mathcal{O}(|t|)$ to run the automaton on t and thereby decide whether t is in the language of that automaton.

In the next section, we will see two particularly useful algebras over graphs.

4 Two Graph Algebras

In this section we will introduce two algebras of (labelled) graphs. The graph algebras VR and HR are closely related to the graph parameters clique-width and tree-width. Graphs have bounded clique-width or bounded tree-width if and only if they can be defined by expressions in these algebras using a bounded number of labels. Later we will see that the definability of a problem in monadic second order logic implies that the problem can be solved in linear time on a graph that is given by a formula in one of the algebras VR and HR. More precisely, we will see that we will see that definability by an MSO sentence (ie the characterization as the set of finite models of an MSO sentence) implies recognizability and we can therefore run the tree-automaton described above for that problem.

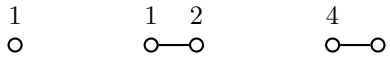
Obviously, to run that tree-automaton on a graph, we need an expression defining the graph. In general we can apply parsing algorithms for graphs of bounded clique-width or bounded tree-width to get an algebraic expression for the graph. Such parsing algorithms need linear time, for graphs of bounded tree-width, to construct an expression in the algebra HR. For graphs of bounded clique-width, the best known parsing algorithm for VR runs in cubic time. This implies that problems that can be defined in monadic second order logic, including NP-complete problems like 3-colourability, can be solved in linear time on graphs of bounded tree-width, and in cubic time on graphs of bounded clique-width.

4.1 The Graph Algebra HR

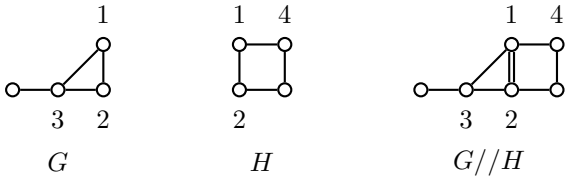
In this section we consider so called graphs with sources. Let $\mathcal{J}$ be the set of graphs (directed or undirected, possibly with loops or multiple edges) up to isomorphism. A **graph with sources** is a pair (G, src_G) where $G \in \mathcal{J}$ is a graph and src_G is a bijection from a finite subset of $\mathbb{N}$ onto a subset of $V(G)$. Thus, it is a graph where certain vertices are labelled by natural numbers, such that each label appears only once. The labelled vertices are called **sources**.

We let $\mathcal{JS}$ denote the set of all graphs with sources. The algebra HR consists of the set $\mathcal{JS}$ equipped with the following operations on sourced graphs:

Basic graphs. There are constants to define all connected graphs with at most one edge in $\mathcal{JS}$, eg.



Parallel composition. For Graphs $G, H \in \mathcal{JS}$ we set $G//H$ to be the disjoint union of G and H where vertices with the same label are fused, which means that, in the resulting graph, these vertices are identified.



G H $G//H$

Forget a source label. If G is a graph in $\mathcal{JS}$ and $a \in \mathbb{N}$ then $\mathrm{forget}_a(G)$ is the graph in $\mathcal{JS}$ which is obtained from G by making the vertex labelled with a (if such exists) a non-source.

Source renaming. For $G \in \mathcal{JS}$ and $a, b \in \mathbb{N}$ the graph $\mathrm{ren}_{a,b}(G)$ is obtained from G by exchanging the labels of a source labelled a and a source labelled b (the operation replaces one label by the other one if one of the labels does not exist).

The graph algebra HR can be used to get control of the tree-width, as is formalized by the following proposition.

Proposition 1. *A graph has tree-width at most k if and only if it can be constructed from basic graphs with at most $k+1$ labels by using the operations $//$,* $\mathrm{ren}_{a,b}$*, and* forget_a*.*

This implies that all graph classes that are equational with respect to the algebra HR necessarily have a bounded tree-width: In every equation system over HR, only a finite number of labels are used, which means that every graph of its language has a construction (following the equations) with a bounded number of labels.

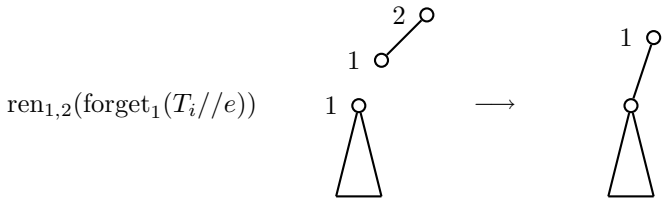


Figure 2: Extending a tree by an edge.

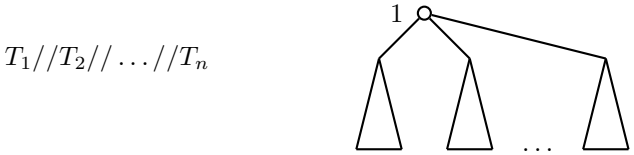


Figure 3: Fusing n trees at their roots.

Example 2. Trees have tree-width 1, they can thus be constructed using the operations from above and only two source labels. Let us explicitly construct a tree $T = (V, E)$ rooted at some arbitrary vertex r.

1. If T is a basic graph, we return the constant defining T, with its root labelled by 1.

2. Otherwise, let $T_1, \ldots, T_n$ denote the subtrees of T rooted at the children of r. Let e denote the constant that defines a graph, consisting of exactly one edge with its ends labelled 1 and 2, respectively. We extend each of the graphs T_i by one edge, forgetting the old root and letting the new vertex become the new root, again labelled with 1 (see Fig. 2).

   Then we successively fuse the extended subtrees at their roots (see Fig. 3).

