Termination of graph rewriting systems through language theory

Guillaume Bonfante, Miguel Couceiro

TERMINATION OF GRAPH REWRITING SYSTEMS THROUGH LANGUAGE THEORY

Guillaume Bonfante
Université de Lorraine, CNRS, LORIA
F-54000 Nancy, France
guillaume.bonfante@univ-lorraine.fr

Miguel Couceiro
Université de Lorraine, CNRS, Inria, LORIA
F-54000 Nancy, France
miguel.couceiro@loria.fr

Dedicated to Maurice Pouzet on the occasion of his 75th birthday

ABSTRACT

The termination issue that we tackle is rooted in Natural Language Processing where computations are performed by graph rewriting systems (GRS) that may contain a large number of rules, often in the order of thousands. This asks for algorithmic procedures to verify the termination of such systems. The notion of graph rewriting that we consider does not make any assumption on the structure of graphs (they are not "term graphs", "port graphs" nor "drags"). This lack of algebraic structure led us to proposing two orders on graphs inspired from language theory: the matrix multiset-path order and the rational embedding order. We show that both are stable by context, which we then use to obtain the main contribution of the paper: under a suitable notion of "interpretation", a GRS is terminating if and only if it is compatible with an interpretation.

Keywords Graph Rewriting · Termination · Order · Natural Language Processing

1 Introduction

Computer linguists rediscovered a few years ago that graph rewriting is a good model of computation for rule-based systems. They used traditionally terms, see for instance Chomsky's Syntagmatic Structures [1]. But usual phenomena such as anaphora do not fit really well within such theories. In such situations, graphs behave much better. For examples of graph rewriting in natural language processing, we refer the reader to the parsing procedure by Guillaume and Perrier [2] or the word ordering modeling by Kahane and Lareau [3]. The first named author with Guillaume and Perrier designed a graph rewriting model called GREW [4] that is adapted to natural language processing.

The rewriting systems developed by the linguists often contain a huge number of rules, e.g., those synthesized from lexicons (e.g. some rules only apply to transitive verbs). For instance, in [2], several systems are presented, some with more than a thousand of rules. Verifying properties such as termination by hand thus becomes intractable. This fact motivates our framework for tackling the problem of GRS termination.

Following the tracks of term rewriting, for which the definition is essentially fixed by the algebraic structure of terms, many approaches to graph rewriting emerged in past years. Some definitions (here meaning semantics) are based on a categorical framework, e.g., the double pushout (DPO) and the single pushout (SPO) models, see [5]. To make use of algebraic potential, some authors make some, possibly weak, hypotheses on graph structures, see for instance the main contribution by Courcelle and Engelfriet [6] where graph decompositions, graph operations and transformations are described in terms of monadic second-order logics (with the underlying decidability/complexity results). In this spirit, Ogawa describes a graph algebra under a limited tree-width condition [7].

Another line of research follows from the seminal work by Lafont [8] on interaction nets. The latter are graphs where nodes have some extra structure: nodes have a label related to some arity and co-arity. Moreover, nodes have some "principal gates" (ports) and rules are actionned via them. One of the main results by Lafont is that rewriting in this setting is (strongly) confluent. This approach has been enriched by Fernandez, Kirchner and Pinaud [9],

who implemented a fully operational system called PORGY with strategies and related semantics. Also, it is worth mentioning the graph rewriting as described by Dershowitz and Jouannaud [10]. Here, graphs are seen as a generalization of terms: symbols have a (fixed) arity, graphs are connected via some sprouts/variables as terms do. With such a setting, a good deal of term rewriting theory also applies to graphs.

Let us come back to the initial problem: termination of graph rewriting systems in the context of natural language processing. We already mentioned that rule sets are large, which makes manual inspection impossible. Moreover, empirical studies fail to observe some of the underlying hypotheses of the previous frameworks. For instance, there is no clear bound on tree-width: even if input data such as dependency graphs are almost like trees, the property is not preserved along computations. Also, constraints on node degrees are also problematic: graphs are usually sparse, but some nodes may be highly connected. To illustrate, consider the sentence "The woman, the man, the child and the dog eat together". The verb "eat" is related to four subjects and there is no a priori limit on this phenomenon. Typed versions (those with fixed arity) are also problematic: a verb may be transitive or not. Moreover, rewriting systems may be intrinsically nondeterministic. For instance, if one computes the semantics of a sentence out of its grammatical analysis, it is quite common there are multiple solutions. To further illustrate nondeterminism consider the well known phrasal construction "He saw a man with a telescope" with two clear readings.

