Logical Description of Context-Free Graph Languages

Joost Engelfriet and Vincent van Oostrom [*]

Department of Computer Science, Leiden University
P.O.Box 9512, 2300 RA Leiden, The Netherlands
email: engelfri@wi.leidenuniv.nl

Abstract. A graph language L is in the class C-edNCE of context-free edNCE graph languages if and only if $L = f(T)$ where f is a function on graphs that can be defined in monadic second-order logic and T is the set of all trees over some ranked alphabet. This logical characterization implies a large number of closure and decidability properties of the context-free edNCE graph languages. Rather than context-free graph grammars we use regular path descriptions to define graph languages.

1 Introduction

Context-free graph grammars are a general formalism to define sets of graphs in a recursive fashion, just as context-free grammars are used to recursively define sets of strings. Since many interesting graph properties are recursive in one way or another, context-free graph grammars provide a means to study such properties in general. As opposed to the case of strings, there are many different types of context-free graph grammars, and there is no agreement on which is the "correct" one. Here we consider the class of C-edNCE graph languages generated by the context-free (or confluent) edNCE graph grammars, which was first investigated in, e.g., [Kau, Bra, Schu, Oos, Eng2]. One advantage of the class C-edNCE is that it is the largest known class of context-free graph languages (where 'context-free' is taken in the sense of [Cou1]). It includes, e.g., the HR (i.e., Hyperedge Replacement) languages of [BC, HK, Hab], the B-NLC languages of [RW1, RW2], and the B-edNCE languages of [ELW, EL]. Thus, results on C-edNCE apply to a quite large class of recursive praph properties. A second advantage of C-edNCE is that it seems to be robust in the sense that it can be characterized in several different ways. It is shown in [CER] that C-edNCE is also generated by a specific type of handle rewriting hypergraph grammars, generalizing hyperedge replacement. It is also shown in [CER] that C-edNCE has a least fixed point characterization in terms of very simple graph operations; in other words, C-edNCE is the class of equational subsets of a certain algebra of graphs (for the notion of 'equational set' see [MW]). In [EO] (see [Oos, Eng2])

[*] The present address of the second author is: Faculty of Mathematics and Computer Science, Vrije Universiteit, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, email: oostrom@cs.vu.nl

it is shown that the C-edNCE graph languages can be described in terms of regular tree and string languages. A "regular path description" of a graph language mainly consists of a regular tree language together with a regular string language for each possible edge label. Each tree t from the regular tree language determines a graph $gr(t)$ in the graph language as follows: $gr(t)$ has the same nodes as t, with the same labels, and there is a γ-labeled edge from node u to node v if the string of labels on the shortest (undirected) path from u to v in t belongs to the regular string language associated with γ. To be precise, $gr(t)$ has in fact only those nodes of t that have certain labels, and the node labels in the graph are obtained from those in the tree by a relabeling. In this paper we will not consider graph grammars but only regular path descriptions, which are easier to understand for readers familiar with formal (tree) language theory. For this reason, the class C-edNCE is also denoted RPD.

The main result of this paper is a characterization of C-edNCE in terms of monadic second-order logic on trees, strengthening our belief that C-edNCE is a robust class of context-free graph languages. We first define the class MSOF of monadic second-order definable functions; they are unary functions that transform graphs into graphs. Then the result is that a graph language L is in C-edNCE if and only if $L = \{f(t) \mid t \in T_\Delta\}$ where f is in MSOF and T_Δ is the set of all trees over a ranked alphabet Δ. Intuitively, the recursive (context-free) aspect of a graph in L is captured by the tree t, whereas the actual construction of the graph $f(t)$ from t is specified in monadic second-order logic. In what follows we abbreviate 'monadic second-order' by MSO. As in the case of a regular path description, the nodes of $f(t)$ are a subset of the nodes of t. Which nodes of t are in $f(t)$ (and which labels they have), and which edges have to be established between these nodes, is described by MSO formulas, to be interpreted on t. To be precise, f is specified by the following formulas: a closed "domain formula" ϕ_{dom} that should be satisfied by t (f is in general a partial function), for each node label σ of $f(t)$ a "node formula" $\phi_\sigma(u)$ expressing that u will be a node of $f(t)$ with label σ, and for each edge label γ an "edge formula" $\phi_\gamma(u,v)$ expressing that there will be a γ-labeled edge from u to v in $f(t)$.

The usefulness of this MSO characterization is that MSO logic is a convenient language to talk about graphs, and hence can be used as a specification language for sets of graphs and functions on graphs. The MSO specification language is more convenient than the (rather technical) formalisms of context-free graph grammars or regular path descriptions, because it allows one to directly express properties of graphs, in the way they are usually defined in graph theory. In particular, the MSO characterization is completely grammar independent, in the sense that there is no need to construct a (graph or tree) grammar to express a recursive property of graphs; instead, the recursion is incorporated in the input trees of the MSO definable function.

MSO characterizations of classes of languages generated by grammars date back to [Buc, Elg], where it is shown that the class of regular string languages equals the class of MSO definable string languages. This was generalized to trees in [Don, TW]: a tree language is regular if and only if it is MSO definable.

For a discussion of such results see Sections 3 and 11 of [Tho], and [Eng4]. Note that these characterizations differ from the one of C-edNCE: by definition, a language L is MSO definable if there exists a closed MSO formula ϕ such that L consists of all strings (trees, graphs) that satisfy ϕ. The class of MSO definable graph languages is incomparable with C-edNCE (cf. [Cou2]). The proof of our MSO characterization is heavily based on the classical results of [Buc, Elg, Don, TW]; for instance, the domain of an MSO definable function is MSO definable (by $\phi_{\rm dom}$) and corresponds directly to the regular tree language of the regular path description. A relationship between context-free graph languages and MSO logic was first established by Courcelle in [Cou2] (see also Section 4 of [Cou3]), where he showed that the class of HR context-free graph languages is closed under intersection with MSO definable graph languages (generalizing the corresponding result for strings). For C-edNCE and several of its subclasses similar results are shown in [Cou1, CER]. These intersection results provide meta-theorems for closure and decidability properties of these classes ('meta' in the sense that, historically, such intersection results were first proved for several specific, MSO definable, properties). Other such meta-theorems (not using MSO logic) are shown, e.g., in [HKV1, HKV2, LW]; as observed above, the advantage of MSO logic is that it is grammar independent, well known, and easy to use. Our MSO characterization of C-edNCE generalizes Courcelle's intersection result for C-edNCE. It implies additional MSO meta-theorems for closure and decidability properties of C-edNCE, in particular generalizing the results of [HKV2] from HR to C-edNCE.

The structure of this paper is as follows. Section 1 contains some preliminary, and mostly well-known definitions on graphs, trees, and strings, on regular tree languages, and on monadic second-order logic. In Section 3 we recall the notion of a regular path description from [EO], together with three of its special cases: type A, type B, and type LIN. The regular tree language of a regular path description of type LIN consists of linear trees (which are close to strings). In Section 4 the class MSOF of MSO definable functions is introduced, together with three of its special cases which are also called type A, type B, and type LIN. MSO definable functions of type LIN are applied to strings rather than trees. It is shown that MSOF is closed under composition. Section 5 contains the main result: the MSO characterization of the class RPD of graph languages that can be described by regular path descriptions (i.e., the class C-edNCE). Similar characterizations hold for types A, B, and LIN. Section 6 contains the closure and decidability results that follow from the MSO characterization. In particular, C-edNCE is closed under MSO definable functions.

The main result of this paper was established in 1988, and presented in [Oos, Eng1] and [Eng3]. Its consequences for closure and decidability properties were obtained in 1990 and presented in [Eng3]. The only newly added result is the MSO characterization of the RPD languages of type A (i.e., the A-edNCE languages) which is based on the results of [EHL] (see [EO]).

In the meantime, an MSO characterization of the HR graph languages has been proved in [CE], which is surprisingly similar to the one of C-edNCE.

Roughly speaking, the vertices of the tree t are turned into both the nodes and the edges of the graph $f(t)$, and the incidence relation between nodes and edges is expressed by an MSO formula. Based on these two characterizations it is shown in [Cou7] that it is decidable whether or not a given C-edNCE language is HR. A general MSO characterization of the equational subsets of certain algebras of relational structures, with a binary gluing operation and all possible quantifier-free first-order definable unary operations, is presented in [Cou5]. For a recent survey on C-edNCE see [ER2]. Other surveys that discuss work on C-edNCE and HR are [Cou8, DHK, Eng5, Eng6]. For graph grammars in general see [Roz, ENRR, EKR, CEER].

2 Preliminaries

$\mathbb{N} = \{0, 1, 2, \dots\}$ and for $m, n \in \mathbb{N}$, $[m, n] = \{m, \dots, n\}$. The domain of a function f is denoted $\text{dom}(f)$.

2.1 Graphs, trees, and strings

The reader is assumed to be familiar with formal language theory (see, e.g., [HU]), in particular tree language theory (see, e.g., [GS]), and with the elementary concepts of graph theory. Strings and trees will (also) be viewed as particular types of graphs.

First we define graphs, i.e., directed graphs with labeled nodes (or vertices) and labeled edges. Let Σ be an alphabet of node labels and Γ an alphabet of edge labels. A *graph* over Σ and Γ is a tuple $H = (V, E, \lambda)$, where V is the finite set of nodes, $E \subseteq \{(v, \gamma, w) \mid v, w \in V, v \neq w, \gamma \in \Gamma\}$ is the set of edges, and $\lambda : V \to \Sigma$ is the node labeling function. The components of H are also denoted as V_H, E_H, and λ_H, respectively. Thus, we consider directed graphs without loops; multiple edges between the same pair of nodes are allowed, but they must have different labels. A graph is undirected if for every $(v, \gamma, w) \in E$, also $(w, \gamma, v) \in E$. Graphs with unlabeled nodes and/or edges can be modeled by taking Σ and/or Γ to be a singleton, respectively.

The set of all graphs over Σ and Γ is denoted $GR_{\Sigma,\Gamma}$. A subset of $GR_{\Sigma,\Gamma}$ is called a *graph language*.

As usual, two graphs H and K are *isomorphic* if there is a bijection $f : V_H \to V_K$ such that $E_K = \{(f(v), \gamma, f(w)) \mid (v, \gamma, w) \in E_H\}$ and, for all $v \in V_H$, $\lambda_K(f(v)) = \lambda_H(v)$. The reader is assumed to be familiar with the way in which concrete graphs are used as representatives of abstract graphs, which are equivalence classes of concrete graphs with respect to isomorphism. We are usually interested in abstract graphs, but mostly discuss concrete ones. For instance, whereas a graph language is defined to be a set of concrete graphs, we usually view it as a set of abstract graphs.

The rooted, ordered trees from tree language theory will be identified (as usual) with a special type of (abstract) graph: each vertex of the tree has a directed edge to each of its k children, $k \geq 0$, and the order of the children is indicated by using the numbers $1, \ldots, k$ to label these edges; the vertex is labeled by a symbol of rank k (from a ranked alphabet). For an example of a tree see Fig. 6(a). A *ranked alphabet* is an alphabet Σ together with a mapping rank : $\Sigma \to \mathbb{N}$. By $\mathrm{rks}(\Sigma)$ we denote the set $[1, m]$ where m is the maximal number $\mathrm{rank}(\sigma)$, $\sigma \in \Sigma$. A *tree* over Σ is a graph $t \in GR_{\Sigma,\mathrm{rks}(\Sigma)}$ with the following two properties: (1) there is a vertex r of t (its root) such that for every vertex v of t there is a unique (directed) path from r to v, and (2) every vertex v of t with label σ has exactly k outgoing edges, where $k = \mathrm{rank}(\sigma)$, and each $i \in [1, k]$ is the label of (exactly) one of these edges. The root of t, i.e., the unique vertex of t that has no incoming edges, is denoted $\mathrm{root}(t)$. The i-th child of a vertex v, i.e., the unique vertex w such that $(v, i, w) \in E_t$, is denoted $v \cdot i$. The *child number* of a vertex v is 0 if $v = \mathrm{root}(t)$, and i if v is the i-th child of its parent. As usual, for trees $t_1, \ldots, t_k$ and $\sigma \in \Sigma$ with $k = \mathrm{rank}(\sigma)$, we denote by $\sigma t_1 \cdots t_k$ the tree consisting of the disjoint union of $t_1, \ldots, t_k$ and a root that has label σ and has an i-labeled edge to the root of each t_i, $1 \leq i \leq k$. In this way every tree over Σ is denoted by a string over Σ. We write T_Σ for the set of all trees over Σ. A subset of T_Σ is called a *tree language*.

