Math. Systems Theory **20**, 83-127 (1987)

Mathematical Systems Theory

©1987 Springer-Verlag New York Inc.

Graph Expressions and Graph Rewritings*

Michel Bauderon[1] and Bruno Courcelle[2]

[1] Département d'Informatique, I.U.T. A, Université Bordeaux I, F-33405 Talence, France

[2] Département d'Informatique, Formation associée au CNRS, Université Bordeaux I, 351 Cours de la Libération, F-33405 Talence, France

Abstract. We define an algebraic structure for the set of finite graphs, a notion of graph expression for defining them, and a complete set of equational rules for manipulating graph expressions. (By a *graph* we mean an oriented hypergraph, the hyperedges of which are labeled with symbols from a fixed finite ranked alphabet and that is equipped with a finite sequence of distinguished vertices). The notion of a context-free graph grammar is introduced (based on the substitution of a graph for a hyperedge in a graph). The notion of an equational set of graphs follows in a standard way from the algebraic structure. As in the case of context-free languages, a set of graphs is context-free iff it is equational. By working at the level of expressions, we derive from the algebraic formalism a notion of graph rewriting which is as powerful as the usual one (based on a categorical approach) introduced by Ehrig, Pfender, and Schneider.

0. Introduction

We define an algebraic structure on the set of finite graphs (actually on the set of finite oriented labeled hypergraphs with a sequence of distinguished vertices). The (hyper)edges play the role of elementary objects with which graphs are built exactly as words are built with letters. This means that we shall consider a graph as a set of (hyper)edges "glued" by means of vertices and *not* as a set of vertices linked by means of edges.

* This work has been supported by the PRC "Mathématiques et Informatique". Reprints can be requested from B. Courcelle by electronic mail at the following address (UUCP network): mcvax!inria!geocub!courcell.

The main definitions and results are the following:

1. *Graph expressions.* Since every graph can be built from basic graphs (the edges) by application of three graph operations (disjoint sum of two graphs, fusion of two vertices, and redefinition of the distinguished vertices), graphs can be defined by graph expressions.
2. *A complete set of algebraic laws.* The graph operations satisfy some laws that we state as equations. This set is complete in the sense that two expressions define the same graph (up to isomorphism) iff they can be transformed into each other by the equations.
3. *An algebraic theory of context-free sets of graphs.* Context-free graph grammars based on the substitution of a graph for an edge in a graph (generalizing the substitution of a word for a letter in a word) can be defined and investigated in the algebraic style of Mezei and Wright [29], Goguen *et al.* [19], or Courcelle [5]. The expected theorem saying that context-free sets of graphs are least fixed points of systems of equations associated with context-free graph grammars can be established. A similar theorem for a slightly different notion of context-free graph grammar has been established by Habel and Kreowski [21].
4. *Graph rewritings associated with rewritings of graph expressions.* Every ground rewriting system on graph expressions defines a rewriting relation on graphs. We prove that these rewritings have the same power as the graph rewritings defined by double pushouts (along the lines of [10]–[12], [14], and [33]) with edge-injective homomorphisms. This result together with the definition of a complete set of equations shows that the theory of graph rewritings (or at least a portion of it) can be done in the framework of term rewriting systems. There is hope of applying, in a fruitful way, the recent results on term rewriting systems (see, for instance, the RTA colloquiums [25], [26]) to graph rewritings.

Let us now explain why we use hypergraphs instead of classical oriented labeled graphs. In the case of words, the definition of context-free grammars is based on the substitution of a word for a letter in a word. We want to define context-free graph grammars in a similar way, with labeled edges playing the role of occurrences of letters. Hence we need graphs having the same "type" as the elementary objects they are built with. We want to be able to substitute an ordinary binary edge with a graph that has two distinguished vertices. The orientation of edges is useful in avoiding ambiguity: in the absence of orientation, a graph could be substituted for an edge in two different ways (clearly, the two distinguished vertices must be distinguished from each other; there is somehow an "entry" and an "exit" vertex). But using a pair of vertices as an "interface" (see [10]) between a graph and a context where to put it is insufficient. Hence we need graphs equipped with a *sequence* of distinguished vertices. Accordingly, we use *hypergraphs*, i.e., graphs with edges having a *sequence* of adjacent vertices the length of which is any nonnegative integer.

Let us finally mention that our algebraic theory is many-sorted with infinitely many sorts. Each integer $n \geq 0$ is a sort and the associated carrier is the set of

all graphs with a sequence of n distinguished vertices. There are infinitely many graph operations described by three "schemes of operations" (i.e., by three operation symbols allowing an infinite overloading). The complete set describing the properties of the graph operations consists of an infinite set of equations, generated from 11 "equation schemes." (From the overloading of the operation symbols, these equation schemes generate infinitely many equations). But for dealing with a given context-free graph grammar or a given graph rewriting system finitely many of these sorts, operations, and equations suffice.

The paper is organized as follows. Section 1 reviews basic algebraic definitions. Section 2 introduces hypergraphs. Section 3 defines the algebraic structure on the set of hypergraphs, the notion of a graph expression, and exhibits a complete set of equational rules for manipulating graph expressions. Section 4 deals with context-free graph grammars. Section 5 investigates graph rewritings defined on graph expressions (it can be read independently of Section 4). Section 6 is a conclusion presenting further research.

1. Definitions and Notations

We review some notations and some basic definitions concerning many-sorted algebras and grammars on many-sorted algebras. They are as in Courcelle [4], [5] and, as in these works, the term *magma* is used for *algebra.*

We denote by $\mathbb{N}$ the set of nonnegative integers and by $\mathbb{N}_+$ the set of positive ones. We denote by $[n]$ the interval $\{1, 2, 3, \ldots, n\}$ for $n \geq 0$ (with $[0] = \emptyset$).

For sets A and B we denote by $A - B$ the set $\{a \in A / a \notin B\}$.

The domain of a partial mapping f: $A \to B$ is denoted by $\mathbf{Dom}(f)$. The restriction of f to a subset A' of A is denoted by $f \restriction A'$.

The partial mapping with an empty domain is denoted by $\emptyset$, as the empty set. If two partial mappings f: $A \to B$ and f': $A' \to B$ coincide on $\mathbf{Dom}(f) \cap \mathbf{Dom}(f')$ we denote by $f \cup f'$ their common extension into a partial mapping: $A \cup A' \to B$ with domain $\mathbf{Dom}(f) \cup \mathbf{Dom}(f')$.

The cardinality of a set A is denoted by $\mathbf{Card}(A)$. The powerset of A is denoted by $\mathcal{P}(A)$.

The set of equivalence relations on A is denoted by $\mathbf{Eq}(A)$.

The set of words written over an alphabet A is denoted by A^*. The empty word is denoted by ε. The length of a word u is denoted by $|u|$.

The symbol $\stackrel{\text{def}}{=}$ means "equal by definition" and is used to introduce new notations.

We now review some definitions on many-sorted algebras that we call *magmas.* (The terms "algebra" and "algebraic" are used in so many contexts with so many different meanings that we want to avoid them as much as possible.) A complete set of definitions can be found in Goguen *et al.* [19] or Ehrig and Mahr [13].

Let $\mathcal{S}$ be a set called the set of *sorts.* An *$\mathcal{S}$-signature* F is a set of symbols given with two mappings α: $F \to \mathcal{S}^*$ (the *arity* mapping) and σ: $F \to \mathcal{S}$ (the *sort*

mapping). We say that the *profile* of f in F is the pair $(\alpha(f), \sigma(f))$ written as usual $\alpha(f) \to \sigma(f)$. The *rank* of f is the integer $|\alpha(f)|$.

A *many-sorted F-magma* is an object $\mathbf{M} = \langle (M_s)_{s \in \mathcal{S}}, (f_{\mathbf{M}})_{f \in F} \rangle$ where M_s is a nonempty set for each s and $f_{\mathbf{M}}$ is a total mapping: $M_{s_1} \times \cdots \times M_{s_k} \to M_s$ for each f of profile $s_1, \ldots, s_k \to s$.

If U is a set of $\mathcal{S}$-*sorted* variables (every u in U has a sort $\sigma(u)$ in $\mathcal{S}$) we denote by $M(F, U)$ the set of terms written with F and U that are well formed with respect to sorts and arities. We denote by $M(F, U)_s$ the set of terms of sort s (i.e., with leading symbol of sort s). If $t \in M(F, \{u_1, \ldots, u_k\})_s$ then $t_{\mathbf{M}}$ denotes classically a mapping $M_{s_1} \times \cdots \times M_{s_k} \to M_s$ also called a *derived operator* of $\mathbf{M}$ (where $s_i = \sigma(u_i)$ for $i = 1, \ldots, k$).

If $t \in M(F) \overset{\text{def}}{=} M(F, \varnothing)$ then $t_{\mathbf{M}} \in M_{\sigma(t)}$.

The mapping $t \mapsto t_{\mathbf{M}}$ is the unique F-homomorphism of $\mathbf{M}(F)$ (the initial F-magma) into $\mathbf{M}$.

Let us augment F into F_+ by adding, for every sort s in $\mathcal{S}$, a new symbol $+_s$ of profile $ss \to s$ and a new constant Ω_s of sort s. With M as above we associate its *power-set magma* $\mathcal{P}(\mathbf{M}) = \langle (\mathcal{P}(M_s))_{s \in \mathcal{S}}, (f_{\mathcal{P}(\mathbf{M})})_{f \in F_+} \rangle$ where, for $A_1, \ldots, A_k \subseteq M_{s_1}, \ldots, M_{s_k}$,

$$A_1 +_{s\mathcal{P}(\mathbf{M})} A_2 = A_1 \cup A_2 \qquad (\text{if } s = s_1 = s_2),$$

$$f_{\mathcal{P}(\mathbf{M})}(A_1, \ldots, A_k) = \{f_{\mathbf{M}}(a_1, \ldots, a_k) / a_1 \in A_1, \ldots, a_k \in A_k\}$$

(where $\alpha(f) = s_1, \ldots, s_k$) and $\Omega_{s\mathcal{P}(\mathbf{M})} = \varnothing$. Hence $\mathcal{P}(\mathbf{M})$ is an F_+-magma.

A polynomial system over F is a sequence of equations $S = \langle u_1 = p_1, \ldots, u_n = p_n \rangle$ where $U = \{u_1, \ldots, u_n\}$ is the $\mathcal{S}$-sorted set of unknowns. Each p_i is a *polynomial* of sort $\sigma(u_i)$. A polynomial of sort s is a term of the form Ω_s or

$$t_1 +_s t_2 +_s \cdots +_s t_m,$$

where the t_j's belong to $M(F \cup U)_s$. The subscript s is usually omitted in $+_s$ or Ω_s.

A *grammar* is a pair $(S, \mathbf{M})$ where S is a polynomial system as above and $\mathbf{M}$ is an F-magma.

A mapping $S_{\mathcal{P}(\mathbf{M})}$ of $\mathcal{P}(M_{\sigma(u_1)}) \times \cdots \times \mathcal{P}(M_{\sigma(u_n)})$ into itself is associated with S and $\mathbf{M}$ as follows: for $A_1 \subseteq M_{\sigma(u_1)}, \ldots, A_n \subseteq M_{\sigma(u_n)}$

$$S_{\mathcal{P}(\mathbf{M})}(A_1, \ldots, A_n) = (A_1', \ldots, A_n'),$$

where

$$A_i' = p_{i\mathcal{P}(\mathbf{M})}(A_1, \ldots, A_n) \qquad \text{for} \quad i = 1, \ldots, n.$$

A *solution* of S in $\mathcal{P}(\mathbf{M})$ is an n-tuple $(A_1, \ldots, A_n)$ such that $(A_1, \ldots, A_n) = S_{\mathcal{P}(\mathbf{M})}(A_1, \ldots, A_n)$.

The system S has a least solution in $\mathcal{P}(\mathbf{M})$ with respect to set inclusion, denoted by $(L((S, \mathbf{M}), u_1), \ldots, L((S, \mathbf{M}), u_n))$. It also has a least solution in $\mathcal{P}(\mathbf{M}(F))$, the powerset magma of the initial F-magma $\mathbf{M}(F)$. Then, for each i, $L((S, \mathbf{M}), u_i) = h_{\mathcal{P}(\mathbf{M})}(L((S, \mathbf{M}(F)), u_i))$ where $h_{\mathcal{P}(\mathbf{M})}$ is the canonical extension to sets of the unique homomorphism $h_{\mathbf{M}}$: $\mathbf{M}(F) \to \mathbf{M}$.

The sets $L((S, \mathbf{M}(F)), u_i)$, also denoted by $L(S, u_i)$, $i \in [n]$, can also be defined by rewriting as follows. Let $D_i(S)$ be the set of rewriting rules

$$u_i \to t_1$$
$$\vdots$$
$$u_i \to t_k$$

associated with the equation $u_i = t_1 + \cdots + t_k$ of S. Let

$$D(S) = \bigcup \{D_i(S) / 1 \leq i \leq n\}.$$

Let

$$L(D(S), u_i) = \{t \in M(F) / u_i \xrightarrow[D(S)]{*} t\},$$

where $\xrightarrow[D(S)]{}$ is the rewriting relation on $M(F \cup U)$ associated with $D(S)$ considered as a ground rewriting system (where the u_i's are considered as constants and not as free, substitutable variables). Then, a classical result states that $L(S, u_i) = L(D(S), u_i)$ for all i. This allows us to define $L((S, \mathbf{M}), u_i)$ as the set of elements m of $M_{\sigma(u_i)}$ such that

$$u_i \xrightarrow[D(S)]{*} t \xmapsto[h_{\mathbf{M}}]{} m \qquad (\text{with } m = h_{\mathbf{M}}(t))$$

for some $t \in M(F)$.

To summarize, the same set $L((S, \mathbf{M}), u_i)$ can be characterized in the following ways:

(1) As the ith component of the least solution of S in $\mathcal{P}(\mathbf{M})$.
(2) As the image under $h_{\mathcal{P}(\mathbf{M})}$ of the ith component of the least solution of S in $\mathcal{P}(\mathbf{M}(F))$.
(3) As the set $\{h_{\mathbf{M}}(t) / t \in M(F), u_i \xrightarrow[D(S)]{*} t\}$.

2. Hypergraphs with Sources

There are numerous notions of graphs. We have chosen to deal with labeled, directed hypergraphs with a sequence of distinguished vertices called the sources. The labels are chosen in a ranked alphabet, i.e., in a set A each element of which has an associated integer (in $\mathbb{N}$) that we call its *type*. The type is defined by a mapping $\tau: A \to \mathbb{N}$. The type of the label of a hyperedge must be equal to the number of vertices of that hyperedge (a same vertex may occur several times on a hyperedge). In order to shorten the statements we shall simply call *graphs* these hypergraphs and *edges* their hyperedges.

(2.1) Definition (Concrete Graphs). Let A be a ranked alphabet with type function τ as above. A *concrete graph* is a quintuple:

$$G = \langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \rangle,$$

where $\mathbf{V}_G$ is a set (the set of vertices of G), $\mathbf{E}_G$ is a set (the set of edges of G), $\mathbf{lab}_G$: $\mathbf{E}_G \to A$ assigns to each edge of G a label in the alphabet A, and $\mathbf{vert}_G$ is a mapping associating, with every edge e of G, a sequence of k vertices where $k = \tau(\mathbf{lab}_G(e))$. By letting $\mathbf{vert}_G(e, i)$ denote the ith element of this sequence we have $\mathbf{vert}_G(e) = (\mathbf{vert}_G(e, 1), \ldots, \mathbf{vert}_G(e, k))$. One may have $\mathbf{vert}_G(e, i) = \mathbf{vert}_G(e, j)$ for $i \neq j$ (in particular in the case of a loop with $k = 2$). One may also have $k = 0$: this corresponds to an edge with no vertex. $\mathbf{src}_G$ is a finite sequence in $\mathbf{V}_G$ also used as a mapping: $[n] \to \mathbf{V}_G$ for some $n \geq 0$. Hence $\mathbf{src}_G(i)$ denotes the ith element of the sequence $\mathbf{src}_G$. It is called a *source.* (If $n = 0$ then G has no source.)

We shall say that an edge e *links* the vertices $\mathbf{vert}_G(e, 1), \ldots, \mathbf{vert}_G(e, k)$, and that $\mathbf{vert}_G(e, i)$ *belongs* to e. If we need to specify the alphabet A, we shall say that G is a *concrete graph over* A. If we need to specify the length n of $\mathbf{src}_G$ we shall refer to G as a *concrete* n*-graph.* The integer n is also called the *type* of G.

With every graph G a 0-graph G^0 is associated by forgetting its sources. We call it the 0-graph *underlying* G. A graph G is *finite* if $\mathbf{V}_G$ and $\mathbf{E}_G$ are both finite.

We denote by $\mathbf{CG}_n$ (or $\mathbf{CG}(A)_n$ if we wish to specify A) the collection of all finite concrete n-graphs over A, and by $\mathbf{CG}$ (or $\mathbf{CG}(A)$) the collection of all finite concrete graphs of all types.

A vertex is *isolated* if it belongs to no edge. An *internal* vertex of G is a vertex that does not appear in the sequence $\mathbf{src}_G$. The ones appearing in $\mathbf{src}_G$ are called *external.*

Similar definitions have been given in Habel and Kreowski [21]. The major difference with theirs is that, here, the sequence $\mathbf{src}_G$ may have repetitions. Hence a vertex may be simultaneously considered as the ith and jth source for $i \neq j$.

(2.2) Examples. The following very simple graphs will be useful in building nontrivial graphs:

(1) The discrete graph $\mathbf{n}$ for $n \geq 0$ is the graph G such that $\mathbf{V}_G = [n]$, $\mathbf{E}_G = \emptyset$, $\mathbf{lab}_G = \emptyset$, $\mathbf{vert}_G = \emptyset$, and $\mathbf{src}_G$ is the sequence $(1, 2, \ldots, n)$. In particular we have the *empty* graph $\mathbf{0}$ which is (necessarily) of type 0.

(2) If a is an element of A of type n, then a is the graph G with a single edge labeled a and defined by $\mathbf{V}_G = [n]$, $\mathbf{E}_G = \{1\}$, $\mathbf{lab}_G(1) = a$, and $\mathbf{vert}_G(1) = \mathbf{src}_G = (1, 2, \ldots, n)$. Note the special case where $n = 0$.

(3) A less trivial example is the 3-graph G such that $\mathbf{V}_G = \{t, u, v, w, x, y, z\}$ and $\mathbf{E}_G$ consists of 12 edges labeled a, a', b, b', c, d, such that

$\tau(a) = \tau(a') = 1,$

$\tau(b) = \tau(b') = 2,$

$\tau(c) = 3,$

$\tau(d) = 0.$

In the drawing of this graph (Figure 1) we have used the following conventions:

(i) Binary edges are represented as usual.