Some hypotheses are rather unusual for standard computations, e.g., fixed number of nodes. Indeed, nodes are usually related to words or concepts (which are themselves closely related to words). A paraphrase may be a little bit longer than its original version, but its length can be easily bounded by the length of the original sentence up to some linear factor. In GREW, node creations are restricted. To take into account the rare cases for which one needs extra nodes, a "reserve" is allocated at the beginning of the computation. All additional nodes are taken from the reserve. Doing so has some efficiency advantages, but that goes beyond the scope of the paper. Also, node and edge labels, despite being large, remain finite sets: they are usually related to some lexicons. These facts together have an important impact on the termination problem: since there are only finitely many graphs of a given size, rewriting only leads to finitely many outcomes. Thus, deciding termination for *a particular input graph* is decidable. However, our problem is to address termination in the class of *all* graphs. The latter problem is often referred to as *uniform termination*, whereas the former is refereed to as *non-uniform*. For word rewriting, uniform termination of non increasing systems constituted a well known problem, and was shown to be undecidable by Sénizergues in [11].

This paper proposes a novel approach for termination of graph rewriting. In a former paper [12], we proposed a solution based on label weights. Here, the focus is on the description (and the ordering) of paths within graphs. In fact, paths in a graph can be seen as good old regular languages. The question of path ordering thus translates into a question of regular language orderings. Accordingly, we define the *graph multi-set path ordering* that is related to that in [13]. Dershowitz and Jouannaud, in the context of drag rewriting, consider a similar notion of path ordering called GPO (see [14]). Our definitions diverge from theirs in that our graph rewriting model is quite different: here, we do not benefit (as they do) from a good algebraic structure. Our graphs have no heads, tails nor hierarchical decomposition. In fact, our ordering is not even well founded! Relating the two notions is nevertheless interesting and left for further work. Plump [15] also defines path orderings for term graphs, but those behave like sets of terms.

One of our graph orderings will involve matrices, and orderings on matrices. Nonetheless, as far as we see, there is no relationship with matrix interpretations as defined by Endrullis, Waldmann and Zantema [16].

The paper is organized as follows. In Section 2 we recall the basic background on graphs and graph rewriting systems (GRS) that we will need throughout the paper, and introduce an example that motivated our work. In Section 3 we consider a language theory approach to the termination of GRSs. In particular, we present the language matrix, and the matrix multiset path order (Subsection 3.4) and the rational embedding order (Subsection 3.5). We also propose the notion of stability by context (Subsection 3.6) and show that both orderings are stable under this condition (Subsection 3.7). In Section 4 we propose notion of graph interpretability and show one of our main results, namely, that a GRS is terminating if and only if it is compatible with interpretations.

Main contributions: The two main contributions of the paper are the following.

1. We propose two orders on graphs inspired from language theory, and we show that both are monotonic and stable by context.

2. We propose a notion of graph interpretation, and show that GRSs that are terminating are exactly those that are compatible with such interpretations.

2 Notation and Graph Rewriting

In this section we recall some general definitions and notation. Given an alphabet Σ, the set of words (finite sequences of elements of Σ) is denoted by Σ^*. The concatenation of two words v and w is denoted by $v \cdot w$. The empty word, being the neutral element for concatenation, is denoted by 1_Σ or, when clear from the context, simply by 1. Note that $\langle \Sigma^*, 1, \cdot \rangle$ constitutes a monoid.

A *language* on Σ is some subset $L \subseteq \Sigma^*$. The set of all languages on Σ is $\mathcal{P}(\Sigma^*)$. The addition of two languages $L, L' \subseteq \Sigma^*$ is defined by $L + L' = \{w \mid w \in L \vee w \in L'\}$. The empty language is denoted by 0 and $\langle \mathcal{P}(\Sigma^*), +, 0 \rangle$ is also a (commutative) monoid. Given some word $w \in \Sigma^*$, we will also denote by w the language made of the singleton $\{w\} \in \mathcal{P}(\Sigma^*)$. Given two languages $L, L' \subseteq \Sigma^*$, their concatenation is defined by $L \cdot L' = \{w \cdot w' \mid w \in L \wedge w' \in L'\}$. In this way, $\langle \mathcal{P}(\Sigma^*), 1, \cdot \rangle$ is also a monoid. Given the distributivity of $\times$ with respect to $+$, the 5-tuple $\langle \mathcal{P}(\Sigma^*), +, 0, 1, \cdot \rangle$ forms a semiring.