Strings over an (ordinary) alphabet will also be viewed as a special type of (abstract) graph: a chain of nodes that are labeled with the symbols of the string (and edges labeled by $*$). Let Σ be an alphabet. A *string* $\sigma_1 \cdots \sigma_n$, $n \geq 0$, with $\sigma_i \in \Sigma$, is identified with the graph $(V, E, \lambda) \in GR_{\Sigma,\{*\}}$ such that $V = \{v_1, \ldots, v_n\}$, $E = \{(v_i, *, v_{i+1}) \mid 1 \leq i < n\}$, and $\lambda(v_i) = \sigma_i$ for every $1 \leq i \leq n$. Note that the empty string is represented by the empty graph. For an example of a string see Fig. 5(a). As usual, Σ^* denotes the set of all strings over Σ. A subset of Σ^* is called a *string language*, or just a language. A language is *regular* if it can be recognized by a finite automaton (or generated by a right-linear grammar).

2.2 Regular tree languages

Regular tree grammars are recognized by finite tree automata and can be generated by regular tree grammars. Let Σ be a ranked alphabet.

A finite (deterministic) bottom-up *tree automaton* over Σ is a tuple $A = (Q, \{\sigma_A\}_{\sigma \in \Sigma}, F)$ where Q is a finite set of states, σ_A is a mapping $Q^k \to Q$ for every $\sigma \in \Sigma$ of rank k (the state transition function for σ), and $F \subseteq Q$ is a set of final states. For a tree $t \in T_\Sigma$, and a vertex $v \in V_t$, the *state reached* by A at v, denoted $\mathrm{state}_{t,A}(v)$, is defined recursively in a bottom-up fashion as follows: if v has label σ of rank k, then $\mathrm{state}_{t,A}(v) = \sigma_A(\mathrm{state}_{t,A}(v \cdot 1), \ldots, \mathrm{state}_{t,A}(v \cdot k))$.

The *language recognized* by A is $L(A) = \{t \in T_\Sigma \mid \mathrm{state}_{t,A}(\mathrm{root}(t)) \in F\}$. A tree language that is recognized by some finite tree automaton, is a *regular tree language*.

For a tree $t \in T_\Sigma$, and a vertex $v \in V_t$, the set of *successful states* of A at v, denoted $\mathrm{succ}_{t,A}(v)$, is defined recursively in a top-down fashion as follows: if v is the root of t, then $\mathrm{succ}_{t,A}(v) = F$, and if v has label σ of rank k and $1 \leq i \leq k$, then $\mathrm{succ}_{t,A}(v \cdot i)$ is the set of all states $q \in Q$ such that $\sigma_A(q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_k) \in \mathrm{succ}_{t,A}(v)$, where $q_j = \mathrm{state}_{t,A}(v \cdot j)$ for

$1 \leq j \leq k$, $j \neq i$. Intuitively, q is in $\text{succ}_{t,A}(v)$ if the automaton, when started at v in state q, arrives in a final state at the root of t. It is easy to see (by a top-down recursion) that for every vertex v of t, $t \in L(A)$ iff $\text{state}_{t,A}(v) \in \text{succ}_{t,A}(v)$.

A *regular tree grammar* over Σ is a context-free grammar $G = (N, \Sigma, P, S)$ (where N is the set of nonterminals, S the initial nonterminal, and P the set of productions) such that the right-hand side of every production is in $T_{\Sigma \cup N}$ (or more precisely, denotes a tree in $T_{\Sigma \cup N}$), assuming the nonterminals to have rank 0. It is easy to see that $L(G)$, the language generated by G, is a subset of T_Σ (or more precisely, denotes a subset of T_Σ). It is well known that a tree language is regular iff it is generated by a regular tree grammar.

2.3 Monadic Second Order Logic

For alphabets Σ and Γ, we define a monadic second-order logical language $\text{MSOL}(\Sigma, \Gamma)$, of which each closed formula expresses a property of the graphs in $GR_{\Sigma,\Gamma}$. The language has *node variables*, denoted $u, v, \ldots$, and *node-set variables*, denoted $U, V, \ldots$. For a given graph H, each node variable ranges over the elements of V_H and each node-set variable over the subsets of V_H. There are four types of atomic formulas in $\text{MSOL}(\Sigma, \Gamma)$: $\text{lab}_\sigma(u)$, for every $\sigma \in \Sigma$, $\text{edge}_\gamma(u, v)$, for every $\gamma \in \Gamma$, $u = v$, and $u \in U$. Their meaning should be clear: node u has label σ, there is an edge with label γ from node u to node v, nodes u and v are the same, and node u is an element of node-set U, respectively. The formulas of the language are constructed from the atomic formulas through the propositional connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, and the quantifiers $\forall$ and $\exists$, in the usual way. Note that not only node variables but also node-set variables may be quantified (which makes the logic monadic second-order rather than first-order). Note also that there are no edge or edge-set variables. As usual, a formula is closed if it has no free variables. For a closed formula ϕ of $\text{MSOL}(\Sigma, \Gamma)$ and a graph H of $GR_{\Sigma,\Gamma}$ we write $H \models \phi$ if ϕ is true for H. If formula ϕ has free variables, say u, v, and U (and no others), then we also write the formula as $\phi(u, v, U)$. If graph H has nodes $x, y \in V_H$ and a set of nodes $X \subseteq V_H$, then we write $H \models \phi(x, y, X)$ to mean that ϕ is true for H when the values x, y, and X are assigned to u, v, and U, respectively. In fact, with an abuse of language, we will usually give the same names u,v, and U to both the variables and the node(set)s x, y, and X. Thus, we write $H \models \phi(u, v, U)$ to mean that the formula ϕ is true for H, nodes $u, v \in V_H$, and node-set $U \subseteq V_H$. This should not lead to confusion.

By $\text{edge}(u, v)$ we denote the disjunction of all formulas $\text{edge}_\gamma(u, v)$, $\gamma \in \Gamma$. It is well known that there exists an MSOL formula $\text{path}(u, v)$ that expresses the existence of a (possibly empty) directed path from u to v (see, e.g., Lemma 1.2 of [Cou3]). In fact, if $\text{closed}(U)$ is the formula $\forall u, v : (\text{edge}(u, v) \wedge u \in U) \rightarrow v \in U$, then we can take $\text{path}(u, v)$ to be the formula $\forall U : (\text{closed}(U) \wedge u \in U) \rightarrow v \in U$. Thus, as a very simple example, $H \models \forall u \forall v : \text{path}(u, v)$ means that H is strongly connected. As another example, bipartiteness of a graph is expressed by the formula $\exists U \exists V : (\text{part}(U, V) \wedge \forall u \forall v : \text{edge}(u, v) \rightarrow (u \in U \wedge v \in V) \vee (u \in$

$V \wedge v \in U)$, where part(U, V) expresses that U and V form a partition of the set of all nodes: $\forall u : (u \in U \vee u \in V) \wedge \neg(u \in U \wedge u \in V)$.

Definition 1. A graph language $L \subseteq GR_{\Sigma,\Gamma}$ is *MSO definable* if there is a closed formula ϕ in MSOL(Σ, Γ) such that $L = \{H \in GR_{\Sigma,\Gamma} \mid H \models \phi\}$. □

Thus, as shown above, the set of strongly connected graphs and the set of bipartite graphs are MSO definable.

For string and tree languages we recall the classical result that MSO definability is the same as regularity, proved in [Buc, Elg] and [Don, TW], respectively. Note that strings and trees are special types of graphs, as explained in Section 2.1.

Proposition 2. *A string language is MSO definable if and only if it is regular. A tree language is MSO definable if and only if it is regular.*

For graph languages there is no generally accepted notion of regularity, but the MSO definable graph languages enjoy several properties that are similar to those of the regular tree and string languages (see, e.g., [Cou3, Eng4] and Theorem 16).

3 Regular Path Descriptions

A way of describing a set of "tree-like" graphs H is by taking a tree t from some regular tree language, defining the nodes of H as a subset of the vertices of t, and defining an edge between nodes u and v of H if the string of vertex labels on the shortest (undirected) path between u and v in t belongs to some regular string language. Note that the nodes of the tree are called 'vertices', in order not to confuse them with the nodes of the defined graph. Such a description of a graph language is called a *regular path description*. This idea was introduced in [Wel], and investigated in [ELW, EO]. It is shown in [EO] that the class of graph languages that can be described by a regular path description is equal to the class C-edNCE of graph languages generated by C-edNCE graph grammars, a particular type of context-free graph grammar. In [Wel, ELW] specific cases of this correspondence were established. In this correspondence, the trees from the regular tree language are related to the derivation trees of the context-free graph grammar.

First we define the string of labels on the shortest undirected path from one vertex u of a tree to another vertex v. That path ascends from u to the least common ancestor of u and v, and then descends to v. In the string this change of direction is indicated by barring the label of the least common ancestor. To suggest the special form of the path, we will denote the corresponding string of labels by bipath(u, v) (as opposed to [EO] where it is denoted path(u, v); here, path(u, v) is an MSOL formula that expresses the existence of a directed path from u to v, cf. Section 2.3).

Definition 3. Let Δ be a ranked alphabet, and let $\overline{\Delta} = \{\overline{\delta} \mid \delta \in \Delta\}$. For $t \in T_\Delta$ and $u, v \in V_t$, we define bipath$_t(u, v) \in \Delta^* \overline{\Delta} \Delta^*$ as follows. Let $z \in V_t$ be the least common ancestor of u and v in t. Let $u_1, \dots, u_m$ ($m \geq 1$) and $v_1, \dots, v_n$ ($n \geq 1$) be the vertices on the directed paths in t from z to u and from z to v, respectively (thus, $z = u_1 = v_1$, $u = u_m$, and $v = v_n$). Then

$$\mathrm{bipath}_t(u, v) = \lambda_t(u_m) \cdots \lambda_t(u_2)\overline{\lambda_t(z)}\lambda_t(v_2) \cdots \lambda_t(v_n).$$

□

Regular path descriptions are defined next.

Definition 4. A *regular path description* is a tuple $R = (\Delta, \Sigma, \Gamma, T, h, W)$, where Δ is a ranked alphabet, Σ and Γ are alphabets (of node and edge labels, respectively), $T \subseteq T_\Delta$ is a regular tree language, h is a partial function from Δ to Σ, and W is a mapping from Γ to the class of regular string languages, such that, for every $\gamma \in \Gamma$, $W(\gamma) \subseteq \Delta^*\overline{\Delta}\Delta^*$.

The *graph language described by* R is $L(R) = \{gr_R(t) \mid t \in T\}$, where $gr_R(t)$ is the graph $H \in GR_{\Sigma,\Gamma}$ such that
V_H is the set of vertices v of t with $\lambda_t(v) \in \mathrm{dom}(h)$,
$\lambda_H(v) = h(\lambda_t(v))$ for $v \in V_H$, and
E_H is the set of all edges (u, γ, v) with $\mathrm{bipath}_t(u, v) \in W(\gamma)$. □

Note that $L(R) \subseteq GR_{\Sigma,\Gamma}$. Note that h is used both to determine which vertices of the tree t are nodes of the graph $gr_R(t)$, and to define their labels in that graph (on the basis of their labels in the tree). Note that for each edge label γ, $W(\gamma)$ is the regular string language that defines the graph edges with label γ.

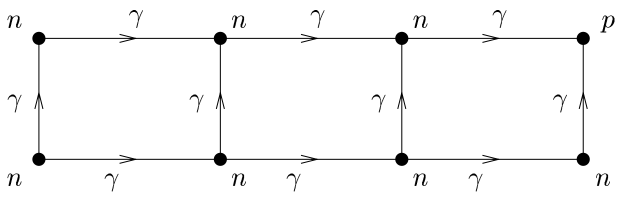


Fig. 1. A “ladder”.

Example 1. (1) A “ladder” is a graph as shown in Fig. 1. The graph language of all such ladders can be described by the regular path description $R_1 = (\Delta, \Sigma, \Gamma, T, h, W)$ with $\Delta = \{n, a, c\}$, $\mathrm{rank}(n) = \mathrm{rank}(a) = 1$, $\mathrm{rank}(c) = 0$, $\Sigma = \{n, p\}$, $\Gamma = \{\gamma\}$, $T = (an)^*ac$ (where we have denoted the trees by strings), h is the total function with $h(n) = h(a) = n$ and $h(c) = p$, and W is given by $W(\gamma) = \{n\overline{a}, c\overline{a}\} \cup \{\overline{n}an, \overline{n}ac, \overline{a}na\}$.

Fig. 2(a) shows the tree $t = anananac$ in T, and Fig. 2(b) shows the graph $gr_{R_1}(t)$, which is in fact the ladder of Fig. 1. In general, if $t = (an)^k ac$ then $gr_{R_1}(t)$ is the ladder with $k + 1$ steps, $k \geq 0$.