Note that the procedure above can be written as an equation system over HR, which means that the class of trees is HR-equational.

4.1.1 HR-expressions provide tree-decompositions

Each expression built from the operations in HR has an expression tree or syntax tree, in which every node corresponds to an operation and the children of that node correspond to the input for the operation.

The syntax tree of an expression corresponds, in a natural way, to the alternative tree-decomposition model (see Fig. 4): For this model, we draw every edge in one of the boxes, it appears in. Furthermore, we draw dotted lines between the appearances of each vertex in different boxes. Using this technique, we obtain a model in which both the tree-structure and the underlying graph are visible.

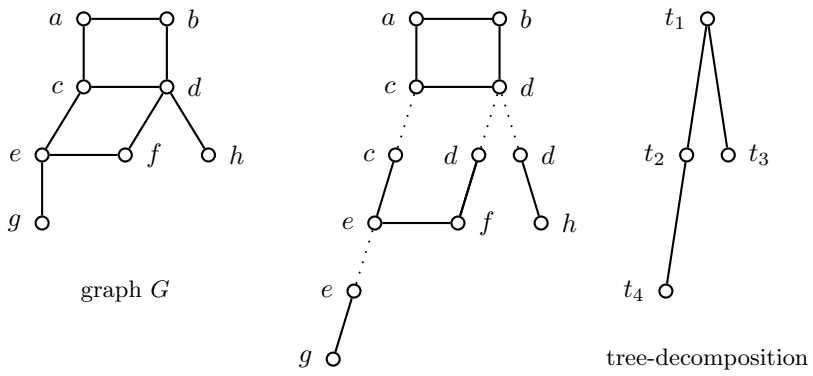


Figure 4: An alternative visualization for tree-decompositions.

Thus, if we can construct a graph by a HR-expression, we get its tree-decomposition for free.

4.2 The Graph Algebra VR

Now we consider vertex-labelled graphs which we will call graphs with ports. Here, let $\mathcal{G}$ denote the set of all simple graphs (directed or not) up to isomorphism. A labelled graph or a **graph with ports** is a pair (G, port_G) where $G \in \mathcal{G}$ is a graph and port_G is a mapping $\mathrm{port}_G : V(G) \to \mathbb{N}$. If $\mathrm{port}_G(v) = p \in \mathbb{N}$ for a vertex $v \in V(G)$ we call v a **p-port**. In contrast to the graphs with sources from the last section, *all* vertices of the graphs considered here are labelled, and different vertices can bear the same label. We let $\mathcal{GP}$ denote the set of graphs with ports up to isomorphism.

We let VR be the algebra on the set $\mathcal{GP}$ with the following operations:

Basic graphs. For every $a \in \mathbb{N}$ there is a constant $\mathbf{a}$ defining a graph consisting of a single vertex labelled with a.

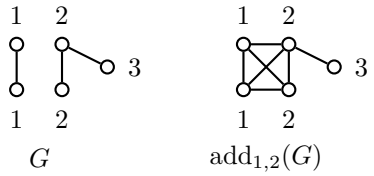
Disjoint union. The graph $G \oplus H$ is the disjoint union of two graphs in $\mathcal{GP}$.

Vertex relabellings. The graph $\mathrm{ren}_{a\to b}(G)$ is the graph obtained from G while every vertex labelled a in G is relabelled into b.

Add edges. For different labels $a, b \in \mathbb{N}$ the graph $\mathrm{add}_{a,b}(G)$ is obtained from G by adding (un-)directed edges from every a-port to every b-port.

Note, that the number of edges that can be added in one operation is not bounded. Thus we can easily create large bicliques in a graph.

A well-formed expression from these operations is called a **k-expression** if it uses at most k different labels. The **clique-width** $\mathrm{cwd}(G)$ of a graph G is



the minimum integer k such that there is a k-expression defining G. Unlabelled graphs are considered as labelled graphs where all vertices bear the same label. Thus the above definition applies also to unlabelled graphs.

4.2.1 Examples and Properties

Obviously every graph that contains at least one edge has clique-width at least 2. It is also easy to see, that cliques have clique-width at most 2: let $t_1 := \mathbf{1}$ denote the 2-expression defining a single vertex labelled with 1, i.e. K_1, and let t_n be a 2-expression defining K_n where all vertices are labelled with 1. Then the expression

$$t_{n+1} = \mathrm{ren}_{2\to 1}(\mathrm{add}_{1,2}(t_n \oplus \mathbf{2}))$$

defines the clique on $n+1$ vertices all labelled with 1.

At this point we should note that the notion of clique-width, unlike tree-width, is sensitive to edge-directions. Cliques have clique-width 2 whereas tournaments (directed cliques) have unbounded clique-width.

The following result was shown in [11]:

Proposition 3. *If a set of simple graphs has bounded tree-width, it has bounded clique-width, but not vice-versa. In particular*

$$\mathrm{cwd}(G) \leq 2^{\mathrm{twd}(G)-1} - 1$$

holds for every undirected simple graph.

As we already know, cliques on n vertices have tree-width $n-1$ but their clique-width is bounded by 2. Therefore, a function bounding the tree-width of graphs in terms of their clique-width can not exist.

It has been shown that the problem of deciding $\mathrm{cwd}(G) \leq k$ for a given integer k and a given graph G is NP-complete. Deciding $\mathrm{cwd}(G) \leq 3$ is possible in polynomial time but for any other fixed integer $k \geq 4$ the complexity of deciding $\mathrm{cwd}(G) \leq k$ is still unknown.