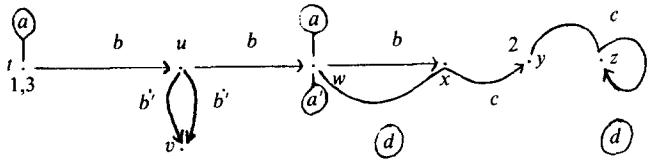
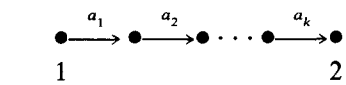


Fig. 1

(ii) Unary edges are labels attached to vertices (a same vertex may be tagged with several labels or several occurrences of the same label).
(iii) Edges of rank greater than 2 are represented as binary edges with intermediate vertices. In Figure 1 there is one edge e such that $\mathbf{vert}_G(e) = wxy$ and another edge e' with $\mathbf{vert}_G(e') = yzz$.
(iv) Nullary edges are represented as floating circles.

The sequence $\mathbf{src}_G$ is *tyt*. On the drawing the positions of the source vertices in $\mathbf{src}_G$ are indicated by the integers 1, 2, and 3. The same conventions will be used in the sequel for representing all graphs.

(2.3) Example (Words as Graphs). We show that words over an alphabet A can be considered as 2-graphs. This example will help us to compare context-free grammars and context-free graph grammars in Section 4. Let A be a finite alphabet. By letting $\tau(a) = 2$ for all a in A we make it into a ranked alphabet. With every word $w = a_1 a_2, \ldots, a_k$ in A^* we associate a 2-graph G better pictured by



than defined formally. Note that the empty word corresponds to a graph with no edge and a single vertex which is, simultaneously, the first and the second source.

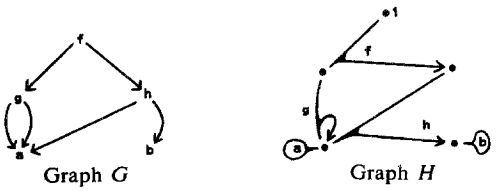
(2.4) Example (Terms and Terms with Shared Subterms). Terms over a fixed signature are usually considered as ordered trees; an implementation of such trees which uses a sharing of identical subtrees can be represented by a graph. Let A be a one-sort signature, i.e., an alphabet given with a rank function $\rho: A \to \mathbb{N}$. The corresponding graphs can be formally defined as 4-tuples of the form

$$G = \langle N, \mathbf{succ}, \mathbf{lab}, v_0 \rangle,$$

where N is the finite set of vertices, $v_0 \in N$ is the root, $\mathbf{lab}$: $N \to A$ defines the label of each vertex, and $\mathbf{succ}$: $N \to N^*$ assigns to each vertex the sequence of its successors. One requires that $|\mathbf{succ}(v)| = \rho(\mathbf{lab}(v))$ for all $v \in N$. These graphs are considered, in particular, in Raoult [32].

Translation into a graph in our sense is easy. We first redefine the rank function. We let $\tau(f) = \rho(f) + 1$ for all f in A. Then with the above graph G, we associate the 1-graph

$$H = \langle N, N, \mathbf{lab}_H, \mathbf{vert}_H, \mathbf{src}_H \rangle,$$



Graph G Graph H

Fig. 2

where

$$\mathbf{lab}_H(v) = \mathbf{lab}(v) \quad \text{for all} \quad v \in N,$$

$$\mathbf{vert}_H(v) = (v, v_1, \ldots, v_k) \quad \text{where} \quad (v_1, \ldots, v_k) = \mathbf{succ}(v),$$

$$\mathbf{src}_H = (v_0).$$

Figure 2 illustrates this construction.

The transformation of G into H is clearly one-to-one. Note that the ordering on the set of successors of any vertex of G is expressed in a natural way in H.

The specific sets $\mathbf{V}_G$ and $\mathbf{E}_G$ chosen to define precisely a graph G are actually irrelevant. We shall not distinguish between two isomorphic graphs. Hence the following definition of an *abstract graph.*

(2.5) Definition (Abstract Graphs). Let G be a concrete n-graph, G' be a concrete n'-graph (both over A). They are *isomorphic* if $n' = n$ and there exist two bijective mappings $h_{\mathbf{V}}$ and $h_{\mathbf{E}}$:

$$h_{\mathbf{V}}: \mathbf{V}_G \to \mathbf{V}_{G'}$$

$$h_{\mathbf{E}}: \mathbf{E}_G \to \mathbf{E}_{G'}$$

such that

$$\mathbf{lab}_G \circ h_{\mathbf{E}} = \mathbf{lab}_{G'},$$

$$h_{\mathbf{V}}(\mathbf{vert}_G(e, i)) = \mathbf{vert}_{G'}(h_{\mathbf{E}}(e), i) \quad \text{for all} \quad i \in [\tau(\mathbf{lab}_G(e))], \quad \text{all } e \text{ in } \mathbf{E}_G,$$

$$h_{\mathbf{V}}(\mathbf{src}_G(i)) = \mathbf{src}_{G'}(i) \quad \text{for all} \quad i \in [n].$$

This means that G and G' are the same up to the specific sets $\mathbf{V}_G$ ($\mathbf{V}_{G'}$) and $\mathbf{E}_G$ ($\mathbf{E}_{G'}$) chosen to define them.

We say that $h = (h_{\mathbf{V}}, h_{\mathbf{E}})$ is an *isomorphism* of G onto G' and we then write $G' = h(G)$. If $G' = G$ we say that h is an *automorphism* of G.

In most cases we shall consider two isomorphic graphs as identical. More precisely we define an *abstract graph* (resp. *abstract n-graph over A*) as the equivalence class of a concrete graph (resp. of a concrete n-graph over A) with respect to isomorphism. And we shall say, in a loose way, "let G be the abstract graph $\langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \rangle$," which means the isomorphism class of the concrete graph $G = \langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \rangle$.

We denote by $\mathbf{G}(A)$ (resp. by $\mathbf{G}(A)_n$) the set of all finite abstract graphs (resp. of all finite abstract n-graphs) over A.

In a concrete graph G the elements of $\mathbf{V}_G$ and $\mathbf{E}_G$ can be considered as *names* given to vertices and edges. These names can be used to distinguish similar items, for instance two edges having the same label and the same sequence of vertices. In the abstract graph associated with G, two such edges cannot be distinguished.

Since we shall mainly be interested in abstract graphs we shall simply call them *graphs* except when it is necessary to emphasize that they are defined "up to an isomorphism."

3. Operations on Graphs

We shall define three operations on graphs that will allow us to build all finite graphs over A starting from the basic graphs $\mathbf{1}$ and a (for a in A) defined in Example (2.2), hence to define them by algebraic expressions.

We first define these operations on concrete graphs, and then we verify that they are well defined for abstract graphs. One of our operations is a disjoint union, needing, for some graphs, the construction of an isomorphic copy, disjoint from some given graph.

Formally, we need a mapping **copy** on concrete graphs such that, for any two concrete graphs G and G', the concrete graph $G'' = \mathbf{copy}(G, G')$ is well defined and satisfies the following conditions:

(1) G'' is isomorphic to G.
(2) $\mathbf{E}_{G''} \cap \mathbf{E}_{G'} = \emptyset$, $\mathbf{V}_{G''} \cap \mathbf{V}_{G'} = \emptyset$.
(3) If $\mathbf{E}_G \cap \mathbf{E}_{G'} = \emptyset$ and $\mathbf{V}_G \cap \mathbf{V}_{G'} = \emptyset$ then $G'' = G$.

There are many ways to define such a mapping **copy** formally but we shall not need any specific construction.

(3.1) Definition (Sum of Two Graphs). Let G' and G'' be two concrete graphs of respective types n' and n''. The sum $G' \oplus G''$ of these two concrete graphs is the concrete $(n'+n'')$-graph G defined as follows;

First case. $\mathbf{V}_{G'} \cap \mathbf{V}_{G''} = \emptyset$ and $\mathbf{E}_{G'} \cap \mathbf{E}_{G''} = \emptyset$: we say that G' and G'' are *disjoint.* Then $\mathbf{V}_G = \mathbf{V}_{G'} \cup \mathbf{V}_{G''}$, $\mathbf{E}_G = \mathbf{E}_{G'} \cup \mathbf{E}_{G''}$, $\mathbf{lab}_G = \mathbf{lab}_{G'} \cup \mathbf{lab}_{G''}$, $\mathbf{vert}_G = \mathbf{vert}_{G'} \cup \mathbf{vert}_{G''}$ (see Section 1 for the notation $\cup$ for functions), and $\mathbf{src}_G = \mathbf{src}_{G'} \cdot \mathbf{src}_{G''} = (\mathbf{src}_{G'}(1), \ldots, \mathbf{src}_{G'}(n'), \mathbf{src}_{G''}(1), \ldots, \mathbf{src}_{G''}(n''))$.

Second case. G' and G'' are not disjoint. The sum of G' and G'' is defined as $G' \oplus G'' \stackrel{\text{def}}{=} G' \oplus \mathbf{copy}(G'', G')$.

If G_1' is isomorphic to G' and G_1'' is isomorphic to G'' it is clear that $G_1' \oplus G_1''$ is isomorphic to $G' \oplus G''$. Hence the sum of two abstract graphs is well defined.

This operation is associative on abstract graphs (see Definition (3.7) below) but not commutative since the sequence of sources of the sum is the *concatenation* of the sequences of sources of the two graphs. If G' and G'' are abstract graphs, at least one of which is of type 0, then $G' \oplus G'' = G'' \oplus G'$.

(3.2) Definition (Redefinition of Sources). For any map α from $[p]$ to $[n]$, we define a *source-redefinition* map σ_α associating a graph of type p with a graph of type n as follows. For any concrete graph G of type n, we let

$$\sigma_\alpha(G) = \langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \circ \alpha \rangle.$$

If $p = 0$ then α is necessarily the empty map (always denoted by $\emptyset$) and $\sigma_\emptyset(G)$ is the 0-graph G^0 obtained from G by "forgetting" its sources. Since the mappings σ_α commute with isomorphisms, the redefinition of sources is well defined for abstract graphs.

(3.3) Definition (Source Fusion). For every equivalence relation δ on $[n]$, we define a mapping θ_δ on concrete graphs of type n as follows. For G, of type n, we let $\theta_\delta(G)$ be the concrete graph G' such that:

(i) $\mathbf{V}_{G'}$ is the quotient of $\mathbf{V}_G$ by the equivalence relation: $v \approx v' \Leftrightarrow v = v'$ or $v = \mathbf{src}_G(i)$ and $v' = \mathbf{src}_G(j)$ for $(i,j) \in \delta$ (with canonical mapping $f: \mathbf{V}_G \to \mathbf{V}_{G'} = \mathbf{V}_G/\approx$).
(ii) $\mathbf{E}_{G'} = \mathbf{E}_G$.
(iii) $\mathbf{vert}_{G'} = f \circ \mathbf{vert}_G$ (i.e., $\mathbf{vert}_{G'}(e, i) = f(\mathbf{vert}_G(e, i))$ for $e \in \mathbf{E}_G, i \in [\tau(\mathbf{lab}_G(e))]$).
(iv) $\mathbf{lab}_{G'} = \mathbf{lab}_G$.
(v) $\mathbf{src}_{G'} = f \circ \mathbf{src}_G$.

If δ is the equivalence relation generated by a single pair (i,j) then we denote θ_δ by $\theta_{i,j}$.

Let Δ_n be the trivial equivalence $\{(i,i)/i \in [n]\}$. Then θ_{Δ_n} is the identity. (This also holds when $n = 0$: then $\Delta_0 = \emptyset$ is considered as an equivalence relation on $[0] = \emptyset$.)

It is clear that if δ is the equivalence relation generated by a set of pairs $\{(i_1, j_1), \ldots, (i_k, j_k)\}$ then

$$\theta_\delta = \theta_{i_1,j_1} \circ \cdots \circ \theta_{i_k,j_k}$$

If δ and δ' are two equivalence relations on $[n]$, then we denote by $\delta \cup \delta'$ the smallest equivalence relation which contains δ and δ'. From the above remark, it is clear that $\theta_{\delta \cup \delta'} = \theta_\delta \circ \theta_{\delta'}$.

As before, this definition also applies to abstract graphs.

It is clear that these operations transform finite graphs into finite graphs. In this paper we consider finite graphs. The case of infinite graphs is considered in Bauderon [2] and Courcelle [9]. From now on "graph" means "finite graph."

(3.4) Definition (Graph Expressions). Let $\mathbb{N}$ be considered as a set of sorts. We define an $\mathbb{N}$-signature H_A consisting of the following symbols:

$\oplus_{n,m}$ of profile $nm \to n+m$ for all $n, m \in \mathbb{N}$.
$\theta_{\delta,n}$ of profile $n \to n$ for all $n \in \mathbb{N}$, all equivalence relations δ on $[n]$.
$\sigma_{\alpha,p,n}$ of profile $n \to p$ for all $n, p \in \mathbb{N}$, all mappings $\alpha: [p] \to [n]$.

a a constant of sort $\tau(a)$ for all a in A.
0 a constant of sort 0.
1 a constant of sort 1.

We shall also use $\mathbb{N}$-sorted sets of variables. If U is such a set, $\tau(u)$ denotes the sort of u in U. We let $E(A, U) \stackrel{\text{def}}{=} M(H_A, U)$ be the set of well-formed finite expressions written with the variables of U, the constants **0**, **1**, and a (for all a in A), and the operation symbols introduced above. We call them *graph expressions*. Each of them has a sort in $\mathbb{N}$ and $E(A, U)_n$ denotes the set of expressions of sort n. We shall use the notations $E(A)$ and $E(A)_n$ when $U = \varnothing$.

An example of a graph expression is

$$g = \sigma_{\alpha,6,9}(\theta_{\delta,7}(a \oplus_{3,4} (u \oplus_{3,1} 1)) \oplus_{7,2} (b \oplus_{1,1} v)),$$

where we assume that

$$a, b \in A, \qquad \tau(a) = 3, \qquad \tau(b) = 1,$$

$$u, v \in U, \qquad \tau(u) = 3, \qquad \tau(v) = 1,$$

$$\delta \in \mathbf{Eq}([7]) \quad \text{and} \quad \alpha : [6] \to [9].$$

It follows that the sort of g is 6.

When writing expressions we shall omit the subscripts n, m, in the operators $\oplus_{n,m}$, $\sigma_{\alpha,p,n}$, and $\theta_{\delta,n}$. Provided the sorts of the variables appearing in an expression are known, its sort can be computed and its well-formedness can be checked. For example, the above expression g will be written

$$\sigma_\alpha(\theta_\delta(a \oplus (u \oplus 1)) \oplus (b \oplus v)).$$

When p is "small", it is convenient to replace $\sigma_{\alpha,p,n}$ (or σ_α) by the symbol $\sigma_{\alpha(1),\ldots,\alpha(p)}$ (see Example (3.5) below).

From Definitions (2.2) and (3.1)-(3.3) we have two many-sorted H_A-magmas, the magma of concrete graphs $\mathbf{CG}(A)$ and the magma of (abstract) graphs $\mathbf{G}(A)$. It follows (from the basic facts recalled in Section 1) that every graph expression g in $E(A)_n$ defines simultaneously

a concrete graph $g_{\mathbf{CG}}$ of type n, and
a graph $g_{\mathbf{G}}$ of type n

(we use the subscripts **CG** and **G** instead of $\mathbf{CG}(A)$ and $\mathbf{G}(A)$). Clearly, $g_{\mathbf{G}}$ is the isomorphism class of $g_{\mathbf{CG}}$. We shall also use the notation $\mathbf{val}(g)$ for $g_{\mathbf{G}}$ when $g \in E(A)$ and we shall say that $\mathbf{val}(g)$ is the *value* of g and that g *defines* or *denotes* $\mathbf{val}(g)$.

Similarly, if $g \in E(A, \{u_1, \ldots, u_m\})_n$ and $G_1, \ldots, G_m$ are concrete (abstract) graphs of respective types $\tau(u_1), \ldots, \tau(u_m)$ then $g_{\mathbf{CG}}(G_1, \ldots, G_m)$ is a concrete n-graph (resp. $g_{\mathbf{G}}(G_1, \ldots, G_m)$ is an abstract n-graph). Let us observe that there is no difference between A and U. Both of them are ranked alphabets. Hence $g \in E(A, U)$ also belongs to $E(A \cup U)$ and hence denotes a graph $\mathbf{val}(g)$ in $\mathbf{G}(A \cup U)$.

In Section 4 we define an operation of substitution on graphs (denoted by $[\ldots]$) such that if $g \in E(A, U)$, where $U = \{u_1, \ldots, u_m\}$, then, for all G_1 in $\mathbf{G}(A)_{\tau(u_1)}$, ..., all G_m in $\mathbf{G}(A)_{\tau(u_m)}$,

$$g_{\mathbf{G}(A)}(G_1, \ldots, G_m) = g_{\mathbf{G}(A \cup U)}[G_1/u_1, \ldots, G_m/u_m].$$

In this formula if $G \in \mathbf{G}(A \cup U)$ then $G[G_1/u_1, \ldots, G_m/u_m]$ denotes the result of the simultaneous substitution in G of G_i for each edge labeled u_i for each $i = 1, \ldots, m$. See Definition (4.1) and Proposition (4.7).

Two expressions g and $g' \in E(A)$ are *equivalent* (we write this $g \equiv g'$) if they denote the same abstract graph, i.e.,

$$g \equiv g' \Leftrightarrow \mathbf{val}(g) = \mathbf{val}(g').$$

(3.5) Example. Let a, b be of type 1, let c, d be of type 2, and e be of type 3. A graph H (with five vertices and six edges) is shown in Figure 3(a). It can be represented by the expression

$$h = \sigma_{1,8,8}(\theta_\delta(a \oplus c \oplus b \oplus d \oplus e \oplus d \oplus \mathbf{1})),$$

where δ is the equivalence relation on [12] generated by

$$\{(1, 2), (3, 4), (4, 5), (5, 6), (7, 1), (9, 4), (10, 11)\}.$$

This expression is formed by source fusion and source redefinition applied to

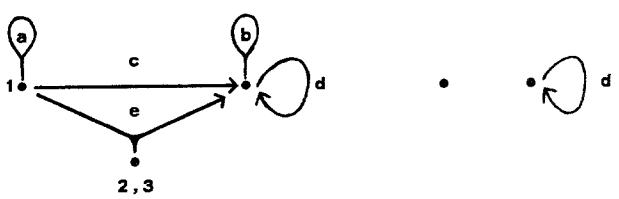
$$h' = a \oplus c \oplus b \oplus d \oplus e \oplus d \oplus \mathbf{1}$$

that represents the graph H' shown in Figure 3(b).

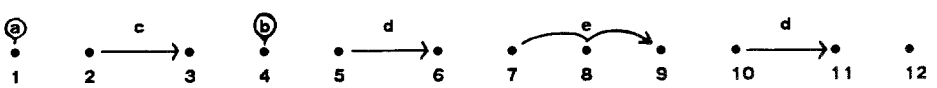
Another expression for H is

$$h'' = \sigma_{1,2,2}(\theta_{\delta''}(e \oplus \sigma_{1,3}(\theta_{\delta'}(a \oplus c \oplus b \oplus d)))) \oplus \sigma_{\varnothing}(\theta_{1,2}(d)) \oplus \sigma_{\varnothing}(\mathbf{1}),$$

where δ' is generated by $\{(1, 2), (3, 4), (4, 5), (5, 6)\}$ and δ'' is generated by $\{(1, 4), (3, 5)\}$.