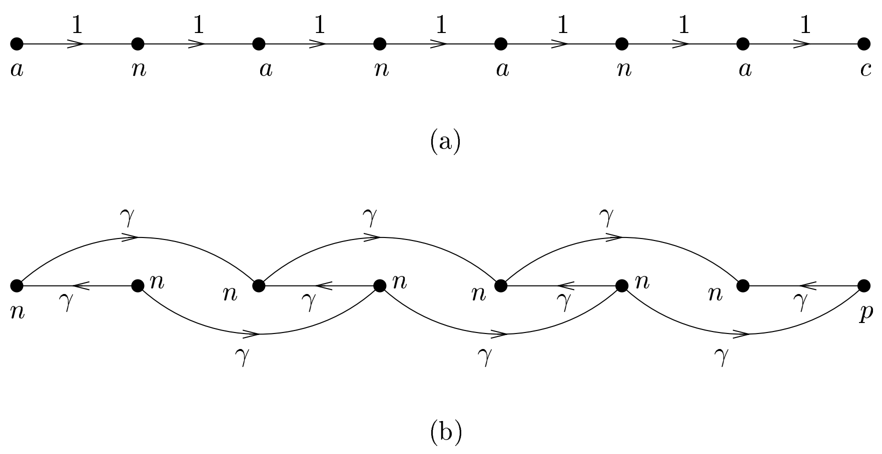


Fig. 2. Regular path description of a "ladder".

(2) As a second example we consider the graph language of all rooted binary trees with γ-labeled edges from each parent to its children, with additional α-labeled edges from each leaf to the root, and with additional β-labeled edges that chain the leaves of the tree. An example of such a graph is given in Fig. 3(b). This graph language is described by the regular path description $R_2 = (\Delta, \Sigma, \Gamma, T, h, W)$ with $\Delta = \{a, b_l, b_r, c_l, c_r\}$, $\text{rank}(a) = \text{rank}(b_l) = \text{rank}(b_r) = 2$, $\text{rank}(c_l) = \text{rank}(c_r) = 0$, $\Sigma = \{n\}$, $\Gamma = \{\gamma, \alpha, \beta\}$, $T = L(G)$, where G is the regular tree grammar with productions $S \to aLR$, $L \to b_lLR$, $R \to b_rLR$, $L \to c_l$, and $R \to c_r$ (with nonterminals S, L, R of which S is the initial one), h is the total function with $h(\delta) = n$ for all $\delta \in \Delta$, and W is defined by $W(\gamma) = (\overline{a} \cup \overline{b_l} \cup \overline{b_r})(b_l \cup b_r \cup c_l \cup c_r)$, $W(\alpha) = (c_l \cup c_r)(b_l \cup b_r)^*\overline{a}$, and $W(\beta) = (c_r b_r^* b_l \cup c_l)(\overline{a} \cup \overline{b_l} \cup \overline{b_r})(b_r b_l^* c_l \cup c_r)$.

Fig. 3(a) shows the tree $t = ab_lc_lb_rc_lc_rc_r$ in $L(G)$, and Fig. 3(b) shows the graph $gr_{R_2}(t)$.

Removing β from Γ (and $W(\beta)$ from W), a regular path description R_2' is obtained of the graph language of all binary trees with additional edges from the leaves to the root (i.e., the same graphs as in $L(R_2)$, but without the β-labeled edges).

(3) A *cograph* is an undirected, unlabeled graph, recursively defined as follows (see [CLS]). A graph consisting of one node is a cograph. If H and K are cographs, then so are $H + K$ and $H \times K$, where $H + K$ is the disjoint union of H and K, and $H \times K$ is obtained from $H + K$ by adding all edges between a node of H and a node of K. A *cotree* is a tree in T_Δ where $\Delta = \{+, \times, n\}$ with $\text{rank}(+) = \text{rank}(\times) = 2$ and $\text{rank}(n) = 0$. Clearly, every cotree is an expression that denotes a cograph (where n is a constant denoting the one-node graph). It is well known that the cograph H denoted by a cotree $t \in T_\Delta$, can be obtained as follows: the nodes of H are the leaves of t, and there is an edge between u and v in H if and only if the least common ancestor of u and v in t has label $\times$.

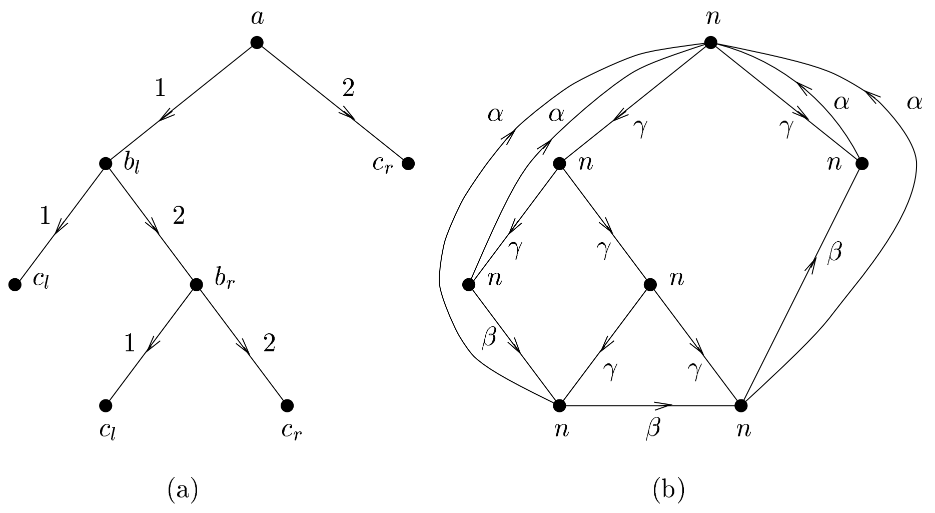


Fig. 3. Regular path description of a binary tree with additional edges.

From this it follows that the set of all cographs is described by the regular path description $R_3 = (\Delta, \Sigma, \Gamma, T, h, W)$ with $\Sigma = \{n\}$, $\Gamma = \{\gamma\}$, $T = T_\Delta$, $h(n) = n$, $h(+)$ and $h(\times)$ are undefined, and $W(\gamma) = n\{+, \times\}^*\overline{\times}\{+, \times\}^*n$.

Fig. 4(a) shows the tree $t = \times + nn + nn$, in T_Δ. The cograph $gr_{R_3}(t)$, which is the square, is shown in Fig. 4(b). □

Let RPD denote the class of graph languages that are described by regular path descriptions. We now define some natural subclasses X-RPD of RPD, by restricting the regular path descriptions to be of type X.

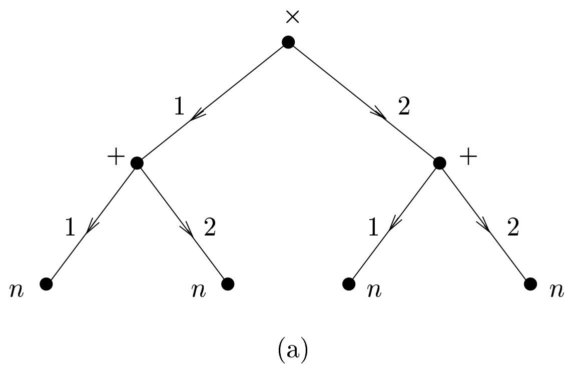
Let B-RPD be the subclass of RPD obtained by restricting every $W(\gamma)$ to be a subset of $\Delta^*\overline{\Delta} \cup \overline{\Delta}\Delta^*$. This means, for a regular path description of type B, that graph edges are only established between tree vertices of which one is a descendant of the other.

Let A-RPD be the subclass of RPD obtained by restricting every $W(\gamma)$ to be finite. Thus, for a regular path description of type A, graph edges can only be established between tree vertices that are at a bounded distance from each other. It is shown in [EO] that A-RPD $\subseteq$ B-RPD and that A-RPD is the class of RPD graph languages of bounded degree.

Let LIN-RPD be the subclass of RPD obtained by restricting the symbols of the ranked alphabet Δ to have rank 1 or 0. This means that the trees in the regular tree language are in fact strings (apart from the edge labels). Thus, intuitively, a regular path description of type LIN uses regular string languages only. Note that, obviously, LIN-RPD $\subseteq$ B-RPD.

In Example 1, R_1 is of type B, A, and LIN, R_2 and R_3 are *not* of type B, A, or LIN, and R_2' is of type B (but not of type A or LIN).

The main result of [EO] is that regular path descriptions have the same power



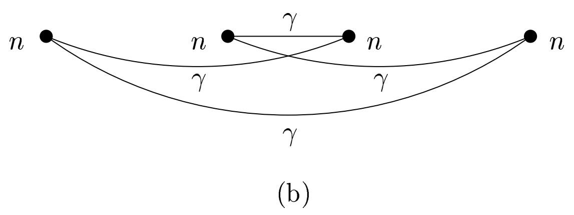


Fig. 4. Regular path description of a cograph.

as C-edNCE graph grammars, a powerful type of context-free graph grammars. Furthermore, the subclasses of RPD defined above correspond to (well-known) subclasses of the class C-edNCE of graph languages generated by C-edNCE grammars. For completeness sake we state this result.

Theorem 5. *C-edNCE = RPD, B-edNCE = B-RPD, A-edNCE = A-RPD, and LIN-edNCE = LIN-RPD.* □

Thus, in the remainder of this paper C-edNCE can be read for RPD, and similarly for the subclasses. Since Theorem 5 is effective, 'C-edNCE graph grammar' can be read for 'regular path description' in decidability results (cf. Section 6). The types B, A, and LIN were first introduced for graph grammars, where they stand for 'boundary' [RW1], 'apex' [EHL], and 'linear' [EL].

4 Monadic Second Order Definable Functions

The main concept in this paper is that of an MSO definable function f on graphs, introduced in [Eng1, Oos], and independently in [Cou4] (for a recent survey see [Cou6]). It is inspired by the notion of interpretability in [ALS], to which we refer for the history of that concept. The idea is that, for a given input graph H, the nodes, edges, and labels of the output graph $H' = f(H)$ are described in terms of MSOL formulas on H. For each node label σ of H' there is a formula $\phi_\sigma(u)$ expressing that u will be a node of H' with label σ. Thus, the nodes of H' are a subset of the nodes of H. For each edge label γ of H' there is a formula $\phi_\gamma(u,v)$ expressing that there will be a γ-labeled edge from u to v in H'. Finally, to allow for partial functions, there is a closed formula ϕ_{dom} that specifies the domain $\text{dom}(f)$ of f (which means that $\text{dom}(f)$ is an MSO definable set of graphs).

Definition 6. Let Σ_i and Γ_i be alphabets, for $i \in \{1,2\}$. An *MSO definable function* $f : GR_{\Sigma_1,\Gamma_1} \to GR_{\Sigma_2,\Gamma_2}$ is specified by formulas in $\text{MSOL}(\Sigma_1,\Gamma_1)$, as follows:

– a closed formula ϕ_{dom}, the *domain formula*,
– a formula $\phi_\sigma(u)$, for every $\sigma \in \Sigma_2$, the *node formulas*,
– a formula $\phi_\gamma(u,v)$, for every $\gamma \in \Gamma_2$, the *edge formulas*.

The domain of f is $\{H \in GR_{\Sigma_1,\Gamma_1} \mid H \models \phi_{\text{dom}}\}$, and for every $H \in \text{dom}(f)$, $f(H)$ is the graph $(V,E,\lambda) \in GR_{\Sigma_2,\Gamma_2}$ such that

– $V = \{u \in V_H \mid$ there is exactly one $\sigma \in \Sigma_2$ such that $H \models \phi_\sigma(u)\}$
– $E = \{(u,\gamma,v) \mid u,v \in V, u \neq v, \gamma \in \Gamma_2,$ and $H \models \phi_\gamma(u,v)\}$, and
– for $u \in V$, $\lambda(u) = \sigma$ where $H \models \phi_\sigma(u)$.

□

The class of MSO definable functions will be denoted MSOF.

Note that a node u of H may *not* be a node of $f(H)$ for two reasons: either there is no σ such that $H \models \phi_\sigma(u)$, or there are more than one such σ. However, it is easy to see that we may always assume the formulas $\phi_\sigma(u)$ to be mutually exclusive (replace $\phi_\sigma(u)$ by the conjunction of $\phi_\sigma(u)$ and all $\neg\phi_{\sigma'}(u)$ with $\sigma' \in \Sigma_2$, $\sigma' \neq \sigma$), in which case only the first reason remains.

Example 2. (0) Consider the function f that is defined for every acyclic graph in $GR_{\Sigma,\Gamma}$ and computes its transitive closure in $GR_{\Sigma,\Gamma\cup\{\tau\}}$, where τ is used to label the new edges. To show that f is MSO definable, we take ϕ_{dom} to be $\neg(\exists u \exists v : \text{edge}(u,v) \wedge \text{path}(v,u))$, expressing that the input graph H is acyclic. For every $\sigma \in \Sigma$, $\phi_\sigma(u)$ is $\text{lab}_\sigma(u)$, i.e., every node of H is a node of $f(H)$ and has the same label. Finally, $\phi_\gamma(u,v)$ is $\text{edge}_\gamma(u,v)$, for every $\gamma \in \Gamma$, and $\phi_\tau(u,v)$ is $\text{path}(u,v)$. Thus, the edges of H remain in $f(H)$, but $f(H)$ also contains all edges that correspond to paths in H.

