Theoretical Computer Science 126 (1994) 53–75
Elsevier

Monadic second-order definable graph transductions: a survey*

Bruno Courcelle

*Laboratoire d'Informatique**, *Université Bordeaux-1, 351 Cours de la Libération, 33405 Talence, France*

Abstract

Courcelle, B., Monadic second-order definable graph transductions: a survey, Theoretical Computer Science 126 (1994) 53–75.

Formulas of monadic second-order logic can be used to specify graph transductions, i.e., multi-valued functions from graphs to graphs. We obtain in this way classes of graph transductions, called *monadic second-order definable graph transductions* (or, more simply, *definable transductions*) that are closed under composition and preserve the two known classes of context-free sets of graphs, namely the class of hyperedge replacement (HR) and the class of vertex replacement (VR) sets. These two classes can be characterized in terms of definable transductions and recognizable sets of finite trees, independently of the rewriting mechanisms used to define the HR and VR grammars. When restricted to words, the definable transductions are strictly more powerful than the rational transductions such that the image of every finite word is finite; they do not preserve context-free languages. We also describe the sets of discrete (edgeless) labelled graphs that are the images of HR and VR sets under definable transductions: this gives a version of Parikh's theorem (i.e., the characterization of the commutative images of context-free languages) which extends the classical one and applies to HR and VR sets of graphs.

0. Introduction

The theory of formal languages investigates finite devices defining sets of finite and countably infinite words and trees, compares their expressive powers, and investigates the solvability of the associated decision problems. These investigations make an essential use of transformations from words or trees to words or trees usually called *transductions*. Of special importance are the *rational* (word to word) transductions;

Correspondence to: B. Courcelle, Université Bordeaux-1, Laboratoire d'Informatique, 351 Cours de la Libération, 33405 Talence, France. Email: courcell@labri.u-bordeaux.fr.

* Supported by the ESPRIT Basic-Research project "Computing by graph transformation" and by the "Programme de Recherches Coordonnées: Mathématiques et Informatique".

** Unité associée au CNRS no. 1304.

SSDI 0304-3975(93)E0123-L

they are closed under composition and inverse, and they preserve the families of recognizable and context-free languages. Tree transductions are more complicated, and there is no unique notion that can be considered as the analogue of that of a rational transduction. For each class of tree transductions, the closure under composition is a major concern, and so is the preservation of recognizability; we refer the reader to the survey by Raoult [36]. Another important transduction is **yield** that maps derivation trees of context-free grammars to the corresponding words. The context-free languages can be characterized as the images of the recognizable sets of (finite) trees under **yield** mappings.

The study of sets of finite and countably infinite graphs (and hypergraphs) by tools like grammars, systems of equations and logical formulas is a relatively recent development in the theory of formal languages. The need for a manageable and powerful notion of graph transduction appears in constructions dealing with graph grammars and is of interest on its own. This paper is a survey presenting the notion of a *monadic second-order definable graph transduction* (a *definable transduction* for short), which has been introduced more or less explicitly and sometimes in restricted forms in several papers [1, 9, 10, 13, 15, 22].

We now introduce these transductions informally. The terms "monadic second-order" refers to a logical language, the *monadic second-order logic*. We recall the role of logic for defining sets of graphs (or hypergraphs; all what we shall say concerning graphs applies to hypergraphs as well). Graphs can be described by relational structures, i.e., by logical structures with no function symbols. The domain of the structure representing a graph is the set of its vertices and edges put together; basic relations describe the incidence of vertices and edges and possible labellings. (This is actually not the only way to represent a graph; see Sections 1 and 4 for more details.) Hence, formulas of any appropriate logical language define properties of this graph. Monadic second-order logic is popular among logicians because of its expressive power and its decidability properties (see [30] for a survey). For dealing with graphs, it is very useful because it can express many fundamental properties (like planarity, connectivity, k-colorability for fixed k) whereas several general decidability results hold. The sets of words characterized by a property expressible in monadic second-order logic are exactly the recognizable sets by the results of Büchi [5] and Elgot [20], presented in [38, Theorem 3.2]. The same property holds for finite trees, as established by Doner [19], see [38, Theorem 11.1]. Sets of graphs defined similarly by characteristic monadic second-order properties behave very much like recognizable sets of words and trees, in particular in constructions involving context-free graph grammars. Since no notion of finite-state graph automaton is known, monadic second-order formulas are essential in such constructions.

We now come to graph transductions. Since we have no good (general) notion of graph automaton, we have no chance to obtain a good notion of graph transduction based on a finite-state machine model. Alternatively, we propose to define transductions of graphs (or more generally of relational structures) by means of monadic second-order formulas. The idea is to transform a structure S into a structure T by

defining T "inside" S by means of such formulas. This is nothing but the classical notion of *semantic interpretation* (see for instance [35]), appropriately extended. In particular, we define T inside an intermediate structure made of k disjoint copies of S (for some fixed k), equipped with a binary relation saying that two elements replicate the same element of S. This makes it possible to construct T with a domain larger than that of S (larger within the factor k).

The family of definable transductions is closed under composition, but not under inverse. These transductions preserve the so-called HR and VR sets of graphs (namely, the two known types of context-free sets of graphs), and yield grammar-independent characterizations of these sets.

This paper is organized as follows. Section 1 reviews relational structures, the way they represent graphs and hypergraphs, and monadic second-order logic; Section 2 introduces (monadic second-order) definable transductions of relational structures and presents a collection of basic examples of such transductions of words, trees and graphs; Section 3 states the main properties of definable transductions of relational structures; Section 4 presents the relationships between definable transductions and the classes of VR and HR sets of graphs (and hypergraphs); Section 5 shows how these transductions make it possible to compare several representations of graphs by relational structures and to code hypergraphs by graphs in a way that fits well with context-free graph grammars; Section 6 compares definable transductions with known transductions of words and trees: Section 7 deals with definable transductions from graphs to commutative words and gives forms of Parikh's theorem that apply to HR and VR graph grammars.

1. Hypergraphs and relational structures

We denote by $\mathbf{Card}(A)$ the cardinality of a set A. We denote by $[n]$ the set of positive integers $\{1, \ldots, n\}$.

For a binary relation $R \subseteq A \times B$, we write aRb for $(a, b) \in R$. The associated mapping from A to the powerset of B (also denoted by R and defined by $R(a) := \{b \in B \mid aRb\}$) is called a *transduction* from A to B. We consider every b such that aRb as *an image of a under* R, hence we consider R as a multivalued function from A to B. The *domain* of R is $\mathbf{Dom}(R) := \{a \in A \mid aRb$ for some b in $B\}$ and the *image* of R is $\mathbf{Im}(R) := \{b \in B \mid aRb$ for some a in $A\}$. For $L \subseteq A$, the *image of* L *under* R is $R(L) := \{b \in B \mid aRb$ for some a in $L\}$. The transduction R^{-1}, associated with the binary relation $\{(b, a) \mid (a, b) \in R\}$ is called the *inverse* of R. If L is a subset of B, then $R^{-1}(L)$ is the *inverse image of* L *under* R. We say that R is *functional* if $\mathbf{Card}(R(\{a\})) \leqslant 1$ for every a in A. We identify functional relations $R \subseteq A \times B$ with partial functions $R : A \to B$, and we write $b = R(a)$ instead of $b \in R(a)$. The composition of a transduction R from A to B and a transduction S from B to C is the transduction from A to C, denoted by $S \circ R$ and associated with the product of the relations R and S, i.e., with the relation $\{(a, c) \mid (a, b) \in R$ and $(b, c) \in S$ for some b in $B\}$. By a *mapping*, we shall mean a total function.

1.1. Hypergraphs

We shall deal with labelled, directed hypergraphs. The labels (intended to label hyperedges) are chosen in a finite *ranked* alphabet A, i.e., a finite alphabet A given with a *rank mapping* $\rho: A \to \mathbb{N}$. The rank of the label of a hyperedge must be equal to the length of its sequence of incident vertices. This rank may be 0, i.e., we allow hyperedges with no vertex. Graphs appear as a special case where all labels are of rank 2 or 1. A hyperedge of length 2 is a (usual) labelled directed edge. It is a *loop* if its two ends are identical. A hyperedge of length 1 can be considered as a piece of information attached to a vertex, hence as a vertex label. However, one may have several, possibly identical labels attached to the same vertex.

A *concrete hypergraph over* A is a 4-tuple $G = \langle \boldsymbol{V}_G, \boldsymbol{E}_G, \mathbf{lab}_G, \mathbf{vert}_G \rangle$, where $\boldsymbol{V}_G$ is the finite set of vertices, $\boldsymbol{E}_G$ is the finite set of (hyper) edges, disjoint with $\boldsymbol{V}_G$ (its elements will hereafter be called *edges* for short), $\mathbf{lab}_G$ is a mapping $\boldsymbol{E}_G \to A$ that defines the *label* of an edge, and $\mathbf{vert}_G$ is a mapping that associates with every edge e the *sequence of its vertices*; this sequence must be of length $\rho(e) := \rho(\mathbf{lab}_G(e))$ (called the *rank* of the edge) and its ith element is denoted by $\mathbf{vert}_G(e, i)$. A concrete hypergraph G is *simple* if, for all e, e' in $\boldsymbol{E}_G$: if $\mathbf{vert}_G(e) = \mathbf{vert}_G(e')$ and $\mathbf{lab}_G(e) = \mathbf{lab}_G(e')$, then $e = e'$. By a *hypergraph*, we mean the isomorphism class of a concrete hypergraph. We denote by $\mathbf{HG}(A)$ the set of hypergraphs over A.