A *preorder* on a set X is a binary relation $\preceq \subseteq X^2$ that is reflexive ($x \preceq x$, for all $x \in X$) and transitive (if $x \preceq y$ and $y \preceq z$, then $x \preceq z$, for all $x, y, z \in X$). A preorder $\preceq$ is a *partial order* if it is anti-symmetric (if $x \preceq y$ and $y \preceq x$, then $x = y$, for all $x, y \in X$). An *equivalence relation* is a preorder that is symmetric ($x \preceq y \Rightarrow y \preceq x$) . Observe that each preorder $\preceq$ induces an equivalence relation $\sim$: $a \sim b$ if $a \preceq b$ and $b \preceq a$. The strict part of $\preceq$ is then the relation: $x \prec y$ if and only if $x \preceq y$ and $\neg(x \sim y)$. We also mention the "dual" preorder $\succeq$ of $\preceq$ defined by: $x \succeq y$ if and only if $y \preceq x$. A preorder $\preceq$ is said to be *well-founded* if there is no infinite chain $\cdots \prec x_2 \prec x_1$ or, equivalently, $x_1 \succ x_2 \succ \cdots$.

The remainder of this section may be found in [4] and we refer the reader to it for an extended presentation. We suppose given a (finite) set Σ_N of *node labels*, a (finite) set Σ_E of *edge labels* and we define graphs accordingly. A graph is a triple $G = \langle N, E, \ell \rangle$ with $E \subseteq N \times \Sigma_\mathrm{E} \times N$ and $\ell : N \to \Sigma_\mathrm{N}$ is the labeling function of nodes. Note that there may be more than one edge between two nodes, but at most one is labeled with some $e \in \Sigma_\mathrm{E}$. In the sequel, we use the notation $m \xrightarrow{e} n$ for an edge $(m, e, n) \in E$.

Given a graph G, we denote by $\mathcal{N}_G$, $\mathcal{E}_G$ and ℓ_G respectively its sets of nodes, edges and labeling function. We we will also (abusively) use the notation $m \in G$ and $m \xrightarrow{e} n \in G$ instead of $m \in \mathcal{N}_G$ and $m \xrightarrow{e} n \in \mathcal{E}_G$ when the context is clear. Furthermore, in 

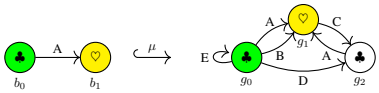
, a, b are nodes, ♣, ♡ are the respective node labels and A is the edge label (here between a and b).

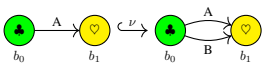
The set of graphs on node labels Σ_N and edge labels Σ_E is denoted by $\mathcal{G}_{\Sigma_\mathrm{N}, \Sigma_\mathrm{E}}$ or $\mathcal{G}$ in short. Two graphs G and G' are said to share their nodes when $\mathcal{N}_G = \mathcal{N}_{G'}$. Given two graphs G and G' such that $\mathcal{N}_G \subseteq \mathcal{N}_{G'}$, set $G \blacktriangleleft G' = \langle \mathcal{N}_{G'}, \mathcal{E}_G \cup \mathcal{E}_{G'}, \ell \rangle$ with $\ell(n) = \ell_G(n)$ if $n \in \mathcal{N}_G$ and $\ell(n) = \ell_{G'}(n)$, otherwise.

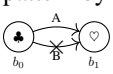
A graph morphism μ between a source graph G and a target graph H is a function $\mu : \mathcal{N}_G \to \mathcal{N}_H$ that preserves edges and labelings, that is, for all $m \xrightarrow{e} n \in G$, $\mu(m) \xrightarrow{e} \mu(n) \in H$ holds, and for any node $n \in G$: $\ell_G(n) = \ell_H(\mu(n))$. A *basic pattern* is a graph, and a *basic pattern matching* is an injective morphism from a basic pattern P to some graph G. Given such a morphism $\mu : G \to G'$, we define $\mu(G)$ to be the sub-graph of G' made of the nodes $\{\mu(n) \mid n \in \mathcal{N}_G\}$, of the edges $\{\mu(m) \xrightarrow{e} \mu(n) \mid m \xrightarrow{e} n \in G\}$ and node labels $\mu(n) \mapsto \ell_G(n)$.

A *pattern* is a pair $P = \langle P_0, \vec{\nu} \rangle$ made of a basic pattern P_0 and a sequence of injective morphisms $\nu_i : P_0 \to N_i$, called *negative conditions*. The basic pattern describes what must be *present* in the target graph G, whereas negative conditions say what must be *absent* in the target graph. Given a pattern $P = \langle P_0, \vec{\nu} \rangle$ and a graph G, a *pattern morphism* is an injective morphism $\mu : P_0 \to G$ for which there is no morphism ξ_i such that $\mu = \xi_i \circ \nu_i$.

Example 1. Consider the basic pattern morphism $\mu : P_0 \to G$ (colors define the mapping):



The pattern $P = \langle P_0, [\nu] \rangle$ with ν defined by 

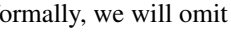
prevents the application of the morphism above. Indeed, $\xi = b_0 \mapsto g_0, b_1 \mapsto g_1$ is such that $\xi \circ \nu = \mu$. When there is only one negative condition, we represent the pattern by crossing nodes and edges which *are not* within the basic pattern. For instance, the pattern P above looks like 

that we hope is self-explanatory.