(a)



(b)

(3.6) Proposition. *For every finite graph G there exists an expression h such that* $G = \mathbf{val}(h)$.

Proof. (An extension of the construction of h in Example (3.5).) If G is the empty graph then $h = \mathbf{0}$. Otherwise let G be an n-graph, with a set of vertices $\mathbf{V}_G = \{v_1, \ldots, v_r\}$ and a set of edges $\mathbf{E}_G = \{e_1, \ldots, e_p\}$. Without loss of generality, we can assume that the set of isolated vertices of G is $\{v_{r-m+1}, \ldots, v_r\}$.

Let a_i be the label of the edge e_i for $1 \leq i \leq p$ (these labels are not necessarily distinct for distinct i's). For each i, $1 \leq i \leq p+1$, we set $\mu(i) = \sum\{\tau(a_j)/1 \leq j \leq i-1\}$. Then $h_1 = a_1 \oplus \cdots \oplus a_p$ is an expression of sort $\mu(p+1)$.

Let δ be the equivalence relation on $[\mu(p+1)]$ defined by the "gluing of edges in the n-graph G":

$$(i,j) \in \delta \Leftrightarrow \begin{cases} i = \mu(i_1) + i_2, \\ j = \mu(j_1) + j_2, \\ \mathbf{vert}_G(e_{i_1}, i_2) = \mathbf{vert}_G(e_{j_1}, j_2) \quad \text{for some } i_1, j_1 \text{ in } [p] \\ \text{and } i_2 \in [\tau(a_{i_1})],\ j_2 \in [\tau(a_{j_1})], \end{cases}$$

and let $h_2 = \theta_\delta(h_1) \oplus \mathbf{m}$, where $\mathbf{m}$ stands for $\mathbf{1} \oplus \mathbf{1} \oplus \cdots \oplus \mathbf{1}$ (m times). This expression defines a $(\mu(p+1)+m)$-graph whose underlying graph is G^0 but which has too many sources.

We define a mapping $\alpha: [n] \to [\mu(p+1)+m]$ such that $\sigma_\alpha(h_2)$ defines G by setting

$$\alpha(i) = \mu(p+1) + k \quad \text{if} \quad \mathbf{src}_G(i) = v_{r-m+k} \quad \text{(i.e., is the } k\text{th isolated vertex)}$$
$$= \mu(j) + k \quad \text{if} \quad \mathbf{src}_G(i) = \mathbf{vert}_G(e_j, k). \qquad \square$$

(3.7) Definitions (Algebraic Properties of Graph Operations). The algebraic properties of the above-defined operations on graphs will be characterized by an infinite set of equations. Actually, these equations are generated by 11 equation schemes listed below and numbered (R1)-(R11).

Since the set A of labels does not play any role in these equations we need not recall it explicitly and we use the notations **CG** and **G** for the magmas $\mathbf{CG}(A)$ and $\mathbf{G}(A)$.

The first equation scheme expresses the associativity of $\oplus$:

$$u \oplus (v \oplus w) = (u \oplus v) \oplus w. \qquad \text{(R1)}$$

For each triple of integers (n, m, p) assigning sorts to the variables u, v, w this equation scheme yields the equation

$$u \oplus_{n,m+p} (v \oplus_{m,p} w) = (u \oplus_{n,p} v) \oplus_{n+m,p} w.$$

It is clear from the definitions that this equation is valid in G.

The subsequent equation schemes concern variables u, v, equivalences δ, δ', and total mappings $\alpha, \beta, \alpha', \beta'$ on finite initial segments of $\mathbb{N}_+$, related by some conditions. To simplify these conditions we shall always assume that $n = \tau(u)$

and $m = \tau(v)$. The integers n, m, p, q can be chosen arbitrarily and independently in $\mathbb{N}$ to generate equations from equation schemes.

$$\sigma_\beta(\sigma_\alpha(u)) = \sigma_{\alpha\circ\beta}(u), \tag{R2}$$

where $\alpha: [p] \to [n]$, $\beta: [q] \to [p]$ (one may have $q = 0$ or $p = q = 0$ or $n = p = q = 0$).

$$\sigma_\alpha(u) = u, \tag{R3}$$

where α is the identity $[n] \to [n]$ (note that $\alpha = \varnothing$ if $n = 0$).

$$\theta_\delta(\theta_{\delta'}(u)) = \theta_{\delta \cup \delta'}(u), \tag{R4}$$

where $\delta, \delta' \in \mathbf{Eq}([n])$.

$$\theta_\Delta(u) = u, \tag{R5}$$

where Δ is the trivial equivalence $\{(i, i)/i \in [n]\}$.

$$\sigma_\alpha(u) \oplus \sigma_{\alpha'}(v) = \sigma_\beta(v \oplus u), \tag{R6}$$

where $\alpha: [p] \to [n]$, $\alpha': [p'] \to [m]$, and $\beta: [p+p'] \to [m+n]$ is such that

$$\beta(i) = m + \alpha(i) \quad \text{for} \quad 1 \le i \le p,$$

$$\beta(i+p) = \alpha'(i) \quad \text{for} \quad 1 \le i \le p'.$$

$$\theta_\delta(u) \oplus \theta_{\delta'}(v) = \theta_{\delta+\delta'}(u \oplus v), \tag{R7}$$

where $\delta \in \mathbf{Eq}([n])$, $\delta' \in \mathbf{Eq}([m])$, and $\delta + \delta'$ is the equivalence on $[n+m]$ generated by $\delta \cup \{(n+i, n+j)/(i, j) \in \delta'\}$.

$$\theta_\delta(u \oplus \mathbf{1}) = \sigma_\alpha(\theta_{\delta'}(u)), \tag{R8}$$

where δ in $\mathbf{Eq}([n+1])$ is such that $(i, n+1) \in \delta$ for some $i \le n$ and where $\delta' = \delta \restriction [n]$, $\alpha: [n+1] \to [n]$, $\alpha(j) = j$ if $j \le n$, $\alpha(n+1) = i$. (By $\delta \restriction [n]$ we denote the equivalence relation $\delta \cap ([n] \times [n])$ on $[n]$.)

$$\theta_\delta(\sigma_\alpha(u)) = \sigma_\alpha(\theta_{\alpha(\delta)}(u)), \tag{R9}$$

where $\delta \in \mathbf{Eq}([n])$, $\alpha: [n] \to [n]$, and $\alpha(\delta)$ denotes the equivalence relation on $[n]$ generated by $\{(\alpha(i), \alpha(j))/(i, j) \in \delta\}$.

$$\sigma_\alpha(\theta_\delta(u)) = \sigma_\beta(\theta_\delta(u)), \tag{R10}$$

where $\alpha, \beta: [p] \to [n]$ and $(\alpha(i), \beta(i)) \in \delta$ for all $i \in [p]$.

$$u \oplus \mathbf{0} = u. \tag{R11}$$

(3.8) Proposition. *The equation schemes* (R1)-(R11) *are valid in* **G**.

In each case, the validity of the equations can be checked from the definitions. We omit the verifications because they are tedious.

(3.9) Remarks. (1) The equation schemes (R1), (R6), (R7), and (R8) are *not* valid in **CG**. This is due to the use of the mapping **copy** in the definition of the sum of two concrete graphs.

(2) Graph expressions can be transformed by means of the equations derived from the equation schemes (R1)-(R11) considered as forming a term rewriting system. We shall write this as $g \overset{*}{\underset{R}{\leftrightarrow}} g'$. (Examples will be given in the proof of Theorem (3.10).) Hence it follows from Proposition (3.8) that if $g, g' \in E(A)$ and $g \overset{*}{\underset{R}{\leftrightarrow}} g'$ then $g \equiv g'$. Moreover, if $g, g' \in E(A, \{u_1, \ldots, u_m\})$ and $g \overset{*}{\underset{R}{\leftrightarrow}} g'$ then, for all $G_1, \ldots, G_m \in G$ of appropriate types,

$$g_{\mathbf{G}}(G_1, \ldots, G_m) = g'_{\mathbf{G}}(G_1, \ldots, G_m).$$

Our next aim will be to establish the converse implications.

(3) Since $\oplus$ is associative in $\mathbf{G}(A)$ (by the validity of (R1)) and is infixed, one can omit the parentheses, but only if the *graph expressions are intended to denote abstract graphs* (see the first remark). For example, the expression g of Definition (3.4) can be written

$$\sigma_\alpha(\theta_\delta(a \oplus u \oplus 1) \oplus b \oplus v)$$

without any ambiguity provided we use it in $\mathbf{G}$. With this convention the use of (R1) (the associativity scheme) becomes implicit in the manipulation of expressions by $\overset{*}{\underset{R}{\leftrightarrow}}$.

(4) We do not know whether the equation schemes (R1)-(R11) form a minimal set. It might be the case that one of them is a consequence of the others.

(5) Here are some *derived* schemes that will be useful in some forthcoming proofs. A special case of scheme (R6) is

$$u \oplus v = \sigma_\beta(v \oplus u) \qquad \text{(R6')}$$

with

$$\beta: [n+m] \to [m+n],$$

$$i \mapsto i+m \quad \text{if} \quad i \in [n]$$

$$i \mapsto i-n \quad \text{otherwise.}$$

By combining (R6), (R6'), and (R2) we get

$$\sigma_\alpha(u) \oplus \sigma_{\alpha'}(v) = \sigma_\tau(u \oplus v) \qquad \text{(R6'')}$$

for a certain τ definable in terms of α and α'. A special case of (R6') is

$$u \oplus v = v \oplus u \quad \text{if } n \text{ or } n' \text{ equals } 0. \qquad \text{(R6''')}$$

Note that we could replace (R6) by (R6') and (R6''). By using (R6) we can use one scheme less than with (R6') and (R6'').

If instead of the operators θ_δ we choose to use the more elementary $\theta_{i,j}$ then the scheme (R4) can be replaced by the following three schemes:

$$\theta_{i,j}(\theta_{k,l}(u)) = \theta_{k,l}(\theta_{i,j}(u)) \quad \text{saying that } \theta_{i,j} \text{ commutes with } \theta_{k,l}, \qquad \text{(R4')}$$

$$\theta_{i,j}(\theta_{i,k}(u)) = \theta_{i,j}(\theta_{j,k}(u)), \qquad \text{(R4'')}$$

$$\theta_{i,j}(u) = \theta_{j,i}(u), \qquad \text{(R4''')}$$

(for all i, j, k, l).

Letting $j = k$ in (R4'') we obtain the idempotence of $\theta_{i,j}$ since $\theta_{j,j}$ coincides with the identity (by scheme (R5)).

Given n, n', and an equivalence δ on $[n+1+n']$ such that $(i, n+1) \in \delta$ for $i \neq n+1$, we can find α and δ' such that, for u and u' of sort n and n', respectively, we have

$$\theta_\delta(u \oplus \mathbf{1} \oplus u') = \sigma_\alpha(\theta_{\delta'}(u \oplus u')). \tag{R8'}$$

The computation proceeds as follows (with $\leftrightarrow$ for $\underset{R}{\leftrightarrow}$):

$$\begin{aligned}
\theta_{\delta'}(u \oplus \mathbf{1} \oplus v) &\leftrightarrow \theta_\delta(\sigma_\beta(v \oplus u \oplus \mathbf{1})) && \text{(by (R6'))} \\
&\leftrightarrow \sigma_\beta(\theta_{\beta(\delta)}(v \oplus u \oplus \mathbf{1})) && \text{(by (R9))} \\
&\leftrightarrow \sigma_\beta\sigma_\tau\theta_{\delta''}(v \oplus u) && \text{(by (R8))} \\
&\leftrightarrow \sigma_\beta\sigma_\tau\theta_{\delta''}\sigma_{\tau'}(u \oplus v) && \text{(by (R6'))} \\
&\underset{R}{\leftrightarrow} \sigma_\alpha\theta_{\delta'}(u \oplus v) && \text{(by (R9) and (R2))}
\end{aligned}$$

for some well-chosen β, δ'', τ, τ', δ', and α.

From (R6''') and (R11) we obtain

$$\mathbf{0} \oplus u = u. \tag{R11'}$$

Our next purpose is to establish the main theorem of this section.

(3.10) Theorem. *Two expressions g and g' of the same sort are equivalent if and only if $g \underset{R}{\leftrightarrow} g'$.*

If follows then from Proposition (3.6):

(3.11) Corollary. *The H_A-magmas $\mathbf{G}(A)$ and $E(A)/\underset{R}{\leftrightarrow}$ are isomorphic.*

Concretely, one can consider a graph as a canonical representative of an equivalence class of $E(A)$ with respect to the congruence $\underset{R}{\leftrightarrow}$.

Here is another consequence of Theorem (3.10). Let $U = \{u_1, \ldots, u_m\}$ be an $\mathbb{N}$-sorted set of symbols with sort mapping τ. Every graph expression g in $E(A, U) = E(A \cup U)$ defines a graph $\mathbf{val}(g) = g_{\mathbf{G}(A \cup U)}$ in $\mathbf{G}(A \cup U)$ and a mapping $g_{\mathbf{G}(A)}$: $\mathbf{G}(A)_{n_1} \times \cdots \times \mathbf{G}(A)_{n_m} \to \mathbf{G}(A)_k$ $(k = \tau(g), n_i = \tau(u_i))$. It is easy to prove that $g_{\mathbf{G}(A)}(u_1, \ldots, u_m) = g_{\mathbf{G}(A \cup U)} = \mathbf{val}(g)$. Hence, for any g and g' in $E(A, U)$,

$$g_{\mathbf{G}(A)} = g'_{\mathbf{G}(A)} \quad \text{iff} \quad g_{\mathbf{G}(A \cup U)} = g'_{\mathbf{G}(A \cup U)} \quad \text{iff} \quad g \underset{R}{\leftrightarrow} g'.$$

The proof of Theorem (3.10) will use expressions of a special type.

(3.12) Definition (Expressions in Canonical Form). An expression g is in *canonical form* if it is of the form $g = \mathbf{0}$ or

$$g = \sigma_\alpha(\theta_\delta(a_1 \oplus \cdots \oplus a_k) \oplus \mathbf{m})$$

with $a_i \in A$ for $i = 1, \ldots, k$, $m \in \mathbb{N}$, $\mathbf{m}$ standing for $\mathbf{1} \oplus \cdots \oplus \mathbf{1}$ (m times). Hence $\mathbf{Dom}(\alpha) = [n]$ where n is the sort the expression g.

Note that k is the number of edges of $\mathbf{val}(g)$ and m is the number of its isolated vertices.

The expressions constructed in the proof of Proposition (3.6) are in canonical form. Two equivalent expressions in canonical form are not necessarily identical (unfortunately) but they are "very close" as we shall see when proving Lemma (3.14) below.

We need two lemmas.

(3.13) Lemma. *Let $n_1, \ldots, n_k$ be a sequence of integers. Let π be a permutation of $[k]$. There exists a permutation α of $[n_1 + \cdots + n_k]$ such that, for all expressions $e_1, \ldots, e_k$ of respective sorts $n_1, \ldots, n_k$, we have*

$$e_{\pi(1)} \oplus \cdots \oplus e_{\pi(k)} \overset{*}{\underset{R'}{\leftrightarrow}} \sigma_\alpha(e_1 \oplus \cdots \oplus e_k),$$

where $R' = \{(\mathrm{R1}), (\mathrm{R2}), (\mathrm{R3}), (\mathrm{R6})\}$.

Proof. We first consider the special case where π is the transposition of two consecutive elements, say i and $i+1$. Let us write $e_{\pi(1)} \oplus \cdots \oplus e_{\pi(k)}$ as $e' \oplus e_{i+1} \oplus e_i \oplus e''$ where $e' = e_1 \oplus \cdots \oplus e_{i-1}$ and $e'' = e_{i+1} \oplus \cdots \oplus e_k$. Then

$$\begin{aligned} e' \oplus e_{i+1} \oplus e_i \oplus e'' &\overset{*}{\leftrightarrow} e' \oplus \sigma_{\beta_1}(e_i \oplus e_{i+1}) \oplus e'' && \text{(by (R6'))} \\ &\overset{*}{\leftrightarrow} \sigma_{\beta_2}(e' \oplus e_i \oplus e_{i+1}) \oplus e'' && \text{(by (R6''))} \\ &\overset{*}{\leftrightarrow} \sigma_{\beta_3}(e' \oplus e_i \oplus e_{i+1} \oplus e'') && \text{(by (R6''))} \end{aligned}$$

for some appropriate mappings $\beta_1, \beta_2, \beta_3$ that are actually permutations (easy to verify). Hence, since π can be decomposed into a sequence of, say, q such transpositions,

$$e_{\pi(1)} \oplus \cdots \oplus e_{\pi(k)} \overset{*}{\leftrightarrow} \sigma_{\gamma_1}(\sigma_{\gamma_2}(\ldots \sigma_{\gamma_q}(e_1 \oplus \cdots \oplus e_k) \ldots))$$

for some permutations $\gamma_1 \ldots \gamma_q$, the result follows by (R2) with $\alpha = \gamma_q \circ \gamma_{q-1} \circ \cdots \circ \gamma_1$. □

Remarks. (1) The permutations $\beta_1, \beta_2, \ldots, \gamma_1, \ldots, \gamma_q$ can be constructed from the integers $n_1, \ldots, n_k$.

(2) There is at most one permutation α such that

$$e_{\pi(1)} \oplus e_{\pi(2)} \oplus \cdots \oplus e_{\pi(k)} \equiv \sigma_\alpha(e_1 \oplus \cdots \oplus e_k) \qquad (*)$$

for all $e_1, \ldots, e_k$. To see this, let α be such a permutation and consider the special case $e_i = \mathbf{0}$ if $\tau(e_i) = 0$ and $e_i = u_i$ with $\tau(u_i) = \tau(e_i)$ if $\tau(e_i) \geq 1$. Then $(*)$ defines the equality of two graphs with edges labeled $u_1, \ldots, u_k$ and such that no two edges have the same label. They have no internal vertices. It follows easily from the consideration of the external vertices that

$$\alpha(\textstyle\sum \{n_{\pi(j)} / 1 \leq j < i\} + i') = n_1 + \cdots + n_{\pi(i)-1} + i'$$

for all $i = 1, \ldots, k$ and all $i' = 1, \ldots, n_i$. This defines α in a unique way.