1.2. Monadic second-order logic

Let R be a finite ranked set of symbols where each element r in R has a rank $\rho(r)$ in $\mathbb{N}_+$. A symbol r in R is considered as a $\rho(r)$-ary relation symbol. An R-*(relational) structure* is a tuple $S = \langle \boldsymbol{D}_S, (r_S)_{r \in R} \rangle$, where $\boldsymbol{D}_S$ is a finite (possibly empty) set, called the *domain* of S, and r_S is a subset of $\boldsymbol{D}_S^{\rho(r)}$ for each r in R. We denote by $\mathscr{S}(R)$ the class of R-structures.

We review *monadic second-order logic* briefly. Its formulas (called *MS formulas* for short), intended to describe the properties of structures S as above, are written with variables of two types, namely lower-case symbols $x, x', y, \ldots$ called *object variables*, denoting elements of $\boldsymbol{D}_S$, and upper-case symbols $X, Y, Y', \ldots$ called *set variables*, denoting subsets of $\boldsymbol{D}_S$. The atomic formulas are of the forms $x = y$, $r(x_1, \ldots, x_n)$ (where r is in R and $n = \rho(r)$), and $x \in X$, and formulas are formed with propositional connectives and quantifications over the two kinds of variables. For every finite set W of object and set variables, we denote by $\mathscr{L}(R, W)$ the set of all formulas that are written with relational symbols from R and have their free variables in W; we also let $\mathscr{L}(R) := \mathscr{L}(R, \emptyset)$ denote the set of closed formulas.

Let S be an R-structure, let $\varphi \in \mathscr{L}(R, W)$, and let γ be a W-assignment in S (i.e., $\gamma(X)$ is a subset of $\boldsymbol{D}_S$ for every set variable X in W, and $\gamma(x) \in \boldsymbol{D}_S$ for every object variable x in W; we write this as $\gamma: W \to S$, to be short). We write $(S, \gamma) \models \varphi$ if and only if φ holds in S for γ. We write $S \models \varphi$ in the case where φ has no free variable. A set of R-structures

L is *definable* if there is a formula φ in $\mathscr{L}(R)$ such that L is the set of all R-structures S such that $S \models \varphi$.

A hypergraph G in $\mathbf{HG}(A)$ can be represented by an R_A-structure, where $R_A := \{\mathbf{edg}_a \mid a \in A\}$ with $\rho(\mathbf{edg}_a) := \rho(a)+1$. The structure representing G is $|G|_2 := \langle \boldsymbol{D}_G, (\mathbf{edg}_{aG})_{a\in A}\rangle$, where $\boldsymbol{D}_G := \boldsymbol{V}_G \cup \boldsymbol{E}_G$ (let us recall that $\boldsymbol{V}_G \cap \boldsymbol{E}_G = \emptyset$), $\mathbf{edg}_{aG}(x, y_1, \ldots, y_n) :\Leftrightarrow x \in \boldsymbol{E}_G$, $\mathbf{lab}_G(x) = a$ and $\mathbf{vert}_G(x) = (y_1, \ldots, y_n)$. Since the domain of this structure consists of vertices *and* edges, quantifications can be done over *two types* of objects or sets of objects. The index 2 refers to these two possibilities, and differenciates this structure from another one defined below where quantifications can be done over vertices and sets of vertices only.

Clearly, $|G|_2$ is isomorphic to $|G'|_2$ as a relational structure if and only if $G = G'$ (i.e., G is isomorphic to G'). We shall consider any two isomorphic structures as equal, like for hypergraphs.

A hypergraph G can be represented by another structure $|G|_1 := \langle \boldsymbol{V}_G, (\mathbf{edg}'_{aG})_{a\in A}\rangle$, where $\mathbf{edg}'_{aG}(y_1, \ldots, y_n)$ holds if and only if $\mathbf{lab}_G(x) = a$ and $\mathbf{vert}_G(x) = (y_1, \ldots, y_n)$ for some edge x in $\boldsymbol{E}_G$. Clearly, two simple hypergraphs G and G' are equal if and only if $|G|_1$ is equal (isomorphic) to $|G'|_1$. Hence, simple hypergraphs are unambiguously represented by these latter structures whereas arbitrary hypergraphs are not.

The representation of hypergraphs by logical structures makes it possible to express their properties by logical formulas. We shall say that a property P of the hypergraphs G of a class $\mathscr{C}$ can be *expressed by a logical formula φ via a representation $|G|$* if, for every G in $\mathscr{C}$, $P(G)$ holds if and only if $|G| \models \varphi$.

A property of hypergraphs is *i-definable* (where i is 1 or 2) if it is expressible by an MS formula, relative to the representation $|\text{-}|_i$. For example, the following formula expresses that a graph G represented by the structure $|G|_1$ is connected:

$$\forall x \forall y \forall X [x \in X \wedge \forall u \forall v ((u \in X \wedge \psi(u, v) \Rightarrow v \in X) \Rightarrow y \in X)],$$

where $\psi(u, v)$ is the disjunction of the formulas $\mathbf{edg}'_a(u, v) \vee \mathbf{edg}'_a(v, u)$ extended to all labels a (this formula expresses that u and v are the two ends of some edge). (All labels are assumed to be of rank 2.)

The structure $|G|_1$ is less expressive than $|G|_2$ for representing properties of a hypergraph G by MS formulas for the obvious reason that one cannot express in $|G|_1$ properties dealing with multiple edges. However, this is also the case if G is assumed to be simple. For instance, the existence of a Hamiltonian circuit in a simple graph is a 2-definable property that is not 1-definable. Some results comparing the expressive powers of MS formulas in the two cases are recalled in Section 4.

2. Monadic second-order definable transductions

We first define *transductions of relational structures*. Let R and Q be two finite ranked sets of relation symbols. Let W be a finite set of set variables, called here the set

of *parameters*. (It is not a loss of generality to assume that all parameters are set variables.) A (Q, R)-*definition scheme* is a tuple of formulas of the form

$$\Delta = (\varphi, \psi_1, \ldots, \psi_k, (\theta_w)_{w \in Q * k}),$$

where

$$k > 0,\ Q * k := \{(q, \vec{j}) \mid q \in Q, \vec{j} \in [k]^{\rho(q)}\},$$

$$\varphi \in \mathscr{L}(R, W),$$

$$\psi_i \in \mathscr{L}(R, W \cup \{x_1\}) \quad \text{for } i = 1, \ldots, k,$$

$$\theta_w \in \mathscr{L}(R, W \cup \{x_1, \ldots, x_{\rho(r)}\}), \quad \text{for } w = (q, \vec{j}) \in Q * k.$$

These formulas are intended to define a structure T in $\mathscr{S}(Q)$ from a structure S in $\mathscr{S}(R)$, and will be used in the following way. The formula φ defines the domain of the corresponding transduction, namely, T is defined only if φ holds true in S. Assuming this condition fulfilled, the formulas $\psi_1, \ldots, \psi_k$, define the domain of T as the disjoint union of the sets $D_1, \ldots, D_k$, where D_i is the set of elements in the domain of S that satisfy ψ_i. Finally, the formulas θ_w for $w = (q, \vec{j}), \vec{j} \in [k]^{\rho(q)}$ define the relation q_T. Here are the formal definitions.

Let $S \in \mathscr{S}(R)$, let γ be a W-assignment in S. A Q-structure T with domain $\boldsymbol{D}_T \subseteq \boldsymbol{D}_S \times [k]$ is *defined in* (S, γ) *by* Δ if

(i) $(S, \gamma) \models \varphi$,
(ii) $\boldsymbol{D}_T = \{(d, i) \mid d \in \boldsymbol{D}_S, i \in [k], (S, \gamma, d) \models \psi_i\}$
(iii) for each q in Q:

$$q_T = \{((d_1, i_1), \ldots, (d_t, i_t)) \in \boldsymbol{D}_T^t \mid (S, \gamma, d_1, \ldots, d_t) \models \theta_{(q, \vec{j})}\},$$

where $\vec{j} = (i_1, \ldots, i_t)$ and $t = \rho(q)$.

By $(S, \gamma, d_1, \ldots, d_t) \models \theta_{(q, \vec{j})}$, we mean $(S, \gamma') \models \theta_{(q, \vec{j})}$, where γ' is the assignment extending γ, such that $\gamma'(x_i) = d_i$ for all $i = 1, \ldots, t$; a similar convention is used for $(S, \gamma, d) \models \psi_i$).

Since T is associated in a unique way with S, γ and Δ whenever it is defined, i.e., whenever $(S, \gamma) \models \varphi$, we can use the functional notation $\mathbf{def}_\Delta(S, \gamma)$ for T.