In this paper we think of graph transformations as sequences of "basic commands".

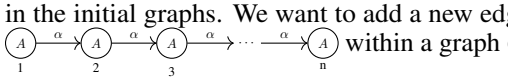
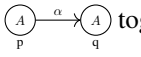
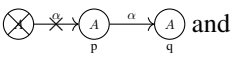
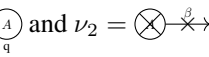
Definition 1 (The command language)**.** There are three basic commands: $\texttt{label}(p, \alpha)$ for node renaming, $\texttt{del\_edge}(p, e, q)$ for edge deletion and $\texttt{add\_edge}(p, e, q)$ for edge creation. In these basic commands, p and q are nodes, α is some node label and e is some edge label. A pattern $\langle P_0, \vec{\nu} \rangle$ is compatible with a command whenever all nodes that it involves belong to P_0.

Definition 2 (Operational semantics)**.** Given a pattern $P = \langle P_0, \vec{\nu} \rangle$ compatible with some command c, and some pattern matching $\mu : P \to G$ where G is the graph on which the transformation is applied, we have the following possible cases: $c = \texttt{label}(p, \alpha)$ turns the label of $\mu(p)$ into α, $c = \texttt{del\_edge}(p, e, q)$ removes $\mu(p) \xrightarrow{e} \mu(q)$ if it exists, otherwise does nothing, and $c = \texttt{add\_edge}(p, e, q)$ adds the edge $\mu(p) \xrightarrow{e} \mu(q)$ if it does not exist, otherwise does nothing. The graph obtained after such an application is denoted by $G \cdot_\mu c$. Given a sequence of commands $\vec{c} = (c_1, \ldots, c_n)$, let $G \cdot_\mu \vec{c}$ be the resulting graph, i.e., $G \cdot_\mu \vec{c} = (\cdots((G \cdot_\mu c_1) \cdot_\mu c_2) \cdot_\mu \cdots c_n)$.

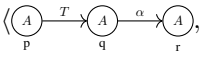
Remark 1. We took a slightly simplified version of patterns. Actually, in [4], we have negative conditions within patterns to avoid some rule applications. But these simplifications should be transparent with respect to termination. Nevertheless, informally, we will omit the right node in a pattern like  to say that there is no edge labeled B from node b_0.

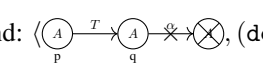
Definition 3. A rule is a pair $R = \langle P, \vec{c} \rangle$ made of a pattern and a (compatible) sequence of commands. Such a rule R applies to a graph G via a pattern morphism $\mu : P \to G$. Let $G' = G \cdot_\mu \vec{c}$, then we write $G \to_{R,\mu} G'$. We define $G \to G'$ whenever there is a rule R and a pattern morphism μ such that $G \to_{R,\mu} G'$.

2.1 The main example

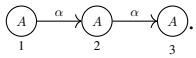
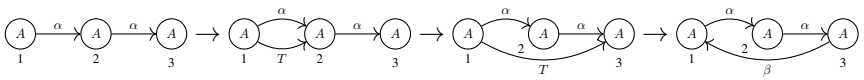
Let $\Sigma_{\mathrm{N}} = \{A\}$ and $\Sigma_{\mathrm{E}} = \{\alpha, \beta, T\}$. For the discussion, we suppose that T is a working label, that is not present in the initial graphs. We want to add a new edge β between node n and node 1 each time we find a maximal chain:  within a graph G. Consider the basic pattern $P_{init} =$  together with its two negative conditions $\nu_1 =$  and $\nu_2 =$  . We consider three rules:

Init: $\langle \langle P_{init}, [\nu_1, \nu_2] \rangle, (\texttt{add\_edge}(p, T, q)) \rangle$ which fires the transitive closure.

Follow: $\langle$  $, (\texttt{add\_edge}(p, T, r), \texttt{del\_edge}(p, T, q)) \rangle$ which follows the chain.

End: $\langle$  $, (\texttt{del\_edge}(p, T, q), \texttt{add\_edge}(q, \beta, p)) \rangle$ which stops the procedure.

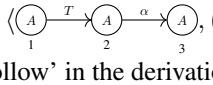
To prevent all pathological cases (e.g., when the edge β is misplaced, when two chains are crossing, and so on), we could introduce more sophisticated patterns. But, since that does not change issues around termination, we avoid obscuring rules with such technicalities.

Example 2. Consider the graph $G =$  . By applying succesively 'Init', 'Follow' and 'End', G rewrites as: 

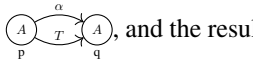
2.2 Three technical facts about Graph Rewriting

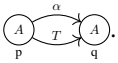
It is well known that the main issue with graph rewriting definitions is the way the context is related to the pattern image and its rewritten part. We shall tackle this issue with Proposition 1.