Nevertheless, there exists a cubic approximation algorithm that, for a given integer k and a graph G, either answers correctly that $\mathrm{cwd}(G) > k$ or produces a 2^{3k+1}-expression, see [13].

This result makes use of the notion of the rank-width. Rank-width is a graph parameter that is equivalent to clique-width in the sense that every graph class of bounded clique-width has also bounded rank-width and vice-versa. More

specifically it has been shown in [12] that for every graph G the following inequalities hold:

$$\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1.$$

As we will see in the next sections, the approximation algorithm for deciding $\mathrm{cwd}(G) \leq k$ yields FPT-algorithms for many hard problems.

5 Monadic Second-Order Logic, Recognizability and Decidability

Our ultimate goal is to check whether a graph G or a whole set of graphs $\mathcal{C}$ satisfies a property φ. Instead of doing this directly, it is equivalent to check whether $G \in L_\varphi$ or $\mathcal{C} \cap L_{\neg\varphi} = \emptyset$, respectively, where L_φ is the set of all graphs satisfying φ:

$$L_\varphi = \big\{ G \text{ a graph} \quad | \quad G \vDash \varphi \big\}.$$

As we will see in this section, the set L_φ is recognizable, so the question $G \in L_\varphi$ can be decided by a tree automaton, which can be constructed and be run in fixed-parameter tractable time. Furthermore, $\mathcal{C} \cap L_{\neg\varphi}$ can be checked for emptiness in finite time if the expressiveness of $\mathcal{C}$ and φ are 'balanced' (with respect to VR/HR-equational and MSO/MSO_2).

5.1 Definability

Let $\mathcal{L}$ denote any logic. A set of graphs L is **$\mathcal{L}$-definable** if there exists some formula $\varphi \in \mathcal{L}$ such that $L_\varphi = L$. As an example, the property that a graph is connected (i.e., the set of connected graphs) is not FO-definable but it is MSO-definable, as we can express this property by saying that a graph $G = (V, E)$ is connected if and only if

> there is *no* set $X \subset V$ such that X is nonempty, $V \setminus X$ is nonempty and there are no edges between X and $V \setminus X$.

In a formal standard syntax, this is:

$$\varphi \quad = \quad \neg\exists X((\exists x X(x)) \wedge (\exists x \neg X(x)) \wedge \forall x \forall y(X(x) \wedge \neg X(y) \Rightarrow \neg E(x, y)))$$

Recall that MSO allows quantification over vertex sets only. We can extend the expressive power of MSO by allowing quantification over edge sets, more precisely by considering edges as elements of the model and replacing the adjacency predicate by an incidence predicate. This extended language we denote by MSO_2. Formally, the signature of our language consits of a single ternary relation inc and we interpret $\mathrm{inc}(x, y, z)$ as 'x is an edge with tail y and head z' or the corresponding statement for undirected graphs. The formula $\exists y, z\, \mathrm{inc}(x, y, z)$ holds if and only if x is an edge and $\exists x\, \mathrm{inc}(x, y, z)$ holds if and only if y and z are adjacent. Thus we can transform every MSO-formula into an equivalent MSO_2-formula.

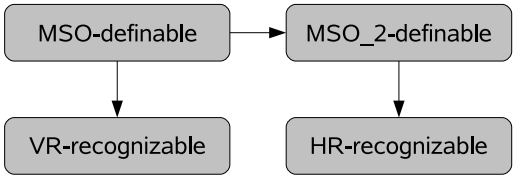


Figure 5: Overview of all properties of graph classes that we consider. An *arrow* indicates an "implies"-relation.

MSO_2 is strictly more expressive than MSO. Hamiltonicity is an example for a property that cannot be expressed in MSO, but which can well be expressed in MSO_2. A graph $G = (V, E)$ has a Hamilton cycle if and only if

> there is a set $Y \subset E$ such that Y covers all vertices, Y forms a 2-regular graph and no subset of Y forms a 2-regular graph.

5.2 Definabiliy implies Recognizability

We have now seen several formal tools for defining sets of graphs. The automata-theoretic notion of recognizability, the concept of expressing a set of graphs as a least solution of a system of equations and now the concept of definability in a logical language (cf. Fig. 5). Of these, the last concept is closest to everyday mathematical language. The following fundamental theorem now states that we can convert a logical formula into a finite-state automaton describing the same set of graphs. Moreover we have an effective method for constructing the automaton corresponding to the given formula.

Theorem 4. *Let L be a set of finite graphs.*

- *If L is* MSO*-definable, then L is* VR*-recognizable.*
- *If L is* MSO_2*-definable, then L is* HR*-recognizable.*

This is an analogoue of the theorem of Doner [15] and Thatcher and Wright [16] concerning languages of words. Note that in contrast to the case of graphs, recognizability *does* imply MSO-definability in the case of words.

Theorem 5. *Let L be a set of words. L is recognizable $\Leftrightarrow$ L is* MSO*-definable.*

The algorithmic consequences of having a description of a set of graphs in terms of a finite-state automaton we will see in section 6, giving us one important application of theorem 4. The other application is that we can derive some positive decidability results.

5.3 Positive Decidability Results

Let $\mathcal{L}$ be a logic and L a set of graphs. The **$\mathcal{L}$-satisfiability problem** on L is the following:
INPUT: A formula $\varphi \in \mathcal{L}$.
QUESTION: Does there exits a $G \in L$ such that $G \vDash \varphi$?

Note that checking whether a formula φ holds for some graph in L is equivalent to checking that $\neg\varphi$ holds for all graphs in L. A problem is **decidable** if there is an algorithm that for any input returns the correct answer in finite time. So if an infinite set of graphs L has a decidable MSO-satisfiability problem, this means that we can decide in finite time whether φ holds universally on the infinite set L, for any MSO formula φ. The following corollaries of the main theorem provide us with rich classes of sets L for which we can do precisely that.