(3.14) Lemma. *Theorem* (3.10) *holds for expressions g and g' in canonical form.*

Proof. Let $g = \sigma_\alpha(\theta_\delta(f) \oplus \mathbf{m})$ and $g' = \sigma_{\alpha'}(\theta_{\delta'}(f') \oplus \mathbf{m}')$ be two expressions in canonical form and of sort n where

$$f = a_1 \oplus a_2 \oplus \cdots \oplus a_l,$$

$$f' = b_1 \oplus b_2 \oplus \cdots \oplus b_k.$$

We let $n_i = \tau(a_i)$ for $i = 1, \ldots, l$. Let us assume that $g \equiv g'$. Let $G = \mathbf{val}(g) = \mathbf{val}(g')$. We wish to prove that $g \underset{R}{\Leftrightarrow} g'$. We can first observe that $l = \mathbf{Card}(\mathbf{E}_G)$, $k = \mathbf{Card}(\mathbf{E}_{G'})$. Hence $k = l$.

First case. $m = m' = 0, f = f'$, and $a_i \neq a_j$ for all $1 \leq i < j \leq k$. If $n_1 + n_2 + \cdots + n_k = 0$ it is clear that $\alpha = \alpha' = \varnothing$, $\delta = \delta' = \varnothing$ hence $g = g'$. So assume that $n_1 + n_2 + \cdots + n_k \neq 0$.

Let $i, j \in [n_1 + \cdots + n_k]$. There is a unique pair (i', i'') such that $i = n_1 + \cdots + n_{i''-1} + i'$ with $i' \in [n_{i''}]$. We let j', j'' be associated similarly with j. This corresponds to saying that i is the i'th vertex of the unique edge in $\mathbf{val}(f)$ labeled $a_{i''}$, and similarly for j, j', and j''. Hence $(i, j) \in \delta$ if and only if in the graph $G = \mathbf{val}(g)$ we have $\mathbf{vert}_G(e, i') = \mathbf{vert}_G(e', j')$ where e (resp. e') is the unique edge labeled $a_{i''}$ (resp. $b_{j''}$). Since G is also the value of g' the same characterization holds for δ'. Hence $\delta' = \delta$. It is then clear that, for all i in $[n]$, we have $(\alpha(i), \alpha'(i)) \in \delta$. Hence (R10) is applicable and gives

$$g = \sigma_\alpha(\theta_\delta(f)) \underset{R}{\leftrightarrow} \sigma_{\alpha'}(\theta_{\delta'}(f')) = g'.$$

Second case (generalizing the first case). $m = m' = 0$, $a_i \neq a_j$ for all $1 \leq i < j \leq k$ and $(b_1, \ldots, b_k)$ is a permutation of $(a_1, \ldots, a_k)$.

By Lemma (3.13) we can find β such that $f' \underset{R}{\Leftrightarrow} \sigma_\beta(f)$. Hence

$$g' = \sigma_{\alpha'}(\theta_{\delta'}(f')) \underset{R}{\Leftrightarrow} \sigma_{\alpha'}(\theta_{\delta'}(\sigma_\beta(f)))$$

$$\underset{R}{\Leftrightarrow} \sigma_{\alpha''}(\theta_{\delta''}(f)) = g''$$

for some α'' and δ'' depending on α', δ', β and by (R2) and (R9). Hence $g' \equiv g''$ and $g \equiv g''$. The first case is applicable to g and g'' and we have $g \underset{R}{\Leftrightarrow} g''$. Hence $g \underset{R}{\Leftrightarrow} g'$.

General case. Let us consider $g = \sigma_\alpha(\theta_\delta(f) \oplus \mathbf{m})$ and let H be a concrete graph isomorphic to $\mathbf{val}(g)$. There exist two mappings

$$h_{\mathbf{V}}: \quad [n_1 + \cdots + n_k + m] \to \mathbf{V}_H,$$

$$h_{\mathbf{E}}: \quad [k] \to \mathbf{E}_H = \{e_1, e_2, \ldots, e_k\}$$

satisfying the following properties:

(1) $h_{\mathbf{E}}$ is a bijection.
(2) $\mathbf{lab}_H(h_{\mathbf{E}}(i)) = a_i$ for all $i = 1, \ldots, k$.
(3) $\mathbf{vert}_H(h_{\mathbf{E}}(i), j) = h_{\mathbf{V}}(n_1 + \cdots + n_{i-1} + j)$ for all $i = 1, \ldots, k$, all $j = 1, \ldots, n_i$.
(4) $h_{\mathbf{V}}$ is onto.
(5) For all $i, j \in [n_1 + \cdots + n_k + m]$, $h_{\mathbf{V}}(i) = h_{\mathbf{V}}(j)$ iff $(i, j) \in \delta$.
(6) For all $i = 1, \ldots, n$, $\mathbf{src}_H(i) = h_{\mathbf{V}}(\alpha(i))$.

Since H is isomorphic to $\mathbf{val}(g')$ similar mappings $h'_{\mathbf{V}}$ and $h'_{\mathbf{E}}$ exist, associated with g'. Their analogous properties are numbered (1')-(6').

We transform H into a graph $\bar{G}$ as follows. We define two new alphabets $U=\{u_1,\ldots,u_n,\ldots\}$ and $W=\{w_1,w_2,\ldots,w_n,\ldots\}$. We let $\mathbf{V}_{\bar{G}}=\mathbf{V}_H$ and $\mathbf{E}_{\bar{G}}=\mathbf{E}_H\cup\mathbf{I}_H$ where $\mathbf{I}_H$ is the set of isolated vertices of H that we assume to be disjoint from $\mathbf{E}_H$ (without loss of generality). Then we assume that $\mathbf{I}_H=\{v_1,\ldots,v_m\}$ and we let

$$\mathbf{lab}_{\bar{G}}(e_i)=u_i,\qquad 1\le i\le k,$$

$$\mathbf{lab}_{\bar{G}}(v_i)=w_i,\qquad 1\le i\le m,$$

$$\mathbf{vert}_{\bar{G}}(e_i)=\mathbf{vert}_H(e_i),\qquad 1\le i\le k,$$

$$\mathbf{vert}_{\bar{G}}(v_i)=(v_i),\qquad 1\le i\le m,$$

$$\mathbf{src}_{\bar{G}}=\mathbf{src}_H.$$

In words we relabel the edges of G and we attach a unary edge with a label from W, to each isolated vertex of G in such a way that no two edges in $\bar{G}$ have the same label.

Consider the expression

$$\bar{g}=\sigma_\alpha(\theta_\delta(u_{i_1}\oplus\cdots\oplus u_{i_k})\oplus w_{j_1}\oplus\cdots\oplus w_{j_m}),$$

where i_t is such that $h_{\mathbf{E}}(t)=e_{i_t}$ for all $t\in[k]$ and j_t is such that $h_{\mathbf{V}}(t)=v_{j_t}$ for all t in $[m]$. We claim that $\mathbf{val}(\bar{g})$ is isomorphic to $\bar{G}$ and that $\bar{g}[\mathbf{lab}_H(e_1)/u_1,\ldots,\mathbf{lab}_H(e_k)/u_k,\ \mathbf{1}/w_1,\ldots,\mathbf{1}/w_m]$ is equal to g. A graph expression $\bar{g}'$ can be similarly obtained from g'. It is equivalent to $\bar{g}$ since both of them define $\bar{G}$.

We now have

$$\begin{aligned}
\bar{g}&=\sigma_\alpha(\theta_\delta(u_{i_1}\oplus\cdots\oplus u_{i_k})\oplus w_{j_1}\oplus\cdots\oplus w_{j_m})\\
&\overset{*}{\underset{R}{\Leftrightarrow}}\sigma_\alpha(\theta_{\delta+\Delta}(u_{i_1}\oplus\cdots\oplus u_{i_k}\oplus w_{j_1}\oplus\cdots\oplus w_{j_m}))\qquad\text{(by (R7) and (R5))}\\
&\overset{*}{\underset{R}{\Leftrightarrow}}\sigma_{\alpha'}(\theta_{\delta'+\Delta}(u_{i'_1}\oplus\cdots\oplus u_{i'_k}\oplus w_{j'_1}\oplus\cdots\oplus w_{j'_m}))\qquad\text{(by the second case above)}\\
&\overset{*}{\underset{R}{\Leftrightarrow}}\sigma_{\alpha'}(\theta_{\delta'}(u_{i'_1}\oplus\cdots\oplus u_{i'_k})\oplus w_{j'_1}\oplus\cdots\oplus w_{k'_m})=\bar{g}'\qquad\text{(by (R7) and (R5)).}
\end{aligned}$$

Since $g=\bar{g}[\mathbf{lab}_H(e_1)/u_1,\ldots,\mathbf{lab}_H(e_k)/u_k,\mathbf{1}/w_1,\ldots,\mathbf{1}/w_m]$ and $g'=\bar{g}'[\mathbf{lab}_H(e_1)/u_1,\ldots,\mathbf{lab}_H(e_k)/u_k,\mathbf{1}/w_1,\ldots,\mathbf{1}/w_m]$ from $\bar{g}\overset{*}{\underset{R}{\Leftrightarrow}}\bar{g}'$ we obtain $g\overset{*}{\underset{R}{\Leftrightarrow}}g'$. □

Theorem (3.10) is an immediate consequence of Lemma (3.14) and the following lemma.

(3.15) Lemma. *For every expression g there exists an expression g_0 in canonical form such that $g\overset{*}{\underset{R}{\Leftrightarrow}}g_0$.*

Proof. By f, f' we shall denote expressions of the form $a_1\oplus a_2\oplus\cdots\oplus a_k$ for some $k\ge 0$, some $a_1,\ldots,a_k\in A$. The proof is by induction on the structure of g.

(1) If $g=\mathbf{m}$ or a we take $g_0=g$.

(2) If $g = \sigma_\alpha(g')$ we take a canonical form $g'_0 = \sigma_{\alpha'}(\theta_{\delta'}(f') \oplus \mathbf{m})$ of g' (such that $g' \overset{*}{\underset{R}{\leftrightarrow}} g'_0$) and $g_0 = \sigma_{\alpha' \circ \alpha}(\theta_{\delta'}(f') \oplus \mathbf{m})$. Then g may be rewritten into g_0 (in canonical form) by (R2).

(3) If $g = \theta_\delta(g')$ we again take a canonical form $g'_0 = \sigma_{\alpha'}(\theta_{\delta'}(f') \oplus \mathbf{m})$ of g' and then

$$g \overset{*}{\leftrightarrow} \theta_\delta(\sigma_{\alpha'}(\theta_{\delta'}(f') \oplus \mathbf{m})) \qquad \text{(by the choice of } g'_0\text{)}$$

$$\leftrightarrow \sigma_{\alpha'}(\theta_{\alpha'(\delta)}(\theta_{\delta'}(f') \oplus \mathbf{m})) = g'' \qquad \text{(by (R9))}.$$

If the vertex fusions defined by $\alpha'(\delta)$ do not concern $\mathbf{m}$ then g'' rewrites into $\sigma_{\alpha'}(\theta_{\delta''}(\theta_{\delta'}(f')) \oplus \mathbf{m})$ for some δ'' by (R7) and (R5) and then into a canonical expression by (R4). Otherwise we write g'' as

$$\sigma_{\alpha'}(\theta_{\alpha'(\delta)}(\theta_{\delta'}(f') \oplus \mathbf{m}_1 \oplus \mathbf{1} \oplus \mathbf{m}_2)),$$

and, by the derived scheme (R8'), we transform it into

$$\sigma_{\alpha'}(\sigma_\beta(\theta_{\delta''}(\theta_{\delta'}(f') \oplus \mathbf{m}_1 \oplus \mathbf{m}_2)))$$

for some β and δ''. This use of (R8') is repeated at most m times, and finally gives an expression in canonical form.

(4) Let us now assume that $g = g' \oplus g''$ and that the expressions g' and g'' have canonical forms

$$g'_0 = \sigma_{\alpha'}(\theta_{\delta'}(f') \oplus \mathbf{m}') \quad \text{and} \quad g''_0 = \sigma_{\alpha''}(\theta_{\delta''}(f'') \oplus \mathbf{m}'').$$

So we need to compute a canonical form for

$$g_1 = \sigma_{\alpha'}(\theta_{\delta'}(f) \oplus \mathbf{m}') \oplus \sigma_{\alpha''}(\theta_{\delta''}(f'') \oplus \mathbf{m}'').$$

Then

$$g_1 \overset{*}{\leftrightarrow} \sigma_{\alpha_1}(\theta_{\delta'}(f') \oplus \mathbf{m}' \oplus \theta_{\delta''}(f'') \oplus \mathbf{m}'') \qquad \text{(by (R6''))}$$

$$\overset{*}{\leftrightarrow} \sigma_{\alpha_1}(\sigma_{\alpha_2}(\theta_{\delta'}(f') \oplus \theta_{\delta''}(f'') \oplus \mathbf{m}' \oplus \mathbf{m}'')) \qquad \text{(by Lemma (3.13))}$$

$$\overset{*}{\leftrightarrow} \sigma_{\alpha_1}(\sigma_{\alpha_2}(\theta_\delta(f' \oplus f'') \oplus \mathbf{m}' \oplus \mathbf{m}'')) \qquad \text{(by (R7))}$$

$$\overset{*}{\leftrightarrow} \sigma_\alpha(\theta_\delta(f' \oplus f'') \oplus \mathbf{m}' \oplus \mathbf{m}'') \qquad \text{(by (R2))}$$

and this expression is the desired canonical form.

In each case the necessary α_1, α_2, δ, and α are appropriately chosen to satisfy the corresponding scheme or derived scheme. □

(3.16) Definition (Widths of Graphs and of Graph-Expressions). We define the *width* of an expression g as the maximum number n such that g has a subexpression of sort n, and we denote it by $\mathbf{wd}(g)$. Our convention that $g_1 \oplus g_2 \oplus g_3$ stands for either $g_1 \oplus_{n,m+p} (g_2 \oplus_{m,p} g_3)$ or $(g_1 \oplus_{n,m} g_2) \oplus_{n+m,p} g_3$ where g_1, g_2, g_3 are expressions of respective sorts n, m, p does not raise any difficulty. In both cases $\mathbf{wd}(g_1 \oplus g_2 \oplus g_3) = \mathbf{Max}\{n+m+p, \mathbf{wd}(g_1), \mathbf{wd}(g_2), \mathbf{wd}(g_3)\}$.

We define the width of a graph $G \in \mathbf{G}(A)$ as $\mathbf{wd}(G) \overset{\text{def}}{=} \mathbf{Min}\{\mathbf{wd}(g) / g \in E(A), \mathbf{val}(g) = G\}$. It is clear that $\mathbf{wd}(G)$ cannot be less than the type of G and the type of any a in A labeling an edge in G.

The expressions produced by the proof of Proposition (3.6) are clearly not of minimal width.

(3.17) Proposition. *If A contains at least one element of type larger than 1, there exist graphs in $\mathbf{G}(A)_0$ of arbitrarily large width.*

Proof. Let n be an integer, $n \geq 1$. Let K be a graph in $\mathbf{G}(A)_0$ having n vertices and such that any two vertices are linked by at least one edge. Let e be a graph expression denoting K.

Our aim is to prove that $\mathbf{wd}(e) \geq n$. Let M be the set of nodes of the syntax tree t of e. For every m in M, the subtree of t issued from m is the syntax tree of a subexpression of e denoted by e/m. (It is clearer *not* to identify e and t.) Let K/m be the graph defined by e/m. There exists an embedding $h_m = (h_{m\mathbf{V}}, h_{m\mathbf{E}})$: $K/m \to K$ (see Definition (5.2) and Proposition (5.6) below).

We denote by $\|m\|$ the cardinality of $h_{m\mathbf{V}}(\mathbf{V}_{K/m})$, i.e., the number of vertices of the subgraph of K that is the image of K/m in K. If m is in the scope of an operator θ_δ some sources of $\mathbf{V}_{K/m}$ may become equal by $h_{m\mathbf{V}}$. Hence $\|m\| \leq \mathbf{Card}(\mathbf{V}_{K/m})$. On the other hand, $h_{m\mathbf{V}}$ is one-to-one on the set of internal vertices of K/m and $h_{m\mathbf{E}}$ is one-to-one on $\mathbf{E}_{K/m}$.

Our proof is based on the following observation. Let $m \in M$ be such that $e/m = (e/m_1) \oplus (e/m_2)$ where m_1 and m_2 are the two successor nodes of m.

Fact 1. At least one of K/m_1 and K/m_2 has no internal vertex.

(Otherwise, if K/m_1 has an internal vertex v and K/m_2 has an internal vertex v', then $h_{m_1\mathbf{V}}(v)$ and $h_{m_2\mathbf{V}}(v')$ are two distinct vertices w and w' of K that cannot be linked by any edge, contradiction.) Without loss of generality we can assume that K/m_1 has no internal vertex. Then

Fact 2. If v is a vertex of K/m_1 and if K/m_2 has internal vertices, then $h_{m_1\mathbf{V}}(v) = h_{m_2\mathbf{V}}(v')$ for some external vertex v' of K/m_2.

(Otherwise, if v'' is an internal vertex of K/m_2, then no edge in K can link $h_{m_1\mathbf{V}}(v)$ and $h_{m_2\mathbf{V}}(v'')$.)

Claim. For every $m \in M$, $\|m\| \leq \mathbf{wd}(e/m)$.

The proof uses a bottom-up induction on the tree t. If m is a leaf, then e/m is either $\mathbf{0}$, $\mathbf{1}$, or a for some a in A and the result holds in all cases. If m is such that $e/m = \sigma_\alpha(e/m_1)$, where m_1 is the successor of m in t, then we can assume that $\|m_1\| \leq \mathbf{wd}(e/m_1)$ and then

$$\|m\| = \|m_1\|, \qquad \mathbf{wd}(e/m) = \mathbf{Max}\{\mathbf{wd}(e/m_1), \tau(e/m)\} \geq \mathbf{wd}(e/m_1).$$

Hence

$$\|m\| = \|m_1\| \leq \mathbf{wd}(e/m_1) \leq \mathbf{wd}(e/m).$$

If m is such that $e/m = \theta_\delta(e/m_1)$, then similarly we have

$$\|m\| = \|m_1\| \leq \mathbf{wd}(e/m_1) = \mathbf{wd}(e/m).$$

Let us finally consider the case where $e/m = (e/m_1) \oplus (e/m_2)$ as in Fact 1. Let $k_i = \tau(e/m_i)$ for $i = 1, 2$.

First case. K/m_1 and K/m_2 have no internal vertex. Hence $\|m_i\| \leq k_i$ for $i = 1, 2$.

$$\|m\| \leq \|m_1\| + \|m_2\|$$
$$\leq k_1 + k_2$$
$$\leq \mathbf{Max}\{k_1 + k_2, \mathbf{wd}(e/m_1), \mathbf{wd}(e/m_2)\} = \mathbf{wd}(e/m).$$

Second case. One and only one of K/m_1 and K/m_2 has internal vertices. Let K/m_2 be that graph. By Fact 2, $h_{m_1 V}(K/m_1) \subseteq h_{m_2 V}(K/m_2)$, hence

$$\|m\| \leq \|m_2\|.$$

By induction $\|m_2\| \leq \mathbf{wd}(e/m_2)$. Hence

$$\|m\| \leq \mathbf{Max}\{\mathbf{wd}(e/m_2), \mathbf{wd}(e/m_1), k_1 + k_2\} = \mathbf{wd}(e/m).$$

This completes the proof of the claim.