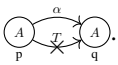
Self-application Let $R = \langle P, \vec{c} \rangle$ be the rule made of a pattern $P = \langle P_0, \vec{\nu} \rangle$ and a sequence of commands $\vec{c}$. There is the identity morphism $1_{P_0} : P_0 \to P_0$, and thus we can apply rule R on P_0 itself, that is, $P_0 \to_{R,1_{P_0}} P_0' = P_0 \cdot_{1_{P_0}} \vec{c}$. We call this latter graph the *self-application of* R.

Rule node renaming To avoid heavy notation, we will use the following trick. Suppose that we are given a rule $R = \langle P, \vec{c} \rangle$, a graph G and a pattern morphism $\mu : P \to G$. Let $P = \langle P_0, \vec{\nu} \rangle$. We define R_μ to be the rule obtained by renaming nodes p in P_0 to $\mu(p)$ (and their references within $\vec{c}$). For instance, the rule 'Follow' can be rewritten as $Follow_\mu = \langle$  $, (\texttt{add\_edge}(1, T, 3), \texttt{del\_edge}(1, T, 2)) \rangle$ where μ denotes the pattern morphism used to apply 'Follow' in the derivation. Observe that: (i) the basic pattern of R_μ is actually $\mu(P_0)$, which is a subgraph of

G, (ii) $\iota : \mu(P_0) \to G$ mapping $n \mapsto n$ is a pattern matching, and (iii) applying rule R_μ with ι is equivalent to applying rule R with μ. In other words, $G \to_{R,\mu} G'$ if (and only if) $G \to_{R_\mu,\iota} G'$. To sum up, we can always rewrite a rule so that its basic pattern is *actually* a subgraph of G.

Uniform rules Let us consider rule 'Init' above. It applies on: 

, and the result is the graph itself: 

. Indeed, we cannot add an already present edge (relative to a label) within a graph. Thus, depending on the graph, the rule will or will not append an edge. Such an unpredictable behavior can be easily avoided by modifying the pattern of 'Init' to: 

. The same issue may come from edge deletions. A *uniform* rule is one for which commands apply (that is, modify the graph) for each rule application. Since this is not the scope of the paper, we refer the reader to [4] for a precise definition of uniformity. We will only observe two facts.

First, any rule can be replaced by a finite set of uniform rules (using negative conditions as above) that operate identically. Thus, we can always suppose that rules are uniform.

Second, the following property holds for uniform rules (see [4]§7 for a proof).

Proposition 1. Suppose that $G \to_{R,\iota} G'$ with $R = \langle P, \vec{c} \rangle$ and $P = \langle P_0, \vec{\nu} \rangle$ (the basic pattern P_0 being a subgraph of G). Let C be the graph obtained from G by deleting the edges in P_0. Then $G = P_0 \blacktriangleleft C$ and $G' = P'_0 \blacktriangleleft C$ with P'_0 being the self-application of the rule. Moreover, $\mathcal{E}_C \cap \mathcal{E}_{P_0} = \emptyset$ and $\mathcal{E}_C \cap \mathcal{E}_{P'_0} = \emptyset$.

Throughout the remainder of the paper we assume that all rules are uniform.

3 Termination of Graph Rewriting Systems

By a *graph rewriting system* (GRS) we simply mean a set of graph rewriting rules (see Section 2). A GRS $\mathbf{R}$ is said to be *terminating* if there is no infinite sequence $G_1 \to G_2 \to \cdots$. Such sequences, whether finite or not, are called *derivations*.

Since there is no node creation (neither node deletion) in our notion of rewriting, any derivation starting from a graph G will lead to graphs whose size is the size of G. Since there are only finitely many such graphs, we can decide the termination for this particular graph G. However, the question we address here is the *uniform termination problem* (see Section 1).

Remark 2. Suppose that we are given a strict partial order $\succ$, not necessarily well founded. If $G \to G'$ implies $G \succ G'$ for all graphs G and G', then the system is terminating. Indeed, suppose it is not the case, let $G_1 \to G_2 \to \cdots$ be an infinite reduction sequence. Since there are only finitely many graphs of size of G_1, it means that there are two indices i and j such that $G_i \to \cdots \to G_j$ with $G_i = G_j$. But then, since $G_i \succ G_{i+1} \succ \cdots \succ G_j$, we have that $G_i \succ G_j = G_i$ which is a contradiction.

A similar argument was exhibited by Dershowitz in [17] in the context of term rewriting. For instance, it is possible to embed rewriting within real numbers rather than natural numbers to prove termination.