Corollary 6. *Every* VR*-equational set of graphs has a decidable* MSO*-satisfiability problem.*

Corollary 7. *Every* HR*-equational set of graphs has a decidable* MSO_2*-satisfiability problem.*

The corollaries can be derived from the theorem via the following two important facts (Theorem 3.51 and 3.58 and Proposition 3.59 of [8]).

Lemma 8. *The intersection of an F-equational and an F-recognizable set is F-equational. The equations defining the intersection can be effectively computed.*

Lemma 9. *Given the defining set of equations, it is decidable whether an F-equational language is empty or not.*

This is analogous to the situation in formal language theory: The intersection of a context-free and a regular language is context-free and a context-free language can be tested for emptiness effectively. (See e.g. [17].)

The two positive decidability results are strong, but graph theorists need not fear that computers will take over their jobs. There are two reasons for that.

First, the proofs are constructive and the algorithms we obtain from them even come with an upper bound on the running time. Yet, the decision algorithms we obtain from them have a running time that is not anywhere near practical.

Second, most infinite sets of graphs that are of interest have unbounded clique-width and thus cannot have a VR or HR description. Furthermore, many interesting graph properties are not $\mathrm{MSO}_{(2)}$-definable. The negative decidability results in section 8 tell us that this is not a shortcoming of the theorem but a fundamental problem: They state that, in a certain sense, the two corollaries are best possible.

6 Inductive Computations and Recognizability: Fixed-Parameter Tractable Algorithms

In this section we will establish the connection between recognizability in graph algebras and fixed parameter tractability for certain problems on graphs, namely for those that are definable in monadic second order logic.

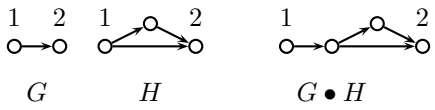
Beforehand, we will introduce the notion of an inductive set of properties, which yields an alternative characterisation for recognizable sets, which might appeal more to our intuition than the definition via automatons. Reading the subsections 6.1 and 6.2 is not necessary in order to understand the FPT-result though. They can as well be skipped.

6.1 Example: Properties of Series-Parallel Graphs

We consider the task of checking properties for certain sets of graphs. Again, we refer to [8] for further details on the proposed definitions, results, and examples. We consider examples for properties on graphs that can be checked by inductive computation if the graph is given by a defining term.

Let $\mathcal{D}_2$ denote the set of finite directed graphs with two distinct vertices marked 1 and 2 up two isomorphism. We consider a sub-algebra of HR on $\mathcal{D}_2$ with the operations $//, \bullet$ and e, where e denotes a constant defining a graph, that consists of exactly one edge with its ends marked 1 and 2 respectively. The operation $//$ is defined as under 4.1 and $\bullet$ is defined by

$$G \bullet H = \text{forget}_3(\text{ren}_{2,3}(G) // \text{ren}_{1,3}(H)).$$



The graphs generated by terms in this algebra are called **series-parallel graphs**. These graphs are defined by the equation

$$S = S//S \cup S \bullet S \cup \{e\}$$

where $S \subseteq \mathcal{D}_2$. We will now show how to prove properties for this set of graphs by induction. Our aim to prove that properties P_i hold for all graphs in S, or to determine if they hold for particular graphs in S, where

$$\begin{array}{lll} P_1(G) & :\Longleftrightarrow & \text{G is connected} \\ P_2(G) & :\Longleftrightarrow & \text{G is planar} \\ P_3(G) & :\Longleftrightarrow & \text{G is 2-colourable.} \end{array}$$

As for the first property, it is routine to see that the following two facts hold:

Basis. e is connected

Induction. $P_1(G) \wedge P_1(H) \Rightarrow P_1(G//H) \wedge P_1(G \bullet H)$.

This implies that every graph in S is connected. Although every graph in S is in fact also planar it is not possible to see this using exactly the same argument as above: Consider as H the complete simple directed graph on 5 vertices minus one edge such that $H//e$ is complete, hence not planar. Therefore, it is not true that the planarity of two graphs G and H in S implies the planarity of $G//H$. However, in order to see that every graph in S is planar we can use the stronger property

$$Q(G) : \Longleftrightarrow \text{ G has a planar drawing with its two sources on the outer face.}$$

This property admits the same inductive prove that we saw for P_1 and this implies that P_2 is also true for all series-parallel graphs.

The property P_3 does not hold for all series-parallel graphs. The graph

$$T = e//(e \bullet e)$$

is a triangle, thus it is not 2-colourable. This example shows also that we can not conclude $P_3(G//H)$ from $P_3(G)$ and $P_3(H)$, since e and $e \bullet e$ are 2-colourable, but T is not. We are therefore interested in checking 2-colourability for a given series-parallel graph, where we assume that such a graph is given by a formula. To achieve this, we introduce the following auxiliary properties:

$$\begin{array}{rcl} \text{same}(G) & : \Longleftrightarrow & \text{G is 2-colourable with sources of the same colour} \\ \text{diff}(G) & : \Longleftrightarrow & \text{G is 2-colourable with sources of different colours} \end{array}$$