Letting m be the root of t we get $\|m\| = n \leq \mathbf{wd}(e/m) = \mathbf{wd}(e)$. Since n was chosen arbitrarily large the proof is complete. □

(3.18) Applications (Derived Theories). Let us go back to the example of words already considered in Example (2.3). We keep the same notations.

We have exhibited a mapping $h: A^* \to \mathbf{G}(A)_2$. This mapping can be defined in terms of expressions as follows, by letting

$$\bar{h}(\varepsilon) = \sigma_{1,1}(\mathbf{1}),$$
$$\bar{h}(a) = a,$$
$$\bar{h}(a \cdot u) = \mathbf{conc}(a, \bar{h}(u)),$$

where, for any two expressions e and e' in $E(A)_2$, $\mathbf{conc}(e, e') = \sigma_{1,4}(\theta_{2,3}(e \oplus e'))$. Hence $h(u) = \mathbf{val}(\bar{h}(u))$ for all u in A^*.

It follows that the algebraic structure $\langle A^*, \varepsilon, (a)_{a \in A}, \bullet \rangle$ is isomorphic to $\langle G'(A)_2, \sigma_{1,1}(\mathbf{1}), (a)_{a \in A}, \mathbf{conc} \rangle$ where $G'(A)_2$ is a certain subset of $\mathbf{G}(A)_2$. Actually $G'(A)_2 = \{\mathbf{val}(g)/g \in M(\{\mathbf{conc}\}, \{\sigma_{1,1}(\mathbf{1})\} \cup A)\}$. It is not difficult to verify the associativity of **conc**, namely that

$$\mathbf{conc}(e, \mathbf{conc}(e', e'')) \underset{R}{\overset{*}{\leftrightarrow}} \mathbf{conc}(\mathbf{conc}(e, e'), e'')$$

for all e, e', and e'' in $E(A)_2$.

Going back to Example (2.4) we now consider the case of trees.

The mapping $k: M(A) \to \mathbf{G}(A)_1$ can be defined as $\mathbf{val} \circ \bar{k}$ where $\bar{k}: M(A) \to E(A)_1$ is such that

$$\bar{k}(a) = a \quad \text{if} \quad a \in A, \quad \rho(a) = 0,$$
$$\bar{k}(a(t_1, \ldots, t_n)) = \mathbf{Const}_a(\bar{k}(t_1), \ldots, \bar{k}(t_n)) \quad \text{if} \quad a \in A, \quad \rho(a) = n,$$
$$t_1, \ldots, t_n \in M(A).$$

In this definition, for each $n \geq 1$, a as above, $e_1, \ldots, e_n \in E(A)_1$ we let

$$\mathbf{Const}_a(e_1, \ldots, e_n) = \sigma_1(\theta_\delta(a \oplus e_1 \oplus e_2 \cdots \oplus e_n)),$$

where the equivalence relation $\delta \in \mathbf{Eq}([2n+1])$ is generated by the set of pairs $\{(i+1, 2i+1)/i = 1, \ldots, n\}$. Hence, dealing with trees corresponds to working in a certain subset of $\mathbf{G}(A)_1$ together with the derived operators $\mathbf{Const}_a$ for a in A.

Other examples can be found in Courcelle [6], [8].

4. Context-Free Graph Grammars

Context-free languages are usually defined and manipulated in terms of derivation sequences, i.e., by considering grammars as rewriting systems. By a theorem of Ginsburg and Rice [18], they can also be characterized as least solutions of certain systems of equations canonically obtained from the grammars. This characterization motivates the alternative terminology of *algebraic* for context-free languages and grammars. Mezei and Wright have shown in [29] that this notion of an equational (or algebraic [15]) set can be defined in any magma and not only in the free monoid as in the case of context-free languages. This theory has been recently developed in Courcelle [5]. Since we have defined an algebraic structure on the set of graphs we immediately obtain the notion of an equational set of graphs from the general results of [29] and [5].

We define a notion of *context-free graph grammar.* It is based on the substitution of a graph for an edge (of the same type) in a graph. Our two main results are:

(1) The characterization of a context-free set of graphs as a component of the least solution of a system of equations in $\mathcal{P}(\mathbf{G}(A))$ canonically associated with the grammar generating it (this is an extension of the theorem of Ginsburg and Rice).
(2) A set of graphs is context-free iff it is equational with respect to $\mathbf{G}(A)$.

The first result can be stated independently of the algebraic structure on $\mathbf{G}(A)$ defined in Section 3, whereas the second explicitly refers to it. Result (1) could be proved "directly." Actually it will follow more easily from the second.

The basic lemma allowing this proof states that the above-mentioned substitution can be done at the level of graph expressions.

(4.1) Definition (Graph Substitutions (for Concrete Graphs)). Let G be a concrete graph, let e be an edge of G of type $p \geq 0$. Let G' be a concrete p-graph. We denote by $G[G'/e]$ the concrete graph G'' defined as follows, with $H = \mathbf{copy}(G', G)$ (H is a copy of G' disjoint from G):

$$\mathbf{E}_{G''} = \mathbf{E}_G \cup \mathbf{E}_H - \{e\},$$

$$\mathbf{lab}_{G''} = (\mathbf{lab}_G \cup \mathbf{lab}_H) \restriction \mathbf{E}_{G''},$$

$$\mathbf{V}_{G''} = V/\sim$$

where $\sim$ is the equivalence, relation on $V \overset{\text{def}}{=} \mathbf{V}_G \cup \mathbf{V}_H$ generated by $\{(\mathbf{src}_H(i),$

$\mathbf{vert}_G(e, i))/i \in [p]\}$ and letting f be the canonical surjection

$$V \to \mathbf{V}_{G''},$$

$$\mathbf{vert}_{G''}(e', i) = f(\mathbf{vert}_G(e', i)) \quad \text{if} \quad e' \in E_G - \{e\},$$

$$= f(\mathbf{vert}_H(e', i)) \quad \text{if} \quad e' \in E_H$$

(for all $i \in [\tau(e')]$ in both cases),

$$\mathbf{src}_{G''}(i) = f(\mathbf{src}_G(i)) \quad \text{for} \quad i \in [\tau(G)].$$

This definition also applies when $p = 0$. In this case e is an edge with no vertex.

If there is in G a single edge e labeled a we can write $G[G'/a]$ for $G[G'/e]$. We can also extend this notation to the case where there is no edge in G labeled a: in that case $G[G'/a] = G$.

Let now $e_1, \ldots, e_k$ be edges of G of respective types $n_1, \ldots, n_k$. Let $G'_1, \ldots, G'_k$ be concrete graphs of respective types $n_1, \ldots, n_k$. Then the result of the simultaneous substitution of G'_1 for $e_1, \ldots, G'_k$ for e_k in G is the concrete graph $G[G'_1/e_1][G'_2/e_2] \cdots [G'_k/e_k]$ also denoted by $G[G'_1/e_1, \ldots, G'_k/e_k]$. If, for all $i = 1, \ldots, k$, e_i is the only edge of G labeled a_i we shall use the notation $G[G'_1/a_1, \ldots, G'_k/a_k]$ for $G[G'_1/e_1, \ldots, G'_k/e_k]$, and, as above, it can also be extended to the case where there is no edge labeled a_i for some $i = 1, \ldots, k$.

With the above notations:

(4.2) Lemma.

(1) *For every permutation π of $[k]$ the concrete graphs $G[G'_1/e_1, \ldots, G'_k/e_k]$ and $G[G'_{\pi(1)}/e_{\pi(1)}] \ldots [G'_{\pi(k)}/e_{\pi(k)}]$ are isomorphic.*

(2) *If $\bar{G}$ is isomorphic to G by h and if $\bar{G}'_i$ is isomorphic to G'_i for all $i = 1, \ldots, k$ then $G[G'_1/e_1, \ldots, G'_k/e_k]$ and $\bar{G}[\bar{G}'_1/h(e_1), \ldots, \bar{G}'_k/h(e_k)]$ are isomorphic.*

We omit the proof which is a straightforward verification. Part (1) of this lemma justifies the qualification of "simultaneous" for the substitution $G[G'_1/e_1, \ldots, G'_k/e_k]$.

(4.3) Definition (Graph Substitutions (for Abstract Graphs)). Lemma (4.2(2)) shows that, in some sense, substitutions can be defined for abstract graphs. But the notation $G[G'_1/e_1, \ldots, G'_k/e_k]$ cannot be used for abstract graphs $G, G'_1, \ldots, G'_k$ since there is no way to designate specific edges (i.e., $e_1, \ldots, e_k$) in abstract graphs. Of course, if, for every $i = 1, \ldots, k$, there is at most one edge labeled a_i the notation $G[G'_1/a_1, \ldots, G'_k/a_k]$ can be used for abstract graphs (provided the type of G'_i is equal to the type of a_i for all i in $[k]$).

(4.4) Definition (Context-Free Graph Grammars). A *context-free graph grammar* is a quadruple $\Gamma = \langle A, U, P, Z \rangle$ consisting of two finite ranked alphabets A and U (the *terminal* and the *nonterminal* one: the rank function is τ: $A \cup U \to \mathbb{N}$) and

a finite set P of pairs of the form (u, G) with $G \in \mathbf{G}(A \cup U)_{\tau(u)}$ and $u \in U$ (the set of *productions*). The term Z is a graph in $\mathbf{G}(A \cup U)$ called the *axiom.*

A binary relation on $\mathbf{CG}(A \cup U)$ is associated with Γ as follows: $H \underset{\Gamma}{\rightarrow} H'$ if there exists a production (u, G) in P and an edge e of H labeled u such that $H' = H[G/e]$. This relation is well defined for abstract graphs and will be used for them in the sequel. We write $H \xrightarrow[(u,G)]{} H'$ if we wish to specify the production used in this rewriting step. Intuitively, $H \xrightarrow[(u,G)]{} H'$ means that G has been substituted in H for some edge labeled u. If H has exactly one edge labeled u we can write more simply $H' = H[G/u]$. Note that if $H \xrightarrow[(u,G)]{} H'$, H and H' are of the same type.

Let $G \in \mathbf{G}(A \cup U)$. The set of graphs *generated by* Γ *from* G is then

$$L(\Gamma, G) \overset{\text{def}}{=} \{H \in \mathbf{G}(A) / G \underset{\Gamma}{\overset{*}{\rightarrow}} H\},$$

and

$$L(\Gamma) \overset{\text{def}}{=} L(\Gamma, Z).$$

Such a set is called a *context-free* set of graphs.

We shall frequently omit the axiom in the definition of the grammar. This means that any graph in $\mathbf{G}(A \cup U)$ can be taken as a possible axiom. In such a case the notation $L(\Gamma)$ is meaningless. In all cases $L(\Gamma, u)$ for $u \in U$ denotes the set of graphs generated from the graph u (reduced to a single hyperedge labeled u) taken as the axiom.

(4.5) Example. Consider the grammar of Figure 4. In this example, *a*, *b*, *c*, and

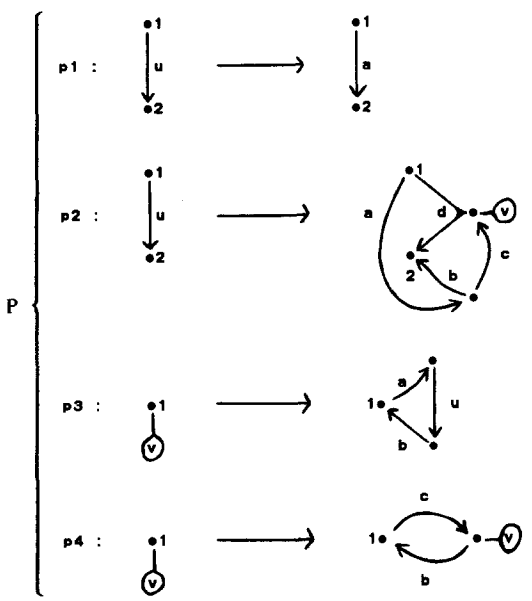


Fig. 4

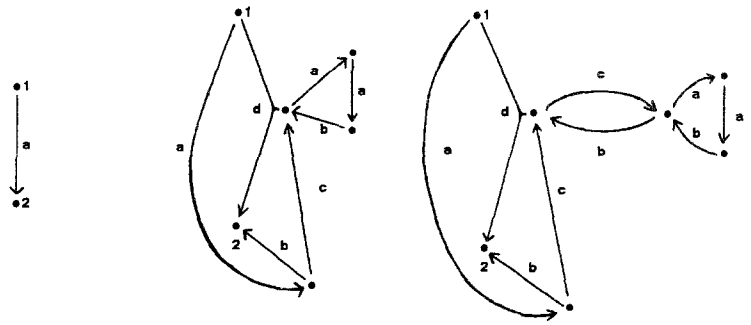


Fig. 5.

d label "terminal edges" (with $\tau(a) = \tau(b) = \tau(c) = 2$, $\tau(d) = 3$) and u and v label "nonterminal edges." The symbol u is of type 2 hence it generates a set of 2-graphs. The distinguished vertices are marked 1 and 2. The symbol v has rank 1 hence generates a set of 1-graphs (see Figure 4).

Examples of graphs generated by Γ from u are shown in Figure 5

(4.6) Definition (Associating a Polynomial System with a Context-Free Graph Grammar). Let $\Gamma = \langle A, U, P\rangle$ be a context-free graph grammar without axiom. Let $U = \{u_1, \ldots, u_m\}$. For every right-hand side G of a production rule of P let $\mathbf{exp}(G)$ be a graph expression denoting G. Let S be the ground rewriting system on $E(A \cup U)$

$$S \overset{\text{def}}{=} \{(u, \mathbf{exp}(G))/(u, G) \in P\}.$$

Let $\bar{S}$ be the polynomial system $\bar{S} \overset{\text{def}}{=} \langle u_i = p_i;\ i \in [m]\rangle$ where $p_i = \sum \{\mathbf{exp}(G)/(u_i, G) \in P\}$. It follows that $D(\bar{S}) = S$ where D is the mapping recalled in Section 1. Then $(\bar{S}, \mathbf{G}(A))$ is a grammar on the magma of graphs $\mathbf{G}(A)$.

We shall prove that $L(\Gamma, u) = L((\bar{S}, \mathbf{G}(A)), u)$ for all $u \in U$. For this purpose we shall prove that graph rewritings can be performed on graph expressions. This will build a bridge between the notion of a context-free graph grammar and the notion of a grammar over the magma of graphs.

We first recall that, for $g \in E(A, \{u_1, \ldots, u_m\})$ and $g_1, \ldots, g_m \in E(A, U)_{\tau(u_1)}, \ldots, E(A, U)_{\tau(u_m)}$, we denote by $g[g_1/u_1, \ldots, g_m/u_m]$ the result of the simultaneous substitution of g_i for u_i for all $i = 1, \ldots, m$. We say that g is *linear in* U if every element of U has at most one occurrence in g.

(4.7) Proposition. *Let* $g \in E(A, \{u_1, \ldots, u_m\})$, *let* $g_1, \ldots, g_m \in E(A)$ *be of respective types* $\tau(u_1), \ldots, \tau(u_m)$. *Let us assume that* g *is linear with respect to* $\{u_1, \ldots, u_m\}$. *Then*

$$\mathbf{val}(g[g_1/u_1, \ldots, g_m/u_m]) = \mathbf{val}(g)[\mathbf{val}(g_1)/u_1, \ldots, \mathbf{val}(g_m)/u_m].$$

Sketch of the Proof. It suffices to perform the proof for $m = 1$. The general result follows from Definitions (4.3) and (4.1) and from the fact that

$$g[g_1/u_1, \ldots, g_m/u_m] = g[g_1/u_1][g_2/u_2] \ldots [g_m/u_m].$$

For $m = 1$, the proof can be done easily by an induction on the structure of g. □

In the following corollary we use the notations of Definition (4.6).

(4.8) Corollary. *Let* $(u, G) \in P$ *and* $g = \mathbf{exp}(G)$. *Let* $h \in E(A \cup U)$ *and* $H = \mathbf{val}(h)$.

(1) *If* $h \xrightarrow[(u,g)]{} h'$ *then* $H \xrightarrow[(u,G)]{} \mathbf{val}(h')$.
(2) *If* $H \xrightarrow[(u,G)]{} H'$ *then there exists* $h' \in E(A \cup U)$ *such that* $h \xrightarrow[(u,g)]{} h'$ *and* $H' = \mathbf{val}(h')$.

Proof. (1) Let $h_1 \in E(A \cup U \cup \{w\})$ where w is a new variable of sort $\tau(u)$ such that $h = h_1[u/w]$, w has exactly one occurrence in h_1 and $h' = h_1[g/w]$. Let H_1 be the concrete graph defined by h_1. It is clear that H is isomorphic to $H_1[u/w]$, and $H_1[u/w] \xrightarrow[(u,G)]{} H_1[G/w]$. By Proposition (4.7) $H_1[G/w]$ is isomorphic to $\mathbf{val}(h_1[g/w]) = \mathbf{val}(h')$. Hence $H \xrightarrow[(u,G)]{} \mathbf{val}(h')$.

(2) The proof is similar. □

Let us recall that by the notations of Section 1 if $g \in E(A, U)$, $U = \{u_1, \ldots, u_m\}$, then $L(S, g)$ denotes $\{g' \in E(A) / g \xrightarrow[S]{*} g'\}$. We denote by $L((\bar{S}, \mathbf{G}), u_i)$ the ith component of the least solution of the system $\bar{S}$ in $\mathscr{P}(\mathbf{G})$ (and $\mathbf{G}$ is a simplified notation for $\mathbf{G}(A)$). Finally, if $L \subseteq E(A)$ then $\mathbf{val}(L) \overset{\text{def}}{=} \{\mathbf{val}(g) / g \in L\}$.

(4.9) Theorem. *Let* $\Gamma = \langle A, U, P, Z \rangle$ *be a context-free graph grammar, let* S *and* $\bar{S}$ *be associated with* $\langle A, U, P \rangle$ *as in* (4.6).

Then:

(1) $L(\Gamma, u) = \mathbf{val}(L(S, u))$
$= L((\bar{S}, \mathbf{G}), u)$ *for all* $u \in U$.

(2) $L(\Gamma) = \mathbf{val}(L(S, \mathbf{exp}(Z)))$
$= Z_{\mathscr{P}(\mathbf{G})}(L_1, \ldots, L_m)$
(*where* $U = \{u_1, \ldots, u_m\}$ *and* $L_i = L((\bar{S}, \mathbf{G}), u_i)$ *for* $i \in [m]$).

Hence every context-free set of graphs over A *is equational with respect to* $\mathbf{G}(A)$.

Intuitively this proposition says that every context-free set $L(\Gamma)$ of graphs can also be defined:

(1) By context-free rewritings of graph expressions followed by evaluations of the "terminal" graph expressions thus generated, i.e., as $\mathbf{val}(L(S, \mathbf{exp}(Z)))$.