The *transduction defined by* Δ is the relation $\mathbf{def}_\Delta := \{(S, T) \mid T = \mathbf{def}_\Delta(S, \gamma)$ for some W-assignment γ in $S\} \subseteq \mathscr{S}(R) \times \mathscr{S}(Q)$. A transduction $f \subseteq \mathscr{S}(R) \times \mathscr{S}(Q)$ is *definable* if it is equal to $\mathbf{def}_\Delta$ for some (Q, R)-definition scheme Δ. In the case where $W = \emptyset$, we say that f is *definable without parameters* (note that it is functional). We shall refer to the integer k by saying that $\mathbf{def}_\Delta$ is *k-copying*. In the special case where $k = 1$, we shall say that $\mathbf{def}_\Delta$ is *noncopying* and we can write more simply D as $(\varphi, \psi, (\theta_q)_{q \in Q})$. In this case:

$$\boldsymbol{D}_T = \{d \in \boldsymbol{D}_S \mid (S, \gamma, d) \models \psi\}$$

and for each q in Q:

$$q_T = \{(d_1, \ldots, d_t) \in \boldsymbol{D}_T^t \mid (S, \gamma, d_1, \ldots, d_t) \models \theta_q\},$$

where $t = \rho(q)$.

Before applying these definitions to general hypergraphs, we give three examples. Our first example is the transduction that associates with a graph the set of its connected components. We consider directed unlabelled graphs. Such a graph G is represented by the structure $|G|_2 = \langle \boldsymbol{D}_G, \mathbf{edg}_G \rangle$, where $\boldsymbol{D}_G = \boldsymbol{V}_G \cup \boldsymbol{E}_G$ and **edg** is a ternary relation symbol. (Since edges have no labels, we use a single relation **edg** instead of several relations $\mathbf{edg}_a$ as in the definition of Section 1.) The definition scheme Δ given below uses a parameter X. It is constructed in such a way that $\mathbf{def}_\Delta(|G|_2, \{x\})$ is the structure $|G'|_2$, where G' is the connected component of G containing the vertex x. We let $(\varphi, \psi, \theta_{\mathbf{edg}})$ be the noncopying definition scheme where

φ is a formula with free variable X expressing that X consists of a unique vertex,

ψ is a formula with free variables X and x expressing that x is a vertex linked by a path (where edges can be traversed in either direction) to the vertex in X, or that x is an edge, one end of which is linked by such a path to the vertex in X.

$\theta_{\mathbf{edg}}$ is the formula $\mathbf{edg}(x_1, x_2, x_3)$.

It is then straightforward to verify that Δ is as desired. It follows in particular from Theorem 4.1(3) that the set of connected components of a HR set of graphs is HR.

Our second example is the functional transduction that maps a word u in $\{a, b\}^+$ to the word u^3. (We denote by $\{a, b\}^+$ the set of nonempty words written with a and b.) In order to define it as a transduction of structures, we represent the words in $\{a, b\}^+$ as structures in the following way. If u has length n, then the associated structure $\|u\|$ is $\langle \{1, 2, \ldots, n\}, \mathbf{suc}, \boldsymbol{p}_a, \boldsymbol{p}_b \rangle$, where

the domain $\{1, 2, \ldots, n\}$ is the set of positions of letters in u,

$\boldsymbol{p}_a(i)$ holds if and only if a is the letter at ith position,

$\boldsymbol{p}_b(i)$ holds if and only if b is the letter at ith position,

$\mathbf{suc}(i, j)$ holds if and only if $j = i + 1$.

We let Δ be the 3-copying definition scheme without parameter $(\varphi, \psi_1, \psi_2, \psi_3, (\theta_{(\mathbf{suc}, i, j)})_{i, j = 1, 2, 3}, (\theta_{(\boldsymbol{p}_a, i)})_{i = 1, 2, 3}, (\theta_{(\boldsymbol{p}_b, i)})_{i = 1, 2, 3})$ such that

φ expresses that the input structure indeed represents a word in $\{a, b\}^+$,

ψ_1, ψ_2, ψ_3 are identical to the Boolean constant **true**,

$\theta_{(\mathbf{suc}, i, j)}(x_1, x_2)$ is $\mathbf{suc}(x_1, x_2)$ if $i = j$,

$\theta_{(\mathbf{suc}, i, j)}(x_1, x_2)$ expresses that x_1 is the last position and that x_2 is the first one if $i = 1$ and $j = 2$, or if $i = 2$ and $j = 3$,

$\theta_{(\mathbf{suc}, i, j)}(x_1, x_2)$ is the constant **false** otherwise,

$\theta_{(\boldsymbol{p}_a, i)}(x_1)$ is $\boldsymbol{p}_a(x_1)$ for $i = 1, 2, 3$,

$\theta_{(\boldsymbol{p}_b, i)}(x_1)$ is $\boldsymbol{p}_b(x_1)$ for $i = 1, 2, 3$.

We claim that $\mathbf{def}_\Delta(S) = T$ if and only if S is a structure of the form $\|u\|$ and T is the corresponding structure $\|u^3\|$. To take a simple example, if $S = \langle \{1, 2, 3, 4\}, \mathbf{suc}, p_a, p_b \rangle$ representing the word *abba*, then $\mathbf{def}_\Delta(S)$ is the structure.

$$T = \langle \{1_1, 2_1, 3_1, 4_1, 1_2, 2_2, 3_2, 4_2, 1_3, 2_3, 3_3, 4_3\}, \mathbf{suc}', p'_a, p'_b \rangle,$$

where we write i_j instead of (i, j), so that i_j is the jth copy of i for $j = 1, 2, 3$, and

$\mathbf{suc}'(i_j, k_m)$ holds if and only if k_m is the successor of i_j in the above enumeration of the domain of T,

$p'_a(i_j)$ holds if $i \in \{1, 4\}$, $j \in \{1, 2, 3\}$ and

$p'_b(i_j)$ holds otherwise.

This is an example of a definable transduction from words to words that is not rational. (This transduction will also be used as a counterexample in Proposition 3.4.)

Our last example is the product of a finite-state automaton $\mathscr{A}$ by a *fixed* finite-state automaton $\mathscr{B}$. A finite-state automaton is defined as a 5-tuple $\mathscr{A} = \langle X, Q, M, I, F \rangle$, where X is the input alphabet (here we take $X = \{a, b\}$), Q is the set of states, M is the transition relation which is a subset of $Q \times X \times Q$ because we consider nondeterministic automata without ε-transitions, I is the set of initial states and F is that of final states. The language it recognizes is denoted by $L(\mathscr{A})$. The automaton $\mathscr{A}$ is represented by the relational structure: $|\mathscr{A}| = \langle Q, \mathbf{trans}_a, \mathbf{trans}_b, \mathbf{initial}, \mathbf{final} \rangle$, where $\mathbf{trans}_a$ and $\mathbf{trans}_b$ are binary relation symbols, **initial** and **final** are unary relation symbols and

$\mathbf{trans}_a(p, q)$ holds if and only if $(p, a, q) \in M$,

$\mathbf{trans}_b(p, q)$ holds if and only if $(p, b, q) \in M$,

$\mathbf{initial}(p)$ holds if and only if $p \in I$,

$\mathbf{final}(p)$ holds if and only if $p \in F$.

Let $\mathscr{B} = \langle X, Q', M', I', F' \rangle$ be a similar automaton, and $\mathscr{A} \times \mathscr{B} = \langle X, Q \times Q', M'', I \times I', F \times F' \rangle$ be the product automaton intended to define the language $L(\mathscr{A}) \cap L(\mathscr{B})$. We let Q' be $\{1, \ldots, k\}$ (let us recall that $\mathscr{B}$ is fixed). We let Δ be the k-copying definition scheme $(\varphi, \psi_1, \ldots, \psi_k, (\theta_w)_{w \in R^* k})$, where $R = \{\mathbf{trans}_a, \mathbf{trans}_b, \mathbf{initial}, \mathbf{final}\}$ and:

φ is the constant **true** (every structure in $\mathscr{S}(R)$ represents an automaton that may have inaccessible states and useless transitions),

$\psi_1, \ldots, \psi_k$ are the constant **true**,

$\theta_{(\mathbf{trans}_a, i, j)}(x_1, x_2)$ is the formula $\mathbf{trans}_a(x_1, x_2)$ if (i, a, j) is a transition of $\mathscr{B}$ and is the constant **false** otherwise,

$\theta_{(\mathbf{trans}_b, i, j)}$ is defined similarly,

$\theta_{(\mathbf{initial}, i)}(x_1)$ is the formula $\mathbf{initial}(x_1)$ if i is an initial state of $\mathscr{B}$ and is **false** otherwise,

$\theta_{(\mathbf{final}, i)}(x_1)$ is defined similarly.

It is not hard to check that $|\mathscr{A} \times \mathscr{B}| = \mathbf{def}_\Delta(|\mathscr{A}|)$. Note that the language defined by an automaton $\mathscr{A}$ is nonempty if and only if there exists a path in $\mathscr{A}$ from some initial state to some final state. This latter property is expressible in monadic second-order logic. Hence, it follows from Proposition 3.2(1) that, for a fixed rational language K, the set of structures representing an automata $\mathscr{A}$ such that $L(\mathscr{A}) \cap K$ is nonempty is definable.

The definitions concerning definable transductions of structures apply to hypergraphs via their representation by relational structures as explained above. However, since we have two representations of hypergraphs by logical structures, we must be more precise. We say that a *hypergraph transduction*, i.e., a binary relation f on hypergraphs is (i,j)-*definable*, where i and j belong to $\{1,2\}$ if and only if the transduction of structures $\{(|G|_i, |G'|_j) | (G, G') \in f\}$ is definable. We shall also use transductions from trees to hypergraphs. Since a tree t is a graph, it can be represented by either $|t|_1$ or $|t|_2$. However, both structures are equally powerful for expressing monadic second-order properties of trees, and definable transductions from trees to trees and from trees to graphs. (This follows from Theorem 5.1). When we specify a transduction involving trees (or words which are special trees) as input or output we shall use the symbol $*$ instead of the integers 1 and 2, in order to recall that the choice of representation is not important in these cases. We shall use *definable* for $(*,*)$-*definable*.