For these properties we can prove the following rules:

$$\begin{array}{rcl} \text{same}(e) & = & \mathit{false} \\ \text{diff}(e) & = & \mathit{true} \\ \text{same}(G//H) & \Longleftrightarrow & \text{same}(G) \wedge \text{same}(H) \\ \text{diff}(G//H) & \Longleftrightarrow & \text{diff}(G) \wedge \text{diff}(H) \\ \text{same}(G \bullet H) & \Longleftrightarrow & (\text{same}(G) \wedge \text{same}(H)) \vee (\text{diff}(G) \wedge \text{diff}(H)) \\ \text{diff}(G \bullet H) & \Longleftrightarrow & (\text{same}(G) \wedge \text{diff}(H)) \vee (\text{diff}(G) \wedge \text{same}(H)) \end{array}$$

Now, for every term t in $< \mathcal{D}_2, //, \bullet, e >$ we can use these rules two compute the pair of boolean values $(\text{same}(\text{val}(t)), \text{diff}(\text{val}(t)))$, where $\text{val}(t)$ denotes the graph that is defined by the term t. The graph is 2-colourable if and only if one of these values is true. Thus, we can check 2-colourability of series-parallel graphs by induction on the structure of the term t.

It is important to note that, using these rules, we can compute the values of $\text{same}(f(G,H))$ and $\text{diff}(f(G,H))$, where $f \in \{//, \bullet\}$, by means of finitely many (in this case 4) values of same and diff of the arguments of f. This example motivates the definition of an inductive set of properties that we will introduce next.

6.2 Inductive Computations and Recognizability

We will now generalize the method we used to check 2-colourability in the algebra of series-parallel graphs to check properties in arbitrary F-algebras $\mathbb{M}$. A **property** P in such an algebra is a mapping $P : \mathbb{M} \to \{true,\ false\}$.

Definition 10. Let $\mathbb{M}$ be a F-algebra and let $\mathcal{P}$ be a finite set of properties. The set $\mathcal{P}$ is **F-inductive** if for every $P \in \mathcal{P}$ and $f \in F$ of arity $n > 0$ there exists a boolean formula B depending only on P and f that computes the boolean value $P(f_{\mathbb{M}}(m_1, \ldots, m_n))$ for all $m_1, \ldots, m_n$ in terms of l (finitely many) values $Q_j(m_i)$ where $Q_l \in \mathcal{P}$ and $i = 1, \ldots, n$,$j = 1, \ldots, l$.

In our previous example the set $\{\text{same}, \text{diff}\}$ is an inductive set of properties in the algebra $< \mathcal{D}_2, //, \bullet, e >$ of series-parallel graphs. If we consider for example the operation $\bullet$ and the property same we can express $\text{same}(G \bullet H)$ by

$$\text{same}(G \bullet H) = B(\text{same}(G), \text{diff}(G), \text{same}(H), \text{diff}(H))$$

where B is the boolean formula

$$B(p_1, p_2, p_3, p_4) = (p_1 \wedge p_3) \vee (p_2 \wedge p_4).$$

In Section 3, we defined the notion of a recognizable set with respect to an arbitrary algebra as the set that is accepted by an F-automaton. The following proposition establishes the connection between recognizable sets and inductive properties:

Proposition 11. *A subset L of an F-algebra $\mathbb{M}$ is recognizable if and only if it is the set of elements that satisfy a property belonging to a finite inductive set of properties $\mathcal{P}$.*

We already saw in 5 that for the graph algebras HR and VR the definability in monadic second order logic implies recognizability in these algebras.

Proposition 11 now shows us that an inductive set of properties can be translated into a tree-automaton. Checking m properties that form an inductive set can thus be implemented by this tree-automaton with 2^m states that work on terms t formed by the operations of the algebra. Such a computation takes time $\mathcal{O}(|t|)$, the running time is thus linear in the size of the expression. We will now see that we get thereby fixed parameter tractability results for graph problems that are definable in MSO Logic.

6.3 Checking MSO-Formulas is Fixed-Parameter Tractable

In this section, we will show that checking MSO-formulas on graphs of bounded tree-width or clique-width is fixed-parameter tractable with respect to the width as a parameter. The theorem is a result from the relationship between MSO logic and recognizability. By a MSO-problem we mean the problem of deciding whether or not a graph satisfies a certain MSO-formula.

Theorem 12. *Every* MSO*-problem on $\mathcal{J}$ is fixed parameter linear with respect to tree-width. Every-*MSO_2 *problem on $\mathcal{G}$ is fixed parameter cubic with respect to clique-width.*

Recall that $\mathcal{J}$ and $\mathcal{G}$ are classes of labelled graphs as defined in Section 4.1 and 4.2. The most important steps of the proof of this theorem are as follows:

1. Given a graph $G \in \mathcal{J}$ we can check for every k in time $\mathcal{O}(g(k)m)$ whether or not G has tree-width at most k and if so produce a term t which uses at most k labels defining G. (For the size of the graph m we can take the number of edges). This result is due to Bodlaender, see [14].

2. For graphs $G \in \mathcal{G}$ we can use a similar result that is due to Oum, see [13]. For these graphs we can we can either verify that a graph has clique-width more than k or construct a 2^{3k+1} expression for it (without knowing the exact value when it is between k and the exponential upper bound) for each k. This takes time $\mathcal{O}(g'(k)n)$, where n denotes the number of vertices of the graph.

3. In section 5 saw that MSO definability implies recognizability.

4. For each k and each MSO formula φ we can built the tree-automaton corresponding to the recognizable set. This construction is independent of the input graph. It can be done once and for all for each pair (φ, k). The automaton checks in linear time $\mathcal{O}(|t|)$ whether or not the input graph defined by t satisfies φ or not.