Let us try to prove the termination of our main example (see Subsection 2.1). Rules such as '$Init$' and 'End' are "simple": we put a weight on edge labels $\omega : \Sigma_{\mathrm{E}} \to \mathbb{R}$ and we say that the weight of a graph is the sum of the weights of its edges labels. Set $\omega(\alpha) = 0, \omega(\beta) = -2$ and $\omega(T) = -1$. Then, rules 'Init' and 'End' decrease the weight by 1 and, since rule 'Follow' keeps the weight constant, it means the two former rules can be applied only finitely many times. Observe that negative weights are no problem with respect to Remark 2.

But how do we handle rule 'Follow'? No weights as above can work.

3.1 A language point of view

Let $G \to G'$ be a rule application. The set of nodes stays constant. Let us think of graphs as automata, and let us forget about node labeling for the time being. Let Σ_{E} be the set of edge labels. Consider a pair of states (nodes), choose one to be the initial state and one to be the final state. Thus the automaton (graph) defines some regular language on Σ_{E}. In fact, the automaton describes n^2 languages (one for each pair of states).

Now, let us consider the effect of graph rewriting in terms of languages. Consider an application of the 'Follow' rule: $G \to G'$. Any word to state r that goes through the transitions $p \xrightarrow{T} q \xrightarrow{\alpha} r$ can be mapped to a shorter one in G' via the transition $p \xrightarrow{T} r$. The languages corresponding to state r contain shorter words. The remainder of this section is

devoted to formalizing this intuition into proper orders on graphs. For that, we will need to *count* the number of paths between any two states. Hence, we shall introduce $\mathbb{N}$-rational expressions, that is, *rational expression with multiplicity*. See, e.g., Sakarovitch's book [18] for an introduction and justifications of the upcoming constructions. We introduce here the basic ideas.

3.2 Formal series

A formal series on Σ (with coefficients in $\mathbb{N}$) is a (total) function $s : \Sigma^* \to \mathbb{N}$. Given a word w, $s(w)$ is the multiplicity of w. The set of words $\underline{s} = \{w \in \Sigma^* \mid s(w) \neq 0\}$ is the *support* of s. Given $n \in \mathbb{N}$, let n be the series defined by $\mathsf{n}(w) = 0$, if $w \neq 1$, and $\mathsf{n}(1) = n$, where 1 denotes the empty word. The empty language is $\underline{0}$, the language made of the empty word is $\underline{1}$. Moreover, for $a \in \Sigma$, the series a is given by $a(w) = 0$ if $w \neq a$ and $a(a) = 1$.

Given two series s and t, their *addition* is the series $s + t$ given by $s + t(w) = s(w) + t(w)$, and their *product* is $s \cdot t$ defined by $s \cdot t(w) = \sum_{u \cdot v = w} s(u)t(v)$. The *star operation* is defined by $s^* = 1 + s + s^2 + \cdots$. The monoïd Σ^* being graded, the operation is correctly defined whenever $s(1) = 0$.

Given a series s, let $s^{\leq k}$ be its restriction to words of length less or equal to k, i.e., $s^{\leq k}(w) = 0$ whenever $|w| > k$ and $s^{\leq k}(w) = s(w)$, otherwise.

An $\mathbb{N}$-rational expression on an alphabet Σ is built upon the grammar [19]:

$$\mathsf{E} ::= a \in \Sigma \mid n \in \mathbb{N} \mid (\mathsf{E} + \mathsf{E}) \mid (\mathsf{E} \cdot \mathsf{E}) \mid (\mathsf{E}^*).$$

Thus, given the constructions mentioned in the previous paragraph, any $\mathbb{N}$-rational expression $E \in \mathsf{E}$ denotes some formal series. To each $\mathbb{N}$-rational expression corresponds an $\mathbb{N}$-automaton, which is a standard automaton with transitions labeled by a non empty linear combination $\sum_{i \leq k} n_i a_i$ with $n_i \in \mathbb{N}$ and $a_i \in \Sigma$ for all $i \leq k$.

3.3 The language matrix

Let us suppose given an edge label set Σ_{E}. Let E denote the $\mathbb{N}$-rational expressions over Σ_{E}. A matrix M of dimension $P \times P$ for some (finite) set P is an array $(M_{i,j})_{i \in P, j \in P}$ whose entries are in E. Let $\mathfrak{M}_E$ be the set of such matrices. Given a graph G, we define the matrix M_G of dimension $\mathcal{N}_G \times \mathcal{N}_G$ as follows: $M_{Gi,j} = T_1 + \cdots + T_\ell$ with $T_1, \ldots, T_\ell$ the set of labels on the transitions between state i and j if such transitions exist, otherwise 0.