Here are a few facts concerning definable transductions of structures.

Fact 2.1. *If f is a definable transduction, there exists an integer k such that,* $\mathbf{Card}(\boldsymbol{D}_T) \leqslant k.\ \mathbf{Card}(\boldsymbol{D}_S)$ *whenever T belongs to $f(S)$.*

Fact 2.2. *The domain of a definable transduction is definable.*

Proof. Let Δ be a definition scheme as in the general definition with $W = \{X_1, \ldots, X_n\}$. Then $\mathbf{Dom}(\mathbf{def}_\Delta) = \{S \mid S \models \exists X_1, \ldots, \exists X_n \varphi\}$. □

Proposition 2.3 states that every k-copying definable transduction is the composition of a noncopying transduction and a k-copying "standard" one which we now define. For $k \geqslant 1$, we let $\boldsymbol{Q}_k := \{s_i \mid 1 \leqslant i \leqslant k\} \cup \{\mathbf{br}\}$, where the s_i's are unary relation symbols and **br** is a binary one. For every S in $\mathscr{S}(R)$, we let $\mathbf{cop}_k(S)$ be the $(R \cup \boldsymbol{Q}_k)$-structure T constructed as follows:

$$\boldsymbol{D}_T = \{(d, i) \mid d \in \boldsymbol{D}_S,\ i \in [k]\}$$

for each r in R of arity t:

$$r_T = \{((d_1, i), \ldots, (d_t, i)) \mid (d_1, \ldots, d_t) \in r_S,\ i \in [k]\},$$

$$s_{iT} := \{(d, i) \mid d \in \boldsymbol{D}_S\},$$

$$\mathbf{br}_T := \{((d, i), (d, j)) \mid d \in \boldsymbol{D}_S,\ 1 \leqslant i, j \leqslant k\}.$$

The unary relation $s_{iT}(x)$ holds if and only if x is "an ith son" (the ith son of d such that $x = (d, i)$); the binary relation $\mathbf{br}_T(x, y)$ holds if and only if x and y are "brothers", i.e., are two "sons" of the same d, with x possibly equal to y.

Proposition 2.3. *A transduction is definable if and only if it is the composition $f \circ \mathbf{cop}_k$ of $\mathbf{cop}_k$ for some k and of some definable noncopying transduction f.*

Proof. It is clear that $\mathbf{cop}_k$ is definable and k-copying. If f is a definable, then Proposition 3.2 yields that $f \circ \mathbf{cop}_k$ is definable and k-copying.

Let us now consider g defined by $\Delta = (\varphi, \psi_1, \ldots, \psi_k, (\theta_w)_{w \in Q * k})$ with set of parameters $W = \{X_1, \ldots, X_n\}$. We aim to construct $\Delta' = (\varphi', \psi', (\theta'_q)_{q \in Q})$ such that $\mathbf{def}_\Delta = \mathbf{def}_{\Delta'} \circ \mathbf{cop}_k$. We shall describe the intended meaning of the formulas that form Δ' (rather than writing them explicitly). Let S be an R-structure, and T be any structure isomorphic to $\mathbf{cop}_k(S)$. We let T' be the R-structure defined as the restriction of T to the set of elements of T that satisfy s_1. Hence, T' is isomorphic to S.

The formula φ' (with free variables in W) expresses (in T) that $X_1, \ldots, X_n$ are subsets of the domain $\boldsymbol{D}_{T'}$ of T' and that $\varphi(X_1, \ldots, X_n)$ holds in T'; the formula ψ' with free variables in $\{X_1, \ldots, X_n\} \cup \{x\}$ expresses (in T) that $s_i(x)$ holds for some i in $[k]$ and that there exists y in $\boldsymbol{D}_{T'}$ such that $\mathbf{br}(x, y)$ holds in T and $\psi_i(X_1, \ldots, X_n, y)$ holds in T'; for each q in Q of arity t, the formula θ'_q has its free variables in $\{X_1, \ldots, X_n\} \cup \{x_1, \ldots, x_t\}$ and expresses (in T) that there exist $i_1, \ldots, i_t$ in $[k]$ such that $s_{i_1}(x_1), \ldots, s_{i_t}(x_t)$ hold and there exist $y_1, \ldots, y_t$ in $\boldsymbol{D}_{T'}$ such that $\mathbf{br}(x_1, y_1), \ldots, \mathbf{br}(x_t, y_t)$ hold in T and $\theta_w(X_1, \ldots, X_n, y_1, \ldots, y_t)$ holds in T', where $w = (q, (i_1, \ldots, i_t))$. It is easy to verify that if $\Delta' = (\varphi', \psi', (\theta'_q)_{q \in Q})$, where $\varphi', \psi', \theta'_q$ are as above, then $\mathbf{def}_\Delta = \mathbf{def}_{\Delta'} \circ \mathbf{cop}_k$. □

The next three propositions list examples of transductions of words and trees that are definable.

Proposition 2.4. *The following mappings are definable transductions*: (1) *word homomorphisms*, (2) *inverse nonerasing word homomorphisms*, (3) *gsm mappings*, (4) *the mirror-image mapping on words*, (5) *the mapping* $\lambda u.[u^n]$ (*where u is a word and n a fixed integer*), (6) *the mapping* **yield** *that maps a derivation tree relative to a fixed context-free grammar to the generated word*, (7) *linear root-to-frontier or frontier-to-root tree transductions.*

Proof. The proofs are easy to do. Let us recall that a *gsm mapping* is a transduction from words to words defined by a *generalized sequential machine*, i.e., a (possibly nondeterministic) transducer that reads at least one input symbol on each move. A special case of (5) has been constructed in detail in our second example before Fact 2.1 (See [29] or the survey by Raoult [36] for tree transductions). □

Fact 2.1 which limits the sizes of the output structures, shows that certain transductions are *not definable*. This is the case of inverse erasing word homomorphisms and of ground tree transducers except in degenerated cases (see Dauchet et al. [18] or [36] on ground tree transducers).

Proposition 2.5. *The transductions that associate with a graph G*: (1) *its spanning forests*, (2) *its connected components*, (3) *its subgraphs satisfying some fixed 2-definable property*, (4) *its maximal subgraphs satisfying some fixed 2-definable property* (*maximal for subgraph inclusion*), (5) *the graph consisting of the union of two disjoint copies of G*, (6) *its*

minors, are all (2, 2)*-definable. The mapping associating with a graph its line graph is* (2, 1)*-definable but not* (2, 2)*-definable.*

We recall that the *line graph* of a graph G has E_G as set of vertices and has undirected edges between any two vertices representing edges sharing a vertex (in G).

Proof. Assertions (1)–(6) are easy consequences of the existence of an MS formula expressing that two given vertices are linked by some path; see [7, 8]. Assertion (6) is proved in [12]. The last assertion is an easy consequence of the definition of a line graph. The non-(2,2)-definability is proved in [15]. □

3. Properties of definable transductions of relational structures

The following proposition is the basic fact behind the notion of semantic interpretation [35]. It says that if $S = \mathbf{def}_\Delta(T, \mu)$ i.e., if S is defined in (T, μ) by Δ, then the monadic second-order properties of S can be expressed as monadic second-order properties of (T, μ).

Let $\Delta = (\varphi, \psi_1, \ldots, \psi_k, (\theta_w)_{w \in Q^* k})$ be a (Q, R)-definition scheme, written with a set of parameters W. Let V be a set of set variables disjoint from W. For every variable X in V, for every $i = 1, \ldots, k$, we let X_i be a new variable. We let $V' := \{X_i / X \in V, i = 1, \ldots, k\}$. For every mapping $\eta: V' \to \mathscr{P}(D)$, we let $\eta \hat{} k: V \to \mathscr{P}(D \times [k])$ be defined by: $\eta \hat{} k(X) = \eta(X_1) \times \{1\} \cup \cdots \cup \eta(X_k) \times \{k\}$. With these notations we can state the following proposition.

Proposition 3.1. *For every formula* β *in* $\mathscr{L}(Q, V)$*, one can construct a formula* β' *in* $\mathscr{L}(R, V' \cup W)$ *such that, for every* T *in* $\mathscr{S}(R)$*, for every assignment* $\mu: W \to T$*, for every assignment* $\eta: V' \to T$*, we have*

> $\mathbf{def}_\Delta(T, \mu)$ *is defined (if it is, we denote it by* S*),* $\eta \hat{} k$ *is a* V*-assignment in* S*, and* $(S, \eta \hat{} k) \models \beta$

if and only if $(T, \eta \cup \mu) \models \beta'$.

Note that, even if S is well-defined, the mapping $\eta \hat{} k$ is not necessarily a V-assignment in S, because $\eta \hat{} k(X)$ is not necessarily a subset of the domain of S which is a possibly proper subset of $D \times [k]$.