(1) Next we consider an MSO definable function f_1 that translates strings into "ladders" (cf. Example 1(1)). Let $\Delta_1 = \{a,n\}$. In Fig. 5 it is shown how f_1 translates the string *anananan* into the ladder of Fig. 1. In general f_1 translates the string $(an)^k an$ into a ladder with $k+1$ steps, $k \geq 0$. Thus, $f_1 : GR_{\Delta_1,\{*\}} \to GR_{\{n,p\},\{\gamma\}}$. The formula ϕ_{dom} expresses that the input string belongs to the regular language $(an)^* an$:

$$\begin{aligned} &\forall u,v : \text{edge}(u,v) \to (\text{lab}_a(u) \wedge \text{lab}_n(v)) \vee (\text{lab}_n(u) \wedge \text{lab}_a(v))) \\ \wedge\ &\forall u : (\text{source}(u) \to \text{lab}_a(u)) \wedge (\text{target}(u) \to \text{lab}_n(u)), \end{aligned}$$

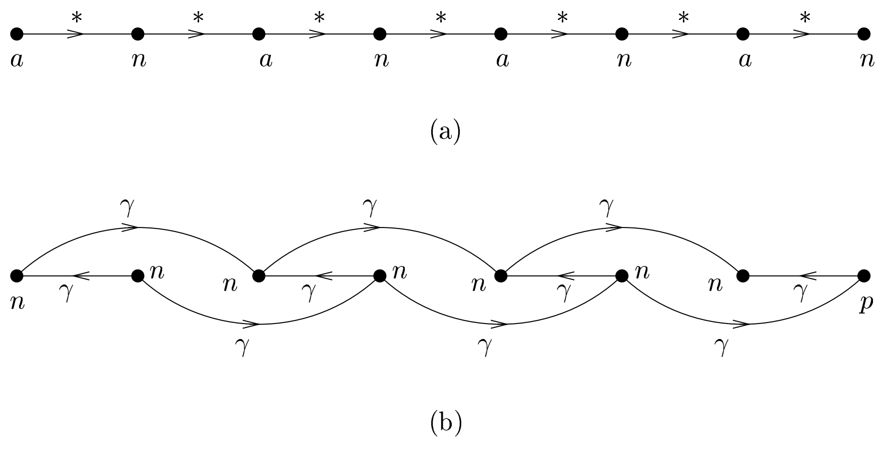


Fig. 5. MSOF description of a "ladder".

where source(u) is $\neg\exists v : \text{edge}(v,u)$ and target(u) is $\neg\exists v : \text{edge}(u,v)$. The node formula $\phi_p(u)$ is target(u) and the node formula $\phi_n(u)$ is $\neg$target(u). Finally, the edge formula $\phi_\gamma(u,v)$ is $(\text{lab}_n(u) \wedge \text{edge}(v,u)) \vee (\exists w : \text{edge}(u,w) \wedge \text{edge}(w,v))$. Note that dom$(f_1)$ also contains graphs that are not strings. If we wish dom(f_1) to be a subset of Δ_1^*, we have to take the conjunction of ϕ_{dom} with a formula expressing that the input graph is a string, e.g., the conjunction of $\exists u : \text{source}(u) \wedge \forall v : \text{path}(u,v)$ with a formula that requires all nodes to be of in- and out-degree at most one: $\forall u,v,w : ((\text{edge}(v,u) \wedge \text{edge}(w,u)) \vee (\text{edge}(u,v) \wedge \text{edge}(u,w))) \rightarrow v = w$.

(2) Let us now consider an MSO definable function f_2 that translates trees into the binary trees with additional edges of Example 1(2). Fig. 6 shows how f_2 translates a tree into such a graph. Let Δ_2 be the ranked alphabet $\{a,c\}$ with rank$(a) = 2$ and rank$(c) = 0$. Then $f_2 : GR_{\Delta_2,\text{rks}(\Delta_2)} \rightarrow GR_{\{n\},\{\gamma,\alpha,\beta\}}$. The domain formula ϕ_{dom} and the node formula $\phi_n(u)$ of f_2 are *true*. The edge formulas of f_2 are as follows: $\phi_\gamma(u,v)$ is edge(u,v), $\phi_\alpha(u,v)$ is target$(u) \wedge$source(v), where 'target' and 'source' are defined in the previous example, and finally $\phi_\beta(u,v)$ is

$$\begin{aligned} &\text{target}(u) \wedge \text{target}(v) \\ \wedge\ &\exists z, z_1, z_2 : \text{edge}_1(z,z_1) \wedge \text{edge}_2(z,z_2) \wedge \text{path}_2(z_1,u) \wedge \text{path}_1(z_2,v), \end{aligned}$$

where path$_i(x,y)$ is a formula which expresses that all the edges on the directed path from x to y have label i (and which can easily be defined analogously to the formula path(x,y)). As in the previous example, if we wish dom(f_2) to consist of trees in T_{Δ_2} only, we should take ϕ_{dom} to be a formula that expresses that the input graph is a tree over Δ_2 (which can easily be found).

If $W(\beta)$ is dropped from W, an MSO definable function f_2' is obtained that translates trees into the same graphs as f_2, but without the β-labeled edges.

(3) As a last example we show that the function f_3 that maps a cotree into the cograph it denotes is MSO definable, see Example 1(3) and Fig. 4. Thus,

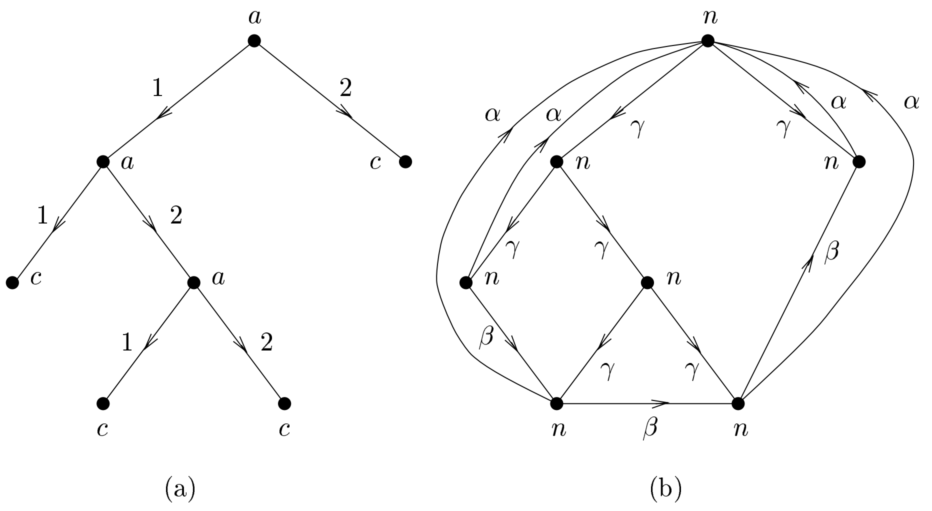


Fig. 6. MSOF description of a binary tree with additional edges.

$f_3 : GR_{\Delta_3,\mathrm{rks}(\Delta_3)} \to GR_{\{n\},\{\gamma\}}$, where $\Delta_3 = \{+, \times, n\}$. The domain formula is *true*. The node formula $\phi_n(u)$ is $\mathrm{lab}_n(u)$; note that this implies that all tree vertices with label $+$ or $\times$ are dropped. The edge formula $\phi_\gamma(u,v)$ of f_3 is $\forall z : \mathrm{lca}(z,u,v) \to \mathrm{lab}_\times(z)$ where $\mathrm{lca}(z,x,y)$ expresses that z is the least common ancestor of u and v:

$$\mathrm{path}(z,u) \wedge \mathrm{path}(z,v) \wedge \forall w : (\mathrm{path}(w,u) \wedge \mathrm{path}(w,v)) \to \mathrm{path}(w,z).$$

□

The main idea of this paper is to use monadic second-order logic for the description of sets of "tree-like" graphs. This is realized by applying MSO definable functions to trees, as in Example 2(2) and (3). Similarly, sets of "string-like" graphs are obtained by applying MSO definable functions to strings, as in Example 2(1).

By MSOF(TREES) we denote the class of all graph languages $f(T_\Delta)$ where Δ is a ranked alphabet and f is an MSO definable function from $GR_{\Delta,\mathrm{rks}(\Delta)}$ to some $GR_{\Sigma,\Gamma}$. Note that, by this definition, $\mathrm{dom}(f)$ need not be a subset of T_Δ; thus, the domain formula of f need not require the input graph to be a tree, cf. Example 2(2). We will show in the next section that RPD = MSOF(TREES). Similar characterizations will also be given for the subclasses of RPD of type B, A, and LIN. To obtain characterizations of B-RPD and A-RPD, we will use the following terminology. A formula $\phi(u,v)$ in $\mathrm{MSOL}(\Delta, \mathrm{rks}(\Delta))$ is of type B if, for every tree $t \in T_\Delta$, $t \models \forall u,v : \phi_\gamma(u,v) \to (\mathrm{path}(u,v) \vee \mathrm{path}(v,u))$. And $\phi(u,v)$ is of type A if there is a number $k \in \mathbb{N}$ such that for every tree $t \in T_\Delta$, $t \models \forall u,v : \phi_\gamma(u,v) \to \mathrm{dist}_k(u,v)$ where $\mathrm{dist}_k(u,v)$ is a formula expressing that the (undirected) distance between u and v is at most k, i.e., that there is an undirected path from u to v of length $< k$. We say that (the specification of) an MSO definable function f is of type B or A, if all its edge formulas are of type B or A, respectively. We now define B-MSOF(TREES) to be the class of all $f(T_\Delta)$ as above, where f is of type B. As in the case of B-RPD, this means that graph edges are only established between tree vertices of which one is a descendant of the other. Similarly, A-MSOF(TREES) is the class of all $f(T_\Delta)$ with f of type A. As for A-RPD, this means that graph edges are only established between tree vertices that are at a bounded distance from each other. For LIN it will be shown that LIN-RPD = MSOF(STRINGS), the class of all graph languages $f(\Delta^*)$ where Δ^* is the set of all strings over some alphabet Δ and f is an MSO definable function from $GR_{\Delta,\{*\}}$ to some $GR_{\Sigma,\Gamma}$. In fact, this will easily follow from the fact that LIN-RPD = MSOF(LIN-TREES), which is defined in the same way as MSOF(TREES) with the restriction that the symbols of Δ all have rank 1 or 0.

In Example 2, f_1 is of type B and A, f_2 and f_3 are *not* of type B or A, and f_2' is of type B (but not of type A). Note that $f_1(\Delta_1^*)$ is in MSOF(STRINGS). Note also that $f_1(\Delta_1^*) = L(R_1)$, $f_2(T_{\Delta_2}) = L(R_2)$, $f_2'(T_{\Delta_2}) = L(R_2')$, and $f_3(T_{\Delta_3}) = L(R_3)$, where the R_i are the regular path descriptions of Example 1(i). Thus $L(R_2)$, $L(R_2')$, and $L(R_3)$ are in MSOF(TREES). As another example we mention that all context-free (string) languages are in MSOF(TREES): for a given context-free grammar G, the function that maps each derivation tree of G into its yield is MSO definable (cf. the β-labeled edges of Example 2(2), and note that the set of derivation trees of G is MSO definable).

A useful property of the MSO definable functions on graphs is that they are closed under composition. This follows from the fact that MSO properties of the output graph can be translated into MSO properties of the input graph, as expressed in the following basic lemma.

Lemma 7. *Let $f : GR_{\Sigma_1,\Gamma_1} \to GR_{\Sigma_2,\Gamma_2}$ be an MSO definable function. For every formula ψ in MSOL(Σ_2,Γ_2) there is a formula $f^{-1}(\psi)$ in MSOL(Σ_1,Γ_1) such that for every graph $H \in \mathrm{dom}(f)$ (and every assignment of nodes and node-sets of $f(H)$ to the free variables of ψ), $f(H) \models \psi \Leftrightarrow H \models f^{-1}(\psi)$.*

Proof. Let f be specified by domain formula ϕ_{dom}, node formulas $\phi_\sigma(u)$, and edge formulas $\phi_\gamma(u,v)$. Let node(u) be the formula in MSOL(Σ_1,Γ_1) that expresses that u will be used as a node of the output graph, i.e., node(u) is the disjunction of all formulas node$_\sigma(u)$, $\sigma \in \Sigma_2$, where node$_\sigma(u)$ is the conjunction of $\phi_\sigma(u)$ and all $\neg\phi_{\sigma'}(u)$ with $\sigma' \in \Sigma_2$, $\sigma' \neq \sigma$.

The formula $f^{-1}(\psi)$ is obtained from the formula ψ by making the following changes:

- Relativize all quantifiers to the formula node(u), i.e., change every subformula $\exists x : \chi$ into $\exists x : \mathrm{node}(x) \wedge \chi$, and every subformula $\exists X : \chi$ into $\exists X : (\forall x : x \in X \to \mathrm{node}(x)) \wedge \chi$, and similarly for the universal quantifiers.
- Change every subformula lab$_\sigma(x)$ into $\phi_\sigma(x)$, and change every subformula edge$_\gamma(x,y)$ into $\phi_\gamma(x,y)$.