(2) Or as a component of the least solution in $\mathscr{P}(\mathbf{G})$ of a polynomial system over H_A (namely, the system $\bar{S} \cup \langle u_0 = \mathbf{exp}(Z)\rangle$ where u_0 is a new unknown of type $\tau(Z)$), and hence is equational.

Proof. By Corollary (4.8) $L(\Gamma, u) = \mathbf{val}(L(S, u))$ and $L(\Gamma) = \mathbf{val}(L(S, Z))$. By the results on polynomial systems recalled in Section 1, we have

$$\mathbf{val}(L(S, u)) = L((\bar{S}, \mathbf{G}), u)$$

and

$$\mathbf{val}(L(S, Z)) = Z_{\mathscr{P}(\mathbf{G})}(L_1, \ldots, L_m)$$

where $(L_1, \ldots, L_m)$ is the least solution of $\bar{S}$ in $\mathscr{P}(\mathbf{G})$. □

(4.10) Remark. Note that $\bar{S}$ is not associated in a unique way with Γ since there exist several expressions denoting the same graph. But, for any alternative system $\bar{S}'$ associated with Γ, the functions $\bar{S}'_{\mathscr{P}(\mathbf{G})}$ and $\bar{S}_{\mathscr{P}(\mathbf{G})}$ are the same. Hence the two systems $\bar{S}'$ and $\bar{S}$ have the same solutions in $\mathscr{P}(\mathbf{G})$.

(4.11) Theorem. *A subset of* $\mathbf{G}(A)_n$ *(for some* A *and* n*) is context-free iff it is equational.*

Proof. The "only if" direction has been obtained in Theorem (4.9). Let us conversely consider a polynomial system $K = \langle u_i = p_i, i \in [n]\rangle$ over $\mathbf{G}(A)$, and $L = L((K, \mathbf{G}), u_1)$. (Again $\mathbf{G}$ stands for $\mathbf{G}(A)$.) Consider $P = \{(u_i, \mathbf{val}(t_{i,j}))/ i \in [m], j \in [n_i]\}$ where each p_i is a sum of the form $t_{i,1} + t_{i,2} + \cdots + t_{i,n_i}$. Then $\Gamma = \langle A, U, P\rangle$ is a context-free graph grammar (without axiom). It is clear that the functions $K_{\mathscr{P}(\mathbf{G})}$ and $\bar{S}_{\mathscr{P}(\mathbf{G})}$ (where $\bar{S}$ is associated with Γ as in Definition (4.6)) are the same. It follows from Theorem (4.9) that

$$L = L((K, \mathbf{G}), u_1) = L((\bar{S}, \mathbf{G}), u_1) = L(\Gamma, u_1)$$

hence that L is context-free. □

To perform this proof it is necessary to allow u_1 to be of type 0 (for the case where $L \subseteq \mathbf{G}(A)_0$). Since in a context-free graph grammar every nonterminal is considered as an edge of the same type we had to introduce edges of type 0, i.e., edges having no vertex in order to have Theorem (4.11).

We shall complete the picture by giving a concrete interpretation of $\bar{S}_{\mathscr{P}(\mathbf{G})}$ in terms of substitutions of sets of graphs. This will give us a theorem for context-free graph grammars analogous to the one of Ginsburg and Rice [18] for context-free (word) grammars. But we first develop Example (4.5).

(4.12) Example (continuation of Example (4.5)). The system S associated with the grammar Γ of Example (4.5) is the following one:

$$S = \begin{cases} q1: & u \to a, \\ q2: & u \to \sigma_\alpha(\theta_\delta(a \oplus b \oplus c \oplus d \oplus v)), \\ q3: & v \to \sigma_{\alpha'}(\theta_{\delta'}(a \oplus u \oplus b)), \\ q4: & v \to \sigma_{\alpha''}(\theta_{\delta''}(c \oplus b \oplus v)) \end{cases}$$

(for appropriate α, δ, α', δ', α'', δ'' such that for each rule $qi\colon x \to g$ of S then $x \to \mathbf{val}(g)$ is the rule pi of P).

The three example graphs displayed in Example (4.5) are the values of the following expressions of $L(S, u)$:

a,

$\sigma_\alpha(\theta_\delta(a \oplus b \oplus c \oplus d \oplus \sigma_{\alpha'}(\theta_{\delta'}(a \oplus a \oplus b))))$, and

$\sigma_\alpha(\theta_\delta(a \oplus b \oplus c \oplus d \oplus \sigma_{\alpha''}(\theta_{\delta''}(c \oplus b \oplus \sigma_{\alpha'}(\theta_{\delta'}(a \oplus a \oplus b))))))$.

They are defined by the following derivations:

$q1$ for the first one,

$q2$; $q3$; $q1$ for the second one, and

$q2$; $q3$; $q4$; $q3$; $q1$ for the third.

(4.13) Definition (Substitution of Sets of Graphs in a Graph). Let $U = \{u_1, \ldots, u_m\}$. Let W be an auxiliary $\mathbb{N}$-sorted set of variables disjoint from U. (The sort mapping is τ and $\tau(u)$ is the type of u.) Let $G \in \mathbf{G}(A \cup U)$. Let $\tilde{G} \in \mathbf{G}(A \cup W)$ satisfy the following:

(1) Each w in W labels at most one edge in $\tilde{G}$.
(2) $G = \tilde{G}[u_{i_1}/w_1, \ldots, u_{i_k}/w_k]$ where $\{w_1, \ldots, w_k\}$ is the set of elements of W occurring in $\tilde{G}$ and $i_1, \ldots, i_k \in [m]$.

There exist several graphs satisfying (1) and (2) but we assume that some uniform way is used to construct $\tilde{G}$ from G. Then for graphs $G_1, \ldots, G_k$ of respective types $\tau(w_1)$ $(= \tau(u_{i_1}))$, $\ldots$, $\tau(w_k)$ $(= \tau(u_{i_k}))$ the graph $\tilde{G}[G_1/w_1, \ldots, G_k/w_k]$ is well defined. For *sets* of graphs $\mathcal{G}_1 \subseteq \mathbf{G}(A)_{\tau(w_1)}, \ldots, \mathcal{G}_k \subseteq \mathbf{G}(A)_{\tau(w_k)}$ we let

$$\tilde{G}[\![\mathcal{G}_1/w_1, \ldots, \mathcal{G}_k/w_k]\!] = \{\tilde{G}[G_1/w_1, \ldots, G_k/w_k] / G_1 \in \mathcal{G}_1, \ldots, G_k \in \mathcal{G}_k\}.$$

We define finally, for $\mathcal{G}_1 \subseteq \mathbf{G}(A)_{\tau(u_1)}, \ldots, \mathcal{G}_m \subseteq \mathbf{G}(A)_{\tau(u)_m}$,

$$G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!] = \tilde{G}[\![\mathcal{G}_{i_1}/w_1, \ldots, \mathcal{G}_{i_k}/w_k]\!].$$

It is easy to verify that the value of $G[\![\cdots]\!]$ does not depend on the specific choice of $\tilde{G}$ (provided (1) and (2) above hold).

Intuitively, $G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!]$ is the set of graphs we obtain by substituting in G arbitrary graphs in $\mathcal{G}_1, \ldots, \mathcal{G}_m$ for the edges labeled $u_1, \ldots, u_m$. Note that distinct elements of $\mathcal{G}_i$ can be substituted for distinct edges both labeled u_i. (This is analogous to the OI-substitution in trees, see Engelfriet and Schmidt [17] or Courcelle [5]).

In the special case where $\mathcal{G}_i = \{G_i\}$ for all $i = 1, \ldots, m$, $G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!]$ is reduced to a single element that we denote by $G[G_1/u_1, \ldots, G_m/u_m]$. Hence this is a new (and actually the last) extension of the notation introduced in

Definition (4.3). With a context-free graph grammar $\Gamma = \langle A, U, P\rangle$ we associate the system $\bar{\Gamma}$ consisting of the following equations:

$$\mathcal{G}_i = \bigcup \{G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!]/(u_i, G) \in P\}, \qquad 1 \leq i \leq m.$$

In this system $\mathcal{G}_i$ denotes a subset of $\mathbf{G}(A)_{\tau(u_i)}$. This system is fully analogous to the system of equations on languages that are classically associated with a context-free grammar. Solving $\bar{\Gamma}$ consists in finding in $E = \mathcal{P}(\mathbf{G}(A)_{\tau(u_1)}) \times \cdots \times \mathcal{P}(\mathbf{G}(A)_{\tau(u_m)})$ the least fixed-point of a certain mapping: $E \to E$, also denoted by $\bar{\Gamma}$. It is clear from Lemma (4.15) below that this mapping is the same as the mapping $\bar{S}_{\mathcal{P}(\mathbf{G})}$. Hence, we have, by Theorem (4.9),

(4.14) Theorem. *$(L(\Gamma, u_1), \ldots, L(\Gamma, u_m))$ is the least solution in $\mathcal{P}(\mathbf{G})$ of the system $\bar{\Gamma}$.*

(4.15) Lemma. *Let $g \in E(A \cup \{u_1, \ldots, u_m\})$, let $G = \mathbf{val}(g)$. Then, for every $\mathcal{G}_1 \subseteq \mathbf{G}(A)_{\tau(u_1)}, \ldots, \mathcal{G}_m \subseteq \mathbf{G}(A)_{\tau(u_m)}$,*

$$G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!] = g_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_1, \ldots, \mathcal{G}_m).$$

Proof. If g is U-linear then by Lemma 10.5 of [5] (p. 58)

$$g_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_1, \ldots, \mathcal{G}_m) = \{g_{\mathbf{G}}(G_1, \ldots, G_m)/G_1 \in \mathcal{G}_1, \ldots, G_m \in \mathcal{G}_m\}$$

and

$$g_{\mathbf{G}}(G_1, \ldots, G_m) = G[G_1/u_1, \ldots, G_m/u_m]$$

by Proposition (4.7) since, for $g_1, \ldots, g_m$ in $E(A)$ defining respectively $G_1, \ldots, G_m$, one has

$$\begin{aligned} g_{\mathbf{G}}(G_1, \ldots, G_m) &= g_{\mathbf{G}}(\mathbf{val}(g_1), \ldots, \mathbf{val}(g_m)) \\ &= \mathbf{val}(g[g_1/u_1, \ldots, g_m/u_m]). \end{aligned}$$

If g is not U-linear, one can find h in $E(A, W)$ (where W is as in Definition (4.13)) which is W-linear and such that $g = h[u_{i_1}/w_1, \ldots, u_{i_k}/w_k]$ for some $i_1, \ldots, i_k$ in $[m]$. Then by Definition (4.13)

$$\begin{aligned} G[\![\mathcal{G}_1/u_1, \ldots, \mathcal{G}_m/u_m]\!] &= \mathbf{val}(h)[\![\mathcal{G}_{i_1}/w_1, \ldots, \mathcal{G}_{i_k}/w_k]\!] \\ &= h_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}) \end{aligned}$$

(by the first part of the proof). On the other hand,

$$\begin{aligned} g_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_1, \ldots, \mathcal{G}_m) &= (h[u_{i_1}/w_1, \ldots, u_{i_k}/w_k])_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_1, \ldots, \mathcal{G}_m) \\ &= h_{\mathcal{P}(\mathbf{G})}(\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}) \end{aligned}$$

by general properties of substitutions. Hence the desired equality holds. □

(4.16) Comparison with Context-Free Word Grammars. We have shown in Example (2.3) that a word in A^* can be considered as a graph in $\mathbf{G}(A)_2$. By extending this construction it is easy to transform a context-free grammar generating $L \subseteq A^*$ into a context-free graph grammar generating the corresponding subset

of $\mathbf{G}(A)_2$. Hence every context-free language "is" a context-free set of graphs. But the set of graphs corresponding to the noncontext-free language $\{a^n b^n c^n / n \geq 0\}$ is a context-free set of graphs. (A construction is given in Habel and Kreowski [2], Example 5.5].)

From the first remark follows the undecidability of many problems like deciding whether $L(\Gamma) \cap L(\Gamma') = \varnothing$ for two context-free graph grammars Γ and Γ'. Some decidability results (like deciding whether $L(\Gamma) = \varnothing$) follow from Theorem (4.9(2)) and the decidability of the corresponding problems for the set of trees $L(S, \mathbf{exp}(Z))$.

(4.17) Proposition. *If $L \subseteq \mathbf{G}(A)$ is context free then $\mathbf{Max}\{\mathbf{wd}(G)/G \in L\}$ is finite.*

Proof. Let $L = L(\Gamma)$. By Theorem (4.9) there exists $g \in E(A \cup U)$ such that $L = \mathbf{val}(L(S, g))$ (we use the notations of Theorem (4.9)). The symbols occurring in the elements of $L(S, g)$ are the finitely many ones occurring in S and in g. It follows that their sorts are bounded by an integer K. Hence $\mathbf{wd}(G) \leq K$ for all $G \in L$. □

(4.18) Corollary. *If $n \geq 0$ and A contains at least one symbol of type larger than 1 then $\mathbf{G}(A)_n$ is not context free.*

Proof. Immediate consequence of Propositions (3.17) and (4.17). □

(4.19) Comparison with the Work of Habel and Kreowski [21]. A very similar notion of context-free graph grammar based on the substitution of hyperedges in labeled hypergraphs has been introduced independently by Habel and Kreowski in [21].

We now discuss two differences:

(1) Instead of one sequence of *sources* they use *two* sequences of distinguished vertices called the *begin* and the *end* sequences. An hypergraph having a *begin* sequence of length k and an *end* sequence of length l is called a (k, l)-hypergraph. The hyperedges also have two distinguished sequences of vertices, a sequence of *sources* and a sequence of *targets*. An hyperedge with k sources and l targets is called a (k, l)-edge. And a (k, l)-hypergraph can be substituted for a (k', l')-edge in an hypergraph provided $k = k'$ and $l = l'$.

(2) The vertices occurring in the *begin* and *end* sequences of an hypergraph must be distinct.

For point (1) we think that introducing two sequences of distinguished vertices instead of one as we do, does not provide any real increase of power. Every grammar in the sense of [21] can be simulated by a grammar in our sense. It suffices to concatenate the *begin* and *end* sequences of graphs and the *source* and *target* sequences of the hyperedges, by remembering the lengths of the *begin* and *end* sequences in order to avoid illegal substitutions.

Point (2) limits the class of grammars dealt with in [21] to a proper subclass of ours.

5. Graph Rewriting Systems

Since every graph expression defines a graph, every binary relation on graph expressions defines a binary relation on graphs. The binary relation on graphs associated with a ground rewriting system on graph expressions will be called a graph rewriting.

The purpose of this section is to compare these graph rewritings with the classical ones defined in terms of double pushouts by Ehrig and co-workers [10]-[12], [14] and Rosen [33]. These pushouts are defined with respect to certain categories $\mathscr{C}$ of graphs. The associated relation on graphs is called a $\mathscr{C}$-rewriting.

We shall establish that our rewritings coincide with the $\mathscr{C}$-rewritings that, roughly speaking, preserve the edges (two distinct edges cannot be fused during the rewriting whereas two distinct vertices can be). This limitation comes from the fact that our basic graph operations also preserve the edges in a certain sense.

(5.1) Definition (Graph Rewriting Rules). A *graph rewriting rule* is a pair of graphs $p = (G, G')$ where G, G' are of the same type.

A *graph rewriting system* is a finite set of graph rewriting rules $S = \{p_1, \ldots, p_k\}$.

The *elementary rewriting relation* on graphs associated with $p = (G, G')$ is defined as follows:

$$H \underset{p}{\rightarrow} H' \qquad \text{iff there exists} \quad H_1 \in \mathbf{G}(A \cup \{u\})$$

where u is a new symbol of type $\tau(u) = \tau(G) = \tau(G')$ having a unique occurrence in H_1 such that $H = H_1[G/u]$ and $H' = H_1[G'/u]$. By Proposition (4.7) this is equivalent to the following:

$$H \underset{p}{\rightarrow} H' \qquad \text{iff} \quad H = \mathbf{val}(h), \quad H' = \mathbf{val}(h')$$

for some expressions h, h' such that $h \underset{p}{\rightarrow} h'$ where

$$h \underset{p}{\rightarrow} h' \qquad \text{iff} \quad h = h_1[g/u], \quad h' = h_1[g'/u]$$

for some $h_1 \in E(A \cup \{u\})$ having a unique occurrence of u, some g and g' such that $G = \mathbf{val}(g)$ and $G' = \mathbf{val}(g')$. The context-free rewritings defined in Definition (4.4) form a special case of these ones.

The *semithuian relation on graphs* associated with $S = \{p_1, \ldots, p_k\}$ is the reflexive and transitive closure of the relation $\underset{s}{\rightarrow}$ defined by $H \underset{s}{\rightarrow} H'$ iff $H \underset{p_i}{\rightarrow} H'$ for some $i = 1, \ldots, k$.

It is important to have a characterization of the fact that $H = H_1[G/u]$ for some H_1 and u since this means that "G appears in H" and that a production of the form $p = (G, G')$ is applicable to H.

(5.2) Definitions (Embeddings, Factors, Occurrences). Let $G, H \in \mathbf{CG}(A)_n$ for some n.

A *homomorphism* $h: G \to H$ is a pair of mappings (h_V, h_E) such that:

(H1) $h_V: V_G \to V_H$.

(H2) $h_E: E_G \to E_H$.

(H3) $h_V(\mathbf{vert}_G(e, i)) = \mathbf{vert}_H(h_E(e), i)$ for all $e \in E_G$ all $i \in [\tau(e)]$.

(H4) $\mathbf{lab}_H(h_E(e)) = \mathbf{lab}_G(e)$ for all $e \in E_G$.

(H5) $h_V(\mathbf{src}_G(i)) = \mathbf{src}_H(i)$ for all $i \in [n]$.

Hence the isomorphisms introduced in Definition (2.5) are the homomorphisms such that h_V and h_E are two bijections.

Let now $G \in \mathbf{CG}(A)_n$ and $H \in \mathbf{CG}(A)_m$ with m not necessarily equal to n. Let us recall that $G^0 \stackrel{\text{def}}{=} \sigma_\varnothing(G)$, i.e., is obtained from G by forgetting its sources.

An *embedding of G into H* is a homomorphism $h: G^0 \to H^0$ satisfying the following additional conditions:

(E1) h_E is one-to-one (we shall say that h is *edge-injective*; see also Definition (5.9) below).

(E2) For all $v, v' \neq v$ in V_G if $h_V(v) = h_V(v')$ then v and v' are external in G.

(E3) For all $v \in V_G$ if $h_V(v)$ belongs to some edge in $E_H - h_E(E_G)$ or is external in H then v is external in G.

These conditions say that the set E_G of edges of G is injectively embedded into E_H, that the set of internal vertices of G is injectively embedded into the set of internal vertices of H, and that the vertices of H common to the image $h(G)$ of G in H and to the edges of H that are not in $h(G)$ are images of the external vertices of G.

If h is an embedding of G into H we say that G is a *factor* of H and that h is an *occurrence* of this factor in H. Note that a graph may have several distinct occurrences in a graph.

