A Reformulation of Matrix Graph Grammars with Boolean Complexes

Pedro Pablo Pérez Velasco, Juan de Lara

Escuela Politécnica Superior
Universidad Autónoma de Madrid, Spain

{pedro.perez, juan.delara}@uam.es

Submitted: Jul 16, 2008; Accepted: Jun 10, 2009; Published: Jun 19, 2009
Mathematics Subject Classifications: 05C99, 37E25, 68R10, 97K30, 68Q42

Abstract

Graph transformation is concerned with the manipulation of graphs by means of rules. Graph grammars have been traditionally studied using techniques from category theory. In previous works, we introduced Matrix Graph Grammars (MGG) as a purely algebraic approach for the study of graph dynamics, based on the representation of simple graphs by means of their adjacency matrices.

The observation that, in addition to positive information, a rule implicitly defines negative conditions for its application (edges cannot become dangling, and cannot be added twice as we work with simple digraphs) has led to a representation of graphs as two matrices encoding positive and negative information. Using this representation, we have reformulated the main concepts in MGGs, while we have introduced other new ideas. In particular, we present (i) a new formulation of productions together with an abstraction of them (so called *swaps*), (ii) the notion of *coherence*, which checks whether a production sequence can be potentially applied, (iii) the minimal graph enabling the applicability of a sequence, and (iv) the conditions for compatibility of sequences (lack of dangling edges) and G-congruence (whether two sequences have the same minimal initial graph).

1 Introduction

Graph transformation [1, 2, 14] is concerned with the manipulation of graphs by means of rules. Similar to Chomsky grammars for strings, a graph grammar is made of a set of rules, each having a left and a right hand side graphs (LHS and RHS) and an initial host graph, to which rules are applied. The application of a rule to a host graph is called a derivation step and involves the deletion and addition of nodes and edges according to the rule specification. Roughly, when an occurrence of the rule's LHS is found in the graph, then it can be replaced by the RHS. Graph transformation has been successfully applied in many areas of computer science, for example, to express the valid structure of graphical languages, for the specification of system behaviour, visual programming, visual simulation, picture processing and model transformation (see [1] for an overview of applications). In particular, graph grammars have been used to specify computations on graphs, as well as to define graph languages (i.e. sets of graphs with certain properties), thus being possible to "translate" *static* properties of graphs such as coloring into equivalent properties of dynamical systems (grammars).

In previous work [9, 10, 11, 12] we developed a new approach to the transformation of *simple* digraphs. Simple graphs and rules can be represented with Boolean matrices and vectors and the rewriting can be expressed using Boolean operators only. One important point of MGGs is that, as a difference from other approaches [2, 14], it explicitly represents the rule dynamics (addition and deletion of elements), instead of only the static parts (pre- and post- conditions). Apart from the practical implications, this fact facilitates new theoretical analysis techniques such as, for example, checking independence of a sequence of arbitrary length and a permutation of it, or obtaining the smallest graph able to fire a sequence. See [12] for a detailed account.

In [11] we improved our framework with the introduction of the *nihilation matrix*, which makes explicit some implicit information in rules: elements that, if present in the host graph, disable a transformation step. These are all edges not included in the left-hand side (LHS), adjacent to nodes deleted by the rule (which would become dangling) and edges that are added by the production, as in simple digraphs parallel edges are forbidden. In this paper, we further develop this idea, as it is natural to consider that a production transforms pairs of graphs, a "positive" one with elements that must exist (identified by the LHS), and a "negative" one, with forbidden elements (identified by the nihilation matrix), which we call a boolean complex. Thus, using boolean complexes, we have provided a new formulation of productions, and introduced an abstraction called *swap* that facilitates rule classification and analysis. Then, we have recasted the fundamental concepts of MGGs using this new formulation, namely: *coherence*, which checks whether a production sequence can be potentially applied, the image of a sequence, the minimal graph enabling the applicability of a sequence, the conditions for compatibility of sequences (lack of dangling edges) and G-congruence (whether two sequences have the same minimal initial graph). Some aspects of the theory are left for further research, such as constraints, application conditions and reachability (see [12]).

The rest of the paper is organized as follows. Section 2 gives a brief overview of the basic concepts of MGG. Section 3 introduces Boolean complexes along with the basic operations defined for them. Section 4 encodes productions as Boolean complexes and relates operations on graphs with operations on Boolean complexes. Section 5 studies coherence of sequences of productions and Section 6 initial digraphs and the image of a sequence. Section 7 generalizes other sequential results of MGG such as compatibility and G-congruence. Finally, Section 8 ends with the conclusions and further research.

2 Matrix Graph Grammars: Basic Concepts

In this section we give a very brief overview of some of the basics of MGGs, for a detailed account and accesible presentation, the reader is referred to [12].

Graphs and Rules. We work with simple digraphs, which we represent as (M, V) where M is a Boolean matrix for edges (the graph *adjacency* matrix) and V a Boolean vector for vertices or nodes. We explicitly represent the nodes of the graph with a vector because rules may add and delete nodes, and thus we mark the existing nodes with a 1 in the corresponding position of the vector. Although nodes and edges can be assigned a type (as in [11]) here we omit it for simplicity.

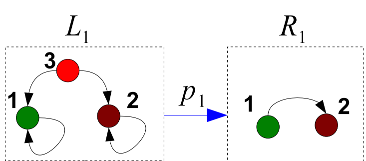
A production, or rule, $p : L \to R$ is a partial injective function of simple digraphs. Using a *static formulation*, a rule is represented by two simple digraphs that encode the left and right hand sides.

Definition 2-1 (Static Formulation of Production). A production $p : L \to R$ is statically represented as $p = (L = (L^E, L^V); R = (R^E, R^V))$, where E stands for edges and V for vertices.

A production adds and deletes nodes and edges; therefore, using a *dynamic formulation*, we can encode the rule's pre-condition (its LHS) together with matrices and vectors to represent the addition and deletion of edges and nodes.

Definition 2-2 (Dynamic Formulation of Production). A production $p : L \to R$ is dynamically represented as $p = (L = (L^E, L^V); e^E, r^E; e^V, r^V)$, where e^E and e^V are the deletion Boolean matrix and vector, r^E and r^V are the addition Boolean matrix and vector (with a 1 in the position where the element is deleted or added respectively).

The right-hand side of a rule p is calculated by the Boolean formula $R = p(L) = r \vee \overline{e}\, L$, which applies to nodes and edges. The $\wedge$ (**and**) symbol is usually omitted in formulae. In order to avoid ambiguity, **and** has precedence over **or**. The **and** and **or** operations between adjacency matrices are defined componentwise.



$$p_1 = \left(\left(L^E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, L^V = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right); \left(R^E = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, R^V = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \right)$$

$$p_1 = \left(\left(L^E, L^V \right); e^E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, r^E = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; e^V = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, r^V = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Figure 1: Simple Production Example (left). Matrix Representation, Static and Dynamic (right)

Example. Figure 1 shows an example rule and its associated matrix representation, in its static (right upper part) and dynamic (right lower part) formulations ■

In MGGs, we may have to operate graphs of different sizes (i.e. matrices of different dimensions). An operation called *completion* [9] rearranges rows and columns (so that the elements that we want to identify match) and inserts zero rows and columns as needed. For example, if we need to operate with graphs L_1 and R_1 in Fig. 1, completion adds a third row and column to R^E (filled with zeros) as well as a third element (a zero) to vector R^V.

A sequence of productions $s = p_n; \ldots ; p_1$ is an ordered set of productions in which p_1 is applied first and p_n is applied last. The main difference with composition $c = p_n \circ \ldots \circ p_1$ is that c is a single production. Therefore, s has $n-1$ intermediate states plus initial and final states, while c has just an initial state plus a final state. Often, sequences are said to be *completed*, because an identification of nodes and edges accross productions has been chosen and the matrices of the rules have been rearranged accordingly. This is a way to decide if two nodes or edges in different productions will be identified to the same node or edge in the host graph (the graph in which the sequence will be applied).

Compatibility. A graph (M, V) is compatible if M and V define a simple digraph, i.e. if there are no dangling edges (edges incident to nodes that are not present in the graph). A rule is said to be *compatible* if its application to a simple digraph yields a simple digraph (see [12] for the conditions). A sequence of productions $s_n = p_n; \ldots ; p_1$ (where the rule application order is from right to left) is compatible if the image of $s_m = p_m; \ldots ; p_1$ is compatible, $\forall m \leq n$.

Nihilation Matrix. In order to consider the elements in the host graph that disable a rule application, rules are extended with a new graph K. Its associated matrix specifies the two kinds of forbidden edges: those incident to nodes deleted by the rule and any edge added by the rule (which cannot be added twice, since we are dealing with simple digraphs).[1]

According to the theory developed in [12], the derivation of the nihilation matrix can be automatized because

$$K = p\left(\overline{D}\right) \quad \text{with} \quad D = \overline{e^V} \otimes \overline{e^V}^t,$$

where transposition is represented by ^{t}. The symbol $\otimes$ denotes the Kronecker product, a special case of tensor product. If A is an m-by-n matrix and B is a p-by-q matrix, then the Kronecker product $A \otimes B$ is the mp-by-nq block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

For example, if $e^V = [0\ 1\ 0]$, then

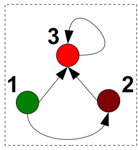
$$D = \overline{e^V} \otimes \overline{e^V}^t = \begin{bmatrix} 1 \cdot [1\ 0\ 1]^t & 0 \cdot [1\ 0\ 1]^t & 1 \cdot [1\ 0\ 1]^t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Please note that given an arbitrary LHS L, a valid nihilation matrix K should satisfy $L^E K = 0$, that is, the LHS and the nihilation matrix should not have common edges.

Example. The left of Fig. 2 shows, in the form of a graph, the nihilation matrix of the rule depicted in Fig. 1. It includes all edges incident to node 3 that were not explicitly deleted and all edges added by p_1. To its right we show the full formulation of p_1 which includes the nihilation matrix ■

[1] Nodes are not considered because their addition does not generate conflicts of any kind.

K



$$p_1 = \left(\left(L^E, L^V\right), K = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}; e^E, r^E; e^V, r^V \right)$$

$p_1^{-1}(K) = Q$

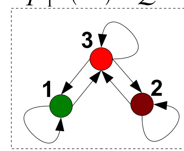


Figure 2: Nihilation Graph (left). Full Formulation of Prod. (center). Evolution of K (right)

As proved in [12] (Prop. 7.4.5), the evolution of the nihilation matrix is fixed by the production. If $R = p(L) = r \vee \overline{e}L$ then

$$Q = p^{-1}(K) = e \vee \overline{r}K, \tag{1}$$

being Q the nihilation matrix[2] of the right hand side of the production p. Hence, we have that $(R, Q) = (p(L), p^{-1}(K))$. Notice that $Q \neq \overline{D}$ in general though it is true that $\overline{D} \subset Q$.

Example. The right of Fig. 2 shows the change in the nihilation matrix of p_1 when the rule is applied. As node 3 is deleted, no edge is allowed to stem from it. Self-loops from nodes 1 and 2 are deleted by p so they cannot appear in the resulting graph ■

We can depict a rule $p : L \to R$ as $R = p(L) = \langle L, p \rangle$, splitting the static part (initial and final states, L and R) from the dynamics (element addition and deletion, p).

Direct Derivation. A direct derivation consists in applying a rule $p \colon L \to R$ to a graph G, through a match $m \colon L \to G$ yielding a graph H. In MGG we use injective matchings, so given $p : L \to R$ and a simple digraph G any $m : L \to G$ total injective morphism is a match for p in G. The match is one of the ways of *completing* L in G. In MGG we do not only consider the elements that should be present in the host graph G (those in L) but also those that should not be (those in the nihilation matrix, K). Hence two morphisms are sought: $m_L \colon L \to G$ and $m_K \colon K \to \overline{G}$, where $\overline{G}$ is the complement of G, which in the simplest case is just its negation. In general, the complement of a graph may take place inside some bigger graph. See [11] or Ch. 5 in [12]. For example, L will normally be a subgraph of G. The negation of L is of the same size ($\overline{L}$ has the same number of nodes), but not its complement inside G which would be as large as G.

Definition 2-3 (Direct Derivation). Given a rule $p : L \to R$ and a graph $G = (G^E, G^V)$ as in Fig. 3(a), $d = (p, m)$ – with $m = (m_L, m_K)$ – is called a direct derivation with result $H = p^* (G)$ if the following conditions are fulfilled:

1. There exist $m_L : L \to G$ and $m_K : K \to \overline{G^E}$ total injective morphisms.

2. $m_L(n) = m_K(n), \forall n \in L^V$.

3. The match m_L induces a completion of L in G. Matrices e and r are then completed in the same way to yield e^* and r^*. The output graph is calculated as $H = p^*(G) = r^* \vee \overline{e^*}G$.

[2] In [12], K is written N_L and Q is written N_R. We shall use subindices when dealing with sequences in Sec. 7, hence the change of notation. In the definition of production, L stands for *left* and R for *right*. The letters that preceed them in the alphabet (K and Q) have been chosen.

$$
\begin{tikzcd}
K \arrow[d, "m_K"'] \\
\overline{G^E}
\end{tikzcd}
\qquad
\begin{tikzcd}
L \arrow[r, "p"] \arrow[d, "m_L"'] \arrow[dr, phantom, "="] & R \arrow[d, dotted, "m_L^*"] \\
G \arrow[r, dotted, "p^*"] & H
\end{tikzcd}
$$

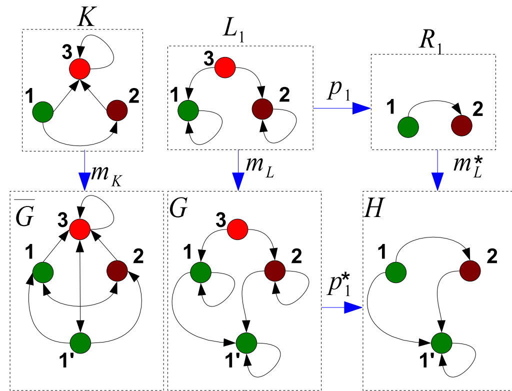


Figure 3: Direct Derivation (left). Example (right)

Remarks. The square in Fig. 3 (a) is a categorical pushout (also known as a fibered coproduct or a cocartesian square). The pushout is a universal construction, hence, if it exists, is unique up to a unique isomorphism. It univoquely defines H, p^* and m_L^* out of L, R and p.

Item 2 in the definition is needed to ensure that L and K are matched to the same nodes in G.