It should be clear that $f^{-1}(\psi)$ satisfies the requirement. □

Note that for closed formulas ψ this lemma means that the MSO definable graph languages are closed under inverse MSO definable functions.

Theorem 8. *MSOF is closed under composition.*

Proof. Let $f : GR_{\Sigma_1,\Gamma_1} \to GR_{\Sigma_2,\Gamma_2}$ and $g : GR_{\Sigma_2,\Gamma_2} \to GR_{\Sigma_3,\Gamma_3}$ be MSO definable functions. The formulas by which f is specified will be indicated by ϕ, and those of g by ψ. The composition $h = g \circ f$ is now specified by formulas χ in MSOL(Σ_1, Γ_1) as follows: χ_{dom} is $\phi_{\text{dom}} \wedge f^{-1}(\psi_{\text{dom}})$; for every $\sigma \in \Sigma_3$, $\chi_\sigma(u)$ is $\text{node}(u) \wedge f^{-1}(\psi_\sigma(u))$, where node$(u)$ is defined as in the proof of Lemma 7; and for every $\gamma \in \Gamma_3$, $\chi_\gamma(u,v)$ is $f^{-1}(\psi_\gamma(u,v))$. □

Lemma 7 and Theorem 8 were proved independently in Proposition 2.5 and Corollary 2.6 of [Cou4] (see also Lemma 4.4 of [ALS]).

5 Characterization

In this section we prove the main result, which we first state.

Theorem 9. *RPD = MSOF(TREES), B-RPD = B-MSOF(TREES), A-RPD = A-MSOF(TREES), and LIN-RPD = MSOF(STRINGS).*

We now turn to the proof. As observed in Section 4, in the LIN case we will first prove that LIN-RPD = MSOF(LIN-TREES).

To prove that RPD $\subseteq$ MSOF(TREES) (which is the easiest part), we use the following lemma. It expresses that, for given $W(\gamma)$, the relation $\text{bipath}_t(u,v) \in W(\gamma)$ between vertices u and v of t (as in Definition 4) can be expressed by an MSOL formula.

Lemma 10. *Let Δ be a ranked alphabet. For every regular language $W \subseteq \Delta^*\overline{\Delta}\Delta^*$ there is a formula $\phi(u,v)$ in MSOL$(\Delta, \text{rks}(\Delta))$ such that for every tree $t \in T_\Delta$ and all vertices $u, v \in V_t$, $t \models \phi(u,v) \Leftrightarrow \text{bipath}_t(u,v) \in W$.*

Proof. To construct ϕ we need the following auxiliary MSOL formulas, to be interpreted for trees in T_Δ. First, lca(z,u,v) is the formula which expresses that z is the least common ancestor of u and v, see the end of Example 2. Second, pathset(U,u,v) is the formula that expresses that U is the set of vertices on the directed path from u to v (if that exists): $\forall z : z \in U \leftrightarrow \text{path}(u,z) \wedge \text{path}(z,v)$.

Now let ψ be the closed formula in MSOL$(\Delta \cup \overline{\Delta}, \{*\})$ that defines the regular language W according to Proposition 2, i.e., $W = \{w \in (\Delta \cup \overline{\Delta})^* \mid w \models \psi\}$. Then we wish $\phi(u,v)$ to express that the formula ψ holds for the string $\text{bipath}_t(u,v)$. Thus, we define $\phi(u,v)$ to be the formula $\forall z, U, V : (\text{lca}(z,u,v) \wedge \text{pathset}(U,z,u) \wedge \text{pathset}(V,z,v)) \to \psi'$, where the formula ψ' is obtained from the formula ψ by the following changes:

– The quantifiers are relativized to the set $U \cup V$, i.e., every subformula $\exists x : \chi$ is changed into $\exists x : (x \in U \vee x \in V) \wedge \chi$, and every subformula $\exists X : \chi$ is changed into $\exists X : (\forall x : x \in X \to (x \in U \vee x \in V)) \wedge \chi$, and similarly for the universal quantifiers.
– For $\delta \in \Delta$, every subformula $\mathrm{lab}_\delta(x)$ is changed into $\mathrm{lab}_\delta(x) \wedge x \neq z$, and every subformula $\mathrm{lab}_{\bar\delta}(x)$ is changed into $\mathrm{lab}_\delta(x) \wedge x = z$.
– Every subformula $\mathrm{edge}_*(x,y)$ is changed into the formula $(x \in U \wedge y \in U \wedge \mathrm{edge}(y,x)) \vee (x \in V \wedge y \in V \wedge \mathrm{edge}(x,y))$; recall that $\mathrm{edge}(x,y)$ is the disjunction of all $\mathrm{edge}_i(x,y)$, $i \in \mathrm{rks}(\Delta)$.

The correctness of the formula $\phi(u,v)$ should be clear. Note that $U \cup V$ is the set of all vertices on the shortest undirected path from u to v, and that $U \cap V = \{z\}$.

This proves the lemma. For the interested reader we observe that this proof may be viewed as a variant of the proof of Lemma 7, as follows. Suppose that the notion of MSO definability is extended to functions of type $f : \{(H,x,y) \mid H \in GR_{\Sigma_1,\Gamma_1}, x,y \in V_H\} \to GR_{\Sigma_2,\Gamma_2}$ by requiring that all formulas in the specification have two additional free node variables x and y (and generalizing Definition 6 in the obvious way). Then it is not difficult to prove that the function $\mathrm{bip}(t,x,y) = \mathrm{bipath}_t(x,y)$ is MSO definable in this sense, and that (with an appropriate generalization of Lemma 7) $\phi(u,v)$ can be defined as $\mathrm{bip}^{-1}(\psi)$. $\square$

Lemma 11. *RPD $\subseteq$ MSOF(TREES), B-RPD $\subseteq$ B-MSOF(TREES), A-RPD $\subseteq$ A-MSOF(TREES), and LIN-RPD $\subseteq$ MSOF(LIN-TREES).*

Proof. Let $R = (\Delta, \Sigma, \Gamma, T, h, W)$ be a regular path description. We specify an MSO definable function $f : GR_{\Delta,\mathrm{rks}(\Delta)} \to GR_{\Sigma,\Gamma}$ such that $f(T_\Delta) = L(R)$. The domain formula ϕ_{dom} of f is the closed formula that defines the regular tree language T according to Proposition 2, i.e., for all $t \in T_\Delta$, $t \models \phi_{\mathrm{dom}} \Leftrightarrow t \in T$. For every $\sigma \in \Sigma$, the node formula $\phi_\sigma(u)$ is the disjunction of all formulas $\mathrm{lab}_\delta(u)$ with $h(\delta) = \sigma$. Finally, for every $\gamma \in \Gamma$, the edge formula $\phi_\gamma(u,v)$ is the formula that corresponds to the regular language $W(\gamma)$ according to Lemma 10. Clearly, f is defined in such a way that it simulates precisely the regular path definition R. Thus, $\mathrm{dom}(f) = T$ and, for every $t \in T$, $f(t) = gr_R(t)$. Hence $f(T_\Delta) = L(R)$. This shows that RPD $\subseteq$ MSOF(TREES) and that LIN-RPD $\subseteq$ MSOF(LIN-TREES).

If R is of type B, then we know that edges are only established between descendants. Hence we can replace the edge formulas $\phi_\gamma(u,v)$ by $\phi_\gamma(u,v) \wedge (\mathrm{path}(u,v) \vee \mathrm{path}(v,u))$, without changing f. Now let R be of type A, and let k be the maximal length of the strings in the $W(\gamma)$, $\gamma \in \Gamma$. Then we know that edges are established only between vertices that are at a distance $< k$ from each other. Hence we can replace $\phi_\gamma(u,v)$ by $\phi_\gamma(u,v) \wedge \mathrm{dist}_k(u,v)$, where $\mathrm{dist}_k(u,v)$ is the formula $\exists x_1, \ldots, x_k : x_1 = u \wedge x_k = v \wedge \psi(x_1, \ldots, x_k)$, and $\psi(x_1, \ldots, x_k)$ is the conjunction of all formulas $\mathrm{edge}(x_i, x_{i+1}) \vee \mathrm{edge}(x_{i+1}, x_i)$ for $1 \leq i < k$. $\square$

The essential part of the proof of the more involved inclusion MSOF(TREES) $\subseteq$ RPD is given in the following key lemma. It says that every MSO definable relation between vertices u and v of a tree s can also be expressed in the form

$\mathrm{bipath}_t(u,v) \in W$ for some regular language W, where t is a change of the node labels of s that should belong to a certain regular tree language.

If Δ and Σ are ranked alphabets, then a *projection* is a total function $\pi : \Delta \to \Sigma$ that is rank preserving, i.e., $\mathrm{rank}(\pi(\delta)) = \mathrm{rank}(\delta)$ for all $\delta \in \Delta$. For a tree $t \in T_\Delta$, $\pi(t)$ denotes the tree in T_Σ that is obtained from t by changing every node label δ into $\pi(\delta)$. It is easy to see that, as a node relabeling, π is MSO definable. Thus, by Lemma 7, for every MSOL formula ϕ there is an MSOL formula $\pi^{-1}(\phi)$ such that $\pi(t) \models \phi \Leftrightarrow t \models \pi^{-1}(\phi)$. In fact, $\pi^{-1}(\phi)$ is the formula ϕ in which every subformula $\mathrm{lab}_\sigma(x)$ is replaced by the disjunction of all $\mathrm{lab}_\delta(x)$ with $\pi(\delta) = \sigma$.

Lemma 12. *Let Σ be a ranked alphabet. For every formula $\phi(u,v)$ in MSOL(Σ, $\mathrm{rks}(\Sigma)$) there are a ranked alphabet Δ, a regular tree language $T \subseteq T_\Delta$, a projection $\pi : \Delta \to \Sigma$, and a regular language $W \subseteq \Delta^*\overline{\Delta}\Delta^*$ such that $\pi(T) = T_\Sigma$ and, for all trees $t \in T$ and vertices $u, v \in V_t$, $\pi(t) \models \phi(u,v) \Leftrightarrow \mathrm{bipath}_t(u,v) \in W$. Moreover, if $\phi(u,v)$ is of type B then $W \subseteq \Delta^*\overline{\Delta} \cup \overline{\Delta}\Delta^*$, and if $\phi(u,v)$ is of type A then W is finite.*

Proof. As in Lemma 10, the proof will be based on Proposition 2. First we construct, in a well-known way, a tree automaton A that "recognizes" the formula $\phi(u,v)$. Let Σ_2 be the ranked alphabet $\Sigma \cup (\Sigma \times \{1,2,12\})$ where the elements of Σ keep their ranks and every $\langle\sigma,i\rangle$ has the same rank as σ. For a tree $s \in T_\Sigma$ and vertices $u, v \in V_s$, the tree $\mathrm{mark}(s,u,v)$ in T_{Σ_2} is obtained from s by "marking" u and v, i.e., adding 1 to the label of u and adding 2 to the label of v (where it is understood that 12 is added in the case that $u = v$). Let ψ be the closed formula $\forall u, v : (\mathrm{mark}_1(u) \wedge \mathrm{mark}_2(v)) \to \phi'(u,v)$, where $\mathrm{mark}_i(x)$ is the disjunction of all $\mathrm{lab}_{\langle\sigma,i\rangle}(x) \vee \mathrm{lab}_{\langle\sigma,12\rangle}(x)$, $\sigma \in \Sigma$, and $\phi'(u,v)$ is obtained from $\phi(u,v)$ by changing every subformula $\mathrm{lab}_\sigma(x)$ into $\mathrm{lab}_\sigma(x) \vee \mathrm{lab}_{\langle\sigma,1\rangle}(x) \vee \mathrm{lab}_{\langle\sigma,2\rangle}(x) \vee \mathrm{lab}_{\langle\sigma,12\rangle}(x)$. Obviously, for $s \in T_\Sigma$ and $u, v \in V_s$, $s \models \phi(u,v) \Leftrightarrow \mathrm{mark}(s,u,v) \models \psi$. Now let $A = (Q, \{\tau_A\}_{\tau \in \Sigma_2}, F)$ be a tree automaton that recognizes the regular tree language corresponding to the closed formula ψ according to Proposition 2. Then $s \models \phi(u,v) \Leftrightarrow \mathrm{mark}(s,u,v) \in L(A)$.

The idea is to incorporate the state behaviour of the tree automaton A into the labels of a tree $s \in T_\Sigma$. Since the vertices of s are not "marked", this can only be the behaviour of A as far as it does not encounter any marks 1, 2, or 12. For given vertices u and v of (the relabeled) s, this information tells us the behaviour of A outside the shortest (undirected) path from u to v. Thus, to find out whether $\mathrm{mark}(s,u,v)$ is recognized by A, it then suffices to simulate the behaviour of A on that path. Since A behaves on paths as a finite automaton, this gives us the regular language W. Then $\mathrm{mark}(s,u,v) \in L(A)$ iff $\mathrm{bipath}_t(u,v) \in W$, where t is the relabeling of s. Note that, to incorporate the behaviour of A into the labels of s we need a new ranked alphabet Δ and, to check that behaviour, a regular tree language $T \subseteq T_\Delta$; then, t should be in T.