Let 1_P be the unit matrix of dimension $P \times P$, that is $(1_P)_{i,j} = \mathsf{0}$ if $i \neq j$ else $\mathsf{1}$. From now on, we abbreviate the notation from 1_P to 1 if the context is clear. Then, let $M_G^* = 1 + M_G + M_G^2 + \cdots$. Each entry of M_G^* is actually an $\mathbb{N}$-regular expression (see Sakarovitch Ch. III, §4 for instance). The (infinite) sum is correctly defined since for all i, j, we have the equality $(M_G)_{i,j} = T_1 + \cdots + T_\ell$. Thus, $1 \notin \underline{(M_G)_{i,j}}$.

The question about termination can be reformulated in terms of matrices whose entries are languages (with words counted with their multiplicity). To prove the termination of the rewriting system, it is then sufficient to exhibit some order $>$ on matrices such that for any two graphs $G \to G'$, we have $M_G^* > M_{G'}^*$. To prove such a property in the infinite class of finite graphs, we will use the notion of "stable orders".

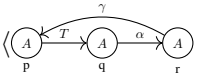
Recall the 'Follow' rule and consider the corresponding basic pattern L and its self-application R. Their respective matrices are:

$$M_L = \begin{pmatrix} 0 & T & 0 \\ 0 & 0 & \alpha \\ 0 & 0 & 0 \end{pmatrix} \quad M_R = \begin{pmatrix} 0 & 0 & T \\ 0 & 0 & \alpha \\ 0 & 0 & 0 \end{pmatrix}.$$

Observe that $(M_R)_{13} > (M_L)_{13}$. This matrix deals with edges/transitions. In order to consider paths, we need to compute M_L^* and M_R^* that are given by:

$$M_L^* = \begin{pmatrix} 1 & T & T \cdot \alpha \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix} \quad M_R^* = \begin{pmatrix} 1 & 0 & T \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix}.$$

Any word within M_R^*'s entries is a sub-word of the corresponding entry in M_L^*.

Example 3. Consider now a variation of 'Follow' made of the pattern $\langle$ 

and commands $(\texttt{add\_edge}(p, T, r), \texttt{del\_edge}(p, T, q))\rangle$. By setting L' as its pattern and R' as its self-application, we get the following

matrices:

$$M^*_{L'} = \begin{pmatrix} (T\alpha\gamma)^* & T(\alpha\gamma T)^* & T\alpha(\gamma T\alpha)^* \\ \alpha\gamma(T\alpha\gamma)^* & (\alpha\gamma T)^* & \alpha(\gamma T\alpha)^* \\ \gamma(T\alpha\gamma)^* & \gamma T(\alpha\gamma T)^* & (\gamma T\alpha)^* \end{pmatrix} \quad M^*_{R'} = \begin{pmatrix} (T\gamma)^* & 0 & T(\gamma T)^* \\ \alpha\gamma(T\gamma)^* & 1 & \alpha(\gamma T)^* \\ \gamma(T\gamma)^* & 0 & (\gamma T)^* \end{pmatrix}.$$

Again, words within $M^*_{R'}$ are sub-words of the corresponding ones in $M^*_{L'}$.

3.4 The matrix multiset path order

The order we shall introduce in this section is inspired by the notion of *multiset path ordering* within the context of term rewriting (see for instance [13]). However, in the present context of graph rewriting (to be compared with Dershowitz and Jouannaud's [14] or with Plump's [15]), the definition is a bit more complicated. Here, we do not consider an order on letters as it is done for terms.

Let $\trianglelefteq$ be the word embedding on Σ^*, that is, the smallest partial order such that $1 \trianglelefteq w$, and if $u \trianglelefteq v$, then $(u \cdot w \trianglelefteq v \cdot w$ and $w \cdot u \trianglelefteq w \cdot v$, for all $u, v, w \in \Sigma^*$. This order $\trianglelefteq$ can be extended to formal series, that is, the multiset-path ordering, see Dershowitz and Manna [20] or Huet and Oppen [21].

Definition 4 (Multiset path order)**.** The *multiset path order* is the smallest partial order on finite series such that

- if there is $w \in \underline{t}$ such that for all $v \in \underline{s}$, $v \triangleleft w$, then $s \trianglelefteq t$, and
- if $r \trianglelefteq s$ and $t \trianglelefteq u$, then $r + t \trianglelefteq s + u$.

We write $s \triangleleft t$ when $s \trianglelefteq t$ and $s \neq t$.

Proposition 2. Addition and product are monotonic with respect to the multiset-path order. Moreover, addition is strictly monotonic with respect to $\trianglelefteq$, and if $r \triangleleft s$, then $r \cdot t \triangleleft s \cdot t$ and $t \cdot r \triangleleft t \cdot s$, whenever $t \neq 0$ (otherwise, we have equality).