Example The right of Fig. 3 depicts a direct derivation example using rule p_1 shown in Fig. 1, which is applied to a graph G yielding graph H. A morphism from the nihilation matrix to the complement of G, $m_K\colon K \to \overline{G}$, must also exist for the rule to be applied ■

Analysis Techniques. In [9, 10, 11, 12] we developed some analysis techniques for MGG. One of our goals was to analyze rule sequences independently of a host graph. For its analysis, we *complete* the sequence by identifying the nodes across rules which are assummed to be mapped to the same node in the host graph (and thus rearrange the matrices of the rules in the sequences accordingly). Once the sequence is completed, our notion of sequence *coherence* [9] allows us to know if, for the given identification, the sequence is potentially applicable, i.e. if no rule *disturbs* the application of those following it. For the sake of completeness:

Definition 2-4 (Coherence of Sequences). The completed sequence $s = p_n; \ldots; p_1$ is *coherent* if the actions of p_i do not prevent those of p_k, $k > i$, for all $i, k \in \{1, \ldots, n\}$.

Closely related to coherence are the notions of minimal and negative initial digraphs, MID and NID, resp. Given a completed sequence, the minimal initial digraph is the smallest graph that allows its application. Conversely, the negative initial digraph contains all elements that should not be present in the host graph for the sequence to be applicable. Therefore, the NID is a graph that should be found in $\overline{G}$ for the sequence to be applicable (i.e. none of its edges can be found in G).

Definition 2-5 (Minimal and Negative Initial Digraphs). Let $s = p_n; \ldots; p_1$ be a completed sequence. A *minimal initial digraph* is a simple digraph which permits all operations of s and does not contain any proper subgraph with the same property. A *negative initial digraph* is a simple digraph that contains all the elements that can spoil any of the operations specified by s.

If the sequence is not completed (i.e. no overlapping of rules is decided) we can give the set of all graphs able to fire such sequence or spoil its application. These are the so-called initial and negative digraph sets in [12]. Nevertheless, they will not be used in the present contribution.

Other concepts aim at checking sequential independence (i.e. same result) between a sequence of rules and a permutation of it. *G-congruence* detects if two sequences (one permutation of the other) have the same MID and NID.

Definition 2-6 (G-congruence). Let $s = p_n; \ldots ; p_1$ be a completed sequence and $\sigma(s) = p_{\sigma(n)}; \ldots ; p_{\sigma(1)}$, being σ a permutation. They are called G-congruent (for graph congruent) if they have the same minimal and negative initial digraphs.

G-congruence conditions return two matrices and two vectors, representing two graphs, which are the differences between the MIDs and NIDs of each sequence. Thus, if zero, the sequences have the same MID and NID. It can be proved that two coherent and compatible completed sequences that are G-congruent are sequentially independent.

All these concepts have been characterized using operators $\triangle$ and $\bigtriangledown$. They extend the structure of sequence, as explained in [12]. Their definition is included here for future reference:

$$\triangle_{t_0}^{t_1} \left(F(x,y) \right) = \bigvee_{y=t_0}^{t_1} \left(\bigwedge_{x=y}^{t_1} \left(F(x,y) \right) \right) \tag{2}$$

$$\bigtriangledown_{t_0}^{t_1} \left(G(x,y) \right) = \bigvee_{y=t_0}^{t_1} \left(\bigwedge_{x=t_0}^{y} \left(G(x,y) \right) \right). \tag{3}$$

As we have seen with the concept of the nihilation matrix, it is natural to think of the LHS of a rule as a pair of graphs encoding positive and negative information. Thus, we extend our approach by considering graphs as pair of matrices, so called Boolean complexes, that will be manipulated by rules. This new representation brings some advantages to the theory, as it allows a natural and compact handling of negative conditions, as well as a proper formalization of the functional notation $\langle L, p \rangle$ as a dot product. In addition, this new reformulation has led to the introduction of new concepts, like *swaps* (an abstraction of the notion of rule), or measures on graphs and rules. Next section introduces the theory of Boolean complexes, while the following ones use this theory to reformulate the MGG concepts we have introduced in this section.

3 Boolean Complexes

In this section we introduce Boolean complexes together with some basic operations defined on them. Also, we shall define the Preliminary Monotone Complex Algebra (*monotone* because the negation of Boolean complexes is not defined), PMCA. This algebra and the Monotone Complex Algebra to be defined in the next section permit a compact reformulation of grammar rules and sequential concepts such as independence, initial digraphs and coherence.

Definition 3-1 (Boolean Complex). A *Boolean complex* (or just a *complex*) $z = (a, b)$ consists of a *certainty* part 'a' plus a *nihil* part 'b', where a and b are Boolean matrices. Two complexes $z_1 = (a_1, b_1)$ and $z_2 = (a_2, b_2)$ are equal, $z_1 = z_2$, if and only if $a_1 = a_2$ and $b_1 = b_2$. A Boolean complex will be called *strict Boolean complex* if its certainty part is the adjacency matrix of some simple digraph and its nihil part corresponds to the nihilation matrix.

Definition 3-2 (Basic Operations). Let $z = (a, b)$, $z_1 = (a_1, b_1)$ and $z_2 = (a_2, b_2)$ be two Boolean complexes. The following operations are defined componentwise:

- Addition: $z_1 \vee z_2 = (a_1 \vee a_2, b_1 \vee b_2)$.
- Multiplication: $z_1 \wedge z_2 = z_1\, z_2 = (a_1\, a_2 \vee b_1\, b_2, a_1\, b_2 \vee a_2 b_1)$.
- Conjugation: $z^* = (\overline{b}, \overline{a})$.
- Dot Product: $\langle z_1, z_2 \rangle = z_1\, z_2^*$.

Here, componentwise means not only that the definition takes place on the certainty and on the nihil parts, but also that we use the standard Boolean operations on each element of the corresponding Boolean matrices. For example, if $a = (a_{jk})_{j,k=1,\ldots,n}$ and $b = (b_{jk})_{j,k=1,\ldots,n}$ are two Boolean matrices, then[3]

$$
\begin{aligned}
a \vee b &= (a_{jk} \vee b_{jk})_{j,k=1,\ldots,n} \\
a \wedge b &= (a_{jk} \wedge b_{jk})_{jk=1,\ldots,n} \\
\overline{a} &= (\overline{a}_{jk})_{jk=1,\ldots,n}
\end{aligned}
$$

The notation $\langle \cdot, \cdot \rangle$ for the dot product is used because it coincides with the functional notation introduced in [9, 12]. Notice however that there is no underlying linear space so this is just a convenient notation. Moreover, the dot product of two Boolean complexes is a Boolean complex and not a scalar value.

The dot product of two Boolean complexes is zero[4] (they are *orthogonal*) if and only if each element of the first Boolean complex is included in both the certainty and nihil parts of the second complex. Otherwise stated, if $z_1 = (a_1, b_1)$ and $z_2 = (a_2, b_2)$, then

$$
\langle z_1, z_2 \rangle = 0 \Longleftrightarrow a_1 \overline{a}_2 = a_1 \overline{b}_2 = b_1 \overline{a}_2 = b_1 \overline{b}_2 = 0. \tag{4}
$$

Given two Boolean matrices, we say that $a \prec b$ if $ab = a$, i.e. whenever a has a 1, b also has a 1 (graph a is *contained* in graph b). The four equalities in eq. (4) can be rephrased as $a_1 \prec a_2$, $a_1 \prec b_2$, $b_1 \prec a_2$ and $b_1 \prec b_2$. This is equivalent to $(a_1 \vee b_1) \prec (a_2 b_2)$. Orthogonality is directly related to the common elements of the certainty and nihil parts.

A particular relevant case – see eq. (8) – is when we consider the dot product of one element $z = (a, b)$ with itself. In this case we get $(a \vee b) \prec (ab)$, which is possible if and only if $a = b$. We shall come back to this issue later.

Definition 3-3 (Preliminary Monotone Complex Algebra, PMCA). The set $\mathfrak{G}' = \{z \,|\, z \text{ is a Boolean complex}\}$ together with the basic operations of Def. 3-2 will be known as *preliminary monotone complex algebra*. Besides, we shall also introduce the subset $\mathfrak{H}' = \{z = (a, b) \in \mathfrak{G}' \,|\, a \wedge b = 0\}$ with the same operations.

[3]Notice that these operations are also well defined for vectors (they are matrices as well).

[4]Zero is the matrix in which every element is a zero, and is represented by 0 or a bolded **0** if any confusion may arise. Similarly, 1 or **1** will represent the matrix whose elements are all ones.

Elements of $\mathfrak{H}'$ are the strict Boolean complexes introduced in Def. 3-1. We will get rid of the term "preliminary" in Def. 4-1, when not only the adjacency matrix is considered but also the vector of nodes that make up a simple digraph. In MGG we will be interested in those $z \in \mathfrak{G}'$ with disjoint certainty and nihil parts, i.e. $z \in \mathfrak{H}'$. We shall define a projection $\mathcal{Z} : \mathfrak{G}' \longrightarrow \mathfrak{H}'$ by $\mathcal{Z}(g) = \mathcal{Z}(a, b) = (a\overline{b}, b\overline{a})$. The mapping $\mathcal{Z}$ sets to zero those elements that appear in both the certainty and nihil parts.

A more *complex-analytical* representation can be handy in some situations and in fact will be preferred for the rest of the present contribution:

$$z = (a, b) \longmapsto z = a \vee i\, b.$$

Its usefulness will be apparent when the algebraic manipulations become a bit cumbersome, mainly in Secs. 5, 6 and 7.

Define one element i – that we will name *nil term* or *nihil term* – with the property $i \wedge i = \mathbf{1}$, being i itself not equal to $\mathbf{1}$. Then, the basic operations of Def. 3-2, following the same notation, can be rewritten:[5]

$$\begin{aligned} z^* &= \overline{b} \vee i\,\overline{a} \\ z_1 \vee z_2 &= (a_1 \vee a_2) \vee i\,(b_1 \vee b_2) \\ z_1 z_2 = z_1 \wedge z_2 &= (a_1 \vee i b_1) \wedge (a_2 \vee i b_2) = (a_1 a_2 \vee b_1 b_2) \vee i\,(a_1 b_2 \vee b_1 a_2) \\ \langle z_1, z_2 \rangle &= (a_1 \vee i b_1) \wedge \left(\overline{b}_2 \vee i \overline{a}_2\right) = \left(a_1 \overline{b}_2 \vee b_1 \overline{a}_2\right) \vee i\left(a_1 \overline{a}_2 \vee b_1 \overline{b}_2\right). \end{aligned}$$

Notice that the conjugate of a complex term $z \in \mathfrak{G}'$ that consists of certainty part only is $z^* = (a \vee i0)^* = 1 \vee i\,\overline{a}$. Similarly for one that consists of nihil part alone: $z^* = (0 \vee ib)^* = \overline{b} \vee i$. If $z \in \mathfrak{H}'$ then they further reduce to $a \vee i0$ and $0 \vee ib$ by applying the projection $\mathcal{Z}$, respectively, i.e. they are invariant.[6] Also, the multiplication reduces to the standard **and** operation if there are no nihil parts: $(a_1 \vee i0)(a_2 \vee i0) = a_1 a_2$.

Proposition 3-4. *Let* $x, y, z \in \mathfrak{G}'$ *and* $z_1, z_2 \in \mathfrak{H}'$. *Then,* $\langle x \vee y, z \rangle = \langle x, z \rangle \vee \langle y, z \rangle$, $\langle z_1, z_2 \rangle = \langle z_2, z_1 \rangle^*$ *and* $(z_1 z_2)^* = z_1^* z_2^*$.

Proof

□The first identity is fulfilled by any Boolean complex and follows directly from the definition. The other two hold in $\mathfrak{H}'$ but not necessarily in $\mathfrak{G}'$. For the second equation just write down the definition of each side of the identity:

$$\begin{aligned} \langle z_1, z_2 \rangle \; &= \left(a_1 \overline{b}_2 \vee \overline{a}_2 b_1\right) \vee i\left(a_1 \overline{a}_2 \vee b_1 \overline{b}_2\right) \\ \langle z_2, z_1 \rangle^* &= \left[a_1 \overline{b}_2 \vee \overline{a}_2 b_1 \vee \left(a_1 b_1 \vee \overline{a}_2\, \overline{b}_2\right)\right] \vee i\left[a_1 \overline{a}_2 \vee b_1 \overline{b}_2 \vee \left(a_1 b_1 \vee \overline{a}_2\, \overline{b}_2\right)\right]. \end{aligned}$$

Terms $a_1 b_1 \vee \overline{a}_2 \overline{b}_2$ vanish as they appear in both the certainty and nihil parts. The third identity is proved similarly. ■

[5]The authors did not manage to prove the existence of such element i in any domain, by any means. In the present contribution, i should be understood just as a very convenient notation that simplifies some manipulations. The reader may however stick to the representation of Boolean complexes as pairs of matrices (a, b). All formulas and final results in this paper have an easy translation from one notation into the other.

[6]Notice that $1 \vee i\overline{a} = (a \vee \overline{a}) \vee i\overline{a} = a \vee i0$ and $\overline{b} \vee i1 = \overline{b} \vee i\left(b \vee \overline{b}\right) = 0 \vee ib$.

Notice however that $(z_1 \vee z_2)^* \neq z_1^* \vee z_2^*$. It can be checked easily as $(z_1 \vee z_2)^* = [(a_1 \vee a_2) \vee i\,(b_1 \vee b_2)]^* = \overline{b}_1\overline{b}_2 \vee i\,\overline{a}_1\overline{a}_2$ but $z_1^* \vee z_2^* = \left(\overline{b}_1 \vee \overline{b}_2\right) \vee i\,(\overline{a}_1 \vee \overline{a}_2)$. This implies that, although $\langle z_1 \vee z_2, z\rangle = \langle z_1, z\rangle \vee \langle z_2, z\rangle$, we no longer have *sesquilineality*, i.e. it is not *linear* in its second component taking into account conjugacy:

$$z\left[\left(\overline{b}_1 \vee \overline{b}_2\right) \vee i\,(\overline{a}_1 \vee \overline{a}_2)\right] = \langle z, z_1 \vee z_2\rangle \neq \langle z, z_1\rangle \vee \langle z, z_2\rangle = z\left[\overline{b}_1\overline{b}_2 \vee i\,\overline{a}_1\overline{a}_2\right].$$

In fact the equality $\langle z, z_1 \vee z_2\rangle = \langle z, z_1\rangle \vee \langle z, z_2\rangle$ holds if and only if $z_1 = z_2$.

The following identities show that the dot product of one element with itself does not have nihil part, returning what one would expect. Equation (7) is particularly relevant as it states that the certainty and nihil parts are in some sense mutually exclusive, which together with eq. (8) suggest the definition of $\mathfrak{H}'$ as introduced in Sec. 3. Notice that this fits perfectly well with the interpretation of L and K in MGG given in Sec. 2.

$$\langle a \vee i\,0, a \vee i\,0\rangle = (a \vee 0\,\overline{a})\,(1 \vee ia) = (a \vee 0\,\overline{a}) \vee i\,(0 \vee a\,\overline{a}) = a \tag{5}$$

$$\langle 0 \vee i\,b, 0 \vee i\,b\rangle = (0 \vee i\,b)\left(\overline{b} \vee i\,1\right) = \left(b \vee 0\,\overline{b}\right) \vee i\left(b\,\overline{b} \vee 0\right) = b \tag{6}$$

$$\langle c \vee i\,c, c \vee i\,c)\rangle = (c \vee i\,c)\,(\overline{c} \vee i\,\overline{c}) = (c\,\overline{c} \vee c\,\overline{c}) \vee i\,(c\,\overline{c} \vee c\,\overline{c}) = 0. \tag{7}$$

The dot product of one element with itself gives rise to the following useful identity:

$$\langle z, z\rangle = z\,z^* = \left(a\overline{b} \vee \overline{a}b\right) \vee i\left(b\overline{b} \vee a\overline{a}\right) = a \oplus b, \tag{8}$$

being $\oplus$ the componentwise **xor** operation. Apart from stating that the dot product of one element with itself has no nihil part (as commented above), eq. (8) tells us how to *factorize* one of the basic Boolean operations: **xor**.