We should note that this result is of purely theoretical interest, since there are large constants involved. This is due to the large number of states in the tree-automaton and to the parsing algorithms. Nevertheless, if the input graph is given by a defining term, we can check MSO properties in linear time for graphs of bounded clique-width or bounded tree-width:

Theorem 13. *For graphs of clique-width at most k,*

- *each monadic second-order property (eg. 3-colourability),*
- *each monadic second-order optimization function (eg. distance), and*
- *each monadic second-order counting function (eg. number of paths),*

can be evaluated in linear time if the graph is given by a VR-expression.

Computing a VR-expression for a graph takes $\mathcal{O}(n^3)$ time in general, and only linear time if the graph has tree-width bounded by k.

7 Monadic Second-Order Transductions

We can use logical formulas to define transformations of graphs into other graphs. Considering such logical transformations is particularly useful as there

is no standard machine model for graph transductions. Actually, logical transformations are defined in a much broader sense not only for graphs, but between any two classes of relational structures. So technically speaking, a logical transduction is binary relation $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ are classes of relational structures and $\mathcal{R}$ will be considered a multivalued partial mapping.

We will only give the more intuitive definition for MSO transductions of graphs here. The full definition for arbitrary structures can be found in [5] or [6].

Definition 14. A definition scheme is a tuple $D_f = (\psi, \delta_1, \ldots, \delta_k, \varphi_{11}, \varphi_{12}, \ldots, \varphi_{kk})$ of MSO formulas for some $k \in \mathbf{N}$ where ψ, δ_i, and φ_{ij} have zero, one, and two free variables, respectively, and may have free set variables from a finite set W of unary relation variables. W is called the set of parameters and may be empty. D_f induces a partial mapping f from graphs and parameters to graphs so that for every graph G the structure $\mathsf{f}(\mathsf{G}, \mathsf{W})$ is defined in the following way:

$$\begin{aligned}
\mathsf{f}(\mathsf{G}, \mathsf{W}) &= \begin{cases} (V_{f(G,W)}, E_{f(G,W)}) \text{ if } \mathsf{W} \models \psi, \\ \text{undefined otherwise} \end{cases} \quad \text{where} \\
V_{f(G,W)} &= \bigcup_{i=1}^{k} \{(u, i) \in V_G \times \{i\} \mid \mathsf{G}, \mathsf{W} \models \delta_i(u)\} \\
E_{f(G,W)} &= \{((u, i), (v, j)) \mid \mathsf{G}, \mathsf{W} \models \delta_i(u) \wedge \delta_j(v) \wedge \varphi_{ij}(u, v)\}
\end{aligned}$$

Then f is called an MSO transduction with parameters W.

Some examples will illustrate this definition. They are taken from [8] where further examples can be found.

Example: Edge Complement

For a simple, undirected graph $G = (V, E)$ the edge compliment $G^* = (V, E^*)$ is the graph on V with edges $uv \in E^* \Leftrightarrow uv \notin E$. The assiciated definition scheme simply consists of the formulas δ and φ with

$$\begin{aligned}
\delta(u) &:= u = u \text{ (Boolean constant } \mathit{True}) \\
\varphi(u, v) &:= u \neq v \wedge \neg E(u, v) \text{ where } E \text{ is viewed as a binary relation on } V.
\end{aligned}$$

As we do not use any parameter dependence, we tacitly assume ψ to be set to *True*, so that our map is total. Notice that in this example, we do not need the power of MSO logic, so that edge complement is really a first-order transduction without parameters.

Example: The largest connected subgraph of G containing X

In this example, X will serve as a parameter so that our map f returns the connected component of an undirected graph G containing $X \neq \emptyset$. The definition

scheme is (ψ, δ, φ) with

$$\begin{aligned} \psi(X) &:= X \neq \emptyset \wedge \exists Y(X \subseteq Y \wedge \mathrm{CONN}(Y)) \\ \delta(X,u) &:= \exists Y(X \subseteq Y \wedge \mathrm{CONN}(Y) \wedge u \in Y) \\ \varphi(X,u,v) &:= \exists Y(X \subseteq Y \wedge \mathrm{CONN}(Y) \wedge u \in Y \wedge v \in Y \wedge E(u,v)) \end{aligned}$$

$\mathrm{CONN}(X)$ is an MSO formula expressing that X is connected.

Another example is the transduction with two (edge and vertex) set parameters X and Y that returns the minor of G after contracting edges in X and deleting edges and vertices in Y.

In the above example, the parameter X is obviously needed in order to specify the connected component of our choice. In general, MSO transductions may have to use a parameter even when its description does not seem to require one. An example is the transduction from a directed graph to its directed acyclic graph where the strongly connected components have been contracted. This transduction needs as a parameter a set that contains exactly one vertex from every strongly connected component because there is no canonical way to find such vertices within MSO formulas. This example is worked out in [8, p.31].

Example: Graph duplication with links between copies

This example demonstrates how the duplication of the graph is useful in MSO transductions. The duplication of an undirected graph $G = (V, E)$ is the graph $G_B = (V_B, E_B)$ on vertices $V_B = V \times \{1, 2\}$ with edges $(u, 1)(v, 2) \in E_B \Leftrightarrow uv \in E$. The associated transduction is given by the definition scheme $(\delta_1, \delta_2, \varphi_{11}, \varphi_{12}, \varphi_{21}, \varphi_{22})$ with

$$\begin{aligned} \delta_1 := \delta_2 &:= \mathit{True} \\ \varphi_{11} := \varphi_{22} &:= \mathit{False} \\ \varphi_{12}(u,v) := \varphi_{21}(u,v) &:= E(u,v) \end{aligned}$$

This justifies the statement that a transduction maps S to a structure "inside" $S \oplus \ldots \oplus S$.