We say that an abstract graph G is a *factor* of an abstract graph H if it is the isomorphism class of some concrete graph which is a factor of some concrete graph isomorphic to H.

The notion of an embedding is closely related with the *gluing condition* introduced in [10]–[12], [14], and [33]. We recall it from [10, p. 24] in order to make a precise comparison, by formulating it for our graphs. (The reader can skip Definition (5.3) and Lemma (5.4) on first reading.)

(5.3) Definition (Gluing Condition). Let B, C, D be 0-graphs in $\mathbf{CG}(A)$. Let (b, c) be a pair of graph homomorphisms $b: B \to C$ and $c: C \to D$. It satisfies the *gluing condition* if the following conditions hold:

(GC1) For any two distinct edges e and e' of C, if $c_E(e) = c_E(e')$ then e and e' belong both to $b_E(E_B)$.

(GC2) For any two distinct vertices v and v' of C, if $c_V(v) = c_V(v')$ then $v, v' \in b_V(V_B)$.

(GC3) If $v \in \mathbf{V}_C$ and $c_{\mathbf{V}}(v)$ belongs to some edge in $\mathbf{E}_D - c_{\mathbf{E}}(\mathbf{E}_C)$ then $v \in b_{\mathbf{V}}(\mathbf{V}_B)$.

If G is an n-graph then the mapping $\mathbf{src}_G: [n] \to \mathbf{V}_G$ can be considered as a homomorphism of the discrete 0-graph $\boldsymbol{n}^0$ into G^0.

We shall use it in this way in the following lemma:

(5.4) Lemma. *Let G be an n-graph, $n \geq 0$, let H be a 0-graph and h be a homomorphism $G^0 \to H$. Then h is an embedding, $G \to H$ iff $(\mathbf{src}_G, h)$ satisfies the gluing condition.*

Proof. Let h be an embedding. Then (GC1) holds by (E1) and the fact that $\mathbf{n}^0$ has no edge. Condition (GC2) holds by (E2). Condition (GC3) holds by (E3) and the fact that H has no external vertex. Hence $(\mathbf{src}_G, h)$ satisfies the gluing condition. The opposite implication can be established similarly. □

This lemma holds even if $n = 0$. In this case $\mathbf{src}_G = \varnothing$, and h is an embedding iff $(\varnothing, h)$ satisfies the gluing condition, iff h is an injective homomorphism such that $h(G)$ is a union of connected components of H.

We now go back to the characterization of G as a factor of H in terms of embeddings. Let us recall from Definition (4.1) that if G and K are disjoint concrete graphs the graph $H = K[G/e]$ can be defined as follows:

$$\mathbf{E}_H = \mathbf{E}_K \cup \mathbf{E}_G - \{e\},$$

$$\mathbf{V}_H = V/\sim,$$

where $\sim$ is the equivalence relation on $V = \mathbf{V}_K \cup \mathbf{V}_G$ generated by the set of pairs $\{(\mathbf{src}_G(i), \mathbf{vert}_K(e, i)) / i \in [\tau(e)]\}$ and letting f be the canonical surjection $V \to \mathbf{V}_H$,

$$\mathbf{vert}_H(e', i) = f(\mathbf{vert}_K(e', i)) \text{ if } e' \in \mathbf{E}_K - \{e\}$$

$$= f(\mathbf{vert}_G(e', i)) \text{ if } e' \in \mathbf{E}_G$$

(for all $i \in [\tau(e')]$ in both cases),

$$\mathbf{src}_H(i) = f(\mathbf{src}_K(i)) \qquad \text{for all} \quad i \in [\tau(K)],$$

$$\mathbf{lab}_H = (\mathbf{lab}_K \cup \mathbf{lab}_G) \restriction \mathbf{E}_H.$$

(5.5) Remark. The triple $(f \restriction \mathbf{V}_K, f \restriction \mathbf{V}_G, \mathbf{V}_H)$ is the pushout in the category of sets of the pair of mappings $\mathbf{src}_G: [n] \to \mathbf{V}_G$ and $\mathbf{vert}_K(e, \cdot): [n] \to \mathbf{V}_K$. If $\mathbf{src}_G$ is injective then so is $f \restriction \mathbf{V}_K$. This is proved in Ehrig [10] but can also be verified directly from the above definition.

Let then $i = (i_{\mathbf{V}}, i_{\mathbf{E}})$ be such that

$$i_{\mathbf{V}} = f \restriction \mathbf{V}_G,$$

$i_{\mathbf{E}}$ is the inclusion map, $\mathbf{E}_G \to \mathbf{E}_H$.

Anticipating the next proposition we shall call i the *embedding of G into $K[G/e]$*.

With these notations

(5.6) Proposition.

(1) *If K and G are disjoint concrete graphs if $H = K[G/e]$ for some edge e of K of type $\tau(G)$, then the pair of maps $i = (i_{\mathbf{V}}, i_{\mathbf{E}})$ defined above is an occurrence of G in H.*

(2) *Conversely, if G and H are concrete graphs such that G has an occurrence h in H there exists a concrete graph K, disjoint from G, with an edge e such that H is isomorphic (by k) to $K[G/e]$ and such that the embedding i of G into $K[G/e]$ is equal to $k \circ h$ (where $h\colon G \to H$ and $k\colon H \to K[G/e]$).*

Proof. (1) It is easy to verify that i is an embedding. For conditions (E2) and (E3) it suffices to note that if $v \in \mathbf{V}_G$, $v' \in V$, v is internal in G, and $v' \sim v$ then $v' = v$. This follows immediately from the definition of $\sim$.

(2) Let $h = (h_{\mathbf{V}}, h_{\mathbf{E}})$ be an embedding of G into H. Without loss of generality we can assume that G and H are disjoint. Let $n = \tau(G)$. Let K be defined as follows:

$$\mathbf{V}_K = (\mathbf{V}_H - h_{\mathbf{V}}(\mathbf{V}_G)) \cup \{h_{\mathbf{V}}(\mathbf{src}_G(1)), \ldots, h_{\mathbf{V}}(\mathbf{src}_G(n))\},$$

$$\mathbf{E}_K = (\mathbf{E}_H - h_{\mathbf{E}}(\mathbf{E}_G)) \cup \{e\}$$

(where e is a new edge of type n),

$$\mathbf{src}_K = \mathbf{src}_H$$

($\mathbf{src}_K$ is well defined by (E3) of Definition (5.2)),

$\mathbf{lab}_K(e) =$ any element of A of type n,

$\mathbf{lab}_K(e') = \mathbf{lab}_H(e')$ for all $e' \in \mathbf{E}_H - h_{\mathbf{E}}(\mathbf{E}_G)$,

$\mathbf{vert}_K(e, i) = h_{\mathbf{V}}(\mathbf{src}_G(i))$ for all $i = 1, \ldots, n$,

$\mathbf{vert}_K(e', i) = \mathbf{vert}_H(e', i)$ for all $e' \in \mathbf{E}_H - h_{\mathbf{E}}(\mathbf{E}_G)$,

$(\mathbf{vert}_H(e', i) \in \mathbf{V}_K$ (by (E3) of Definition (5.2)).

It is clear that G and K are disjoint. The remaining verifications are easy. □

Remark. In (2) the graph K can be written $\sigma_\alpha(\theta_\delta(K' \oplus u))$ (up to isomorphism) where u is a new symbol of type n and K' is a graph (essentially K minus the edge e) of type $n+p$ where $p = \tau(K)$, α is the inclusion map, $[p] \to [p+2n]$, and δ is the equivalence relation on $[p+2n]$ generated by $\{(p+1, p+n+1), \ldots, (p+n, p+2n)\}$. Hence $H = K[G/e]$ can be written in the special form $\sigma_\alpha(\theta_\delta(K' \oplus G))$.

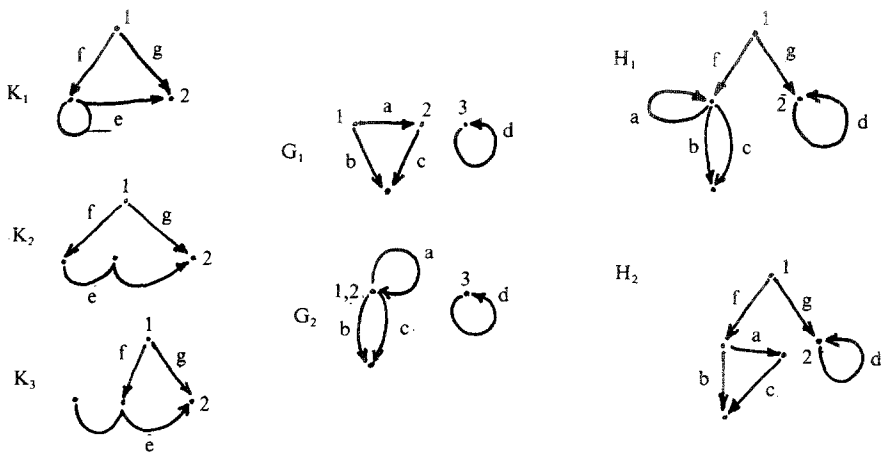


Fig. 6

(5.7) Examples. Consider the graphs shown in Figure 6. It is easy to verify the following equalities (of abstract graphs):

$$H_1 = K_1[G_1/e], \qquad H_2 = K_2[G_1/e],$$

$$H_1 = K_1[G_2/e] = K_2[G_2/e] = K_3[G_2/e].$$

It is clear that G_2 has no occurrence in H_2. Although G_2 has exactly one occurrence in H_1 three different "context" graphs K_i, $i = 1, 2, 3$, can be defined such that $H_1 = K_i[G_2/e]$. This is due to the fact that the sequence of sources of G_2 has a repetition ($\mathbf{src}_{G_2}(1) = \mathbf{src}_{G_2}(2)$). In the case of G_1 there is a unique "context" graph K such that $H_1 = K[G_1/e]$. This is precisely shown in the following proposition.

(5.8) Proposition. *Let G, H, and h be as in Proposition* (5.6(2)). *Let us assume that G has pairwise distinct sources. Let (K, e, k, i) and (K', e', k', i') be two quadruples satisfying Proposition* (5.6(2)). *There exists an isomorphism j: $K \to K'$ such that $j_E(e) = e'$.*

The proof is a straightforward verification that we omit.

The following proposition shows that the notions of embeddings and factors (and by Lemma (5.4), the gluing condition) can be expressed at the level of graph expressions.

(5.9) Proposition. *For abstract graphs H and G the following conditions are equivalent*:

(1) *G is a factor of H.*

(2) *There exist graph expressions g and h such that g is a subexpression of h,* $\mathbf{val}(g) = G$, *and* $\mathbf{val}(h) = H$.

(3) *There exist* $K \in \mathbf{G}(A \cup \{u\})$ *where* u *has a unique occurrence in* K *such that* $H = K[G/u]$.

If, furthermore, H *is of type* 0 *then these conditions are equivalent to*:

(4) *There is a homomorphism* j: $G^0 \to H$ *such that* $(\mathbf{src}_G, h)$ *satisfies the gluing condition.*

Proof. (2)⇔(3) by Proposition (4.7).
(3)⇔(1) by Proposition (5.6).
(1)⇔(4) by Lemma (5.4). □

The following proposition states what is called the embedding of derivations in [10], [11], and [33].

(5.10) Proposition. *Let* (G, G') *be a graph rewriting rule and let* $H \xrightarrow[(G,G')]{} H'$. *Let* H *be a factor of some other graph* K *say* $K = K_1[H/u]$. *Then one has, with* $K' = K_1[H'/u]$,

$$K \xrightarrow[(G,G')]{} K' \quad \text{and} \quad K \xrightarrow[(H,H')]{} K'.$$

Proof. We let g, g', h, h', k be graph expressions denoting G, G', H, H', K and such that, for some $h_1 \in E(A, \{w\})$ and $k_1 \in E(A, \{u\})$,

$$h = h_1[g/w],$$

$$h' = h_1[g'/w],$$

$$k = k_1[h/u] = (k_1[h_1/w])[g/w],$$

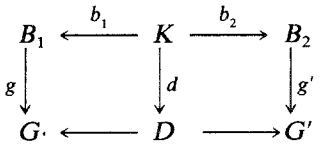
by the associativity of substitution. For the same reason $k' = k_1[h'/u] = (k_1[h_1/w])[g'/w]$. Hence the result holds with $K' = \mathbf{val}(k')$. □

We now compare these graph rewritings with the classical ones formalized in terms of push-out diagrams first by Ehrig *et al.* [14] and later by Rosen [33], Ehrig [10], and Ehrig *et al.* [12]. A complete bibliography can be found in Nagl [30].

We assume that the reader knows the few definitions in category theory used in the above-cited works, see Manes [28] or McLane [27]. We begin with a very general definition.

(5.11) Definition (Rewriting Rules in a Category). Let $\mathscr{C}$ be a category. A *$\mathscr{C}$-rewriting rule* is a pair of morphisms having the same domain. Actually, it is convenient to write such a pair as a quintuple $p = (B_1 \xleftarrow{b_1} K \xrightarrow{b_2} B_2)$ by also indicating the domain K and the codomains B_1 and B_2 of b_1 and b_2.

If G and G' are objects in $\mathscr{C}$ we say that G *rewrites into* G' *via* p iff there exists a commutative diagram



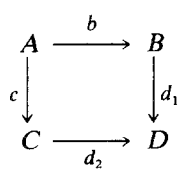
where both squares are push-outs. We write this $G \underset{p}{\Rightarrow} G'$.

Definition 3.1 of Ehrig [10, p. 18] is the instance of this notion associated with the category $\mathscr{C}$ of finite oriented graphs (not hypergraphs), the vertices and the edges of which are "colored," i.e., labeled in a finite fixed set. The homomorphisms are the natural ones (defined by (H1)-(H4)). In Ehrig *et al.* [14], the extra condition that d is injective is also imposed. See Rosen [33] for a review of these slightly different definitions.

In what follows we shall use this definition for a category of graphs with edge-injective homomorphisms. It is concretely used in the following way: given G, $p = (B_1 \xleftarrow{b_1} K \xrightarrow{b_2} B_2)$, and $g: B_1 \to G$ one must construct G' such that $G \underset{p}{\Rightarrow} G'$. It is proved in Ehrig [10] that there exist D and d such that (K, B_1, D, G) forms a push-out iff (b_1, g) satisfies the gluing condition. Since a push-out (K, D, B_2, G') can always be constructed (from K, b_2, d) the gluing condition is necessary and sufficient for the existence of G' such that $G \underset{p}{\Rightarrow} G'$.

In our approach, $G \underset{p}{\to} G'$ iff there exists an embedding $B_1 \to G$. This is equivalent to the gluing condition by Proposition (5.9). If b_1 is injective then D is uniquely defined and so is G' [10, p. 25, Lemma 3.7]. This fact corresponds to the unicity result of Proposition (5.8). The efficiency of an algorithm for constructing G' from p and $g: B_1 \to G$ is considered in [12] and [33].

In order to compare the two notions of graph rewritings precisely, we introduce some notations for manipulating square diagrams. A diagram Δ of the form



is linearly denoted by

$$\Delta = (A \xrightarrow{b} B \xrightarrow{d_1} D, A \xrightarrow{c} C \xrightarrow{d_2} D).$$

The object A is called the *pivot* of Δ. Two diagrams Δ and Δ' where Δ is as above and

$$\Delta' = (A' \xrightarrow{b'} B' \xrightarrow{d_1'} D', A' \xrightarrow{c'} C' \xrightarrow{d_2'} D')$$

are *isomorphic* if there exist four isomorphisms $h_X: X \to X'$ (for $X = A, B, C, D$) such that all diagrams analogous to

$$(A \xrightarrow{h_A} A' \xrightarrow{c'} C', A \xrightarrow{c} C \xrightarrow{h_C} C')$$

commute. We say that $h = (h_A, h_B, h_C, h_D)$ is an isomorphism $\Delta \to \Delta'$.

In the double square diagram of Definition (5.11) the homomorphism g formalizes the fact that B_1 "appears in G" and that B_2 can be substituted for it. To make this precise we must choose the category $\mathscr{C}$. We take for $\mathscr{C}$ the category $\mathbf{CG}(A)_0$ of concrete 0-graphs over A with edge-injective homomorphisms (they have been defined in Definition (5.2)). We call it $\mathscr{E}$ in the sequel by assuming that A is known from the context.

Note that an embedding $h: G \to G'$ where G is an n-graph and G' is an n'-graph is an edge-injective homomorphism $G^0 \to G'^0$. We comment on the restriction to edge-injective homomorphisms as follows.

We consider a graph as a set of edges "glued" together by means of vertices (and *not* as a set of vertices connected by edges). Hence we consider that G "appears in G'" if the edges of G can be mapped *injectively* into $\mathbf{E}_{G'}$. Of course, the "gluings" of the edges and their labels must be preserved in this mapping. This is ensured by conditions (E1) and (E2) of the definition of an embedding. New gluings can occur in the embedding of G into G' so that we do not require that h_V is injective too.

Our next aim is to establish that the $\mathscr{E}$-rewritings and the rewritings of Definition (5.1) have the same power (up to the isomorphism of graphs). A few technical lemmas are needed.

Let $H, K \in \mathbf{CG}(A)_0$, let $G \in \mathbf{CG}(A)_n$, let e be an edge of type n in K, and let K_e denote the graph K minus the edge e (formally $\mathbf{E}_{K_e} = \mathbf{E}_K - \{e\}$, $\mathbf{V}_{K_e} = \mathbf{V}_K$, $\mathbf{lab}_{K_e} = \mathbf{lab}_K \restriction \mathbf{E}_{K_e}$, $\mathbf{vert}_{K_e} = \mathbf{vert}_K \restriction \mathbf{E}_{K_e}$). Let us assume that $H = K[G/e]$, and let us consider the following diagram in $\mathscr{E}$:

$$\begin{array}{ccc} [n] & \xrightarrow{\mathbf{src}_G} & G \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle i} \\ K_e & \xrightarrow[j]{} & H = K[G/e] \end{array}$$

where $[n]$ is the discrete 0-graph $\mathbf{n}^0$ $(= \sigma_\varnothing(\mathbf{n}))$ (we may have $n = 0$), i is the embedding of G into $H = K[G/e]$ (see Lemma (5.4)), $\mathbf{src}_G$ also denotes the homomorphism $(\mathbf{src}_G, \varnothing)$, $k = (k_V, \varnothing)$ where $k_V(i) = \mathbf{vert}_K(e, i)$ for $i \in [n]$, and $j = (j_V, j_E)$ where j_E is the inclusion map $\mathbf{E}_K - \{e\} \to \mathbf{E}_H$ and $j_V = f \restriction \mathbf{V}_K$. (Here f is the mapping $\mathbf{V}_K \cup \mathbf{V}_G \to \mathbf{V}_H$ used to define $H = K[G/e]$, as in Remark (5.5).) We denote this diagram by $\Delta(G, K, e)$.

(5.12) Lemma.