We shall introduce the notation

$$\|z\| = \langle z, z\rangle. \tag{9}$$

In some sense, $\|z\|$ *measures* how big (closer to $\mathbf{1}$) or small (closer to $\mathbf{0}$) the Boolean complex z is. It follows directly from the definition that $\|i\| = 1$ (this is just a formal identity) and $\|z^*\| = \|z\|$.

4 Production Encoding

In this section we introduce the Monotone Complex Algebra, which not only considers edges but also nodes. Compatibility issues may appear so we study compatibility for a simple digraph and also for a single production (compatibility for sequences will be addressed in Sec. 7). Next we turn to the characterization of MGG productions using the dot product of Def. 3-2. The section ends introducing *swaps*, which can be thought of as a generalization of productions. This concept will allow us to reinterpret productions as introduced in [12].

To get rid of the "preliminary" term in the definition of $\mathfrak{G}'$ and $\mathfrak{H}'$ (Def. 3-3) we shall consider an element as being composed of a (strict) Boolean complex and a vector of nodes. Hence, we have that $\mathcal{L} = \left(L^E \vee iK^E, L^V \vee iK^V\right)$ where E stands for *edge* and V for *vertex*.[7] Notice that $L^E \vee iK^E$ are matrices and $L^V \vee iK^V$ are vectors.

[7] If an equation is applied to both edges and nodes then the superindices will be omitted. They will also be omitted if it is clear from the context which one we refer to.

Definition 4-1 (Monotone Complex Algebra). The *Monotone Complex Algebra* is the set $\mathfrak{G} = \{(L^E \vee iK^E, L^V \vee iK^V) \mid L^E \vee iK^E$ and $L^V \vee iK^V$ are Boolean complexes as introduced in the paragraph above$\}$ together with the operations in Def. 3-2. Let $\mathfrak{H}$ be the subset of $\mathfrak{G}$ in which certainty and nihil parts are disjoint.

This definition extends Def. 3-3. The intuition behind $\mathfrak{G}$ (and $\mathfrak{H}$) is that $L^E \vee iK^E$ keeps track of edges while $L^V \vee iK^V$ keeps track of nodes.

Concerning $\mathfrak{G}$, a production $p : \mathfrak{G} \to \mathfrak{G}$ consists of two independent productions $p = (p_C, p_N)$ – being p_C, p_N MGG productions; see Defs. 2-1 and 2-2 – one acting on the certainty part and the other on the nihil part:

$$\mathcal{R} = p(\mathcal{L}) = p(L \vee iK) = p_C(L) \vee ip_N(K) = R \vee iQ, \tag{10}$$

where R is introduced in Def. 2-1 and Q in eq. (1). As p_C and p_N are not related to each other if we stick to $\mathfrak{G}$, it is true that $\forall g_1, g_2 \in \mathfrak{G}$, $\exists p$ such that $p(g_1) = g_2$. However, productions as introduced in MGG do relate p_C and p_N: they must fulfill $p_N = p_C^{-1}$. Also, in MGG, the certainty and nihil parts have to be disjoint. Hence, we will consider $p = (p_C, p_N) : \mathfrak{H} \to \mathfrak{H}$ for the rest of the paper unless otherwise stated.

We want p_N to be a production so we must split it into two parts: the one that acts on edges and the one that acts on vertices. Otherwise there would probably be dangling edges in the nihil part as soon as the production acts on nodes. The point is that the image of the nihil part with the operations specified by productions are not graphs in general, unless we restrict to edges and keep nodes apart. This behaviour is unimportant and should not be misleading.

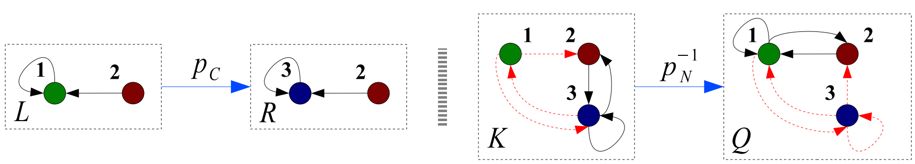


Figure 4: Potential Dangling Edges in the Nihilation Part

Example.□The left of Fig. 4 shows the certainty part of a production p that deletes node 1 (along with two incident edges) and adds node 3 (and two incident edges). Its nihil counterpart for edges is depicted to the right of the same figure. Notice that node 1 should not be included in K because it appears in L and we would be simultaneously demanding its presence and its absence. Therefore, edges $(1, 3)$, $(1, 2)$ and $(3, 1)$ – those with a red dotted line – would be dangling in K (red dotted edges do belong to the graphs they appear on). The same reasoning shows that something similar happens in Q but this time with edges $(1, 3)$, $(3, 1)$, $(3, 2)$ and $(3, 3)$ and node 3.

This is the reason to consider nodes and edges independently in the nihil parts of graphs and productions. In K, as nodes 1 and 3 belong to L, it should not make much sense to include them in K too, for if K dealt with nodes we would be demanding their presence and their abscense. In Q the production adds node 3 and something similar happens. ■

Now that nodes are considered, compatibility issues in the certainty part may show up. The determination of compatibility for a simple digraph is almost straightforward. Let $g =$

$(g_C^E \vee i g_N^E, g_C^V \vee i g_N^V) \in \mathfrak{H}$. Potential dangling edges are given by

$$\overline{D}_g = \overline{\overline{g_C^V} \otimes \overline{g_C^V}}, \tag{11}$$

so the graph g will be compatible if $g_C^E \, \overline{D}_g = 0$. As $g \in \mathfrak{H}$, there are no common elements between the certainty and nihil parts and $\overline{D}_g \prec g_N^E$.

A production $p(\mathcal{L}) = p(L \vee iK) = R \vee iQ = \mathcal{R}$ is compatible if it preserves compatibility, i.e. if it transforms a compatible digraph into a compatible digraph. This amounts to saying that $RQ = 0$.

Recall from Sec. 2 that grammar rule actions are specified through *erasing* and *addition* matrices, e and r respectively. Because e acts on elements that must be present and r on those that should not exist, it seems natural to encode a production as

$$p = e \vee ir. \tag{12}$$

Our next objective is to use the dot product – see Def. 3-2 – to represent the application of a production. This way, a unified approach would be obtained. To this end define the operator $P : \mathfrak{G} \to \mathfrak{G}$ by

$$p = e \vee ir \longmapsto P(p) = \overline{e}\,\overline{r} \vee i\,(e \vee r)\,. \tag{13}$$

Proposition 4-2 (Production). *Let $\mathcal{L}$ and $\mathcal{R}$ be the left and right hand sides, resp., as in Def. 4-1 and eq.* (10)*, and P as defined in eq.* (13)*. Then,*

$$\mathcal{R} = \langle \mathcal{L}, P(p) \rangle\,. \tag{14}$$

Proof

□The proof is a short exercise that makes use of some identities which are detailed below:

$$\begin{aligned}
\langle \mathcal{L}, P(p) \rangle &= \langle (L \vee iK), \overline{e}\,\overline{r} \vee i(e \vee r) \rangle = \\
&= [\overline{e}\,\overline{r} L \vee (e \vee r)K] \vee i\,[\overline{e}\,\overline{r} K \vee (e \vee r)L] = \\
&= (r \vee \overline{e} L)\, e \vee i\,(\overline{r} K) = p(L) \vee i p^{-1}(K) = \mathcal{R}.
\end{aligned} \tag{15}$$

In addition to $\overline{r} L = L$, we have used the following identities:

$$\begin{aligned}
(e \vee r)K &= eK \vee rK = rK = r(r \vee \overline{e}\overline{D}) = r. \\
\overline{e}\,\overline{r}\,K &= \overline{r}\left(\overline{e}\,r \vee \overline{e}\,\overline{e}\overline{D}\right) = \overline{r}\,K. \\
(e \vee r)L &= eL \vee rL = eL = e.
\end{aligned}$$

We have also used that $r\overline{e} = r$, $r\overline{D} = r$ due to compatibility and $rL = 0$ almost by definition. Besides, Prop. 7.4.5 in [12] has also been used, which proves that transformation of the nihil parts evolves according to the inverse of the production, i.e. $Q = p^{-1}\,(K)\,.$ ■

The production is defined through the operator P instead of directly as $p = \overline{e}\,\overline{r} \vee i(e \vee r)$ for several reasons. First, eq. (12) and its interpretation seem more natural. Second, $P(p)$ is self-adjoint, i.e. $P(p)^* = P(p)$, which in particular implies that $\|P(p)\| = \mathbf{1}$, $\forall p$ (see eq. (16) below). Therefore, $\|\cdot\|$ would not *measure* the size of productions (interpreted as graphs according to eq. (12) and as long as $\|\cdot\|$ measures sizes of Boolean complexes) and we would be forced to introduce a new norm. This is because

$$\begin{aligned} \|P(p)\| &= \langle P(p), P(p)\rangle = \\ &= (\overline{e}\,\overline{r} \vee i(e \vee r))\,(\overline{e}\,\overline{r} \vee i(e \vee r))^* = \overline{e}\,\overline{r} \vee e \vee r = \mathbf{1}. \end{aligned} \tag{16}$$

By way of contrast, $\|p\| = e \oplus r = e \vee r$. With operator P the size of a production is the number of changes it specifies, which is appropriate for MGG.[8]

The proposed encoding puts into a single expression the application of a grammar rule, both to L and to K. Also, it links the functional notation introduced in [12] and the dot product of Sec. 3.

Theorem 4-3 (Surjective Mapping). *There exists a surjective mapping from the set of MGG productions on to the set of self-adjoint graphs in $\mathfrak{H}$.*

Proof
□It is not difficult to check that z is self-adjoint if and only if $\|z\| = \mathbf{1}$: on the one hand, if $z = a \vee i\overline{a}$ then $\langle z, z\rangle = zz^* = (a \vee i\overline{a})(a \vee i\overline{a}) = a \vee \overline{a} = \mathbf{1}$. On the other hand, if we have $z = a \vee ib$ and $\|z\| = a \oplus b = \mathbf{1}$ then $a = \overline{b}$.

The surjective morphism is given by operator P. Clearly, P is well-defined for any production. To see that it is surjective, fix some graph $g = g_1 \vee ig_2$ such that $\|g\| = \mathbf{1}$. Then, $g = g_1 \vee i\overline{g}_1$. Any partition of $\overline{g}_1$ as **or** of two disjoint digraphs would do. Recall that productions (as graphs) have the property that their certainty and nihil parts must be disjoint. ■

The operator P is surjective but not necessarily injective. It defines an equivalence relation and the corresponding quotient space. In this way, we introduce the notion of *swap* which allows a more abstract view of the concept of production. Their importance stems from the fact that swaps summarize the dynamics of a production, independently of its left hand side. They allow us to study a set of actions, indepedently of the actual graph they are going to be applied to.

Definition 4-4 (Swap). The swap space is defined as $\mathcal{W} = \mathfrak{H}/P(\mathfrak{H})$. An equivalence class in the swap space will be called a *swap*. The swap w associated to a production $p : \mathfrak{H} \to \mathfrak{H}$ is $w = w_p = P(p)$, i.e. $p \in \mathfrak{H} \longmapsto w_p \in \mathcal{W}$.[9]

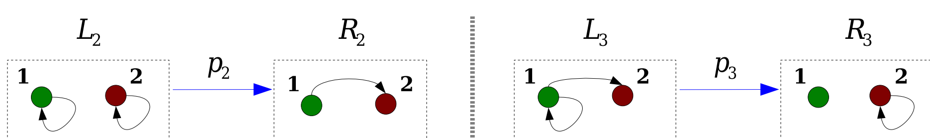


Figure 5: Example of Productions

Example □Let p_2 and p_3 be two productions as those depicted in Fig. 5. Their images in $\mathcal{W}$ are:

$$P(p_2) = P(p_3) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \vee i \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = w. \tag{17}$$

[8] Eventually, in complexity theory, one is interested in looking for an appropriate measure of the number of actions that transform one state (graph) into another.

[9] Acording to eq. (12) any element in $\mathfrak{H}$ can be interpreted as a production and viceversa.

They appear to be very different if we look at their defining matrices L_2, L_3 and R_2, R_3 or at their graph representation. Also, they seem to differ if we look at their erasing and addition matrices:

$$e_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad e_3 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad r_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad r_3 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

However, they are the same swap as eq. (17) shows, i.e. they belong to the same equivalence class. Notice that both productions act on edges $(1,1)$, $(2,2)$ and $(1,2)$ and none of them touches edge $(2,1)$. This is precisely what eq. (17) says as we will promptly see.

Swaps can be helpful in studying and classifying the productions of a grammar. For example, there are 16 different simple digraphs with 2 nodes. Hence, there are 256 different productions that can be defined. However, there are only 16 different swaps. From the point of view of the number of edges that can be modified, there is 1 swap that does not act on any element (which includes 16 productions), 4 swaps that act on 1 element, 6 swaps that act on 2 elements, 4 swaps that act on 3 elements and 1 swap that acts on all elements. ■

We can reinterpret actions specified by productions in Matrix Graph Grammars in terms of swaps: instead of adding and deleting elements, they interchange elements between the certainty and nihil parts, hence the name.

Notice that, because swaps are self-adjoint, it is enough to keep track of the certainty or nihil parts. So one production is fully specified by, for example, its left hand side and the nihil part of its associated swap.[10]

5 Coherence

So far we have extended MGG by defining the transformations (productions) in $\mathfrak{G}$ and $\mathfrak{H}$. The theory will be more interesting if we are able to develop the necessary concepts to deal with sequences of applications rather than productions alone. Among the two most basic notions are coherence and the initial digraph, which have been introduced in Sec. 2. We shall reformulate and extend them in this and the next sections.

Recall that the coherence of the sequence $s = p_n; \ldots; p_1$ guarantees that the actions of one production p_i do not prevent the actions of those sequentially behind it: $p_{i+1}, \ldots, p_n$. The first production to be applied in s is p_1 and the last one is p_n. The order is as in composition, from right to left.