The formal construction is as follows. The alphabet Δ consists of all tuples (σ, g, G, j) such that $\sigma \in \Sigma$, g is a mapping $[1,k] \to Q$, where k is the rank of σ, $G \subseteq Q$, and $j = 0$ or $j \in \mathrm{rks}(\Sigma)$. The rank of (σ, g, G, j) is the one of σ. The projection π is defined by $\pi(\sigma, g, G, j) = \sigma$. The tree language T consists of all trees $t \in T_\Delta$ such that for every vertex $v \in V_t$, if v has label (σ, g, G, j) in t and σ has rank k, then $g(i) = \mathrm{state}_{\pi(t),A}(v \cdot i)$ for every $i \in [1, k]$, $G = \mathrm{succ}_{\pi(t),A}(v)$, and j is the child number of v. For the terminology used, see Section 2.2. Thus, g contains the states reached by A at the children of v and G is the set of successful states of A at v, both with respect to the tree $\pi(t)$ which is called s in the intuitive discussions above. The need for child number j can be seen by comparing Examples 2(2) and 1(2), where the childnumbers 1 and 2 are the subscripts l and r of b and c. Using the recursive definitions of 'state' and 'succ' it is straightforward to show that T is a regular tree language. In fact, it suffices to check for each vertex and its children, whether their labels "fit" (i.e., T is even a "local" tree language). The details are left to the reader. It should also be clear that for every tree $s \in T_\Sigma$ there is a tree $t \in T_\Delta$ (in fact a unique one) with $\pi(t) = s$, and so $\pi(T) = T_\Sigma$. It remains to define W. The language W consists of all strings

$$(\sigma_1, g_1, G_1, j_1) \cdots (\sigma_n, g_n, G_n, j_n)\overline{(\sigma, g, G, j)}(\sigma'_{n'}, g'_{n'}, G'_{n'}, j'_{n'}) \cdots (\sigma'_1, g'_1, G'_1, j'_1),$$

$n, n' \geq 0$, such that there exist states $q_1, \ldots, q_n, q, q'_{n'}, \ldots, q'_1$ in Q with the following six properties. Intuitively, $q_1, \ldots, q_n, q$ are the states reached by A (in the tree $\mathrm{mark}(\pi(t), u, v)$) on the path from u to z and $q'_1, \ldots, q'_{n'}, q$ are the states reached by A on the path from v to z, where z is the least common ancestor of u and v. By k, k_i, and k'_i we denote the rank of σ, σ_i and σ'_i, respectively.

1. If $n \geq 1$, then $\langle\sigma_1, 1\rangle_A(g_1(1), \ldots, g_1(k_1)) = q_1$.
   And similarly with primes and with mark 2 instead of mark 1:
   if $n' \geq 1$, then $\langle\sigma'_1, 2\rangle_A(g'_1(1), \ldots, g'_1(k'_1)) = q'_1$.
2. For $2 \leq i \leq n$, $\sigma_{iA}(g_i(1), \ldots, g_i(m-1), q_{i-1}, g_i(m+1), \ldots, g_i(k_i)) = q_i$, where $m = j_{i-1}$. And similarly with primes.
3. If $n, n' \geq 1$, then $j_n \neq j'_{n'}$ and, depending on whether j_n is smaller or larger than $j'_{n'}$, either
   $\sigma_A(g(1), \ldots, g(m-1), q_n, g(m+1), \ldots, g(m'-1), q'_{n'}, g(m'+1), \ldots, g(k)) = q$,
   or
   $\sigma_A(g(1), \ldots, g(m'-1), q'_{n'}, g(m'+1), \ldots, g(m-1), q_n, g(m+1), \ldots, g(k)) = q$,
   where $m = j_n$ and $m' = j'_{n'}$.
4. If $n = 0$ and $n' \geq 1$, then
   $\langle\sigma, 1\rangle_A(g(1), \ldots, g(m'-1), q'_{n'}, g(m'+1), \ldots, g(k)) = q$, where $m' = j'_{n'}$.
   And similarly, if $n \geq 1$ and $n' = 0$, then
   $\langle\sigma, 2\rangle_A(g(1), \ldots, g(m-1), q_n, g(m+1), \ldots, g(k)) = q$, where $m = j_n$.
5. If $n = 0$ and $n' = 0$, then $\langle\sigma, 12\rangle_A(g(1), \ldots, g(k)) = q$.
6. Finally, $q \in G$.

This ends the definition of W. It is straightforward to show that W is regular. A finite automaton recognizing W should simulate the state behaviour of A on the unprimed part of the string, and (nondeterministically) simulate A backwards on the primed part of the string, checking properties (1) and (2). At the barred symbol it should simulate A for that symbol by property (3), (4), or (5), and it should check property (6). The details are left to the reader. If $\phi(u,v)$ is of type B, then we additionally require that $n = 0$ or $n' = 0$ in the above definition of W (and hence property (3) can be omitted), and if $\phi(u,v)$ is of type A for some upper bound k on the distance, then we restrict W to contain strings of length $\leq k$ only.

From the construction it should be clear that for every tree $t \in T$ and all vertices $u, v \in V_t$, $\mathrm{mark}(\pi(t),u,v) \in L(A)$ iff $\mathrm{bipath}_t(u,v) \in W$. Note that the condition $j_n \neq j'_{n'}$ in property (3) guarantees that the barred symbol labels the least common ancestor z of u and v. Note also that property (6) expresses that $\mathrm{state}_{\mathrm{mark}(\pi(t),u,v),A}(z) \in \mathrm{succ}_{\mathrm{mark}(\pi(t),u,v),A}(z)$, i.e., that $\mathrm{mark}(\pi(t),u,v) \in L(A)$. Hence $\pi(t) \models \phi(u,v) \Leftrightarrow \mathrm{mark}(\pi(t),u,v) \in L(A) \Leftrightarrow \mathrm{bipath}_t(u,v) \in W$.

□

Lemma 13. *MSOF(TREES) $\subseteq$ RPD, B-MSOF(TREES) $\subseteq$ B-RPD, A-MSOF(TREES) $\subseteq$ A-RPD, and MSOF(LIN-TREES) $\subseteq$ LIN-RPD.*

Proof. Let f be an MSO definable function $GR_{\Delta,\mathrm{rks}(\Delta)} \to GR_{\Sigma,\Gamma}$, where Δ is a ranked alphabet, specified by domain formula ϕ_{dom}, node formulas $\phi_\sigma(u)$, and edge formulas $\phi_\gamma(u,v)$. We will define a regular path description $R = (\Delta', \Sigma, \Gamma, T, h, W)$ such that $L(R) = f(T_\Delta)$. For every $\gamma \in \Gamma$, let Δ_γ, $T_\gamma \subseteq T_{\Delta_\gamma}$, $\pi_\gamma : \Delta_\gamma \to \Delta$, and $W_\gamma \subseteq \Delta_\gamma^* \overline{\Delta_\gamma} \Delta_\gamma^*$ be as given by Lemma 12 for the formula $\phi_\gamma(u,v)$. Thus $\pi_\gamma(T_\gamma) = T_\Delta$ and, for all $t \in T_\gamma$ and $u, v \in V_t$, $\pi_\gamma(t) \models \phi_\gamma(u,v) \Leftrightarrow \mathrm{bipath}_t(u,v) \in W_\gamma$. The idea of the construction of R is to add the information to the labels of the trees in T_Δ that allows the "recognition" of the edge formula $\phi_\gamma(u,v)$ (as explained in Lemma 12), for all $\gamma \in \Gamma$ simultaneously, and, moreover, add information that indicates for each vertex u whether or not the node formula $\phi_\sigma(u)$ is satisfied, for all $\sigma \in \Sigma$.

We define Δ' to consist of all symbols $(\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$ where $\delta \in \Delta$, g_{e} is a mapping $\Gamma \to \bigcup_{\gamma\in\Gamma} \Delta_\gamma$ such that $g_{\mathrm{e}}(\gamma) \in \Delta_\gamma$ and $\pi_\gamma(g_{\mathrm{e}}(\gamma)) = \delta$ for every $\gamma \in \Gamma$, and g_{n} is a mapping $\Sigma \to \{0,1\}$. The rank of $(\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$ is the one of δ. The domain of the function h consists of all $(\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$ such that there is exactly one $\sigma \in \Sigma$ with $g_{\mathrm{n}}(\sigma) = 1$, and then $h(\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$ equals that unique σ.

To describe T we need the following projections (the notion of a projection is defined just before Lemma 12): the projection $\pi : \Delta' \to \Delta$ with $\pi(\delta, g_{\mathrm{e}}, g_{\mathrm{n}}) = \delta$, and for every $\gamma \in \Gamma$ the projection $\rho_\gamma : \Delta' \to \Delta_\gamma$ with $\rho_\gamma(\delta, g_{\mathrm{e}}, g_{\mathrm{n}}) = g_{\mathrm{e}}(\gamma)$. Note that, by the definition of Δ', $\pi_\gamma \circ \rho_\gamma = \pi$, for every $\gamma \in \Gamma$. The tree language T is defined to consist of all trees $t \in T_{\Delta'}$ such that (1) $\pi(t) \models \phi_{\mathrm{dom}}$, (2) $\rho_\gamma(t) \in T_\gamma$ for every $\gamma \in \Gamma$, and (3) for every vertex $u \in V_t$, if u has label $(\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$ then, for every $\sigma \in \Sigma$, $g_{\mathrm{n}}(\sigma) = 1 \Leftrightarrow \pi(t) \models \phi_\sigma(u)$. Regularity of T can be shown using Proposition 2 by presenting an MSOL formula that defines T: it is the conjunction of (1) the formula $\pi^{-1}(\phi_{\mathrm{dom}})$, (2) all formulas $\rho_\gamma^{-1}(\psi_\gamma)$, where ψ_γ defines T_γ according to Proposition 2, and (3) the formula $\forall u : \chi(u)$ where $\chi(u)$ is the disjunction of all formulas $\chi_{\delta'}(u)$, $\delta' \in \Delta'$, and for each $\delta' = (\delta, g_{\mathrm{e}}, g_{\mathrm{n}})$, $\chi_{\delta'}(u)$ is the conjunction of $\mathrm{lab}_{\delta'}(u)$, all formulas $\pi^{-1}(\phi_\sigma(u))$ with $g_{\mathrm{n}}(\sigma) = 1$, and all formulas $\pi^{-1}(\neg\phi_\sigma(u))$ with $g_{\mathrm{n}}(\sigma) = 0$.

Finally, for every $\gamma \in \Gamma$, we define $W(\gamma) = \rho_\gamma^{-1}(W_\gamma)$, i.e., $W(\gamma)$ consists of all strings $w \in (\Delta' \cup \overline{\Delta'})^*$ such that $\rho_\gamma(w) \in W_\gamma$, where ρ_γ is interpreted on strings in the obvious way, i.e., as a length-preserving homomorphism (with $\rho_\gamma(\overline{\delta'}) = \overline{\rho_\gamma(\delta')}$). Since the regular languages are closed under inverse homomorphisms, $W(\gamma)$ is regular. Obviously, if all edge formulas of f are of type B then, by Lemma 12, $W_\gamma \subseteq \Delta^*\overline{\Delta} \cup \overline{\Delta}\Delta^*$ and hence $W(\gamma) \subseteq (\Delta')^*\overline{\Delta'} \cup \overline{\Delta'}(\Delta')^*$, which means that R is of type B. Similarly, if all edge formulas of f are of type A, then every $W(\gamma)$ is finite and hence R is of type A.

This ends the definition of the regular path description R. To show that $L(R) = f(T_\Delta)$, it suffices to prove that $\pi(T) = \mathrm{dom}(f)$ and that $gr_R(t) = f(\pi(t))$ for every $t \in T$. The inclusion $\pi(T) \subseteq \mathrm{dom}(f)$ is immediate from point (1) in the definition of T. The inclusion $\mathrm{dom}(f) \subseteq \pi(T)$ follows from the definition of T and from the equality $\pi_\gamma(T_\gamma) = T_\Delta$ for every $\gamma \in \Gamma$. In fact, for every $s \in \mathrm{dom}(f)$ there is a unique tree $t \in T$ such that $\pi(t) = s$. Now consider some $t \in T$. By the definition of h and point (3) in the definition of T, $gr_R(t)$ and $f(\pi(t))$ have the same nodes, with the same labels. They also have the same edges (u, γ, v) because

$$\begin{aligned}
&\mathrm{bipath}_t(u,v) \in W(\gamma)\\
&\Leftrightarrow \ \text{(by the definition of } W(\gamma))\\
&\mathrm{bipath}_{\rho_\gamma(t)}(u,v) \in W_\gamma\\
&\Leftrightarrow \ \text{(by point (2) in the definition of } T \text{ and by Lemma 12)}\\
&\pi_\gamma(\rho_\gamma(t)) \models \phi_\gamma(u,v)\\
&\Leftrightarrow \ \text{(because } \pi_\gamma \circ \rho_\gamma = \pi)\\
&\pi(t) \models \phi_\gamma(u,v).
\end{aligned}$$

This proves the lemma. □

It now remains to prove that strings are equivalent with linear trees, cf. Examples 2(1) and 1(1).