(1) *$\Delta(G, K, e)$ is a push-out in $\mathscr{E}$.*

(2) *Every push-out diagram in $\mathscr{E}$ with a discrete pivot is isomorphic to some $\Delta(G, K, e)$.*

Proof. (1) We have pointed out in Remark (5.5) that

$$([n] \xrightarrow{\mathrm{src}_G} \mathbf{V}_G \xrightarrow{i_\mathbf{V}} \mathbf{V}_H, [n] \xrightarrow{k_\mathbf{V}} \mathbf{V}_K \xrightarrow{j_\mathbf{V}} \mathbf{V}_H)$$

forms a push-out in the category of sets. Since $i_\mathbf{E}(\mathbf{E}_G) \cup j_\mathbf{E}(\mathbf{E}_{K_e}) = \mathbf{E}_H$ the diagram

$$(\varnothing \xrightarrow{\varnothing} \mathbf{E}_{K_e} \xrightarrow{j_\mathbf{E}} \mathbf{E}_H, \varnothing \xrightarrow{\varnothing} \mathbf{E}_G \xrightarrow{i_\mathbf{E}} \mathbf{E}_H)$$

is also a push-out in the category of sets. Together with some verifications concerning **lab** and **vert**, this suffices to prove that $\Delta(G, K, e)$ is a push-out in the category $\mathscr{C}$. It is also one in the category of concrete 0-graphs over A with arbitrary (not necessarily edge-injective) homomorphisms. The proof of a similar result can be found in [33, Lemma 2.3, p. 339].

(2) Given a push-out diagram in $\mathscr{C}$ with a discrete (possibly empty) pivot, it is easy to construct G, K, and e. The isomorphism of $\Delta(G, K, e)$ and the given diagram follow from the universality property (i.e., the unicity up to isomorphisms) of push-outs. □

Remark. We have defined $\mathscr{C}$ as a category of *concrete* graphs and an $\mathscr{C}$-rewriting rule as a quintuple $p = (B_1 \xleftarrow{b_1} I \xrightarrow{b_2} B_2)$. Since push-outs are defined up to isomorphism, applying p to some concrete graph H produces a graph H' that is defined up to isomorphism, hence is actually an abstract graph. Hence $H \underset{p}{\Rightarrow} H'$ is defined for abstract graphs.

In rule p the graphs B_1, I, and B_2 can also be considered as abstract graphs. However, for proofs involving constructions on graphs, it is frequently convenient to assume that B_1, I, and B_2 are concrete graphs.

(5.13) Proposition. *Every graph rewriting rule is equivalent to an $\mathscr{C}$-rewriting rule.*

Proof. Let $p = (G, G')$ be a graph rewriting rule where G and $G' \in \mathbf{G}(A)_n$. Let $\bar{p}$ be the $\mathscr{C}$-production $(G^0 \xleftarrow{\mathrm{src}_G} [n] \xrightarrow{\mathrm{src}_{G'}} G'^0)$. Let $H, H' \in \mathbf{G}(A)_0$ be such that $H \underset{p}{\rightarrow} H'$. This means that, for some $K \in \mathbf{G}(A \cup \{u\})_0$ with a unique occurrence of u, we have

$$H = K[G/u],$$

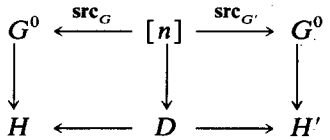
$$H' = K[G'/u],$$

hence by Lemma (5.12) a double push-out as follows:

$$\begin{array}{ccccc} G^0 & \xleftarrow{\mathrm{src}_G} & [n] & \xrightarrow{\mathrm{src}_{G'}} & G'^0 \\ \downarrow & & \downarrow & & \downarrow \\ K[G/u] = H & \longleftarrow & K_u & \longrightarrow & H' = K[G'/u] \end{array}$$

In this diagram by K_u we mean the graph K_e where e is the unique edge in K labeled u. Hence H rewrites into H' via $\bar{p}$. Let us conversely assume that H

rewrites into H' via $\bar{p}$ by means of a double push-out diagram of the form below (where all graphs are concrete):



We can build K, e such that $D = K_e$ and by part 2 of Lemma (5.12) this double square is isomorphic to the one constructed in the first part of the proof (over G^0, G'^0, K) □

It is clear that every $\mathscr{E}$-rewriting rule of the form $p = (B_1 \xleftarrow{b_1} I \xrightarrow{b_2} B_2)$ where I is discrete (we shall say that p is *discrete*) can be considered (up to isomorphism) as associated with a graph rewriting rule (G_1, G_2). It suffices to take a bijection $i: [n] \to \mathbf{V}_I$ and to define G_j as the 0-graph B_j enriched with a source mapping $\mathbf{src}_{G_j}(s) = b_j(i(s))$ for $s \in [n]$, $j = 1, 2$. Hence in order to prove that the graph rewriting rules are as powerful as the $\mathscr{E}$-rewriting rules we need only prove that the interface graph in each rule may be restricted to the discrete underlying graph without changing the power of the rule.

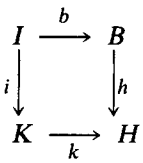
This result is actually proved in Ehrig *et al.* [14, Proposition 3.3, p. 173] and mentioned in Ehrig [10, p. 18]. (The existence of edges in interface graphs is important for Church-Rosser properties: see pp. 28-37 of [10]). We reestablish it since our category is not exactly the same.

For this purpose we introduce some notations:

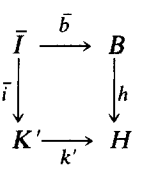
(1) For every graph G we denote by $\bar{G}$ the discrete graph obtained by deleting all its edges.

(2) If $b = (b_{\mathbf{V}}, b_{\mathbf{E}})$ is a homomorphism $G \to G'$ then we denote by $\bar{b}$ the homomorphism $(b_{\mathbf{V}}, \varnothing)$: $\bar{G} \to G'$.

(3) If Δ is a diagram in $\mathscr{E}$ of the form



we denote by $\bar{\Delta}$ the diagram



where K' is K minus the edges in $i_{\mathbf{E}}(\mathbf{E}_I)$ and k' is the restriction of k to K'.

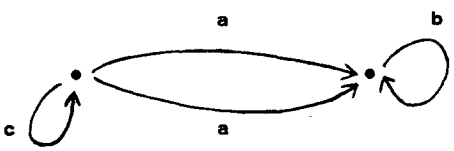
It is clear that $\bar{\Delta}$ is commutative if Δ is commutative.

(5.14) Lemma. *Let Δ be a commutative diagram. Then Δ is a push-out in $\mathscr{C}$ iff $\bar{\Delta}$ is a push-out in $\mathscr{C}$.*

Proof. Let us assume that Δ is a push-out. Consider a pair of morphisms $m_1: B \to M$ and $m_2': K' \to M$ for some M such that $m_1 \circ \bar{b} = m_2' \circ \bar{i}$. There is a unique way to extend m_2' into $m_2: K \to M$ such that $m_1 \circ b = m_2 \circ i$; for e in $i_E(\mathbf{E}_I)$, say of the form $i_E(e_1)$, we take $m_{2E}(e) = m_{1E}(b_E(e_1))$. Since i_E is injective, m_{2E} is well defined. Then, since Δ is a push-out, there is a unique morphism $j: H \to M$ such that $j \circ h = m_1$ and $j \circ k = m_2$, and j is the required morphism establishing that $\bar{\Delta}$ is a push-out.

We omit the remaining technical details. □

(5.15) Remark. This lemma is not valid in the category of graphs with arbitrary homomorphisms. It suffices to consider the example below (see Figure 7). The diagram $\bar{\Delta}$ is *not* a push-out. The graph resulting from the push-out has two edges labeled a instead of one:



If $p = (B_1 \xleftarrow{b_1} I \xrightarrow{b_2} B_2)$ is an $\mathscr{C}$-rewriting rule, we define

$$\bar{p} = (B_1 \xleftarrow{\bar{b}_1} \bar{I} \xrightarrow{\bar{b}_2} B_2).$$

(5.16) Lemma. *For all graphs H_1 and H_2, H_1 rewrites into H_2 via p iff H_1 rewrites into H_2 via $\bar{p}$.*

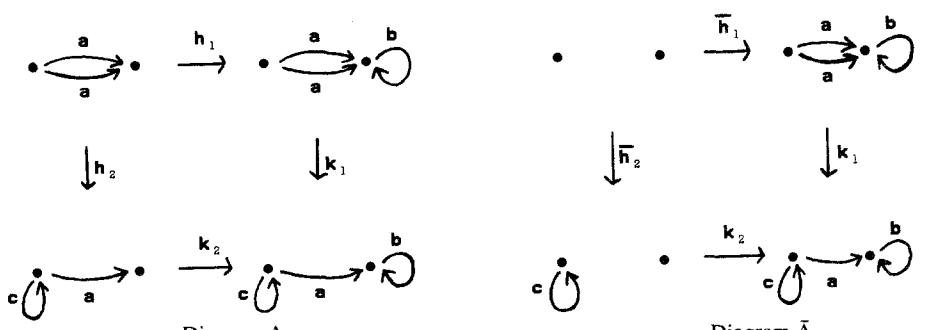
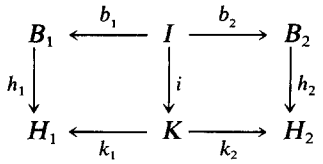


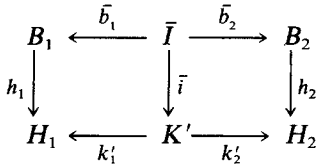
Diagram Δ Diagram $\bar{\Delta}$

Fig. 7

Proof. Let us consider a double push-out $\Delta_1\Delta_2$:



It follows from Lemma (5.12) that the associated diagram $\bar{\Delta}_1\bar{\Delta}_2$



is also a double push-out. This proves the "only-if" part of the lemma. Conversely, let us assume that a diagram of the form $\bar{\Delta}_1\bar{\Delta}_2$ above is a double push-out. We first define K, i, k_1, and k_2. Consider an edge e in I with a sequence of vertices $v_1, \ldots, v_m$. We make it into an edge $\hat{e}$ in K having the label e and the sequence of vertices $\bar{i}_{\mathbf{V}}(v_1), \ldots, \bar{i}_{\mathbf{V}}(v_m)$. We do this for all e in $\mathbf{E}_I$. Hence, formally:

$$\mathbf{V}_K = \mathbf{V}_{K'},$$

$$\mathbf{E}_K = \mathbf{E}_{K'} \cup \{\hat{e} / e \in \mathbf{E}_I\},$$

$$i_{\mathbf{V}} = \bar{i}_{\mathbf{V}},$$

$$i_{\mathbf{E}}(e) = \hat{e} \qquad \text{for all } e \text{ in } \mathbf{E}_I,$$

$$k_{j\mathbf{V}} = k'_{j\mathbf{V}} \qquad \text{for } j = 1, 2,$$

$$k_{j\mathbf{E}}(e) = k'_{j\mathbf{E}}(e) \qquad \text{if } e \in \mathbf{E}_K, \quad j = 1, 2,$$

$$k_{j\mathbf{E}}(\hat{e}) = h_j(b_j(e)) \qquad \text{if } e \in \mathbf{E}_I, \quad j = 1, 2.$$

This defines a commutative double-square $\Delta_1\Delta_2$. Hence it follows from Lemma (5.14) that $\Delta_1\Delta_2$ is a double push-out. This proves the "if" part of the lemma. □

The following theorem follows from Lemma (5.16) and Proposition (5.13).

(5.17) Theorem. *The graph rewriting rules and the $\mathscr{E}$-rewriting rules have the same power.*

(5.18) Remark. Lemma (5.16) holds if one requires in the double push-out defining the rewriting of H_1 into H_2 via p that i is edge injective. In Ehrig *et al.* [14] such a restriction is imposed and they obtain Proposition 3.3 which is nothing else than our Lemma (5.16).

6. Conclusions

This paper presents the first steps of a truly algebraic theory of graphs. The main directions for future research are the following:

(1) Constructing derived operations allowing the interpretation of the theory of magmoids [1], the theory of flowcharts *à la* Elgot and others [3], [16], or Schmeck [34] as subtheories (or derived theories) of the present one.
(2) Investigating the recognizable sets of graphs (following the terminology of Mezei and Wright [29]). This is done in Courcelle [6] where the relations between the recognizability and the monadic second-order definability of a set of graphs are investigated.
(3) Constructing an algebraic theory of infinite graphs, in particular of regular infinite graphs extending the theory of infinite trees developed in [4]. This is done in [2] and [9].
(4) Using the powerful existing tools [22], [23], [25], [26] on term rewriting systems modulo equivalence (namely $\underset{R}{\leftrightarrow}$) in order to obtain confluence properties for graph rewriting systems (as investigated by Ehrig [10] and also by Raoult [32] but in a different way).
(5) NLC graph grammars (Janssens and Rozenberg [24]) do not fit in the present formalism, but a similar algebraic theory can be developed for them. Some progress has been made in this direction in [7] where the notion of a context-free NLC graph grammar is introduced.

Acknowledgments

We thank J. Engelfriet, H. Ehrig, and the referees for their numerous comments and suggestions.

References

[1] Arnold, A., and Dauchet M., Théorie des magmoïdes, *RAIRO Inform. Théor.*, **12** (1978), 235-257.
[2] Bauderon M., On infinite graphs defined by equations, Research Report, Bordeaux I University (to appear).
[3] Bloom, S., Elgot, C., and Wright J., Solutions to the iteration equation and extensions of the scalar iteration operation, *SIAM J. Comput.*, **9** (1980), 25-45.
[4] Courcelle, B., Fundamental properties of infinite trees, *Theoret. Comput. Sci.*, **25** (1983), 95-169.
[5] Courcelle, B., Equivalences and transformations of regular systems. Applications to recursive program schemes and grammars, *Theoret. Comput. Sci.*, **46** (1986), 1-122.
[6] Courcelle, B., Recognizability and 2nd order definability for sets of finite graphs, Research Report 8634, Bordeaux I University, 1986.
[7] Courcelle, B., An axiomatic definition of context-free rewriting and its application to NLC graph grammars, *Theoret. Comput. Sci.*, **55** (1988) (to appear).
[8] Courcelle, B., A representation of graphs by algebraic expressions and its use for graph rewriting systems, *Proc. 3rd International Workshop on Graph Grammars*, Warrenton, Virginia, 1986 (to appear in 1987 in Lecture Notes Comput. Sci, Springer-Verlag, Berlin,).
[9] Courcelle, B., On some sets of countable graphs having a decidable monadic second-order theory, Research Report 8729, Bordeaux I University, 1987.

[10] Ehrig, H., Introduction to the algebraic theory of graphs, in *Proc. 1st International Workshop on Graph Grammars*, Lecture Notes Comput. Sci., Vol. 73, Springer-Verlag, Berlin, 1979, pp. 1-69.

[11] Ehrig, H., and Kreowski, H. J., Push-out properties: an analysis of gluing constructions for graphs, *Math. Nachr.*, **91** (1979), 135-149.

[12] Ehrig, H., Kreowski, H. J., Maggiolo-Schettini, A., Rosen, B. R., and Winkowski, J., Transformations of structures: an algebraic approach, *Math. Systems Theory*, **14** (1981), 305-334.

[13] Ehrig, H., and Mahr, B., *Fundamentals of Algebraic Specifications*, EATCS Monographs, Vol. 6, Springer-Verlag, Berlin, 1985.

[14] Ehrig, H., Pfender, M., and Schneider, H., Graph grammars: An algebraic approach, *Proc. 14th IEEE Symp. on Switching and Automata Theory*, Iowa City, 1973, pp. 167-180.

[15] Eilenberg, S., and Wright, J., Automata in general algebras, *Inform. and control*, **11** (1967), 52-70.

[16] Elgot, C., Monadic computation and iterative algebraic theories, *Proc. Logic Colloq. 73*, North Holland, Amsterdam, 1975, pp. 175-230.

[17] Engelfriet, J., Schmidt, E., IO and OI, *J. Comput. System Sci.*, **15** (1977), 328-353 and **16** (1978), 67-99.

[18] Ginsburg, S., and Rice, H., Two families of languages related to ALGOL, *J. Assoc. Comput. Mach.*, **9** (1962), 350-371.

[19] Goguen, J., Thatcher, J., Wagner, E., and Wright, J., Initial algebra semantics and continuous algebras, *J. Assoc. Comput. Mach.*, **24** (1977), 68-95.

[20] Habel, A., and Kreowski, H. J., On context-free graph languages generated by edge replacements, in *Proc. 2nd International Workshop on Graph Grammars*, Lecture Notes Comput. Sci., Vol. 153, Springer-Verlag, Berlin, 1983, pp. 143-158.

[21] Habel, A., and Kreowski, H. J., Some structural aspects of hypergraph languages generated by hyperedge replacements, Preprint, October 1985 (a shortened version can be found in Lecture Notes Comput. Sci., Vol. 247, Springer-Verlag, Berlin, 1987, pp. 207-219).

[22] Huet, G., Confluent reduction: abstract propertiies and applications to term rewriting systems, *J. Assoc. Comput. Mach.*, **27** (1980), 797-821.

[23] Huet, G., and Oppen, D., Equations and rewrite rules, a survey, in *Formal Languages, Perspectives and Open Problems* (R. Book, ed.), Academic Press, New York, 1980.

[24] Janssens D., and Rozenberg, G., A survey of NLC grammars, in *Proc. CAAP' 83*, L'Aquila, Lecture Notes Comput. Sci., Vol 159, Springer-Verlag, Berlin, 1983, pp. 114-128.

[25] Jouannaud, J. P. (ed.), *Rewriting Techniques and Applications* (Colloquium held in Dijon, May 1985), Lecture Notes Comput. Sci., Vol. 202, Springer-Verlag, Berlin, 1985.

[26] Lescanne, P. (ed.), *Proc. Second Conference on Rewriting Techniques and Applications*, Bordeaux, May 1987, Lecture Notes Comput. Sci., Vol. 256, Springer-Verlag, Berlin, 1987.

[27] McLane, S., *Category Theory for the Working Mathematician*, Springer-Verlag, Berlin, 1971.

[28] Manes, E., *Algebraic Theories*, Springer-Verlag, Berlin, 1976.

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

[30] Nagl, M., Bibliography on graph-rewriting systems (graph grammars), in *Proc. 2nd International Workshop on Graph Grammars*, Lecture Notes Comput. Sci., Vol. 153, Springer-Verlag, Berlin, pp. 415-448.

[31] Petrov, S., Graph grammars and automata (survey), *Automat. Remote Control*, **39** (1978), 1034-1050.

[32] Raoult, J. C., On graph rewritings, *Theoret. Comp. Sci.*, **32** (1984), 1-24.

[33] Rosen, B., Deriving graphs from graphs by applying a production, *Acta Inform.*, **4** (1975), 337-357.

[34] Schmeck, H., Algebraic characterization of reducible flowcharts, *J. Comput. System Sci.*, **27**(2) (1983), 165-199.

Received October, 1985, and in revised form October 30, 1986, and July 22, 1987.