Theorem 5-1 (Coherence). *The sequence of productions $s = p_n; \ldots; p_1$ is coherent if the Boolean complex $C \equiv C^+ \vee iC^- = 0$, where*

$$C^+ = \bigvee_{j=1}^{n} \left(R_j \bigtriangledown_{j+1}^{n} \left(\overline{e}_x r_y\right) \vee L_j \bigtriangleup_{1}^{j-1} \left(e_y \, \overline{r}_x\right)\right) \tag{18}$$

[10]Given a swap and a complex $\mathcal{L}$, it is not difficult to calculate the production having $\mathcal{L}$ as left hand side and whose actions agree with those of the swap.

and

$$C^- = \bigvee_{j=1}^{n} \left(Q_j \bigtriangledown_{j+1}^{n} \left(e_y \, \overline{r}_x \right) \vee K_j \bigtriangleup_1^{j-1} \left(r_y \, \overline{e}_x \right) \right). \tag{19}$$

with $\bigtriangleup$ and $\bigtriangledown$ as defined in eqs. (2) and (3), resp.

Proof

□The definition of equality of Boolean complexes in Def. 3-1 states that $C^+ \vee iC^- = 0$ if and only if $C^+ = C^- = 0$. The certainty part C^+ and the nihil part $C^- = 0$ can be proved similarly.[11] We shall start with the certainty part C^+.

Certainty part C$^+$

Consider $s_2 = p_2; p_1$ a sequence of two productions. In order to decide whether the application of p_1 does not exclude p_2, we impose three conditions on edges:

1. The first production – p_1 – does not delete (e_1) any element used (L_2) by the second production:
$$e_1 L_2 = 0. \tag{20}$$
2. p_2 does not add (r_2) any element preserved (used but not deleted, $\overline{e}_1 L_1$) by p_1:
$$r_2 L_1 \overline{e}_1 = 0. \tag{21}$$
3. No common elements are added by both productions:
$$r_1 r_2 = 0. \tag{22}$$

The first condition is needed because if p_1 deletes an edge used by p_2, then p_2 would not be applicable. Regarding edges, the last two conditions are mandatory in order to obtain a simple digraph (with at most one edge in each direction between two nodes).

Conditions (21) and (22) are equivalent to $r_2 R_1 = 0$ because, as both are equal to zero, we can do

$$0 = r_2 L_1 \overline{e}_1 \vee r_2 r_1 = r_2 \left(r_1 \vee \overline{e}_1 L_1 \right) = r_2 R_1,$$

which may be read "p_2 does not add any element that comes out from p_1's application". All conditions can be synthesized in the following identity:

$$r_2 R_1 \vee e_1 L_2 = 0. \tag{23}$$

To obtain a closed formula for the general case, we may use the fact that $r\overline{e} = r$ and $e\overline{r} = e$. Equation (23) can be transformed to obtain:

$$R_1 \overline{e}_2 r_2 \vee L_2 e_1 \overline{r}_1 = 0. \tag{24}$$

$D_2; D_1$	(20)	$D_2; P_1$	$\surd$	$D_2; A_1$	$\surd$
$P_2; D_1$	(20)	$P_2; P_1$	$\surd$	$P_2; A_1$	$\surd$
$A_2; D_1$	$\surd$	$A_2; P_1$	(21)	$A_2; A_1$	(22)

Table 1: Possible Actions for Two Productions

Now we chack that eq. (24) covers all possibilities. Call **D** the action of deleting an element, **A** its addition and **P** its preservation, i.e. the edge appears in both the LHS and the RHS. Table 1 comprises all nine possibilities for two productions.

A tick means that the action is allowed, while a number refers to the condition that prohibits the action. For example, $P_2; D_1$ means that first production p_1 deletes the element and second p_2 preserves it (in this order). If the table is looked up we find that this is forbidden by eq. (20).

Now we proceed with three productions. Consider the sequence $s_3 = p_3; p_2; p_1$. We must check that p_2 does not disturb p_3 and that p_1 does not prevent the application of p_2. Notice that both of them are covered in our previous explanation (in the two productions case). Thus, we just need to ensure that p_1 does not exclude p_3, taking into account that p_2 is applied in between.

1. p_1 does not delete (e_1) any element used (L_3) by p_3 and not added ($\overline{r}_2$) by p_2:
$$e_1 L_3 \overline{r}_2 = 0. \tag{25}$$

2. Production p_3 does not add (r_3) any edge stemming from p_1 (this is R_1) and not deleted (e_2) by p_2:
$$r_3 R_1 \overline{e}_2 = 0. \tag{26}$$

Again, regarding edges, the last condition is needed in order to obtain a simple digraph. Performing similar manipulations to those carried out for s_2 we get the full condition for s_3, given by the equation:

$$L_2 e_1 \vee L_3 \left(e_1 \overline{r}_2 \vee e_2\right) \vee R_1 \left(\overline{e}_2 r_3 \vee r_2\right) \vee R_2 r_3 = 0. \tag{27}$$

Proceeding as before, identity (27) is “extended” to represent the general case using operators $\triangle$ and $\bigtriangledown$:

$$L_2 e_1 \overline{r}_1 \vee L_3 \overline{r}_2 \left(e_1 \overline{r}_1 \vee e_2\right) \vee R_1 \overline{e}_2 \left(r_2 \vee \overline{e}_3 r_3\right) \vee R_2 \overline{e}_3 r_3 = 0. \tag{28}$$

This part of the proof can be finished by induction.

Nihil part $\mathbf{C}^-$

We proceed as for the certainty part. First, let’s consider a sequence of two productions $s_2 = p_2; p_1$. In order to decide whether the application of p_1 does not exclude p_2 (regarding elements that appear in the nihil parts) the following conditions must be demanded:

1. No common element is deleted by both productions:
$$e_1 e_2 = 0. \tag{29}$$

[11] The reader is invited to consult the proof of Th. 4.3.5 in [12] plus Lemma 4.3.3 and the explanations that follow Def. 4.3.2 in the same reference. Diagrams and examples therein included can be of some help.

2. Production p_2 does not delete any element that the production p_1 demands not to be present and that besides is not added by p_1:

$$e_2 K_1 \overline{r}_1 = 0. \tag{30}$$

3. The first production does not add any element that is demanded not to exist by the second production:

$$r_1 K_2 = 0. \tag{31}$$

Altogether we can write

$$e_1 e_2 \vee \overline{r}_1 e_2 K_1 \vee r_1 K_2 = e_2 (e_1 \vee \overline{r}_1 K_1) \vee r_1 K_2 = e_2 Q_1 \vee r_1 K_2 = 0, \tag{32}$$

which is equivalent to

$$e_2 \overline{r}_2 Q_1 \vee \overline{e}_1 r_1 K_2 = 0 \tag{33}$$

due to basic properties of MGG productions (see e.g. Prop. 4.1.4 in [12] for further details).

In the case of a sequence that consists of three productions, $s_3 = p_3; p_2; p_1$, the procedure is to apply the same reasoning to subsequences $p_2; p_1$ (restrictions on p_2 actions due to p_1) and $p_3; p_2$ (restrictions on p_3 actions due to p_1) and **or** them. Finally, we have to deduce which conditions have to be imposed on the actions of p_3 due to p_1, but this time taking into account that p_2 is applied in between. Again, we can put all conditions in a single expression:

$$Q_1 \left(e_2 \vee \overline{r}_2 e_3 \right) \vee Q_2 e_3 \vee K_2 r_1 \vee K_3 \left(r_1 \overline{e}_2 \vee r_2 \right) = 0. \tag{34}$$

$D_2; D_1$	(31)	$D_2; P_1$	$\surd$	$D_2; A_1$	$\surd$
$P_2; D_1$	(31)	$P_2; P_1$	$\surd$	$P_2; A_1$	$\surd$
$A_2; D_1$	$\surd$	$A_2; P_1$	(30)	$A_2; A_1$	(29)

Table 2: Possible Actions for Two Productions

We now check that eqs. (33) and (34) do imply coherence. To see that eq. (33) implies coherence we only need to enumerate all possible actions on the nihil parts. It might be easier if we think in terms of the negation of a potential host graph to which both productions would be applied $\left(\overline{G}\right)$ and check that any problematic situation is ruled out. See table 2 where **D** is deletion of one element from $\overline{G}$ (i.e., the element is added to G), **A** is addition to G and **P** is preservation (These definitions of **D**, **A** and **P** are opposite to those given for the certainty case above).[12] For example, action $A_2; A_1$ tells that in first place p_1 adds one element ε to $\overline{G}$. To do so this element has to be in e_1 (or incident to a node that is going to be deleted). After that, p_2 adds the same element, deriving a conflict between the rules. This proves $C^- = 0$ for the case $n = 2$.

When the sequence has three productions, $s = p_3; p_2; p_1$, there are 27 possible combinations of actions. However, some of them are considered in the subsequences $p_2; p_1$ and $p_3; p_2$. Table 3 summarizes them.

[12]Preservation means that the element is demanded to be in $\overline{G}$ because it is demanded not to exist by the production (it appears in K_1) and it remains as non-existent after the application of the production (it appears also in Q_1).

$D_3; D_2; D_1$	(31)	$D_3; D_2; P_1$	(31)	$D_3; D_2; A_1$	(31)
$P_3; D_2; D_1$	(31)	$P_3; D_2; P_1$	(31)	$P_3; D_2; A_1$	(31)
$A_3; D_2; D_1$	(31)	$A_3; D_2; P_1$	$\surd$	$A_3; D_2; A_1$	$\surd$
$D_3; P_2; D_1$	(31)	$D_3; P_2; P_1$	$\surd$	$D_3; P_2; A_1$	$\surd$
$P_3; P_2; D_1$	(31)	$P_3; P_2; P_1$	$\surd$	$P_3; P_2; A_1$	$\surd$
$A_3; P_2; D_1$	(31)/(30)	$A_3; P_2; P_1$	(30)	$A_3; P_2; A_1$	(30)
$D_3; A_2; D_1$	$\surd$	$D_3; A_2; P_1$	(30)	$D_3; A_2; A_1$	(29)
$P_3; A_2; D_1$	$\surd$	$P_3; A_2; P_1$	(30)	$P_3; A_2; A_1$	(29)
$A_3; A_2; D_1$	(29)	$A_3; A_2; P_1$	(29)	$A_3; A_2; A_1$	(29)

Table 3: Possible Actions for Three Productions

There are four forbidden actions:[13] $D_3; D_1$, $A_3; P_1$, $P_3; D_1$ and $A_3; A_1$. Let's consider the first one, which corresponds to $r_1 r_3$ (the first production adds the element – it is erased from $\overline{G}$ – and the same for p_3). In Table 3 we see that related conditions appear in positions $(1,1)$, $(4,1)$ and $(7,1)$. The first two are ruled out by conflicts detected in $p_2; p_1$ and $p_3; p_2$, respectively. We are left with the third case which is in fact allowed. The condition $r_3 r_1$ taking into account the presence of p_2 in the middle in eq. (34) is contained in $K_3 r_1 \overline{e}_2$, which includes $r_1 \overline{e}_2 r_3$. This must be zero, i.e. it is not possible for p_1 and p_3 to remove from $\overline{G}$ one element if it is not added to $\overline{G}$ by p_2. The other three forbidden actions can be checked similarly.

The proof can be finished by induction on the number of productions. The induction hypothesis leaves again four cases: $D_n; D_1$, $A_n; P_1$, $P_n; D_1$ and $A_n; A_1$. The corresponding table changes but it is not difficult to fill in the details. ■

There are some duplicated conditions, so it could be possible to "optimize" C. The form considered in Th. 5-1 is preferred because we may use $\triangle$ and $\triangledown$ to synthesize the expressions. Some comments on previous proof follow:

1. Notice that eq. (29) is already in C through eq. (18) which demands $e_1 L_2 = 0$ (as $e_2 \subset L_2$ we have that $e_1 L_2 = 0 \Rightarrow e_1 e_2 = 0$).

2. Condition (30) is $e_2 K_1 \overline{r}_1 = e_2 \overline{r}_1 r_1 \vee e_2 \overline{r}_1 \overline{e}_1 \overline{D}_1 = e_2 \overline{e}_1 \overline{D}_1$, where we have used that $K_1 = p\left(\overline{D}_1\right)$. Note that those $\overline{e}_1 \overline{D}_1 \neq 0$ are the dangling edges not deleted by p_1.

3. Equation (31) is $r_1 K_2 = r_1 p_2 \left(\overline{D}_2\right) = r_1 \left(r_2 \vee \overline{e}_2 \overline{D}_2\right) = r_1 r_2 \vee r_1 \overline{e}_2 \overline{D}_2$. The first term $(r_1 r_2)$ is already included in C and the second term is again related to dangling edges.

4. Potential dangling edges appear in coherence and this may seem to indicate a possible link between coherence and compatibility.[14]

An easy remark is that the complex $C^+ \vee iC^-$ in Th. 5-1 provides more information than just settling coherence as it measures non-coherence: *problematic* elements (i.e. those that prevent coherence) would appear as ones and the rest as zeros.

[13]Those actions appearing in table 1 updated for p_3.

[14]Compatibility for sequences is characterized in Sec. 7. Coherence takes into account dangling edges, but only those that appear in the "actions" of the productions (in matrices e and r).

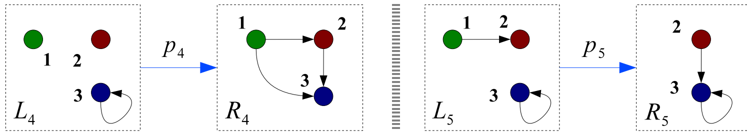


Figure 6: Example of Coherence

Example.□Let's consider the sequence $s = p_5; p_4$. Recall that the order of application is from right to left so p_4 is applied first and p_5 right afterwards. Let p_4 and p_5 be those productions depicted in Fig. 6. Once simplified, its coherence complex is

$$\begin{aligned} C(s) &= C^+(s) \vee iC^-(s) = (R_4 r_5 \vee L_5 e_4) \vee i\,(Q_4 e_5 \vee K_5 r_4) = \\ &= \left(\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \vee \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \vee \\ &\vee i \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \vee \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \right) = \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \vee i \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Coherence problems appear in this example for several reasons. Edge $(2, 3)$ is added twice while self-loop $(3, 3)$ is first deleted in p_4 and then used in p_5. Edge $(1, 3)$ becomes dangling because production p_5 deletes node 1. Edge $(2, 3)$ appears in $C^-(s)$ for the same reason that makes it appear in $C^+(s)$. ■

6 Initial Digraph

The minimal initial digraph $M(s)$ for a completed sequence $s = p_n; \ldots ; p_1$ was introduced in [12] as a simple digraph that permits all operations of s and that does not contain a proper subgraph with the same property. The negative initial digraph has a similar definition but for the nihil part (see Sec. 2 for both definitions).

In this section, in Th. 7-1 we encode, as a Boolean complex, the minimal and negative initial digraphs, renaming it to *initial digraph*. Also, a closed formula for its image under the action of a sequence of productions is provided.

Coherence and initial digraphs are closely related. The coherence of a sequence of productions depends on how nodes are identified across productions. This identification defines the minimum digraph needed to apply a sequence, which is the initial digraph.