Proof (*sketch*). Let us first consider the case where $\mathbf{def}_\Delta$ is noncopying. In order to transform β into β', one replaces every atomic formula $q(u_1, \ldots, u_n)$ by the formula $\theta_q(u_1, \ldots, u_n)$ which defines it in terms of the relations of T. One also restricts quantifications to the domain of S, i.e. one replaces $\exists x[\mu]$ by $\exists x[\psi(x) \wedge \mu]$ and $\exists X[\mu]$ by $\exists X[\forall x\{x \in X \Rightarrow \psi(x)\} \wedge \mu]$. In the case where $k > 1$, one replaces $\exists X[\mu]$ by a formula of the form $\exists X_1, \exists X_2, \ldots, \exists X_k[\mu']$, where μ' is an appropriate

transformation of μ based on the fact that $X = X_1 \times \{1\} \cup \cdots \cup X_k \times \{k\}$. The reader will find a complete construction in [10, Proposition 2.5, p. 166]. □

From this proposition, we get easily the following proposition.

Proposition 3.2. (1) *The inverse image of a definable set of structures under a definable transduction is definable.*

(2) *The composition of two definable transductions is definable.*

Proof. (1) Let $L \subseteq \mathscr{S}(Q)$ be defined by a closed formula β and $\mathbf{def}_\Delta$ be a transduction as in Proposition 3.1. Then $\mathbf{def}_\Delta^{-1}(L) \subseteq \mathscr{S}(R)$, is defined by the formula $\exists Y_1, \ldots, \exists Y_n[\beta']$, where $Y_1, \ldots, Y_n$ are the parameters and β' is constructed from β as in Proposition 3.1.

(2) Let $\Delta = (\varphi, \psi_1, \ldots, \psi_k, (\theta_w)_{w \in Q * k})$ be a k-copying definition scheme and $\Delta' = (\varphi', \psi'_1, \ldots, \psi'_{k'}, (\theta'_w)_{w \in P * k'})$ be k'-copying such that $\mathbf{def}_\Delta$ is a transduction from $\mathscr{S}(R)$ to $\mathscr{S}(Q)$ and $\mathbf{def}_{\Delta'}$ is a transduction from $\mathscr{S}(Q)$ to $\mathscr{S}(P)$. Let f be the transduction $\mathbf{def}_{\Delta'} \circ \mathbf{def}_\Delta$ from $\mathscr{S}(R)$ to $\mathscr{S}(P)$: we shall construct a definition scheme Δ'' for it. Just to simplify the notation we shall assume that the parameters of Δ are Y and Y' and that those of Δ' are Z and Z'. We shall also assume that the relations of P are all binary. The general case will be an obvious extension.

In order to describe Δ'' we shall denote by T an R-structure, we shall denote by S the Q-structure $\mathbf{def}_\Delta(T, Y, Y')$, where Y and Y' are subsets of $\boldsymbol{D}_T$ and we denote by U the P-structure $\mathbf{def}_{\Delta'}(S, Z, Z')$, where Z and Z' are subsets of $\boldsymbol{D}_S$. Hence $\boldsymbol{D}_S$ is a subset of $\boldsymbol{D}_T \times [k]$, and $\boldsymbol{D}_U$ is a subset of $\boldsymbol{D}_T \times [k] \times [k']$ which is canonically isomorphic to a subset of $\boldsymbol{D}_T \times [kk']$. Hence Δ'' will be kk'-copying.

The parameters Z and Z' represent sets of the respective forms $Z = Z_1 \times \{1\} \cup \cdots \cup Z_k \times \{k\}$ and $Z' = Z'_1 \times \{1\} \cup \cdots \cup Z'_k \times \{k\}$. Hence, the definition scheme Δ'' will be written in terms of parameters $Y, Y', Z_1, \ldots, Z_k, Z'_1, \ldots, Z'_k$. It will be of the form

$$(\varphi'', (\psi''_{i,j})_{i \in [k]\, j \in [k']}, (\theta''_{p,(i,i'),(j,j')})_{p \in P,\, i,i' \in [k],\, j,j' \in [k']}),$$

so that the domain of $\boldsymbol{D}_U$ will be handled as a subset of $\boldsymbol{D}_T \times [k] \times [k']$ and not of $\boldsymbol{D}_\tau \times [kk']$. The formulas forming Δ'' will be obtained from those forming Δ' by the transformation of Proposition 3.1. We first consider φ'' which should express that

$$\mathbf{def}_\Delta(T, Y, Y') \text{ is defined, i.e., that } (T, Y, Y') \models \varphi,$$

and that if $Z = Z_1 \times \{1\} \cup \cdots \cup Z_k \times \{k\}$ and $Z' = Z'_1 \times \{1\} \cup \cdots \cup Z'_k \times \{k\}$, then

$$\mathbf{def}_{\Delta'}(S, Z, Z') \text{ is defined, i.e., that } (S, Z, Z') \models \varphi',$$

which is equivalent to:

$$(T, Y, Y', Z_1, \ldots, Z_k, Z'_1, \ldots, Z'_k) \models \tau,$$

where τ is obtained from φ' by the transformation of Proposition 3.1. Hence, φ'' is the conjunction of φ and τ. We omit the constructions of the other formulas because they are quite similar. □

We could define more powerful transductions by which a structure S would be constructed "inside" $T \times T$ instead of "inside" a structure formed of a fixed number of disjoint copies of T linked as explained in the definition of $\mathbf{cop}_k$ used in Proposition 2.3. However, with this variant, one could construct a *second-order* formula β' as in Proposition 3.1 (with quantifications on binary relations), but not a *monadic* second-order one (at least in general). We wish to avoid (full) second-order logic because most constructions and decidability results (like those of [8]) break down. In the third example given before Fact 2.1, we have shown that the transduction associating the automaton $\mathscr{A} \times \mathscr{B}$ with an automaton $\mathscr{A}$ is definable (via the chosen representation of finite-state automata by relational structures) *for fixed* $\mathscr{B}$.

Here are some other closure properties of the class of definable transductions.

Proposition 3.3. *The union of two definable transductions is a definable transduction. So is the intersection of a definable transduction with a transduction of the form* $A \times B$, *where* A *and* B *are definable sets.*

Proof. See [9] for the first assertion and [10] for the second. □

Here are now some negative closure properties.

Proposition 3.4. (1) *The image of a definable set under a definable transduction is a set that is not definable in general.*

(2) *The inverse of a definable transduction is a transduction that is not definable in general.*

(3) *The intersection of two definable transductions is a transduction that is not definable in general.*

Proof. (1) The transduction of words that maps $a^n b$ to $a^n b a^n b a^n b$ for $n \geqslant 0$ is definable (this follows from Proposition 3.3 and the second example given before Fact 2.1). The image of the definable language a^*b is a language that is not regular (and even not context-free), hence not definable by the basic result of Büchi and Elgot ([38, Theorem 3.2]) recalled in the introduction.

(2) The inverse of this transduction is not definable since, if it would be, its domain would be definable (by Fact 2.2), hence regular, which is not the case.

(3) The intersection of the definable transductions of words that map $a^n b^m$ to c^n, and $a^n b^m$ to c^m is the one that maps $a^n b^n$ to c^n. It is not definable because its domain is not a definable language. □

4. Definable transductions and context-free graph grammars

There are two classes of context-free graph (and hypergraph) grammars, the class of HR (Hyperedge Replacement) and the class of VR (Vertex Replacement) grammars.

Both kinds of grammars can be described by rewriting rules: *HR grammars* (which generate the *HR sets* of graphs and hypergraphs) are based on the replacement in a hypergraph of a hyperedge by a hypergraph: the labels of edges play the role of terminal and nonterminal symbols in context-free grammars; the *VR grammars* can be described by a complicated rewriting of vertices. Grammars of both types are *context-free* in the sense of [6]. This means in particular that they are *confluent*, i.e., that independent derivation steps can be permuted and that the equivalence classes of derivation sequences w.r.t. permutations of independent steps can be characterized by *derivation trees* (see [6] for a formal definition). Another characteristic property of context-free graph grammars is that the generated sets can be characterized as forming the least solutions of systems of equations canonically associated with the considered grammar. Grammars of both types can actually be defined in an easier way as systems of fixed-point equations, written with set union and appropriate operations on hypergraphs that we need not recall here.

We refer the reader to [2, 7, 10, 31, 32] for definitions and basic properties of HR grammars. The relations between HR grammars and definable transductions are collected in the following theorem.

Theorem 4.1. (1) *The mapping* **yield** *that associates with a derivation tree of a (fixed) HR grammar the generated hypergraph is* $(*, 2)$*-definable.*

(2) *A set of hypergraphs is HR if and only if it is the image of a recognizable set of finite trees under a* $(*, 2)$*-definable transduction.*

(3) *The class of HR sets of hypergraphs is closed under* $(2, 2)$*-definable transductions.*

Proof. (1) is proved in [10]; the "if" part of (2) is a difficult theorem established in [15], whereas the "only if" part follows immediately from (1); (3) follows immediately from (2) since the composition of a $(*, 2)$-definable transduction from trees to hypergraphs and a $(2, 2)$-definable transduction from hypergraphs to hypergraphs is $(*, 2)$-definable. □

Remark. In Condition (2), one can replace *a recognizable set of finite trees* by *the set* $\mathbb{B}$ *of finite binary unlabelled trees*, because every recognizable set of finite trees is the image of $\mathbb{B}$ under some $(*, *)$-definable transduction [15, Lemma 2.4].