Generally, MSO transductions are defined for general structures. For example, there is an MSO transduction between cograph-terms built from $\{\oplus_l, \otimes_l, \mathbf{1}_l\}$ and their associated cographs. This example is worked out in [8, p.33].

Properties of MSO transductions

Proposition 15. *The composition of two MSO transductions is an MSO transduction.*

The proof consists in the construction of the resulting transduction by conjunction of the original MSO formulas. Writing the proof down is not trivial, though, because one has to take care of parameters and copies of structures.

Next is the so-called fundamental property of MSO transductions.

Theorem 16. *Let τ be an MSO transduction that associates to every structure S a structure $\tau(S)$. Then for every MSO formula ψ there is an effectively computable MSO formula $\tau^{\#}(\psi)$ such that*

$$S \models \tau^{\#}(\psi) \Leftrightarrow \tau(S) \models \psi \tag{1}$$

$\tau^{\#}(\psi)$ is called the backwards translation of ψ.

Intuitively, the structure S contains enough information to describe $\tau(S)$. For graph transductions without duplication, $\tau^{\#}(\psi)$ is constructed from ψ by replacing the edge relation $E(u, v)$ with $\delta(u) \wedge \delta(v) \wedge \varphi(u, v)$ and restricting quantifiers $\exists x\mu$ and $\exists X\mu$ by replacing them with $\exists x(\delta(x) \wedge \mu)$ and $\exists X(\forall x(x \in X \Rightarrow \delta(x)) \wedge \mu)$, respectively. When the transduction uses duplication of the graph, the construction of the formula uses disjunction over the several vertex and edge conditions within restrictions and edge relation replacements. A full construction for general structures is contained in [2].

Corollary 17. *If the MSO satisfiability problem is decidable on a class of structures L, then it is decidable on the image of L under any MSO transduction.*

This corollary gives us more detailed insight on the relationship between MSO and MSO_2 logic. Any MSO_2 statement about a graph G transforms into an MSO statement about G's incidence graph $\mathrm{inc}(G)$. The following theorem tells us that on many classes of graphs, the incidence graphs are just images of MSO transductions:

Theorem 18. *The mapping $G \to inc(G)$ is an MSO transduction on the following classes of graphs:*

- *degree $< d$,*
- *tree-width $< k$,*
- *planar graphs,*
- *graphs without some fixed graph H as a minor,*
- *uniformly k-sparse graphs (graphs of average degree $< k$).*

Thus, on the classes of graphs from Theorem 18, the MSO_2 satisfiability problem is decidable if and only if the MSO satisfiability problem is decidable.

Here are some more results that involve MSO transductions:

Proposition 19. *A set of graphs is* VR-equational *if and only if it is the image of (all) binary trees under an* MSO transduction. *VR-equational sets of graphs are stable under* MSO *transductions. A set of graphs has bounded* clique-width *if and only if it is the image of a set of binary trees under an* MSO *transduction.*

Proposition 20. *A set of graphs is* HR-equational *if and only if it is the image of (all) binary trees under an* MSO_2 *transduction. HR-equational sets of graphs are stable under* MSO_2 *transductions. A set of graphs has bounded* tree-width *if and only if it is the image of a set of binary trees under an* MSO_2 *transduction.*

8 Links between MSO logic and combinatorics: Seese's theorem and conjecture

In the context of Theorem 18, a result from Seese [4, 10] is interesting:

Theorem 21. *If the MSO_2 satisfiability problem is decidable on a class of graphs L (i.e., the MSO satisfiability problem is decidable on $inc(L)$), then it has bounded tree width.*

Analogously, Seese conjectured that if MSO satisfiability is decidable on L, then L has bounded clique-width. This is known as Seese's Conjecture. In 2004, Courcelle and Oum proved the following [7]:

Theorem 22. *If C_2MSO satisfiability is decidable on a class of graphs L, then L has bounded clique-width.*

Here C_2MSO denotes MSO logic augmented with a predicate that expresses that a set has even cardinality. This statement is weaker than Seese's conjecture because a smaller amount of classes satisfies the premise. Seese's conjecture can be considered a converse to the statement that on the classes L_k of graphs with clique-width $\leq k$ MSO satisfiability is decidable.

Here is a proof sketch of Theorem 21:

- Suppose L has unbounded tree-width, then L's set of minors contains all $k \times k$ grids by a result from Robertson and Seymour.
- If a class contains all $k \times k$ grids, then its MSO_2 satisfiability problem is undecidable.
- All minors of L constitute the image of L under an MSO_2 transduction.
- Similarly as in Corollary 17, this implies that if L's MSO_2 satisfiability problem is decidable, then so is that of its minors, which yields the contradiction.

The proof of Theorem 22 follows a similar pattern. For clique-width, a special kind of minor relation, called *vertex minors*, is introduced (see e.g. [9]). A vertex minor of a graph G is constructed by repeated vertex deletions and *local complementations* where the edge relations between the neighbors of a vertex v are reversed. Courcelle and Oum prove that if L's C_2MSO satisfiability problem is decidable, then so is that of the class of all vertex-minors of L because constructing the vertex-minors of L is a C_2MSO transduction. Also, if a class of bipartite graphs C has unbounded clique-width, then the set of its vertex-minors contains all S_k graphs, which can be further transformed by an MSO transduction into the $k \times k$ grids, on which MSO is not decidable. They finally establish an equivalence between (directed and undirected) graphs and bipartite undirected graphs that finishes the proof.