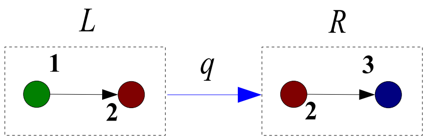
Now we are interested in what elements will be forbidden and which ones will be available once every production is applied.[15] Matrix $\overline{D} = \overline{\overline{e} \otimes \overline{e}^t}$ specifies what edges can not be present

[15]Whenever the tensor (Kronecker) product is used, we refer to the vector of nodes so the V superscript is

because at least one of their incident nodes have been deleted. Let's introduce the dual concept:

$$T = \left(\overline{\overline{r} \otimes \overline{r}^t}\right) \wedge \left(\overline{e} \otimes \overline{e}^t\right). \tag{35}$$

T are the newly available edges after the application of a production due to the addition of nodes.[16] The first term, $\overline{\overline{r} \otimes \overline{r}^t}$, has a one in all edges incident to a vertex that is added by the production. We have to remove those edges that are incident to some node deleted by the production, which is what $\overline{e} \otimes \overline{e}^t$ does.



$$\overline{e}\otimes\overline{e}^t = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \overline{\overline{r}\otimes\overline{r}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Figure 7: Available and Unavailable Edges After the Application of a Production

Example.□Figure 7 depicts to the left a production q that deletes node 1 and adds node 3. Its nihil term and its image are

$$K = q\left(\overline{D}\right) = r \vee \overline{e}\overline{D} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad Q = q^{-1}(K) = e \vee \overline{r}K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

To the right of Fig. 7, matrix T is included. It specifies those elements that are not forbidden once production q has been applied. ■

It is worth stressing that matrices $\overline{D}$ and T do not tell actions of the production to be performed in the complement of the host graph, $\overline{G}$. Actions of productions are specified exclusively by matrices e and r.

Theorem 6-1 (Initial Digraph). *The* initial digraph $M(s)$ *for the completed coherent sequence of productions* $s = p_n; \ldots; p_1$ *is given by*

$$M(s) = M_C(s) \vee i M_N(s) = \bigtriangledown_1^n \left(\overline{r}_x L_y \vee i\, \overline{e}_x \overline{T}_x K_y\right). \tag{36}$$

Proof

□We shall first prove the theorem for the certainty part. This will give us the main ideas to proceed with the nihil part. In both cases we shall use induction on the number of productions.

Certainty part $\mathbf{M_C = \bigtriangledown_1^n \left(\overline{r}_x L_y\right)}$

As the sequence is coherent and has been completed (i.e. nodes are related across productions) the graph $L = \bigvee_{j=1}^n L_i$ has enough elements to carry out all operations specified in

omitted. For example $R \otimes R' \equiv R^V \otimes \left(R^V\right)^t$. The ^{t} symbol stands for transposition.

[16]This is why T does not appear in the calculation of the coherence of a sequence: coherence takes care of real actions (e, r) and not of potential elements that may or may not be available $\left(\overline{D}, T\right)$.

the sequence.[17] Hence, in order to check if $M(s)$ has enough elements it suffices to see that $s(L) = s(M(s))$.

If we had a sequence consisting of only one production $s_1 = p_1$ then it should be clear that the minimal digraph needed to apply the sequence is L_1. This is almost by definition.

In the case of a sequence of two productions, say $s_2 = p_2; p_1$, what p_1 uses (L_1) is again needed. All edges that p_2 uses (L_2), except those added ($\overline{r}_1$) by the first production, are also mandatory. Note that the elements added (r_1) by p_1 are not considered in the initial digraph. If an element is preserved (used and not erased, $\overline{e}_1 L_1$) by p_1, then it should not be taken into account:

$$L_1 \vee L_2 \overline{r}_1 \, \overline{(\overline{e}_1 L_1)} = L_1 \vee L_2 \overline{r}_1 \left(e_1 \vee \overline{L}_1 \right) = L_1 \vee L_2 \overline{R}_1. \tag{37}$$

This formula can be paraphrased as "elements used by p_1 plus those needed by p_2's left hand side, except the ones resulting from p_1's application". Let's see that it provides enough elements to s_2:

$$\begin{aligned} p_2; p_1 \left(L_1 \vee L_2 \overline{R}_1 \right) &= r_2 \vee \overline{e}_2 \left(r_1 \vee \overline{e}_1 \left(L_1 \vee L_2 \overline{R}_1 \right) \right) = \\ &= r_2 \vee \overline{e}_2 \left(R_1 \vee r_1 \overline{R}_1 L_2 \vee \overline{e}_1 \overline{R}_1 L_2 \right) = \\ &= r_2 \vee \overline{e}_2 \left(R_1 \vee r_1 L_2 \vee \overline{e}_1 L_2 \right) = \\ &= r_2 \vee \overline{e}_2 \left(r_1 \vee \overline{e}_1 \left(L_1 \vee L_2 \right) \right) = p_2; p_1 \left(L_1 \vee L_2 \right). \end{aligned}$$

Let's move one step forward with the sequence of three productions $s_3 = p_3; p_2; p_1$. The minimal digraph needs what s_2 needed ($L_1 \vee L_2 \overline{R}_1$) but even more so. We have to add what the third production uses (L_3) except what comes out from p_1 and is not deleted by production p_2 (this is $R_1 \overline{e}_2$) to finally remove what comes out (R_2) from p_2:

$$M(s_3) = L_1 \vee L_2 \overline{R}_1 \vee L_3 \overline{(\overline{e}_2 \, R_1)} \, \overline{R}_2 = L_1 \vee L_2 \overline{R}_1 \vee L_3 \overline{R}_2 \left(e_2 \vee \overline{R}_1 \right). \tag{38}$$

Similarly to what has already been done for s_2, we check that the initial digraph has enough elements such that it is possible to apply p_1, p_2 and p_3:

$$\begin{aligned} p_3; p_2; p_1 \left(M(s_3) \right) &= r_3 \vee \overline{e}_3 \left(r_2 \vee \overline{e}_2 \left(r_1 \vee \overline{e}_1 \left(L_1 \vee L_2 \overline{R}_1 \vee L_3 \overline{R}_2 \left(e_2 \vee \overline{R}_1 \right) \right) \right) \right) = \\ &= r_3 \vee \overline{e}_3 \left(r_2 \vee \overline{e}_2 \left(\overline{e}_1 L_2 \vee \overline{e}_1 e_2 L_3 \overline{R}_2 \vee \underbrace{R_1 \vee L_3 \overline{e}_1 \overline{R}_1 R_2}_{= R_1 \vee L_3 \overline{e}_1 R_2} \right) \right) = \\ &= r_3 \vee \overline{e}_3 \left(\underbrace{\overline{e}_2 r_1 \vee \overline{e}_2 \overline{e}_1 L_1}_{= \overline{e}_2 R_1} \vee \overline{e}_2 \overline{e}_1 L_2 \vee \underbrace{r_2 \vee L_3 \overline{e}_1 \overline{e}_2 \overline{r}_2 \overline{L}_2}_{= r_2 \vee L_3 \overline{e}_1 \overline{e}_2 \overline{L}_2} \right) = \\ &= r_3 \vee \overline{e}_3 \left(r_2 \vee \overline{e}_2 \left(r_1 \vee \overline{e}_1 \left(L_1 \vee L_2 \vee L_3 \right) \right) \right) = \\ &= p_3; p_2; p_1 \left(L_1 \vee L_2 \vee L_3 \right). \end{aligned}$$

[17] It is also possible to interpret L as a non-completed graph, whose completion will avoid any coherence issue. If for example we had coherence issues with every single element in p_i then L would be the disjoint union of every L_i.

The same reasoning applied to the case of four productions gives the equation:

$$M_4 = L_1 \vee L_2\overline{R}_1 \vee L_3\overline{(\overline{e}_2\, R_1)}\,\overline{R}_2 \vee L_4\overline{(\overline{e}_3\,\overline{e}_2 R_1)}\ \overline{(\overline{e}_3 R_2)}\ \overline{R}_3. \tag{39}$$

Minimality is inferred by construction, because for each L_i all elements added by a previous production and not deleted by any production p_j, $j < i$, are removed. If any other element is erased from the initial digraph, then some production in s_n would miss some element.

Now we want to express previous formulas using operators $\triangle$ and $\bigtriangledown$. The expression

$$L_1 \vee \bigvee_{i=2}^{n} \left[L_i \triangle_1^{i-1} \left(\overline{R}_x\, e_y\right)\right] \tag{40}$$

is close but we would be adding terms that include $\overline{R}_1 e_1$, and clearly $\overline{R}_1 e_1 \neq \overline{R}_1$, which is what we have in the initial digraph.[18] Therefore, considering the fact that $\overline{a}b \vee \overline{a}\,\overline{b} = \overline{a}$ in propositional logics, we eliminate them by performing **or** operations:

$$\overline{e}_1 \bigtriangledown_1^{n-1} \left(\overline{R}_x L_{y+1}\right). \tag{41}$$

Thus we have a formula for the initial digraph which is slightly different from that in the theorem:

$$M(s) = L_1 \vee \overline{e}_1 \bigtriangledown_1^{n-1} \left(\overline{R}_x L_{y+1}\right) \vee \bigvee_{i=2}^{n} \left[L_i \triangle_1^{i-1} \left(\overline{R}_x e_y\right)\right]. \tag{42}$$

Our next step is to show that previous identity is equivalent to

$$M(s) = L_1 \vee \overline{e}_1 \bigtriangledown_1^{n-1} \left(\overline{r}_x L_{y+1}\right) \vee \bigvee_{i=2}^{n} \left[L_i \triangle_1^{i-1} \left(\overline{r}_x e_y\right)\right] \tag{43}$$

by illustrating the way to proceed for $n = 3$. To this end, the identity $\overline{r}L = L$ is used as well as the fact that $a \vee \overline{a}b = a \vee b$ in propositional logics:

$$\begin{aligned} M_3 &= L_1 \vee L_2\overline{R}_1 \vee L_3\overline{R}_2 \left(e_2 \vee \overline{R}_1\right) = \\ &= L_1 \vee L_2\overline{r}_1 \left(e_1 \vee \overline{L}_1\right) \vee \left(L_3\overline{r}_2 e_2 \vee L_3\overline{r}_2\overline{L}_2\right) \left(e_2 \vee \overline{r}_1 e_1 \overline{r}_1 \overline{L}_1\right) = \\ &= L_1 \vee L_2\overline{r}_1\overline{L}_1 \vee L_2 e_1 \vee L_3 e_2 \vee \underbrace{L_3 e_2 e_1 \vee L_3 e_2 \overline{r}_1 \overline{L}_1 \vee L_3 e_2 \overline{L}_2}_{\text{disappears due to } L_3 e_2} \vee \\ &\qquad \vee L_3\overline{r}_2\overline{L}_2\overline{r}_1\overline{L}_1 \vee L_3\overline{r}_2\overline{L}_2 e_1 = \\ &= L_1 \vee L_2\left(\overline{r}_1 \vee e_1\right) \vee L_3\overline{L}_2\overline{r}_2\overline{r}_1 \vee L_3 e_2 \vee L_3\overline{L}_2\overline{r}_2 e_1 = \\ &= L_1 \vee L_2\overline{r}_1 \vee L_3\overline{r}_2\left(e_2 \vee \overline{r}_1\right). \end{aligned}$$

But (43) is what we have in the theorem, because as the sequence is coherent, the third term in (43) is zero:

$$\bigvee_{i=2}^{n} \left[L_i \triangle_1^{i-1} \left(\overline{r}_x\, e_y\right)\right] = 0. \tag{44}$$

[18]Not in formula (36) but in expressions derived up to now for minimal initial digraph: formulas (37) and (38).

Finally, as $L_1 = L_1 \vee e_1$, it is possible to omit $\overline{e}_1$ and obtain (36), recalling again that $\overline{r}L = L$.

Nihil part $\mathbf{M_N(s)} = \bigtriangledown_{\mathbf{1}}^{\mathbf{n}} \left(\overline{\mathbf{e}}_{\mathbf{x}} \overline{\mathbf{T}}_{\mathbf{x}} \mathbf{K}_{\mathbf{y}}\right)$

We shall go a little bit faster as the proof proceeds along the lines of that for the certainty part, which in essence started with a big enough graph and removed as many elements as possible. However, for edges in the nihil part, besides the actions of the productions on edges, we need to keep track of the actions of the productions on nodes because some potential dangling edges may become available (if their incident nodes are added by some grammar rule then they stop being potential dangling edges).

Think of G as an "ambient graph" in which the operations are taking place. For the nihil term $\bigtriangledown_1^n \overline{e}_x \overline{T}_x K_y$ it is easier to think in what must be or must not be found in $\overline{G}$, rather than in G.

We once more proceed by induction on the number of productions. For the time being, for simplicity, we omit the effect of adding nodes which may turn potential dangling edges into available ones, i.e. we ignore $\overline{T}_x$. In a sequence with a single production it should be obvious that K_1 (and only K_1) needs to be demanded.

For a sequence of two productions $s_2 = p_2; p_1$, K_1 is again necessary. It is clear that $K_1 \vee K_2$ with $K_1^V K_2^V = 0$ – i.e. all nodes and hence edges unrelated – would be enough, but it may include more elements than strictly needed. Among them, those already deleted by p_1 and those that already appear in K_1 and that are not added by p_1 – $\overline{r}_1 K_1$ –. If these elements of K_2 are not going to be considered, we need to **and** their negation: $\overline{e}_1 \overline{(\overline{r}_1 K_1)} K_2$. Altogether, we get $K_1 \vee \overline{e}_1 \overline{(\overline{r}_1 K_1)} K_2$. Some simple manipulations prove that:

$$\begin{aligned} K_1 \vee K_2 \overline{e}_1 \overline{(\overline{r}_1 K_1)} &= K_1 \vee K_2 \overline{e}_1 \left(r_1 \vee \overline{K}_1\right) = \\ &= K_1 \vee K_2 \overline{(e_1 \vee \overline{r}_1 K_1)} = K_2 \overline{Q}_1. \end{aligned} \tag{45}$$

Minimality is inferred by construction. If any other element was removed then either p_1^{-1} or p_2^{-1} could not be applied (and still consider dangling edges). It is not difficult to check that the sequence $p_2^{-1}; p_1^{-1}$ can be applied to $K_1 \vee K_2 \overline{Q}_1$. The expressions for sequences of three, four, . . ., n poductions are:

$$N_3 = N_2 \vee K_3 \overline{\overline{r}_2 Q_1}\, \overline{Q}_2 \tag{46}$$

$$N_4 = N_3 \vee K_4 \overline{\overline{r}_3 \overline{r}_2 Q_1}\, \overline{\overline{r}_2 Q_2}\, \overline{Q}_3 \tag{47}$$

$$\ldots$$

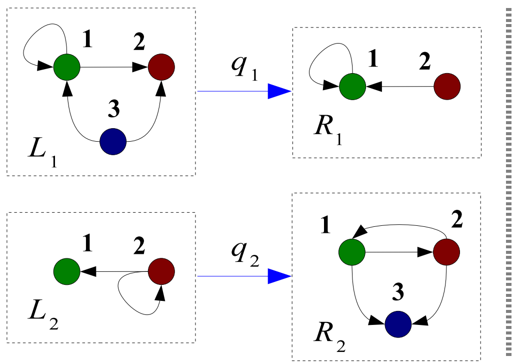
$$N_n = K_1 \vee \overline{r}_1 \bigtriangledown_1^{n-1} \left(\overline{Q}_x K_{y+1}\right) \vee \bigvee_{j=2}^{n} \left[K_j \bigtriangleup_1^{j-1} \left(\overline{Q}_x r_y\right)\right] \tag{48}$$

$$N_n = K_1 \vee \overline{r}_1 \bigtriangledown_1^{n-1} \left(\overline{e}_x K_{y+1}\right) \vee \bigvee_{j=2}^{n} \left[K_j \bigtriangleup_1^{j-1} \left(\overline{e}_x r_y\right)\right]. \tag{49}$$

There are two tricky steps. The first one is how to derive N_n in eq. (48) and the second is how to obtain its equivalent expression (49). The reader is referred to the proof for the certainty part above, where detailed explanations have been provided.