Let us say that a HR grammar is *linear* if each right-hand side of a production rule has at most one nonterminal, and that a set of hypergraphs is *linear HR* if there exists a linear HR grammar generating it. From the constructions establishing Theorem 4.1, we get that a set of hypergraphs is linear HR if and only if it is the image of a recognizable language under a $(*, 2)$-definable transduction, and that the class LIN-HR of linear HR sets of hypergraphs is closed under $(2, 2)$-definable transductions.

We now come to the more complex class of VR sets of *simple* graphs. A few words of history are in order because the definition of this class has emerged from a sequence of

papers. It comes out of the NLC grammars introduced by Janssens and Rozenberg [33]; (NLC means "node labelled controlled"). Not all NLC grammars are context-free because they are not always confluent (independent derivation steps cannot in general be permuted). The BNLC ("boundary" NLC) grammars form a restriction of the general case and are confluent [37]. Whereas NLC grammars generate undirected graphs with unlabelled edges, the more general edNCE grammars can generate directed labelled graphs, and edge labels can be modified during the rewriting. Only the confluent ones (the C-edNCE grammars) are context-free. Other types of grammars, the S-HH ("separated handle hypergraph") grammars, that are always context-free, generate exactly the same sets of graphs as the C-edNCE grammars and give more easily equivalent systems of fixed-point equations [16]. See also [26] for more details and references. A grammar independent characterization of C-edNCE sets of graphs, slightly weaker than Theorem 4.2(2) is given by Engelfriet in [26].

A *VR set of graphs* is a set of directed or undirected graphs generated by either a C-edNCE or a S-HH grammar, or defined as a component of the least solution of a systems of fixed-point equations written with certain appropriate operations (see [16]). Actually, the simplest way to define them is by systems of equations (see also [13, 14] for these systems). Since these grammars are confluent, their derivation sequences can be represented by derivation trees [6].

Theorem 4.2. (1) *The mapping* **yield** *that maps a derivation tree of a fixed C-edNCE or S-HH grammar to the generated graph is* $(*, 1)$*-definable.*

(2) *A set of simple graphs is VR if and only if it is the image of a recognizable set of finite trees under a* $(*, 1)$*-definable transduction.*

(3) *The class of VR sets of graphs is closed under* (1,1)*-definable transductions.*

Proof. (1) See [13] or [22]. (2) The "if" part is proved in [22] for a restricted notion of transduction; the full form is Theorem 3.2 of [15], an alternative proof is given in [13]. (3) follows from (2) like do the corresponding assertions of Theorem 4.1. □

As in Theorem 4.1, and for the very same reason, we can replace in Theorem 4.2(2) *a recognizable set of finite trees* by *the set* $\mathbb{B}$. Since a $(*, 2)$-definable transduction is $(*, 1)$-definable, it follows also from Theorems 4.1(2) and 4.2(2) that every HR set of simple graphs is VR, which gives another proof of a result known from [16]. As another consequence we get, using Proposition 2.5, that the set of line graphs of the graphs of a HR set is VR.

The *linear* VR sets of graphs, i.e., those defined by *linear* C-edNCE or S-HH grammars (with at most one nonterminal in each right-hand side of a production) can be characterized as the images of recognizable languages under $(*, 1)$-definable transductions, and their class (let us denote it by LIN-VR) is closed under $(1, 1)$-definable transductions.

As another application of Theorem 4.2, we shall establish that the set of *chordal graphs* is not VR. A chordal graph is a simple undirected graph in which every cycle of

length at least 4 has a chord (see [28]). These graphs can also be described as "tree-shaped" combinations of cliques. Considering also the fact that the set of all cliques is VR, one might think that the set of chordal graphs is VR. But this is not the case.

Proposition 4.3. *The set of chordal graphs is not VR.*

Proof. In the following proof, all graphs are simple loop-free and undirected. Let L be the set of graphs constructed in the following way. They consist of a clique K having at least four vertices augmented with a possibly empty set V of additional vertices and with edges such that every vertex in V is linked to two distinct vertices of K.

For every G in L, we let $H:=f(G)$ be the graph constructed as follows: its vertices are those of the clique K upon which G is constructed (they are actually the vertices of G of degree more than two); two such vertices are linked in H if and only if they are both linked in G to a same vertex v of V (i.e., a vertex of degree two). It follows from this definition that the transduction

$$\{(|G|_1, |f(G)|_1) \mid G \in L\}$$

is definable. Since every graph in L is chordal and every graph with at least 4 vertices is $f(G)$ for some G in L, we obtain that the image under f of the set of chordal graphs is the set of all graphs having at least four vertices. This latter set is not VR (because every VR set contains at most finitely many square grids by, e.g., [15]). It follows from Theorem 4.2(3) that the set of chordal graphs is not VR either. □

What about VR sets of simple *hypergraphs*? We *define* them as the images of recognizable sets of finite trees (equivalently of the set of finite binary trees $\mathbb{B}$) under $(*,1)$-definable transductions. Courcelle [13, Theorem 4.6] gives a characterization in terms of systems of fixed point equations built with appropriate operations. A decidable characterization of the VR sets of hypergraphs that are not HR is given in [14] (and its proof uses definable transductions). Let us finally mention that S-HH grammars generate certain VR sets of simple hypergraphs, but not all of them [16].

5. Comparing representations of graphs and hypergraphs by relational structures

Transductions of structures can be used for the following two purposes: (1) for comparing several representations of the same object by relational structures, and (2) for encodings of hypergraphs by graphs, or more generally, of combinatorial objects by others. We illustrate successively these two uses.

We have defined two relational structures, $|G|_1$ and $|G|_2$ able to represent unambiguously simple hypergraphs. For every set $\mathscr{C}$ of simple hypergraphs, we let $\mathbf{tr}(\mathscr{C})$ be the transduction: $\mathbf{tr}(\mathscr{C}) = \{(|G|_1, |G|_2) \mid G \in \mathscr{C}\}$. It is functional since we are dealing with simple hypergraphs and since any two isomorphic structures are considered as

equal. If $\mathscr{C}$ is the set of all (finite) simple graphs, then $\mathbf{tr}(\mathscr{C})$ is not definable (otherwise, by Theorems 4.1 and 4.2, the classes of HR and VR sets of graphs would be the same which is not the case).

The *tree-width* of a graph is an integer that characterizes how close it is to be a tree: forests have tree-width 1, the tree-width of a clique with n vertices is n. The graphs generated by a HR-grammar have a tree-width bounded by some integer computable from the grammar (see [7, 11, 12]).

Theorem 5.1. *Let k be an integer. Let $\mathscr{C}$ be either the set of simple graphs of degree at most k or that of simple graphs of tree-width at most k or that of simple hypergraphs of tree-width at most k. The transduction $\mathbf{tr}(\mathscr{C})$ is definable.*

The proof is given in [9] and [15, Lemma 3.8] for graphs, and in [14] for hypergraphs. This theorem has consequences relevant to the comparison between HR and VR sets of graphs.

Corollary 5.2. *If a VR set of graphs has bounded degree, bounded tree-width, or is a subset of some HR set, then it is HR. If a VR set of hypergraphs has bounded tree-width, or is a subset of some HR set, then it is HR.*

Proof. Immediate application of Theorems 4.1, 4.2, and the fact that a HR set of hypergraphs has bounded tree-width. The case of sets of graphs of bounded degree is also known from [24, 4]. □

We now consider encodings of hypergraphs by graphs, along the lines of [25]. For H in $\mathbf{HG}(A)$, we denote by $\mathbf{gra}(H)$ the graph G such that

$$V_G = V_H \cup E_H \text{ (recall that } V_H \cap E_H = \emptyset),$$

$$E_G = \{(e, i, v) \mid e \in E_H, v \in V_H, v = \mathbf{vert}_H(e, i)\}$$
$$\cup \{(e, a) \mid e \in E_H, \mathbf{lab}_H(e) = a\} \cup \{(v, *) \mid v \in V_H\},$$

$$\mathbf{vert}_G((e, i, v)) = (e, v) \text{ and } \mathbf{lab}_G((e, i, v)) = i,$$

$$\mathbf{vert}_G((e, a)) = (e) \text{ and } \mathbf{lab}_G((e, a)) = a, \text{ and,}$$

$$\mathbf{vert}_G((v, *)) = (v) \text{ and } \mathbf{lab}_G((v, *)) = *.$$

Thus, $\mathbf{gra}(H)$ belongs to $\mathbf{HG}(A \cup [m] \cup \{*\})$, where m is the maximum rank of an element of A. In this graph, the vertices represent the vertices of H as well as the edges of H. If $\mathbf{vert}_H(e) = (v_1, \ldots, v_n)$ then, in $\mathbf{gra}(H)$, there is an i-labelled edge from the vertex representing e to the one representing v_i. If $\mathbf{lab}_H(e) = b$, then the vertex representing e is incident with an edge of rank 1 having the label b in $\mathbf{gra}(H)$. The vertices of G representing vertices of H are incident with an edge of rank 1 having the "new" label $*$. Note that the mapping **gra** is injective.

The following result from [15] is a generalization of [25, Theorem 1] and the given proof is independent of the original one. It shows that the C-edNCE grammars generate the same sets of hypergraphs (via the coding **gra**) as the HR grammars.