In this specific setting, the cardinality predicate in C_2MSO gives us the power to speak about edge relations in $G * v$, the graph G after local complementation

at v. If we apply local complementation to n different vertices that all have two vertices u and v as neighbors, then the edge relation between u and v is reversed exactly when n is odd.

The definition of the bipartite graphs S_k is as follows: $S_k = (A, B, E)$ with $A = \{1, \ldots, k^2 - 1\}$ and $B = \{1, \ldots, k^2 - k\}$ so that $ij \in E \Leftrightarrow i \leq j \leq i + k - 1$. Figure 6 shows an S_3 and a 3×4 grid that can be constructed from S_3 using an MSO transduction.

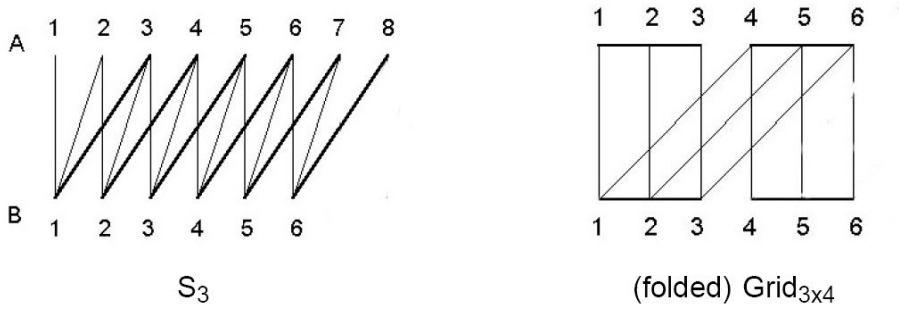


Figure 6: An S_3 and the corresponding grid.

The C_2MSO transduction from S_k is then defined by first using C_2MSO to define an ordering on A and B by x *is consecutive to* $y \Leftrightarrow |n_G(x) \triangle n_G(y)| = 2$. Thus, we can identify edges from $i \in B$ to $i \in A$ and from $i \in B$ to $i + k - 1 \in A$ in C_2MSO. Thus we can add and delete the relevant edges in order to make the S_k into a $k \times k$ grid.

This technique uses the parity predicate only to define a linear order. If we are given a natural order on the vertices in another way, MSO is just as powerful as C_2MSO. This is illustrated by the following Corollary:

Corollary 23. *If MSO satisfiability is decidable in a class of directed acyclic graphs with Hamiltonian directed paths, then the class has bounded clique-width and it is the image of a class of trees under an MSO transduction.*

9 Open Questions

Question 1 (Blumensath, Weil, Courcelle)

Which operations, quantifier-free definable or not, yield equivalent extensions of VR, HR, that is, alternative signatures giving the same equational and recognizable sets?

Question 2

Under which operations, quantifier-free definable or not, are REC(VR) and REC(HR) closed? The case of HR is considered in [3].

Question 3

Is it true that the decidability of MSO satisfiability implies bounded clique-width, as conjectured by Seese?

References

[1] Achim Blumensath and Bruno Courcelle (2006) *Recognizability, hypergraph operations, and logical types*. Information and Computation, 204 (853-919).

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

[3] Bruno Courcelle (1994) *Recognizable sets of graphs: equivalent definitions and closure properties*. Math. Struct. Comput. Sci. 4:1-32.

[4] Bruno Courcelle (1995) *The monadic second-oder logic of graphs VIII: orientations*. Annals of Pure Applied Logic 72 (103-143).

[5] Bruno Courcelle (1997) *The Expression of Graph Properties and Graph Transformations in Monadic Second-Order Logic*. In G. Rozenberg, Ed., Handbook of graph grammars and computing by graph transformations, vol. 1: Foundations (313-400), World Scientific, London.

[6] Bruno Courcelle (2003) *The monadic second-order logic of graphs XIV: uniformly sparse graphs and edge set quantifications*. Theoretical Computer Science, 299 (1-3).

[7] Bruno Courcelle and Sang-Il Oum (2007) *Vertex-Minors, Monadic Second-Order Logic, and a Conjecture by Seese*. J. Combin. Theory, Ser. B 97 (91-126).

[8] Bruno Courcelle (to appear) *Graph algebras and monadic second-order logic*. Retrieved on October 24, 2007, from http://www.labri.fr/perso/courcell/ActSci.html.

[9] Sang-Il Oum (2005) *Rank-width and vertex-minors*. J. Combin. Theory, Ser. B 95 (79-100).

[10] Detlef Seese (1991) *The structure of the models of decidable monadic theories of graphs*. Annals of Pure Applied Logic, 53:169:195.

[11] Bruno Courcelle and Stephan Olariu (2000) *Upper bounds to the clique width of graphs*. Discrete Appl. Math., 101:77-114.

[12] Sang-il Oum and Paul Seymour(2006) *Approximating clique-width and branch-width*. J. Combin. Theory Ser. B, 96(4):514-528.

[13] Sang-il Oum (2005) *Approximating Rank-Width and Clique-Width Quickly*. Springer Lecture Notes in Computer Science 3787 (49-58).

[14] Hans Bodlaender (1996) *A linear time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput. 25 (1305-1317).

[15] J. Doner (1970) *Tree acceptors and some of their applications*, J. Comput. System Sci. 4 (406-451).

[16] J.W. Thatcher and J.B. Wright (1968) *Generalized finite automata theory with an application to a decision problem of secondorder logic*, Math. Systems Theory 2 (57-81).

[17] Michael A. Harrison, *Introduction to Formal Language Theory*, Addison Wesley Longman Publishing Co, 1978.