Once we get here it is easy to obtain $\bigtriangledown_1^n \left(\overline{e}_x K_y\right)$. First, note that the sequence is coherent so the third term in eq. (49) is zero. Second, as $K_1 = K_1 \vee r_1$, the $\overline{r}_1$ can be simplified because $a \vee \overline{a}b = a \vee b$ in propositional logics.

Finally, the same reasoning applies for those nodes that are added. So we do not only need to remove elements erased by previous productions but also edges that are not incident to any non-existent edge, $\bigtriangledown_1^n \left(\overline{e}_x K_y\right) \longmapsto \bigtriangledown_1^n \left(\overline{e}_x \overline{T}_x K_y\right)$ ■



$$K_1 = r_1 \vee \overline{e}_1 \overline{D}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \vee \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_2 = r_2 \vee \overline{e}_2 \overline{D}_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\overline{T}_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \overline{T}_2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 8: Sequence of Two Productions

Example.□Figure 8 includes two productions with their nihilation matrices K_1 and K_2. The initial digraph of the sequence $s = q_2; q_1$ is

$$\begin{aligned} M(s) &= \bigtriangledown_1^2 \left(\overline{r}_x L_y \vee i \overline{e}_x \overline{T}_x K_y\right) = (\overline{r}_1 L_1 \vee \overline{r}_1 \overline{r}_2 L_2) \vee i \left(\overline{e}_1 \overline{T}_1 K_1 \vee \overline{e}_1 \overline{e}_2 \overline{T}_1 \overline{T}_2 K_2\right) = \\ &= (L_1 \vee \overline{r}_1 L_2) \vee i \left(\overline{T}_1 K_1 \vee \overline{e}_1 \overline{T}_1 \overline{T}_2 K_2\right) \\ &= \left(\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \vee i \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \vee \right. \\ &\quad \left. \vee \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \right) = \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \vee i \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \equiv M_C(s_2) \vee i M_N(s_2). \end{aligned}$$

$M_C(s_2)$ has the minimal set of edges and nodes necessary to apply productions p_1 and p_2, in this precise order. $M_N(s_2)$ has the minimal amount of edges that must be missing.

We have represented $M_C(s_2) \vee i M_N(s_2)$ to the left of Fig. 9 together with its evolution as well as the final state, $s_2\left(M(s_2)\right)$. To the right of the same figure there is the same evolution but limited to edges and from the point of view of swaps. With black solid lines we have represented the edges that are present and with red dotted lines those that are absent. Recall that swaps interchange them. ■

As above, think of G as an "ambient graph" in which operations take place. A final remark is that $\overline{T}$ makes the number of edges in $\overline{G}$ as small as possible. For example, in $\overline{e}_1 \overline{e}_2 \overline{T}_1 \overline{T}_2 K_2$ we

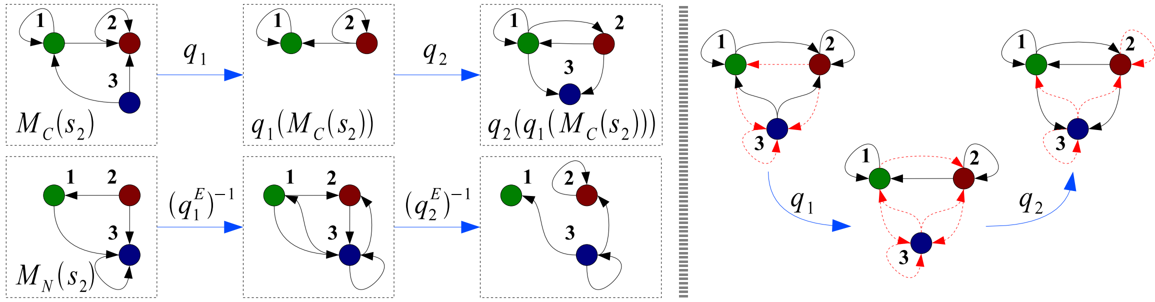


Figure 9: Initial Digraph of a Sequence of Two Productions Together with its Evolution

are in particular demanding $\overline{e}_1\overline{T}_1\overline{T}_2r_2$ (because $K_2 = r_2 \vee \overline{e}_2\overline{D}_2$). If we start with a compatible host graph, it is not necessary to ask for the absence of edges incident to nodes that are added by a production (we called them *potentially available* above). Notice that these edges could not be in the host graph as they would be dangling edges or we would be adding an already existent node. Summarizing, if compatibility is assumed or demanded by hypothesis, we may safely ignore $\overline{T}_x$ in the formula for the initial digraph. This remark will be used in the proof of the G-congruence characterization theorem in Sec. 7.

We end this section with a closed formula for the effect of the application of a coherent concatenation to an initial digraph. It can be useful if we want to operate in the general case.

Corollary 6-2. *Let $s = p_n; \ldots ; p_1$ be a coherent sequence of productions, and $M(s)$ its initial digraph. Considering only the certainty parts, then*

$$s(M(s)) = \bigwedge_{i=1}^{n} \left(\overline{e}_i M(s)\right) \vee \triangle_1^n \left(\overline{e}_x r_y\right) \tag{50}$$

$$\overline{s(M(s))} = \bigwedge_{i=1}^{n} \left(\overline{r}_i \overline{M(s)}\right) \vee \triangle_1^n \left(\overline{r}_x e_y\right) \tag{51}$$

Proof
□Theorem 6-1 proves that $s\left(M(s)\right) = s\left(\bigvee_{i=1}^{n} L_i\right)$. To derive the formulas apply induction on the number of productions and $\overline{e}r = r$. ■

Notice that eqs. (50) and (51) have the same shape as a single production $p = r \vee \overline{e}L$, where for eq. (50)

$$e = \bigvee_{i=1}^{n} e_i \qquad r = \triangle_1^n \left(\overline{e}_x\, r_y\right) \tag{52}$$

and for eq. (51)

$$e = \bigvee_{i=1}^{n} r_i \qquad r = \triangle_1^n \left(\overline{r}_x\, e_y\right). \tag{53}$$

The e's in eqs. (52) and (53) are those elements not deleted by any production and the r's are what a grammar rule adds and no previous production deletes (*previous* with respect to the order of application).

For the application of a single production p, the order of deletion and addition is unimportant: $p(L) = r \vee \overline{e}L = \overline{e}(r \vee L)$. This is because $\overline{e}r = r$. However, the order of application does matter in the case of a sequence s if we write it using eqs. (52) and (53), $s(G) = r \vee \overline{e}G$. Here it is necessary to carry out deletion first and addition afterwards.

Equation (50) is closely related to composition of a sequence of productions as defined in Sec. 4.5 in [12]. This explains why it is possible to interpret a coherent sequence of productions as a single production. Recall that any sequence is coherent if the appropriate identifications on nodes are performed.

The negation of the minimal initial digraph which appears in identity (51) can be explicitly calculated in terms of the operator nabla:

$$\overline{M(s)} = \nabla_1^{n-1}\left(\overline{L_x}\, r_y\right) \vee \bigwedge_{i=1}^{n} \overline{L_i}. \tag{54}$$

Corollary 6-3. *Let $s = p_n; \ldots; p_1$ be a coherent sequence of productions, and $M(s)$ its initial digraph. Then*

$$s(M(s)) = s\left(M_C(s)\right) \vee i\ \overline{s\left(\overline{M_N(s)}\right)}. \tag{55}$$

7 Compatibility and Congruence

This section reviews some more sequential results, adapting and extending them. The notions we cope with are compatibility[19] and G-congruence. By the end of the section we will very briefly touch on sequential independence, application conditions and graph constraints.

Compatibility asks for "closedness" of the space (graphs) with respect to the specified operations. In essence, it demands the lack of dangling edges. Definitions of compatibility for increasingly general concepts can be found in Sec. 2: single simple digraph, production and sequence. According to Prop. 4-2 productions act on edges and on vertices. They are obviously related but this relation has not been demonstrated. It is of importance in order to study the evolution of the nihil part of (strict) Boolean complexes. What one production forbids, another production may need or even can make accessible again.

Proposition 7-1 (Compatibility). *Let $s = p_n; \ldots; p_1$ be a sequence consisting of compatible productions. If*

$$\nabla_1^n\left(\overline{e}_x \overline{r}_x M_C(s_x) M_N(s_x)\right) = 0 \tag{56}$$

then s is compatible, where $M_C(s_m)$ and $M_N(s_m)$ are the certainty and nihil parts of the initial digraphs of $s_m = p_m; \ldots; p_1$, $m \in \{1, \ldots, n\}$.

Proof (Sketch)
□Equation (56) is a restatement of the definition of compatibility for a sequence of productions. The condition appears when the certainty and nihil parts are demanded to have no common elements. Compatibility of each production is used to simplify terms of the form $L_i K_i$ ■

[19]Compatibility has been defined in Sec. 2 and used for a single production for example in the proof of Prop. 4-2, Sec. 4.

As happened with coherence and the complex $C^+ \vee iC^-$ in Th 5-1, eq. (56) for compatibility provides information on which elements may prevent it.

Compatibility and coherence are related notions but only to some extent. Coherence deals with actions of productions, while compatibility with potential presence or abscense of elements. This might be better understood if we think in terms of sequences: when the left hand side $L \vee iK$ of a grammar rule p is matched in a host graph $G \vee i\overline{G}$, all elements of L must be found in G and all edges of K must be found in $\overline{G}$. When p is applied, a new graph $H \vee i\overline{H}$ is derived. Again, all elements of R have to be found in H and all edges in Q will be in $\overline{H}$, no matter if some of them are now potentially usable (say p adds some nodes and some potentially dangling edges are not dangling edges anymore).

Now we turn to G-congruence, which studies equality of initial digraphs for a sequence $s = p_n; \ldots; p_1$ and a permutation of it, $s' = \sigma(s) = p_{\sigma(n)}; \ldots; p_{\sigma(1)}$. As previously commented, this is closely related to sequential indepence, which is a fundamental concept in graph rewriting.

We limit ourselves to the advancement and delaying of a single production: permutations ϕ and δ. Advancement[20] is $\phi = (1\ 2\ \ldots\ n{-}1\ n)$ and delaying is $\delta = (n\ n{-}1\ \ldots\ 2\ 1)$, i.e $\phi(s) = p_{n-1}; p_{n-2} \ldots; p_1; p_n$ and $\delta(s) = p_1; p_n; \ldots p_2$.

We first calculate what we call congruence conditions. Then, some technical lemmas are proved and, finally, the section ends stating and proving the main result regarding sameness of initial digraphs. As commented right after the example on p. 25, assuming compatibility allows us to safely ignore $\overline{T}$. We shall do so for the rest of the section.

Congruence conditions (abbreviated as **CC**, *positive CC* if they refer to the certainty part of the initial digraph and *negative CC* for the nihil part of the initial digraph) are the formulas which should check the differences between the initial digraphs of two sequences, one being a permutation of the other. For its calculation we proceed by induction on the number of productions, starting with $n = 2$.

Let's start by just considering the certainty part of the initial digraph. Suppose we have a coherent sequence made up of two productions $s_2 = p_2; p_1$ with initial digraph $M_C(s_2)$ and, applying the (only possible) permutation $\sigma = (1\ 2)$, get another coherent concatenation $s'_2 = p_1; p_2$ with initial digraph $M_C(s'_2)$. Production p_1 does not delete any element added by p_2 because, otherwise, if p_1 in s_2 deleted something, it would mean that it already existed (as p_1 is applied first in s_2) while p_2 adding that same element in s'_2 would mean that this element was not present (because p_2 is applied first in s'_2). This condition can be written:

$$e_1 r_2 = 0. \tag{57}$$

A similar reasoning states that p_1 can not add any element that p_2 is going to use:

$$r_1 L_2 = 0. \tag{58}$$

Analogously for p_2 against p_1, i.e. for $s'_2 = p_1; p_2$, we have:

$$e_2 r_1 = 0 \tag{59}$$

$$r_2 L_1 = 0. \tag{60}$$

[20]Numbers in the permutation refers to the position that the production occupies inside the sequence, not to its subindex.

As a matter of fact two equations are redundant – (57) and (59) – because they are already contained in the other two. Note that $e_i L_i = e_i$, i.e. in some sense $e_i \subset L_i$, so it is enough to ask for:

$$r_1 L_2 \vee r_2 L_1 = 0. \tag{61}$$

It is easy to check that these conditions make $M_C(s_2) = M_C(s_2')$. In detail:

$$\begin{aligned} M_C(s_2) &= M_C(s_2) \vee r_1 L_2 = L_1 \vee \overline{r}_1 L_2 \vee r_1 L_2 = L_1 \vee L_2 \\ M_C(s_2') &= M_C(s_2') \vee r_2 L_1 = L_2 \vee \overline{r}_2 L_1 \vee r_2 L_1 = L_2 \vee L_1. \end{aligned}$$

Let's now turn to the nihil part of the initial digraph, for which the first production should not delete any element forbidden for p_2 (in such a case these elements would be in $\overline{G}$ for $p_1; p_2$ and in G for $p_2; p_1$):

$$0 = e_1 K_2 = e_1 r_2 \vee e_1 \overline{e}_2 \overline{D}_2. \tag{62}$$

Note that we already had $e_1 r_2 = 0$ in eq. (57). A symmetrical reasoning yields $e_2 \overline{e}_1 \overline{D}_1 = 0$, and altogether:

$$e_1 \overline{e}_2 \overline{D}_2 \vee e_2 \overline{e}_1 \overline{D}_1 = 0. \tag{63}$$

First monomial in (63) simply states that no potential dangling edge for p_2 (not deleted by p_2) can be deleted by p_1.

It is not difficult to show that eq. (63) guarantees the same nihil part of the initial digraph. In $p_2; p_1$ the nihil part of the initial digraph is given by $K_1 \vee \overline{e}_1 K_2$. Condition (62) demands $e_1 K_2 = 0$ so we can **or** them to get:

$$K_1 \vee \overline{e}_1 K_2 \vee e_1 K_2 = K_1 \vee K_2. \tag{64}$$

A similar reasoning applies to $p_1; p_2$, obtaining the same result.