Theorem 5.3. (1) *The transduction* **gra** *is injective and* $(2,2)$*-definable; the transduction* $\mathbf{gra}^{-1}$ *is* $(1,2)$*-definable.*

(2) *For every subset* L *of* $\mathbf{HG}(A)$*, the set* $\mathbf{gra}(L)$ *is VR if and only if it is HR if and only if the set* L *is HR.*

Proof. We have already observed that **gra** is injective; it is easy to verify that it is $(2,2)$-definable. Hence, if L is HR, then the set $\mathbf{gra}(L)$ is HR, hence also VR. Conversely, the transduction $\mathbf{gra}^{-1}$ is $(1,2)$-definable. Hence, by Theorems 4.1 and 4.2, $\mathbf{gra}^{-1}(L')$ is HR if L' is VR. Hence, if $\mathbf{gra}(L)$ is VR, the set $L=\mathbf{gra}^{-1}(\mathbf{gra}(L))$ (because **gra** is injective) is HR. □

6. Definable transductions of words and trees

We review some relations between definable transductions and classical notions in language theory. The following proposition shows that the class of definable transductions and that of rational transductions are incomparable.

Proposition 6.1. *A rational transduction is definable if and only if the image of every word is a finite language.*

Proof. The "only if" part follows from Fact 2.1. Conversely, a rational transduction such that every word has a finite image is of the form $\lambda u.[h(k^{-1}(u)\cap K)]$, where K is a recognizable language, h and k are homomorphisms with k nonerasing (i.e., the image of a letter is not the empty word): see [3, Exercise 7.2, p. 87]. It is thus definable by Propositions 2.4, 3.2 and 3.3. □

We have observed that **yield**, the transduction from a derivation tree relative to some fixed context-free grammar to the generated word, is definable. Its inverse is not definable in general even if the grammar is unambiguous (otherwise the set of Polish prefix notations of terms over a finite ranked alphabet would be the domain of a definable transduction and would be definable whence recognizable which is not the case).

For some (but not all) HR grammars Γ, there exists a $(2,*)$-definable transduction associating with every hypergraph G generated by Γ one of its derivation tree relative to Γ. In other words, for these grammars, the definable transduction **yield** has a definable inverse (see [10]). However, these grammars, analogous to left-linear context-free (word) grammars, are less powerful than the general HR grammars.

The example used in the proof of Proposition 3.4 shows that the image of a context-free language under a definable transduction is not context-free. The following is more precise.

Theorem 6.2. (1) *The following classes of languages are identical*:

(1.1) *the sets of words defined by HR grammars, or, equivalently, by VR grammars,*

(1.2) *the images of HR (resp. VR) sets of hypergraphs by* $(2,*)$*-definable (resp.* $(1,*)$*-definable) transductions from hypergraphs to words.*

(2) *The following classes of languages are identical*:

(2.1) *the images of regular languages under definable transductions from words to words,*

(2.2) *the languages in LIN-HR, or equivalently, in LIN-VR,*

(2.3) *the images of LIN-HR (resp. LIN-VR) sets of hypergraphs by* $(2,*)$*-definable (resp.* $(1,*)$*-definable) transductions, from hypergraphs to words.*

Proof. That (1.1)=(1.2) follows from Theorems 4.1(3) and 4.2(3); the other equalities follow from previous remarks. □

The main result of [23] characterizes the HR (or VR) languages as the output languages of certain transducers from trees to words (namely the *deterministic tree-walking tree-to-word transducers*). Another result of the same paper characterizes the languages in LIN-HR (equivalently, in LIN-VR) as the output languages of *deterministic 2-way gsm mappings.*

Here are some open questions: What is the class of images of context-free languages under definable (word-to-word) transductions? It is in between the two classes considered in assertions (1) and (2) of Theorem 6.2, but is it *strictly in between*? It is closely related, perhaps identical, to the class of images of context-free languages under deterministic 2-way gsm mappings. This later class has been considered in [27] where it is proved (Corollary 4.10) that it equals the class of *context-free controlled ETOL systems of finite index*, also considered in [34]. It would also be interesting to have "machine" characterizations of definable word-to-word transductions (in terms of 2-way generalized sequential machines or of related devices) and of definable tree-to-word transductions.

7. Parikh's theorem and definable transductions from graphs to commutative words

Let $A=\{a_1,\dots,a_n\}$ be an alphabet. A *commutative word* over A, say w, can be identified with the n-tuple of numbers of occurrences of letters $a_1,\dots,a_n$ in w or with the edgeless, vertex labelled graph with one vertex labelled by a_i for each occurrence of a_i in w, for each $i=1,\dots,n$. We shall denote by $A^{\S}$ the set of such commutative words, the set of tuples of nonnegative integers and the set of such graphs which are all canonically isomorphic.

In Theorem 6.2, we have characterized the images of HR and VR sets of hypergraphs under definable transductions from hypergraphs to words. We do the same here for their images under definable transductions from hypergraphs to commutative words.

A subset of $A^{\$}$ is *semilinear* if it is semilinear in the usual sense when considered as a set of tuples of numbers, i.e., if it is a finite union of sets of the form $\{\vec{w}+\lambda_1\vec{z}_1+\cdots+\lambda_k\vec{z}_k \mid \lambda_1,\ldots,\lambda_k\in\mathbb{N}\}$, where $\vec{w},\vec{z}_1,\ldots,\vec{z}_k\in\mathbb{N}^n$.

From [27, Corollary 3.2.7] saying that the commutative images of the output languages of deterministic tree-walking tree-to-word transducers are semilinear sets and the main result of [23], we obtain the following result.

Theorem 7.1. *The following classes of sets of commutative words are identical:*

(1) *the class of semilinear sets,*

(2) *the class of HR (equivalently of VR) sets of commutative words,*

(3) *the images of HR (resp. of VR) sets of hypergraphs under* $(2,*)$*-definable (resp. under* $(1,*)$*-definable) transductions from hypergraphs to commutative words.*

A proof that does not rest on the constructions of [27] can be given with the help of the following lemma of independent interest. Let φ be a formula in $\mathscr{L}(R,\{X_1,\ldots,X_k\})$ for some ranked set R of relational symbols. Let S be an R-relational structure. We define

$$\mathbf{sat}(\varphi,S):=\{(X_1,\ldots,X_k)\mid X_1,\ldots,X_k\subseteq \boldsymbol{D}_S,(S,X_1,\ldots,X_k)\models\varphi\}.$$

With φ and an $n\times k$ matrix B of nonnegative integers, we can form the transduction $f_{\varphi,B}$ from R-structures to $A^{\$}$ (identified with $\mathbb{N}^n$) such that

$$f_{\varphi,B}(S):=\{(\mathbf{Card}(X_1),\ldots,\mathbf{Card}(X_k)).B\mid(X_1,\ldots,X_k)\in\mathbf{sat}(\varphi,S)\}.$$

Lemma 7.2. *A transduction from structures to commutative words is definable if and only if it is of the form* $f_{\varphi,B}$ *for some* φ *and* B.

Proof. Let P be the set of unary relation symbols $\boldsymbol{p}_1,\ldots,\boldsymbol{p}_n$. A commutative word over A will be represented by a P-structure $T=\langle \boldsymbol{D}_T,\boldsymbol{p}_{1T},\ldots,\boldsymbol{p}_{nT}\rangle$ such that for every d in $\boldsymbol{D}_T$ the relation $\boldsymbol{p}_{iT}(d)$ holds for exactly one index i. Considering T as a graph with set of vertices $\boldsymbol{D}_T$, we make $\boldsymbol{p}_{iT}(d)$ hold if and only if d is labelled by a_i.

Let φ and B be given as in the statement. Let b be the value of the largest coefficient $B_{i,j}$ of B. We shall construct T belonging to $f_{\varphi,B}(S)$ as $\mathbf{def}_{\Delta}(S,X_1,\ldots,X_k)$, where Δ is an $n.k.b$-copying definition scheme:

$$\langle\varphi,(\psi_{i,j,t})_{i\in[n],j\in[k],t\in[b]},(\theta_w)_{w\in P\times[n]\times[k]\times[b]}\rangle.$$

The definition scheme Δ will be such that

$$\boldsymbol{D}_T=\{(x,i,j,t)/x\in\boldsymbol{D}_S,\ i\in[n],j\in[k],\ t\in[B_{i,j}],\ x\in X_j\},\tag{1}$$

$$\boldsymbol{p}_{iT}((x,i',j,t))\text{ holds if and only if }i'=i.\tag{2}$$

Since the ith component of the tuple in $f_{\varphi,B}(S)$ associated with $X_1,\ldots,X_k$ is $\mathbf{Card}(X_1).B_{i,1}+\cdots+\mathbf{Card}(X_k).B_{i,k}$, the structure T is actually equal to $\mathbf{def}_{\Delta}(S,X_1,\ldots,X_k)$ if it satisfies (1) and (2) for $X_1,\ldots,X_k$ satisfying φ in S. Hence, we

need only take $\psi_{i,j,t}$ to be the formula "$x \in X_j$" if $t \leqslant B_{i,j}$ and the formula **false** otherwise. For $w = (\boldsymbol{p}_i, i', j, t)$ we take for θ_w the formula **true** if $i = i'$ and **false** otherwise.