Lemma 14. *MSOF(LIN-TREES) = MSOF(STRINGS).*

Proof. To show that MSOF(LIN-TREES) $\subseteq$ MSOF(STRINGS), let f be an MSO definable function $GR_{\Delta,\mathrm{rks}(\Delta)} \to GR_{\Sigma,\Gamma}$, where Δ is a ranked alphabet of which all symbols have rank 1 or 0. Note that every tree in T_Δ is also a string in Δ^*, apart from the edge labels 1 that should be $*$ (note that $\mathrm{rks}(\Delta) = \{1\}$). Vice versa, a string $w \in \Delta^*$ is in T_Δ if it is nonempty, the last symbol of w has rank 0, and all other symbols have rank 1. Define the function $f' : GR_{\Delta,\{*\}} \to GR_{\Sigma,\Gamma}$ by changing every $\mathrm{edge}_1(x,y)$ in the formulas of f into $\mathrm{edge}_*(x,y)$, and by adding to the domain formula of f the above requirement that there is at least one node, that the label of the (unique) node with no outgoing edges has rank 0, and that the labels of all other nodes have rank 1 (which can easily be expressed in an MSOL formula). Then $f'(\Delta^*) = f(T_\Delta)$ and so $f(T_\Delta)$ is in MSOF(STRINGS).

To show that MSOF(STRINGS) $\subseteq$ MSOF(LIN-TREES), let f be an MSO definable function $GR_{\Delta,\{*\}} \to GR_{\Sigma,\Gamma}$, where Δ is an ordinary alphabet. Define the ranked alphabet $\Delta' = \Delta \cup \{e\}$ where every element of Δ has rank 1 and e is a new symbol of rank 0. Then $T_{\Delta'} = \{we \mid w \in \Delta^*\}$. It is easy to see that the function $g : T_{\Delta'} \to \Delta^*$ with $g(we) = w$ is MSO definable (take the domain formula to be *true*, take the node formula $\phi_\delta(u)$ to be $\mathrm{lab}_\delta(u)$, for every $\delta \in \Delta$, and take the edge formula $\phi_*(u, v)$ to be $\mathrm{edge}_1(u, v)$). Clearly, $f(\Delta^*) = f(g(T_{\Delta'}))$ and so, since $f \circ g$ is MSO definable by Theorem 8, $f(\Delta^*)$ is in MSOF(LIN-TREES). □

6 Closure and Decidability Properties

From Theorem 9 a lot of closure properties and decidability results for RPD can be deduced, which we will now discuss. We start with *closure properties.* The main result is that RPD is closed under MSO definable functions, i.e., if $L \in$ RPD and $f \in$ MSOF, then $f(L) = \{f(H) \mid H \in L, H \in \mathrm{dom}(f)\}$ is in RPD. This is immediate from Theorem 9 and the closure of MSOF under composition (Theorem 8).

Theorem 15. *RPD and LIN-RPD are closed under MSO definable functions.*

Every tree language T_Δ is in A-MSOF(TREES) because the identity on T_Δ is MSO definable with edge formulas of type A (viz. the formulas $\mathrm{edge}_\gamma(u, v)$). Consequently, RPD is the closure of A-RPD under the MSO definable functions. Hence, since B-RPD is a proper subclass of RPD (see Theorem 22 of [EO]), neither A-RPD nor B-RPD is closed under arbitrary MSO definable functions.

As a corollary of Theorem 15 we reobtain Courcelle's intersection result (cf. the Introduction) for RPD: RPD is closed under intersection with MSO definable graph languages (Theorem 6.9 of [CER]). This is because for every MSO definable language R the identity function on R is MSO definable.

Theorem 16. *RPD, B-RPD, A-RPD, and LIN-RPD are closed under intersection with MSO definable graph languages.*

Proof. Let $R \subseteq GR_{\Sigma,\Gamma}$ be an MSO definable graph language, defined by a closed formula ϕ in $\mathrm{MSOL}(\Sigma, \Gamma)$. Obviously id_R, the identity function on R, is MSO definable: the domain formula is ϕ, and the node and edge formulas are $\mathrm{lab}_\sigma(u)$ and $\mathrm{edge}_\gamma(u, v)$, respectively. Since for every language L, $L \cap R = \mathrm{id}_R(L)$, the statement follows from Theorem 15 for RPD and LIN-RPD. Let $L = f(T_\Delta)$ be a graph language in B-MSOF(TREES), where the edge formulas of f are of type B. Then $L \cap R = (\mathrm{id}_R \circ f)(T_\Delta)$. From the proofs of Theorem 8 and Lemma 7 it can easily be seen that $\mathrm{id}_R \circ f$ has the same edge formulas as f. Thus, $L \cap R$ is in B-MSOF(TREES). The same argument holds for A-MSOF(TREES). □

The closure under intersection with MSO definable sets was first proved by Courcelle in Corollary 4.8 of [Cou2] for the case of Hyperedge Replacement graph grammars, which generate a subclass of RPD. This was the first result that related context-free graph languages to monadic second-order logic, and was the main source of inspiration for the present paper.

From Theorem 15 we also obtain the known result that RPD is closed under edge complement (cf. the discussion before Theorem 22 in [EO]): assuming that there is just one edge label γ, define f by taking $\neg\mathrm{edge}_\gamma(u,v)$ as edge formula (and all $\mathrm{lab}_\sigma(u)$ as node formulas, and *true* as domain formula). Example 2(1) shows that RPD is closed under taking the transitive closure of each graph, and similarly it is easy to see that one can also throw away all existing transitive edges, i.e., turn an acyclic graph into its Hasse diagram. In general one can add or remove edges that satisfy certain MSO properties (or rather their incident nodes satisfy them), and similarly one can remove (but not add) nodes that satisfy MSO properties, such as removing all isolated nodes from all graphs of the language.

As another consequence of Theorem 15 we show that if L is an RPD language, then so is the language of all induced subgraphs (of graphs of L) that satisfy a given MSO property. For a graph H and $U \subseteq V_H$, we denote by $H[U]$ the subgraph of H induced by U. Let $\phi(U)$ be an MSOL formula. We say that $H[U]$ is an *induced ϕ-subgraph* of H if $H \models \phi(U)$.

Theorem 17. *Let $\phi(U)$ be a formula in MSOL(Σ, Γ) and let $L \subseteq GR_{\Sigma,\Gamma}$. If L is in RPD, then the set of all induced ϕ-subgraphs of graphs in L is in RPD. The same holds for B-RPD, A-RPD, and LIN-RPD.*

Proof. The idea of the proof is to add 0 or 1, nondeterministically, to the labels of a graph H in L, and let U be the set of nodes which are labeled 1. From this graph, $H[U]$ can easily be obtained by an MSO definable function.

Let $\Sigma' = \Sigma \times \{0,1\}$, and let $\rho : GR_{\Sigma',\Gamma} \to GR_{\Sigma,\Gamma}$ be the mapping that changes every node label (σ, i) into σ. Let us now show that the graph language $\rho^{-1}(L) \subseteq GR_{\Sigma',\Gamma}$ is in RPD. By Theorem 9 we know that RPD = MSOF(TREES). Let $L = f(T_\Delta)$ for some f in MSOF and some ranked alphabet Δ. Let Δ' be the ranked alphabet $\Delta \times \{0,1\}$ where the rank of (δ, i) is the one of δ, and let $\pi : \Delta' \to \Delta$ be the projection such that $\pi(\delta, i) = \delta$. For $i = 0,1$ let $\mathrm{bit}_i^\Delta(u)$ be the disjunction of all formulas $\mathrm{lab}_{(\delta,i)}(u)$, $\delta \in \Delta$. Then $\rho^{-1}(L) = f'(T_{\Delta'})$ where the defining formulas of f' are obtained from those of f as follows: ϕ'_{dom} is $\pi^{-1}(\phi_{\mathrm{dom}})$, $\phi'_{(\sigma,i)}(u)$ is $\pi^{-1}(\phi_\sigma(u)) \wedge \mathrm{bit}_i^\Delta(u)$, and $\phi'_\gamma(u,v)$ is $\pi^{-1}(\phi_\gamma(u,v))$. Hence $\rho^{-1}(L) \in$ RPD.

It is easy to see that there is an MSO definable function g that translates $\rho^{-1}(L)$ into the required language $\{H[U] \mid H \in L, U \subseteq V_H, H \models \phi(U)\}$ (and then the result follows from Theorem 15). In fact, the domain formula ϕ_{dom} of g expresses the fact that the set U of all nodes that have bit 1 in their label satisfies $\phi(U)$, i.e., ϕ_{dom} is $\forall U : (\forall u : u \in U \leftrightarrow \mathrm{bit}_1^\Sigma(u)) \to \phi(U)$, where $\mathrm{bit}_1^\Sigma(u)$ is defined just as $\mathrm{bit}_1^\Delta(u)$. The node formula $\phi_\sigma(u)$ of g is $\mathrm{lab}_{(\sigma,1)}(u)$, and the edge formula $\phi_\gamma(u,v)$ of g is $\mathrm{edge}_\gamma(u,v)$.

It is left to the reader to verify that the constructions preserve the types B, A, and LIN. □

As an example, if L is in RPD, then the set of all connected components of graphs in L is also in RPD, because there is an MSOL formula $\phi(U)$ expressing that U is a connected component (see Example 3(2)).

The MSO definable functions of [Cou4, Cou5] are more general in two ways. First, they allow the addition of nodes: each node of the input graph is used to represent (at most) k nodes of the output graph, where k is fixed. As an example, the function f that maps each graph H into the disjoint union of H with itself is then MSO definable (with $k = 2$); clearly this function is not MSO definable in our sense, because $f(H)$ has more nodes than H. Second, by admitting free variables in the defining formulas, MSO definable relations are obtained. As an example, a graph can be translated into its connected components. RPD is also closed under these generalized MSO definable relations, and, in fact, Theorem 17 is a special case of this (see Theorems 3.2 and 3.4 of [CE]).

We now turn to *decidability properties*. As observed after Theorem 5, the results that follow also hold for C-edNCE graph grammars instead of regular path descriptions. We start showing that the emptiness and finiteness problems are decidable for RPD. Although this is well known for C-edNCE grammars, it is proved here for completeness sake.

Proposition 18. *It is decidable for an arbitrary regular path description R whether or not $L(R) = \emptyset$, and whether or not $L(R)$ is finite.*

Proof. Let $R = (\Delta, \Sigma, \Gamma, T, h, W)$. Clearly, $L(R) = \emptyset$ if and only if $T = \emptyset$. Since emptiness of regular tree languages is decidable, emptiness of $L(R)$ can be decided.

To show the decidability of finiteness of $L(R)$ we construct a context-free grammar G such that $L(R)$ is finite iff $L(G)$ is finite (and then use the decidability of the finiteness problem for context-free grammars). Since there are only finitely many graphs with a given number of nodes, it suffices to construct G in such a way that, for every $n \in \mathbb{N}$, $a^n \in L(G)$ iff there is a tree $t \in T$ such that $gr_R(t)$ has n nodes. Let (N, Δ, P, S) be a regular tree grammar generating T (see Section 2.2). We define $G = (N, \{a\}, P', S)$ where P' is constructed as follows. If $A \to t$ is a production in P (where t is in $T_{\Delta \cup N}$, or more precisely, denotes a tree in $T_{\Delta \cup N}$) and if every symbol $\delta \in \Delta$ that occurs in t is in $\mathrm{dom}(h)$, then $A \to w$ is in P', where the string w is obtained from t by changing every symbol of Δ into a (and leaving the nonterminals as they are). □

It is well known that Proposition 18 and Theorem 16 can be combined in an obvious way: it is decidable for an RPD language L and an MSO definable graph language R whether or not $L \cap R = \emptyset$, and whether or not $L \subseteq R$ (and similarly for finiteness). Proposition 18 and Theorem 15 can be combined in the same way: it is decidable for an MSO definable function f and an RPD language L whether or not $f(L)$ is empty, and whether or not $f(L)$ is finite. Concentrating on finiteness, this implies that certain boundedness problems are decidable for RPD, cf. [HKV2] where boundedness problems are investigated for Hyperedge Replacement grammars (see also the recent [Eng5, Dre]). As an example, let f be the MSO definable function that transforms every graph into the discrete graph consisting of all its isolated nodes. Then the above result shows that it is decidable for an RPD language L whether there is a bound on the number of isolated nodes in the graphs of L. We now show two general decidability results for boundedness problems.

For an MSOL formula $\phi(U)$ and a graph H, we denote by $size_\phi(H)$ the maximal number of nodes of an induced ϕ-subgraph of H.