We will proceed with three productions so, following a consistent notation, we set $s_3 = p_3; p_2; p_1$, $s_3' = p_2; p_1; p_3$ with permutation $\sigma_3 = (\,1\ 3\ 2\,)$ and their corresponding certainty part of the initial digraphs $M_C(s_3) = L_1 \vee \overline{r}_1\, L_2 \vee \overline{r}_1\, \overline{r}_2\, L_3$ and $M_C(s_3') = \overline{r}_3\, L_1 \vee \overline{r}_3\, \overline{r}_2\, L_2 \vee L_3$. Conditions are deduced similarly to the two productions case:[21]

$$r_3 L_1 = 0 \qquad r_3 L_2 \overline{r}_1 = 0 \qquad r_1 L_3 = 0 \qquad r_2 L_3 \overline{e}_1 = 0. \tag{65}$$

Let's interpret them all. $r_3 L_1 = 0$ says that p_3 cannot add an edge that p_1 uses. This is because this would mean (by s_3) that the edge is in the host graph (it is used by p_1) but s_3' says that it is not there (it is going to be added by p_3). The second condition is almost equal but with p_2 playing the role of p_1, which is why we demand p_1 not to add the element ($\overline{r}_1$). Third equation is symmetrical with respect to the first. The fourth states that we would derive a contradiction if the second production adds something (r_2) that production p_3 uses (L_3) and p_1 does not delete $(\overline{e}_1)$. This is because by s_3 the element was not in the host graph. Note that s_3' says the opposite, as p_3 (to be applied first) uses it. All can be put together in a single expression:

$$L_3\,(r_1 \vee \overline{e}_1 r_2) \vee r_3\,(L_1 \vee \overline{r}_1 L_2) = 0. \tag{66}$$

[21] As far as we know, there is no rule of thumb to deduce the conditions for G-congruence. They depend on the operations that productions define and their relative order.

For the sake of completeness let's point out that there are other four conditions but they are already considered in eq. (66):

$$e_1 r_3 = 0 \qquad r_3 e_2 \overline{r}_1 = 0 \qquad e_3 r_1 = 0 \qquad r_2 e_3 \overline{e}_1 = 0. \tag{67}$$

Now we deal with those elements that must not be present. Four conditions similar to those for two productions – compare with eq. (62) – are needed:

$$\begin{aligned} e_1 K_3 &= e_1 r_3 \vee e_1 \overline{e}_3 \overline{D}_3 = 0 \\ e_3 K_1 &= e_3 r_1 \vee e_3 \overline{e}_1 \overline{D}_1 = 0 \\ e_3 K_2 \overline{e}_1 &= e_3 r_2 \overline{e}_1 \vee e_3 \overline{e}_1 \overline{e}_2 \overline{D}_2 = 0 \\ e_2 K_3 \overline{r}_1 &= e_2 r_3 \overline{r}_1 \vee e_2 \overline{r}_1 \overline{e}_3 \overline{D}_3 = 0. \end{aligned} \tag{68}$$

Note that the first monomial in every equation can be discarded as they are already considered in identity (66). We put them altogether to get:

$$\begin{aligned} & e_1 \overline{e}_3 \overline{D}_3 \vee e_3 \overline{e}_2 \overline{e}_1 \overline{D}_2 \vee e_3 \overline{e}_1 \overline{D}_1 \vee e_2 \overline{e}_3 \overline{r}_1 \overline{D}_3 = \\ & \quad = e_3 \left(\overline{e}_1 \overline{D}_1 \vee \overline{e}_1 \overline{e}_2 \overline{D}_2 \right) \vee \overline{e}_3 \overline{D}_3 \left(e_1 \vee \overline{r}_1 e_2 \right). \end{aligned} \tag{69}$$

Moving one production three positions forward in a sequence of four productions, i.e. $p_4; p_3; p_2; p_1 \mapsto p_3; p_2; p_1; p_4$, while maintaining the certainty part of the initial digraph has as associated conditions those given by the equation:

$$L_4 \left(r_1 \vee \overline{e}_1\, r_2 \vee \overline{e}_1\, \overline{e}_2\, r_3 \right) \vee r_4 \left(L_1 \vee \overline{r}_1\, L_2 \vee \overline{r}_1\, \overline{r}_2\, L_3 \right) = 0, \tag{70}$$

and the nihil part of the initial digraph by:

$$e_4 \left(\overline{e}_1\, \overline{D}_1 \vee \overline{e}_1\, \overline{e}_2 \overline{D}_2 \vee \overline{e}_1\, \overline{e}_2\, \overline{e}_3 \overline{D}_3 \right) \vee \overline{e}_4\, \overline{D}_4 \left(e_1 \vee \overline{r}_1\, e_2 \vee \overline{r}_1\, \overline{r}_2\, e_3 \right) = 0. \tag{71}$$

By induction it can be proved that for advancement of one production $n-1$ positions inside the sequence of n productions $s_n = p_n; \ldots; p_1$, the equation which contains all *positive CC* can be expressed in terms of operator ∇ and has the form:

$$CC^{+} \left(\phi_n, s_n \right) = L_n \nabla_1^{n-1} \left(\overline{e}_x r_y \right) \vee r_n \nabla_1^{n-1} \left(\overline{r}_x L_y \right) = 0. \tag{72}$$

and for the *negative CC*:

$$CC^{-} \left(\phi_n, s_n \right) = \overline{D}_n \overline{e}_n \nabla_1^{n-1} \left(\overline{r}_x e_y \right) \vee e_n \nabla_1^{n-1} \left(\overline{e}_x \overline{D}_y \right) = 0. \tag{73}$$

Some monomials were discarded in eq. (68) because they were already considered in eq. (66). If (73) is not used in conjunction with (72), then the more complete form

$$CC^{-} \left(\phi_n, s_n \right) = K_n \nabla_1^{n-1} \left(\overline{r}_x e_y \right) \vee e_n \nabla_1^{n-1} \left(\overline{e}_x K_y \right) \tag{74}$$

should be preferred. Recall that $K_j = r_j \vee \overline{e}_j \overline{D}_j$. The point is that $\overline{e}_j \overline{D}_j$ considers potential dangling edges while K_j also includes those to be added.

It is possible to write eqs. (72) and (73) in terms of L_i and K_i. We will do it for sequences s_3 and s'_3. One illustrating example should suffice:

$$\begin{aligned} r_3\overline{r}_1L_1 \vee \overline{D}_3\overline{e}_3\overline{r}_1e_1 &= \overline{r}_1L_1\left(r_3 \vee e_1\overline{e}_3\overline{D}_3\right) = \\ &= \overline{r}_1L_1\left(r_3e_1 \vee r_3\overline{e}_1 \vee e_1\overline{e}_3\overline{D}_3\right) = \\ &= \overline{r}_1L_1\left(e_1K_3 \vee r_3\overline{e}_1\right) = \overline{r}_1L_1K_3\left(e_1 \vee r_3\right). \end{aligned} \tag{75}$$

Last equality holds because $K_ir_i = r_i \vee r_i\overline{D}_i = r_i$ and $a \vee \overline{a}b = a \vee b$ in propositional logics. We have also used that $K_i\overline{e}_i = \overline{e}_i\left(r_i \vee \overline{e}_i\overline{D}_i\right) = K_i$.

A formula considering the positive (72) and the negative (73) parts can be derived by induction. It is presented as a proposition:

Proposition 7-2. *Positive and negative congruence conditions for sequences s_n and $s'_n = \phi_n(s_n)$ are given by:*

$$CC\left(\phi_n, s_n\right) = L_n\nabla_1^{n-1}\overline{e}_xK_y\left(r_y \vee e_n\right) \vee K_n\nabla_1^{n-1}\overline{r}_xL_y\left(e_y \vee r_n\right). \tag{76}$$

Proof

□■

G-congruence is obtained when $CC\left(\phi_n, s_n\right) = 0$. An equivalent reasoning does it for production delaying $n-1$ positions, giving very similar formulas. Suppose that production p_1 is moved backwards in concatenation s_n to get $s''_n = p_1; p_n; \ldots; p_2$, i.e. δ_n is applied. The positive part of the condition is:

$$CC^+\left(\delta_n, s_n\right) = L_1\nabla_2^n\left(\overline{e}_xr_y\right) \vee r_1\nabla_2^n\left(\overline{r}_xL_y\right) = 0 \tag{77}$$

and the negative part:

$$CC^-\left(\delta_n, s_n\right) = \overline{D}_1\overline{e}_1\nabla_2^n\left(\overline{r}_xe_y\right) \vee e_1\nabla_2^n\left(\overline{e}_x\overline{D}_y\right) = 0. \tag{78}$$

As in the positive case it is possible to merge equations (77) and (78) to get a single expression:

Proposition 7-3. *Positive and negative congruence conditions for sequences s_n and $s''_n = \delta_n(s_n)$ are given by:*

$$CC\left(\delta_n, s_n\right) = L_1\nabla_2^n\overline{e}_xK_y\left(r_y \vee e_1\right) \vee K_1\nabla_2^n\overline{r}_xL_y\left(e_y \vee r_1\right). \tag{79}$$

Proof

□■

It is necessary to show that these conditions guarantee sameness of initial digraphs, but first we need two technical lemmas that provide us with some identities used to transform the initial digraphs. Advancement and delaying are very similar so only advancement is considered for the rest of the section.

Lemma 7-4. *Let $s_n = p_n; \ldots; p_1$ be a sequence and $s'_n = \sigma\left(s_n\right) = p_{n-1}; \ldots; p_1; p_n$ and that $CC^+\left(\phi_n, s_n\right)$ is satisfied. Then the following identity may be* ***or****ed to $M_C(s_n)$ without changing it:*

$$DC^+(\phi_n, s_n) = L_n\nabla_1^{n-2}\left(\overline{r}_xe_y\right). \tag{80}$$

Proof

□Let's start with three productions. Recall that $M_C(s_3) = L_1 \vee$ *other_terms* and that $L_1 = L_1 \vee e_1 = L_1 \vee e_1 \vee e_1 L_3$ (last equality holds for any formula $a \vee ab = a$ in propositional logics). Note that $e_1 L_3$ is eq. (80) for $n = 3$.

For $n = 4$, apart from $e_1 L_4$, we need to get $e_2 \overline{r}_1 L_4$ (as the full condition is $DC^+(\phi_4, s_4) = L_4 \left(e_1 \vee \overline{r}_1 e_2\right)$). Recall again the minimal initial digraph for four productions whose first two terms are $M_C(s_4) = L_1 \vee \overline{r}_1 L_2$. It is not necessary to consider all terms in $M_C(s_4)$ to get $DC^+(\phi_4, s_4)$:

$$\begin{aligned} M_C(s_4) &= (L_1 \vee e_1) \vee (\overline{r}_1 L_2 \vee \overline{r}_1 e_2) \vee \ldots = \\ &= (L_1 \vee e_1 \vee e_1 L_4) \vee (\overline{r}_1 L_2 \vee \overline{r}_1 e_2 \vee \overline{r}_1 e_2 L_4) \vee \ldots = \\ &= (L_1 \vee e_1 L_4) \vee (\overline{r}_1 L_2 \vee \overline{r}_1 e_2 L_4) \vee \ldots = \\ &= M_C(s_4) \vee DC^+(\phi_4, s_4). \end{aligned}$$

The proof can be finished by induction. ■

Next lemma states a similar result for the nihil part of initial digraphs. We will need it to prove invariance of the nihil part of the initial digraph.

Lemma 7-5. *With notation as above and assuming that $CC^-(\phi_n, s_n)$ is satisfied, the following identity may be* ***or****ed to $M_N(s_n)$ without changing it:*

$$DC^-(\phi_n, s_n) = \overline{e}_n \overline{D}_n \nabla_1^{n-2} \left(\overline{e}_x r_y\right). \tag{81}$$

Proof

□We follow the same scheme as in the proof of Lemma 7-4. Let's start with three productions. Recall that $M_N(s_3) = K_1 \vee$ *other_terms* and that $K_1 = K_1 \vee r_1 = K_1 \vee r_1 \vee r_1 \overline{e}_3 \overline{D}_3$. Note that $r_1 \overline{e}_3 \overline{D}_3$ is eq. (81) for $n = 3$.

For $n = 4$, besides the term $r_1 \overline{e}_4 \overline{D}_4$ we need to get $\overline{e}_1 r_2 \overline{e}_4 \overline{D}_4$ (because we have that $DC^-(\phi_4, s_4) = \overline{e}_4 \overline{D}_4 \left(r_1 \vee \overline{e}_1 r_2\right)$). The first two terms of the negative initial digraph for four productions are $M_N(s_4) = K_1 \vee \overline{e}_1 K_2$. Again, it is not necessary to consider the whole formula for $M_N(s_4)$:

$$\begin{aligned} M_N(s_4) &= (K_1 \vee r_1) \vee (\overline{e}_1 K_2 \vee r_2 \overline{e}_1) \vee \ldots = \\ &= \left(K_1 \vee r_1 \vee r_1 \overline{e}_4 \overline{D}_4\right) \vee \left(\overline{e}_1 K_2 \vee \overline{e}_1 r_2 \vee \overline{e}_1 r_2 \overline{e}_4 \overline{D}_4\right) \vee \ldots = \\ &= \left(K_1 \vee r_1 \overline{e}_4 \overline{D}_4\right) \vee \left(\overline{e}_1 K_2 \vee \overline{e}_1 r_2 \overline{e}_4 \overline{D}_4\right) \vee \ldots = \\ &= M_N(s_4) \vee DC^-(\phi_4, s_4). \end{aligned}$$

The proof can be finished by induction. ■

If conditions $CC^-(\phi_n, s_n)$ and $DC^-(\phi_n, s_n)$ are applied independently of $CC^+(\phi_n, s_n)$ and $DC^+(\phi_n, s_n)$ then the expression

$$DC^-(\phi_n, s_n) = K_n \nabla_1^{n-2} \left(\overline{e_x}\, r_y\right) \tag{82}$$

should be used instead of the definition given by equation (81).

We are ready to formally state a characterization of G-congruence in terms of congruence conditions CC.