We now prove the converse. Given a (P, R)-definition scheme Δ, we shall construct φ and B such that $f_{\varphi,B} = \mathbf{def}_\Delta$. Let Δ be m-copying with parameters $X_1, \ldots, X_k$. Let φ' be its first formula, which defines the domain of $\mathbf{def}_\Delta$. We shall use the new variables $Y_{i,j}$ for $i = 1, \ldots, m, j = 1, \ldots, n$. From Δ it is not hard to construct a formula φ expressing the following in any R-structure S:

> $(X_1, \ldots, X_k)$ holds and, for every i and j, $Y_{i,j}$ is the set of elements x of $\boldsymbol{D}_S$ such that (x, i) belongs to the domain of T where $T = \mathbf{def}_\Delta(S, X_1, \ldots, X_k)$ and $\boldsymbol{p}_{jT}((x, i))$ holds.

The number of elements of T that satisfy $\boldsymbol{p}_j$ is thus the sum of cardinalities of the sets $Y_{i,j}$, for $i = 1, \ldots, m$. From this observation follows the definition of B, a $(k + mn) \times n$-matrix with coefficients in $\{0, 1\}$. □

Any set of the form (3) in Theorem 7.1 is thus the image of a recognizable set of trees under a transduction of the form $f_{\varphi,B}$. By reduction to Parikh's theorem for context-free languages (see [14, Lemma 3.1]), one proves that it is semi-linear. This gives a proof of Theorem 7.1 that does not depend on the results of [27]. Here is an extension of Parikh's theorem.

Corollary 7.3. *Let L be a HR (resp. VR) set of hypergraphs. Let L' be the associated set of structures (of type 2 or 1, resp.). Then $f_{\varphi,B}(L')$ is semilinear.*

This result extends the version of Parikh's theorem of [31] in the sense that it does not only count vertex or edge labels but cardinalities of sets satisfying MS formulas. It is used in [14] to decide whether a VR set of hypergraphs is HR.

8. Conclusion

Definable transductions form a quite powerful class of graph transformations, that is nevertheless manageable, as shown by the closure theorems we have stated above. We do not review algorithmic aspects here. Let us only mention that these transductions form a key tool in [1] where testing properties of tree-structured graphs (a derivation tree relative to a context-free grammar is a typical example of structuring) is reduced (via the inverse of a definable transduction) to testing properties of trees by means of finite-state tree automata. Let us also mention that for every input graph G of tree-width at most some k, an output graph (relative to a fixed definable transduction) can be constructed in time $O(\mathbf{size}(G))$ by the results of [17].

A natural question is whether the class of definable transductions can be extended so as to be closed under inverse. The answer is no if one wishes that the extended transductions preserve the classes HR and VR, because, roughly speaking, the set of

all finite graphs is neither HR nor VR. This is a striking difference with the case of words (since the set of all words over a finite alphabet is context-free): there is no hope to generalize everything from words (or trees) to graphs.

Acknowledgments

I thank A. Arnold, J. Engelfriet, M. Mosbah and J.-C. Raoult for useful remarks and suggestions on preliminary versions of this paper.

References

[1] S. Arnborg, J. Lagergren and D. Seese, Problems easy for tree-decomposable graphs, *J. Algorithms* **12** (1991) 308–340.
[2] M. Bauderon and B. Courcelle, Graph expressions and graph rewritings, *Math. Systems Theory* **20** (1987) 83–127.
[3] J. Berstel, *Transductions and Context-free Languages* (Teubner, Stuttgart, 1979).
[4] F.-J. Brandenburg, The equivalence of boundary and confluent graph grammars on graph languages with bounded degree, Lecture Notes in Computer Science, Vol. 488 (Springer, Berlin, 1991) 312–322.
[5] J. Büchi, Weak second-order logic and finite automata, *Z. Math. Logik Grundlagen Math.* **5** (1960) 66–92.
[6] B. Courcelle, An axiomatic definition of context-free rewriting and its application to NLC graph grammars, *Theoret. Comput. Sci.* **55** (1987) 141–181.
[7] B. Courcelle, Graph rewriting: an algebraic and logic approach, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. B (Elsevier, Amsterdam, 1990) 193–242.
[8] B. Courcelle, The monadic second-order logic of graphs I: recognizable sets of finite graphs, *Inform. Comput.* **85** (1990) 12–75.
[9] B. Courcelle, The monadic second-order logic of graphs VI: on several representations of graphs by relational structures, Discrete Appl. Math. to appear; *Proc. Logic in Computer Science 1990*, Philadelphia (1990) 190–196.
[10] B. Courcelle, The monadic second order logic of graphs V: on closing the gap between definability and recognizability, *Theoret. Comput. Sci.* **80** (1991) 153–202.
[11] B. Courcelle, Graph grammars, monadic second-order logic and the theory of graph minors, in: N. Robertson and P. Seymour, eds., *Graph Structure Theory*, Contemporary Mathematics, Vol. 147 (Amer. Math. Soc., Philadelphia, PA, 1993) 565–590.
[12] B. Courcelle, The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues, *Inform. Théorique Appl.* **26** (1992) 257–286.
[13] B. Courcelle, The monadic second-order logic of graphs VII: graphs as relational structures, *Theoret. Comput. Sci.* **101** (1992) 3–33.
[14] B. Courcelle, Structural properties of context-free sets of graphs generated by vertex replacement, Research Report 91-44, Bordeaux-1 Univ.; *Inform. Comput.* (1994), to appear.
[15] B. Courcelle and J. Engelfriet, A logical characterization of the sets of hypergraphs generated by hyperedge replacement grammars, Research Report 91-41, Bordeaux-1 Univ. 1991, *Math. Systems Theory*, to appear.
[16] B. Courcelle, J. Engelfriet and G. Rozenberg, Handle-rewriting hypergraph grammars, *J. Comput. System Sci.* **46** (1993) 218–270.
[17] B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, *Theoret. Comput. Sci.* **109** (1993) 49–82.
[18] M. Dauchet, T. Heuillard, P. Lescanne and S. Tison, Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems, *Inform. Comput.* **88** (1990) 187–201.

[19] J. Doner, Tree acceptors and some of their applications, *J. Comput. System Sci.* **4** (1970) 406–451.
[20] C. Elgot, Decision problems of finite automata design and related arithmetics, *Trans. AMS* **98** (1961) 21–52.
[21] J. Engelfriet, Context-free NCE graph grammars, *Proc. FCT' 89*, Lecture Notes in Computer Science, Vol. 380, (Springer, Berlin, 1989) 148–161.
[22] J. Engelfriet, A characterization of context-free NCE graph languages by monadic second-order logic on trees, Lecture Notes in Computer Science, Vol. 532 (Springer, Berlin, 1991) 311–327.
[23] J. Engelfriet and L. Heyker, The string generating power of context-free hypergraph grammars, *J. Comput. System Sci.* **43** (1991) 328–360.
[24] J. Engelfriet and L. Heyker, Hypergraph languages of bounded degree, Report 91-01, Univ. Leiden, 1991: *J. Comput. System Sci.*, to appear.
[25] J. Engelfriet and G. Rozenberg, A comparison of boundary graph grammars and context-free hypergraph grammars, *Inform. Comput.* **84** (1990) 163–206.
[26] J. Engelfriet and G. Rozenberg, Graph grammars based on node rewriting: an introduction to NLC graph grammars, Lecture Notes in Computer Science, Vol. 532 (Springer, Berlin, 1991) 12–23.
[27] J. Engelfriet, G. Rozenberg and G. Slutzki, Tree transducers, L-systems and two-way machines, *J. Comput. System Sci.* **20** (1980) 150–202.
[28] F. Gavril, Algorithms for minimum coloring, maximum clique, and maximum independent set of a chordal graph, *SIAM J. Comput.* **1** (1972) 180–187.
[29] F. Gecseg and M. Steinby, *Tree Automata* (Akademiai Kiado, Budapest, 1984).
[30] Y. Gurevich, Monadic second-order theories, in: J. Barwise and S. Feferman, eds., *Model Theoretic Logic* (Springer, Berlin, 1985) 479–506.
[31] A. Habel, Hyperedge replacement: grammars and languages, Lecture Notes in Computer Science, Vol. 643 (Springer, Berlin, 1992).
[32] A. Habel and H.J. Kreowski, May we introduce to you: hyperedge replacement?, *Proc. 3rd Internat. Workshop on Graph Grammars*, Lecture Notes in Computer Science, Vol. 291 (Springer, Berlin, 1987) 15–26.
[33] D. Janssens and G. Rozenberg, A survey of NLC grammars, Lecture Notes in Computer Science, Vol. 159 (Springer, Berlin, 1983) 114–128.
[34] K.-J. Lange, Context-free controlled ETOL systems, *Proc. 10th ICALP*, Lecture Notes in Computer Science, Vol 154 (Springer, Berlin, 1980) 723–733.
[35] M. Rabin, A simple method for undecidability proofs and some applications, in: Y. Bar-Hillel, ed., *Logic, Methodology and Philosophy of Science II* (North-Holland, Amsterdam, 1965) 58–68.
[36] J.-C. Raoult, A survey of tree transductions, in: M. Nivat and A. Podelski, eds., *Tree Automata and Languages* (Elsevier, Amsterdam, 1992) 311–326.
[37] G. Rozenberg and E. Welzl, Boundary NLC grammars, basic definitions, normal forms and complexity, *Inform. Control* **69** (1986) 136–167.
[38] W. Thomas, Automata on infinite objects, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. B (Elsevier, Amsterdam, 1990) 133–192.