Theorem 19. *Let $\phi(U)$ be an MSOL formula. It is decidable for a regular path description R whether or not there exists a natural number b such that $size_\phi(H) \leq b$ for all $H \in L(R)$.*

Proof. Let L_ϕ be the set of all induced ϕ-subgraphs of graphs in $L(R)$. It should be clear that $\mathrm{size}_\phi(H)$ is bounded on $L(R)$ if and only if L_ϕ is finite. Since Theorem 17 is effective, a regular path description for the language L_ϕ can be constructed, and L_ϕ can be tested on finiteness by Proposition 18. $\square$

A similar result can be shown for the number of induced ϕ-subgraphs rather than their size. For an MSOL formula $\phi(U)$ and a graph H, we denote by $num_\phi(H)$ the number of induced ϕ-subgraphs of H (where isomorphic ones are *not* identified).

Theorem 20. *Let $\phi(U)$ be an MSOL formula. It is decidable for a regular path description R whether or not there exists a natural number b such that $num_\phi(H) \leq b$ for all $H \in L(R)$.*

Proof. For a graph H and two nodes u and v of H, define u and v to be ϕ-equivalent, denoted by $u \equiv_\phi v$, if for all $U \subseteq V_H$ with $H \models \phi(U)$: $u \in U \leftrightarrow v \in U$. In other words, u and v are ϕ-equivalent if they belong to the same induced ϕ-subgraphs of H. Let $\mathrm{eq}_\phi(H)$ be the number of equivalence classes of the equivalence relation $\equiv_\phi$ on V_H. Clearly, $\mathrm{eq}_\phi(H) \leq 2^{\mathrm{num}_\phi(H)}$ and $\mathrm{num}_\phi(H) \leq 2^{\mathrm{eq}_\phi(H)}$. Hence, $\mathrm{num}_\phi(H)$ is bounded on $L(R)$ if and only if $\mathrm{eq}_\phi(H)$ is bounded on $L(R)$. To decide boundedness of $\mathrm{eq}_\phi(H)$ we consider representatives of the equivalence classes of $\equiv_\phi$. For a graph H, define a ϕ-representative set to be a subset of V_H that contains exactly one node from each equivalence class of $\equiv_\phi$. Let $\psi(V)$ be the MSOL formula that expresses that V is a ϕ-representative set: $(\forall u\ \exists v : v \in V \wedge u \equiv_\phi v) \wedge (\forall u, v : (u \in V \wedge v \in V \wedge u \equiv_\phi v) \rightarrow u = v)$, where $u \equiv_\phi v$ is expressed by the formula $\forall U : \phi(U) \rightarrow (u \in U \leftrightarrow v \in U)$. Then, clearly, $\mathrm{eq}_\phi(H)$ is bounded on $L(R)$ if and only if $\mathrm{size}_\psi(H)$ is bounded on $L(R)$. The latter is decidable by Theorem 19. $\square$

It follows from these theorems that (almost) all concrete decidability results proved for Hyperedge Replacement grammars in [HKV2] are also decidable for regular path descriptions (and hence for C-edNCE grammars). Moreover, our results seem to be easier to use than those in [HKV2]. In fact, the general results of [HKV2] are formulated in terms of compatible functions rather than MSO formulas, and it is usually much easier to express a certain graph property in MSOL than to show its compatibility: the graph theoretical definition of the property can usually be written directly in the logic. Let us give some concrete examples.

Example 3. (1) It is decidable whether an RPD language is of bounded degree. In fact, let us say that a set U of nodes of a graph H is a "neighbourhood" if it consists of all neighbours of some node of H. An MSOL formula $\phi(U)$ expressing that U is a neighbourhood is $\exists u\, \forall v : v \in U \leftrightarrow \text{edge}(u,v) \vee \text{edge}(v,u)$. Thus, the bounded degree property is decidable for RPD languages by Theorem 19.

(2) Let $\text{upath}(u,v)$ be a formula expressing that there is an undirected path from u to v. It can be defined in the same way as the formula $\text{path}(u,v)$ in Section 2.3, using $\text{edge}(u,v) \vee \text{edge}(v,u)$ instead of $\text{edge}(u,v)$ in the formula $\text{closed}(U)$. Then the formula $\exists u : u \in U \wedge (\forall v : v \in U \leftrightarrow \text{upath}(u,v))$ expresses the fact that U is a connected component. Hence it is decidable for an RPD graph language whether or not there is a bound on the size of the connected components of its graphs, and also, whether or not there is a bound on the number of connected components of its graphs (by Theorems 19 and 20, respectively). The same holds for the strongly connected components.

(3) Let $\phi(U)$ be a formula for the property that U is a (maximal) clique. For instance $\phi(U)$ is $\forall v : v \in U \leftrightarrow (\forall u : u \in U \to \text{edge}(u,v) \wedge \text{edge}(v,u))$. By Theorem 19 it is decidable for $L \in \text{RPD}$ whether there is a bound on the size of cliques in the graphs of L. This was shown for (a subclass of) B-RPD in [RW2]. By Theorem 20 it is decidable whether the number of cliques is bounded for the graphs in L. □

Note that, of course, all decidability results of this section hold for classes of graph grammars that are effectively contained in RPD. Thus, apart from the C-edNCE grammars, they also hold, e.g., for the Hyperedge Replacement grammars (see, e.g., [BC, HK, ER1]), the B-NLC graph grammars (see [RW1]), and the C-NLC graph grammars (see [Cou1]).

Acknowledgement:

The first author wishes to thank Bruno Courcelle, George Leih, and Roderick Bloem for stimulating discussions.

References

[ALS] S.Arnborg, J.Lagergren, D.Seese; Easy problems for tree-decomposable graphs, J. of Algorithms 12 (1991), 308-340

[BC] M.Bauderon, B.Courcelle; Graph expressions and graph rewritings, Math. Systems Theory 20 (1987), 83-127

[Bra] F.J.Brandenburg; On polynomial time graph grammars, Proc. STACS 88, Lecture Notes in Computer Science 294, Springer-Verlag, Berlin, 1988, pp.227-236

[Buc] J.Büchi; Weak second-order arithmetic and finite automata, Z. Math. Logik Grundlag. Math. 6 (1960), 66-92

[CE] B.Courcelle, J.Engelfriet; A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars, Math. Systems Theory 28 (1995), 515-552

[CEER] J.Cuny, H.Ehrig, G.Engels, G.Rozenberg (eds.); *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science 1073, Springer-Verlag, Berlin, 1996

[CER] B.Courcelle, J.Engelfriet, G.Rozenberg; Handle-rewriting hypergraph grammars, J. of Comp. Syst. Sci. 46 (1993), 218-270

[CLS] D.G.Corneil, H.Lerchs, L.Stewart Burlingham; Complement reducible graphs, Discr. Appl. Math. 3 (1981), 163-174

[Cou1] B.Courcelle; An axiomatic definition of context-free rewriting and its application to NLC graph grammars, Theor. Comput. Sci. 55 (1987), 141-181

[Cou2] B.Courcelle; The monadic second-order logic of graphs I: Recognizable sets of finite graphs, Inform. and Comput. 85 (1990), 12-75

[Cou3] B.Courcelle; Graph rewriting: an algebraic and logic approach, in *Handbook of Theoretical Computer Science*, Vol.B (J.van Leeuwen, ed.), Elsevier, 1990, pp.193-242

[Cou4] B.Courcelle; The monadic second-order logic of graphs V: On closing the gap between definability and recognizability, Theor. Comput. Sci. 80 (1991), 153-202

[Cou5] B.Courcelle; The monadic second-order logic of graphs VII: Graphs as relational structures, Theor. Comput. Sci. 101 (1992), 3-33

[Cou6] B.Courcelle; Monadic second-order definable graph transductions: a survey, Theor. Comput. Sci. 126 (1994), 53-75

[Cou7] B.Courcelle; Structural properties of context-free sets of graphs generated by vertex replacement, Inform. and Comput. 116 (1995), 275-293

[Cou8] B.Courcelle; The expression of graph properties and graph transformations in monadic second-order logic, Chapter in [Roz]

[DHK] F.Drewes, A.Habel, H.-J.Kreowski; Hyperedge replacement graph grammars, Chapter in [Roz]

[Don] J.Doner; Tree acceptors and some of their applications, J. Comp. Syst. Sci. 4 (1970), 406-451

[Dre] F.Drewes; Computation by tree transductions, Ph. D. Thesis, University of Bremen, February 1995

[EHL] J.Engelfriet, L.M.Heyker, G.Leih; Context-free graph languages of bounded degree are generated by apex graph grammars, Acta Informatica 31 (1994), 341-378

[EKR] H.Ehrig, H.-J.Kreowski, G.Rozenberg (eds.); *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science 532, Springer-Verlag, Berlin, 1991

[EL] J.Engelfriet, G.Leih; Linear graph grammars: power and complexity, Inform. and Comput. 81 (1989), 88-121

[Elg] C.C.Elgot; Decision problems of finite automata and related arithmetics, Trans. Amer. Math. Soc. 98 (1961), 21-51

[ELW] J.Engelfriet, G.Leih, E.Welzl; Boundary graph grammars with dynamic edge relabeling, J. of Comp. Syst. Sci. 40 (1990), 307-345

[Eng1] J.Engelfriet; Monadic second-order logic for graphs, trees, and strings, Copies of transparencies, November 1988

[Eng2] J.Engelfriet; Context-free NCE graph grammars, Proc. FCT '89, Lecture Notes in Computer Science 380, Springer-Verlag, Berlin, 1989, pp.148-161

[Eng3] J.Engelfriet; A characterization of context-free NCE graph languages by monadic second-order logic on trees, in [EKR], pp.311-327

[Eng4] J.Engelfriet; A regular characterization of graph languages definable in monadic second-order logic, Theor. Comput. Sci. 88 (1991), 139-150.

[Eng5] J.Engelfriet; Graph grammars and tree transducers, Proc. CAAP'94 (S.Tison, ed.), Lecture Notes in Computer Science 787, Springer-Verlag, Berlin, 1994, pp.15-36

[Eng6] J.Engelfriet; Context-free graph grammars, Chapter for the *Handbook of Formal Languages* (G.Rozenberg, A.Salomaa, eds.), Volume III, Springer-Verlag, to appear.

[ENRR] H.Ehrig, M.Nagl, G.Rozenberg, A.Rosenfeld (eds.); *Graph-Grammars and their Application to Computer Science*, Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, 1987

[EO] J.Engelfriet, V.van Oostrom; Regular description of context-free graph languages, Tech. Report 95-34, Leiden University, 1995, to appear in J. of Comp. Syst. Sci.

[ER1] J.Engelfriet, G.Rozenberg; A comparison of boundary graph grammars and context-free hypergraph grammars, Inform. and Comput. 84 (1990), 163-206

[ER2] J.Engelfriet, G.Rozenberg; Node replacement graph grammars, Chapter in [Roz]

[GS] F.Gécseg, M.Steinby; *Tree automata*, Akadémiai Kiadó, Budapest, 1984

[Hab] A.Habel; *Hyperedge Replacement: Grammars and Languages*, Lecture Notes in Computer Science 643, Springer-Verlag, 1992

[HK] A.Habel, H.-J.Kreowski; May we introduce to you: hyperedge replacement, in [ENRR], pp.15-26

[HKV1] A.Habel, H.-J.Kreowski, W.Vogler; Metatheorems for decision problems on hyperedge replacement graph languages, Acta Informatica 26 (1989), 657-677

[HKV2] A.Habel, H.-J.Kreowski, W.Vogler; Decidable boundedness problems for sets of graphs generated by hyperedge replacement, Theor. Comput. Sci. 89 (1991), 33-62

[HU] J.E.Hopcroft, J.D.Ullman; *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979

[Kau] M.Kaul; Syntaxanalyse von Graphen bei Präzedenz-Graph-Grammatiken, Dissertation, Universität Osnabrück, 1985

[LW] T.Lengauer, E.Wanke; Efficient decision procedures for graph properties on context-free graph languages, J. of the ACM 40 (1993), 368-393

[MW] J.Mezei, J.B.Wright; Algebraic automata and context-free sets, Inform. and Control 11 (1967), 3-29

[Oos] V.van Oostrom; Graph grammars and 2nd order logic (in Dutch), M. Sc. Thesis, Leiden University, January 1989

[Roz] G.Rozenberg (ed.), *Handbook of Graph Transformations*, Volume I: Foundations, World Scientific, to appear.

[RW1] G.Rozenberg, E.Welzl; Boundary NLC graph grammars - basic definitions, normal forms, and complexity, Inform. and Control 69 (1986), 136-167

[RW2] G.Rozenberg, E.Welzl; Combinatorial properties of boundary NLC graph languages, Discr. Appl. Math. 16 (1987), 59-73

[Schu] R.Schuster; Graphgrammatiken und Grapheinbettungen: Algorithmen und Komplexität, Technical Report MIP-8711, Universität Passau, 1987

[Tho] W.Thomas; Automata on infinite objects, in *Handbook of Theoretical Computer Science*, Vol.B (J.van Leeuwen, ed.), Elsevier, 1990, pp.133-192

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

[Wel] E.Welzl; Boundary NLC and partition controlled graph grammars, in [ENRR], pp.593-609

This article was processed using the LaTeX macro package with LLNCS style