Theorem 7-6 (G-congruence). *With notation as above and assuming compatibility and coherence, sequence s_n and $\phi(s_n)$ are G-congruent if $CC^+(\phi_n, s_n) \vee i\, CC^-(\phi_n, s_n) = 0$, where*

$$CC^+(\phi_n, s_n) = L_n \nabla_1^{n-1} \overline{e}_x K_y \left(r_y \vee e_n\right) \tag{83}$$
$$CC^-(\phi_n, s_n) = K_n \nabla_1^{n-1} \overline{r}_x L_y \left(e_y \vee r_n\right). \tag{84}$$

Also, s_n and $\delta(s_n)$ are G-congruent if $CC^+(\delta_n, s_n) \vee i\, CC^-(\delta_n, s_n) = 0$, with

$$CC^+(\delta_n, s_n) = L_1 \nabla_2^{n} \overline{e}_x K_y \left(r_y \vee e_1\right) \tag{85}$$
$$CC^-(\delta_n, s_n) = K_1 \nabla_2^{n} \overline{r}_x L_y \left(e_y \vee r_1\right). \tag{86}$$

Proof
□First, using $CC^+(\phi_i, s_i)$ and $DC^+(\phi_i, s_i)$, we will prove $M_C(s) = M_C(s')$ for three and five productions. Identities $a \vee \overline{a}\, b = a \vee b$ and $\overline{a} \vee a\, b = \overline{a} \vee b$ will be used.

$$\begin{aligned} M_C(s_3) \vee CC^+(\phi_3, s_3) \vee DC^+(\phi_3, s_3) = {} & \left[L_1 \vee \overline{r}_1 L_2 \vee \overline{r}_1 \overline{r}_2 L_3\right] \vee \\ & \vee \left[r_1 L_3 \vee \overline{e}_1 r_2 L_3 \vee r_3 L_1 \vee \overline{r}_1 r_3 L_2\right] \vee \left[e_1 L_3\right] = \\ & = L_1 \vee \overline{r}_1 L_2 \vee \cancel{\overline{r}}_1 \overline{r}_2 L_3 \vee r_1 L_3 \vee \cancel{\overline{e}}_1 r_2 L_3 \vee e_1 L_3 = \\ & = L_1 \vee \overline{r}_1 L_2 \vee \cancel{\overline{r}}_2 L_3 \vee r_2 L_3 \vee L_3 \left(r_1 \vee e_1\right) = \\ & = L_1 \vee \overline{r}_1 L_2 \vee L_3. \end{aligned}$$

In our first step, as neither $r_3 L_1$ nor $\overline{r}_1 r_3 L_2$ are applied to $M_C(s_3)$, they have been omitted (for example, $L_1 \vee r_3 L_1 = L_1$). Once $r_1 L_3$, $e_1 L_3$ and $r_2 L_3$ have been used, they are omitted as well.

Let's check out $M_C(s_3')$, where in the second equality $r_1 L_3$ and $r_2 \overline{e}_1 L_3$ are ruled out since they are not used:

$$\begin{aligned} M_C(s_3') \vee CC^+(\phi_3, s_3') = {} & \left[\overline{r}_3 L_1 \vee \overline{r}_1 \overline{r}_3 L_2 \vee L_3\right] \vee \left[r_1 L_3 \vee r_2 \overline{e}_1 L_3 \vee r_3 L_1 \vee \overline{r}_1 r_3 L_2\right] = \\ & = \cancel{\overline{r}}_3 L_1 \vee \overline{r}_1\ \cancel{\overline{r}}_3 L_2 \vee L_3 \vee r_3 L_1 \vee \overline{r}_1 r_3 L_2 = \\ & = L_1 \vee \overline{r}_1 L_2 \vee L_3. \end{aligned}$$

The case for five productions is almost equal to that of three productions but it is useful to illustrate in detail how $CC^+(\phi_5, s_5)$ and $DC^+(\phi_5, s_5)$ are used to prove that $M_C(s_5) = M_C(s_5')$ in a more complex situation. The key point is the transformation $\overline{r}_1 \overline{r}_2 \overline{r}_3 \overline{r}_4 L_5 \longmapsto L_5$ and the following identities show the way to proceed:

$$\begin{aligned} \cancel{\overline{r}}_1\, \overline{r}_2\, \overline{r}_3\, \overline{r}_4\, L_5 \vee r_1 L_5 &= \overline{r}_2\, \overline{r}_3\, \overline{r}_4\, L_5 \\ \cancel{\overline{r}}_2\, \overline{r}_3\, \overline{r}_4\, L_5 \vee \cancel{\overline{e}}_1\, r_2 L_5 \vee e_1 L_5 &= \overline{r}_3\, \overline{r}_4\, L_5 \\ \cancel{\overline{r}}_3\, \overline{r}_4\, L_5 \vee \cancel{\overline{e}}_1\ \cancel{\overline{e}}_2 r_3 L_5 \vee e_1 L_5 \vee \cancel{\overline{r}}_1 e_2 L_5 \vee r_1 L_5 &= \overline{r}_4\, L_5 \\ \cancel{\overline{r}}_4 L_5 \vee \cancel{\overline{e}}_1\ \cancel{\overline{e}}_2\ \cancel{\overline{e}}_3 r_4 L_5 \vee e_1 L_5 \vee \cancel{\overline{r}}_1 e_2 L_5 \vee r_1 L_5 & \\ \vee\ \cancel{\overline{r}}_1\ \cancel{\overline{r}}_2 e_3 L_5 \vee \cancel{\overline{e}}_1 r_2 L_5 &= L_5. \end{aligned}$$

Note that we are in a kind of iterative process: what we get on the right of the equality is inserted and simplified on the left of the following one, until que get L_5. For L_4 the process is similar but shorter.

Now one example with three productions for the nihil part of the initial digraph is studied, $M_N(s_3) \vee CC^-(\phi_3, s_3) \vee DC^-(\phi_3, s_3) = M_N(s_3') \vee CC^-(\phi_3, s_3)$:

$$\begin{aligned}
M_N(s_3) \vee CC^-(\phi_3, s_3) \vee DC^-(\phi_3, s_3) = & \\
&= [K_1 \vee \overline{e}_1\, K_2 \vee \overline{e}_1\, \overline{e}_2\, K_3] \vee [e_3 K_1 \vee \overline{e}_1\, e_3 K_2 \vee e_1 K_3 \vee \\
&\vee \overline{r}_1\, e_2 K_3] \vee [r_1 K_3] = K_1 \vee \overline{e}_1\, K_2 \vee \cancel{\overline{e}_1}\, \overline{e}_2\, K_3 \vee e_1 K_3 \vee \\
&\vee \cancel{\overline{r}_1}\, e_2 K_3 \vee r_1 K_3 = K_1 \vee \overline{e}_1\, K_2 \vee \cancel{\overline{e}_2}\, K_3 \vee e_2 K_3 \vee \\
&\vee K_3\,(r_1 \vee e_1) = K_1 \vee \overline{e}_1\, K_2 \vee\ K_3. \\
M_N(s_3') \vee CC^-(\phi_3, s_3) &= [\overline{e}_3 K_1 \vee \overline{e}_1 \overline{e}_3\, K_2 \vee K_3] \vee [e_1 K_3 \vee e_2 \overline{r}_1 K_3 \vee e_3 K_1 \vee \\
&\vee \overline{e}_1 e_3 K_2] = \cancel{\overline{e}_3}\, K_1 \vee \overline{e}_1\ \cancel{\overline{e}_3} K_2 \vee K_3 \vee e_3 K_1 \vee \overline{e}_1 e_3 K_2 = \\
&= K_1 \vee \overline{e}_1\, K_2 \vee K_3.
\end{aligned}$$

Notice that the procedure followed to show $M_N(s_3) = M_N(s_3')$ is completely analogous to that of $M_C(s_3) = M_C(s_3')$. ■

As happened with coherence in Th 5-1 and compatibility in Prop. 7-1, $CC^+ \vee iCC^-$ in Th. 7-6 provides information on which elements may prevent congruence.

There are some relevant topics that we have not mentioned such as sequential independence,[22] application conditions and graph constraints.[23] We briefly discuss them now.

Swaps and productions are closely related, but one does not substitute the other. With respect to the image of a sequence and sequential independence, sticking to swaps to the detriment of productions has its effect on the interpretation of operations. On the positive side, among many other things, swaps are a nice redefinition and generalization of productions that take into account the certainty and nihil parts; on the negative side, our intuition needs to be adjusted. For example, consider a production p that only deletes edge $(1, 2)$ and does nothing else. Suppose that it is applied twice to the graph G that consists of nodes 1, 2 and edge $(1, 2)$. In this case $p; p(G) = G$ which is algebraically correct. However, it does not encode "delete edge $(1, 2)$ twice". The point here is that of completion: we would rather have considered its application to G', made up of nodes 1, $1'$ and 2 and edges $(1, 2)$ and $(1', 2)$. A similar reasoning shows that sequential independence is "granted" if we rely only on algebraic operations and do not pay attention to completion:

$$p_2; p_1(\mathcal{L}) = \langle\langle \mathcal{L}, P(p_1)\rangle, P(p_2)\rangle = \mathcal{L}P(p_1)P(p_2) = \mathcal{L}P(p_2)P(p_1) = p_1; p_2(\mathcal{L}).$$

Previous comments highlight some of the reasons why coherence, compatibility, initial digraph and G-congruence are so valuable, justifying their inclusion and also linking Secs. 5, 6 and 7 to Sec. 4.

Regarding application conditions and graph constraints, they are not difficulty related to what has been presented so far. If they are allowed to be applied to $g \in \mathfrak{G}$ instead of the restricted case that we have studied ($\mathfrak{H}$), we may impose limits on what elements cannot be

[22]Productions p_1 and p_2 are sequentially independent if $p_2; p_1$ and $p_1; p_2$ output the same result. This concept can be generalized to more than two productions. This topic is studied in detail in [12].

[23]They are both means to establish restrictions on the application of productions. These topics are studied in detail in [12].

added or deleted by sequences of productions (swaps). This is because if one edge is in the certainty part and in the nihil part, it cannot be deleted by any swap. On the contrary, if one edge does not appear either in the certainty or in the nihil parts, it is not possible for a swap to add it.

If we call any of these situations a *swap restriction*, it can be guaranteed that a sequence will not add or delete (or both) some element, despite the actual definition of the productions that make up the sequence or the grammar.

8 Conclusions and Future Work

In this paper we have given a comprehensive study of MGG. Besides, some new concepts such as swaps and (strict) Boolean complexes have been introduced and almost all results have been generalized. We believe it is a natural representation in the MGG context, as productions act on pairs of graphs $(L, K) \xrightarrow{p} (p(L), p^{-1}(K))$. Relevant algebraic structures for their study have been introduced (PMCA, $\mathfrak{G}$, $\mathfrak{H}$). Swaps allow studying and classifying productions according to their dynamic behaviour, defining a surjective morphism into the self-adjoint graphs in $\mathfrak{H}$.

With respect to other similar approaches to MGGs, in [15] the DPO approach was implemented using Mathematica. In that work, (simple) digraphs were represented by Boolean adjacency matrices. This is the only similarity with our work, as our goal is to develop a theory for (simple) graph rewriting based on Boolean matrix algebra. Other somehow related work is the relational approaches of [5, 6], but they rely on category theory for expressing the rewriting. It is also worth mentioning the set-theoretic approaches to graph transformation [3, 13]. Even though some of these approaches have developed powerful analysis techniques and efficient tool implementations, the rewriting is usually limited (e.g. a node or edge can be replaced by a subgraph). Altogether, our work is original as we encode in rules not only static information (pre- and post-conditions), but also the dynamics (element addition and deletion). Moreover, our new formalization allows a compact representation of positive and implicit negative information. This new approach to graph transformation has made possible to analyse new concepts in the literature (like e.g. the initial digraphs, nihilation matrix, swaps, congruence) and extend others to sequences of arbitrary finite length (e.g, sequential independence).

Our main interest for further research is complexity theory through MGGs. Complexity theory [4, 8] is concerned with the study of the *intrinsic complexity* of computational tasks. Traditionally, it has been studied through abstract devices able to represent the notion of algorithm, such as Turing Machines or Boolean Circuits [16]. Our proposal is to use MGG instead, as its algebraic nature allows using results from different branches of mathematics such as logics, group theory and Boolean algebra. For this purpose, it is first necessary to study MGG as a model of computation. Also, measures of complexity and the like are mandatory. Identities like (8) suggest the use of **xor** metrics.

Another promising idea might be to encode properties of graphs (such as coloring) using graph grammars, translating *static* properties into equivalent *dynamic* properties of associated sequences. We are also working in the introduction of abstract harmonic analysis in MGG. Finally, there are many more topics for further research, e.g. graph constraints, applicability, reachability, confluence and infinite graphs, some of which we have already commented on.

Acknowledgements: The authors would like to thank an anonymous referee for his/her extremely useful comments. Pedro Pablo wants to thank the open source and the copyleft communities. *SAGE* (http://www.sagemath.org/) has been used for some calculations. *OpenOffice Drawing* (http://www.openoffice.org/) has been used with Figs. 4, 5, 7, 8 and 9. *The Gimp* (http://www.gimp.org/) has helped with some finishing touches. *Emacs* (http://www.gnu.org/software/emacs/) is unvaluable for typing and *teTeX* for LaTeX. This work has been partially sponsored by the Spanish Ministry of Science and Innovation, project METEORIC (TIN2008-02081/TIN).

References

[1] Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. 1999. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 2 (Applications, Languages and Tools)*. World Scientific.

[2] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G. 2006. *Fundamentals of Algebraic Graph Transformation.* Springer.

[3] Engelfriet, J., Rozenberg, G. 1997. *Node Replacement Graph Grammars.* In [14], pp.: 1-94.

[4] Goldreich, O. 2008. *Computational Complexity: A Conceptual Approach.* Cambridge University Press.

[5] Kahl, W. 2002. *A Relational Algebraic Approach to Graph Structure Transformation*. Tech. Rep. 2002-03, Universitat der Bundeswehr Munchen.

[6] Mizoguchi, Y., Kuwahara, Y. 1995. *Relational Graph Rewritings*. TCS 141:311–328, Elsevier.

[7] Mulmuley, K., Sohoni, M. A. 2008. *On P vs. NP, Geometric Complexity Theory, and the Flip I: a high level view*. arXiv:0709.0748v1.

[8] Papadimitriou, C. 1994. *Computational Complexity.* Addison-Wesley.

[9] Pérez Velasco, P. P., de Lara, J. 2006. *Matrix Approach to Graph Transformation: Matching and Sequences*. LNCS 4178, pp.:122-137. Springer.

[10] Pérez Velasco, P. P., de Lara, J. 2006. *Petri Nets and Matrix Graph Grammars: Reachability*. EC-EAAST(2).

[11] Pérez Velasco, P. P., de Lara, J. 2007. *Using Matrix Graph Grammars for the Analysis of Behavioural Specifications: Sequential and Parallel Independence.* ENTCS 206, pp.:133-152. Elsevier.

[12] Pérez Velasco, P. P. 2008. *Matrix Graph Grammars*. E-book available at: `http://www.mat2gra.info/`, CoRR abs/0801.1245.

[13] Raoult, J.-C., Vosisin, F. 1992. *Set-Theoretic Graph Rewriting.* INRIA Rapport de Recherche no. 1665.

[14] Rozenberg, G. (ed.) 1997. *Handbook of Graph Grammars and Computing by Graph Transformation.* Vol.1 (Foundations), World Scientific.

[15] Valiente, G. 1998. *Grammatica: An Implementation of Algebraic Graph Transformation on Mathematica*. Proc. 6^{th} Works on Theory and Application of Graph Transformations. pp. 261–267.

[16] Vollmer, H. 1999. *Introduction to Circuit Complexity. A Uniform Approach.* Springer.