Proof. Addition is monotonic by definition. Actually, we prove now that it is strictly monotonic. Suppose that $r \triangleleft s$. We prove that $r + t \triangleleft s + t$, by induction (see Definition 4). Suppose that there is $w \in \underline{s}$ such that for all $v \in \underline{r}$ we have $v \triangleleft w$, then $r \triangleleft s$. Since $r(w) = 0$, then $(r + t)(w) = t(w) < s(w) + t(w) = (s + t)(w)$, and we are done. Otherwise, $r = r_0 + r_1$ and $s = s_0 + s_1$ with $r_0 \trianglelefteq s_0$ and $r_1 \trianglelefteq s_1$. One of the two inequalities must be strict (otherwise $r = s$). Suppose $r_0 \triangleleft s_0$. By definition, observe that $r_1 + t \trianglelefteq s_1 + t$. But then, $r + t = r_0 + (r_1 + t)$ and $s = s_0 + (s_1 + t)$ and we apply induction on (r_0, s_0). As addition is commutative, the result holds.

For the product, suppose that $r \trianglelefteq s$ and let t be some series. We prove $r \cdot t \trianglelefteq s \cdot t$; the other inequality $t \cdot r \trianglelefteq t \cdot s$ is similar. Again, we proceed by induction on Definition 4:

- Suppose there is $w \in \underline{s}$ such that for all $v \in \underline{r}, v \triangleleft w$. By induction on t, if $t = 0$, $r \cdot t = 0 \trianglelefteq 0 = s \cdot t$. Otherwise, $t = t_0 + v_0$ for a word v_0. Observe that $r \cdot v_0 = \sum_{v \in \underline{r}} r(v) v \cdot v_0$. Since for all $v \in \underline{r}$, $v \cdot v_0 \triangleleft w \cdot v_0$, we have $r \cdot v_0 \triangleleft w \cdot v_0 \trianglelefteq s \cdot v_0$. Now, $r \cdot t = r \cdot (t_0 + v_0) = r \cdot t_0 + r \cdot v_0$ and $s \cdot t = s \cdot t_0 + s \cdot v_0$. By induction, $r \cdot t_0 \trianglelefteq s \cdot t_0$ and since $r \cdot v_0 \trianglelefteq s \cdot v_0$, the result holds.
- Otherwise, $r = r_0 + r_1$. In this case, $s \cdot r = s \cdot r_0 + s \cdot r_1$ and $t \cdot r = t \cdot r_0 + t \cdot r_1$. The result then follows by induction.

To show strict monotonicity, suppose $r \triangleleft s$ and again proceed by case analysis. Suppose that there is some $w \in \underline{s}$ such that for all $v \in \underline{r}, v \triangleleft w$. Since $t \neq 0$, it contains at least one word v_0 such that $t = t_0 + v_0$. By $r \triangleleft s$, $r \cdot v_0 = \sum_{v \in \underline{r}} r(v) v \cdot v_0 \triangleleft \sum_{v \in \underline{s}} s(v) v \cdot v_0 = s \cdot v_0$. Now, $r \cdot t_0 \trianglelefteq s \cdot t_0$ by monotonicity. Thus $r \cdot t = r \cdot t_0 + r \cdot v_0 \triangleleft s \cdot t_0 + s \cdot v_0 = s \cdot t$ where the strict inequality is due to strict monotonicity of addition. □

Definition 5 (Matrix multiset-path order)**.** Let M and M' be two matrices with dimension $P \times P$. Write $M \trianglelefteq M'$ if for all $k \geq |P|$ and for all $(i, j) \in P \times P$, we have $M_{i,j}^{\leq k} \trianglelefteq M'_{i,j}{}^{\leq k}$.

Corollary 1. The addition and the multiplication are monotonic with respect to the matrix multiset-path order.

Proof. It follows from Proposition 2 since addition and product of matrices are defined as addition and product of their entries. □

3.5 The Rational Embedding Order

Let Σ be some fixed alphabet. A finite state *transducer* is a finite state automaton whose transitions are decorated by pairs (a, w), $a \in \Sigma$, $w \in \Sigma^*$. We refer the reader to the book of Sakarovitch [18] for details. To give the intuition, a transducer works as follows. At the beginning, the current state is set to the initial state of the transducer. The reading "head" is put on the first letter of the input and the output is set to the empty word. At each step of the computation, given some state q, some letter a read and some transition $q' \in \delta(q, (a, w))$, the transducer appends w to the output, shifts the input head to the "right" and sets the current state to q'. The computation ends when a) the input word is completely read in which case the result is the output or b) if there is no transition compatible with the current state and the read letter. In this latter case, the computation is said to fail. Thus, compared to a finite state automaton whose computations end on True (a final state) or False (not a final state) given some input word, the transducer output words, thus defining a relation in Σ^*. Given some transducer τ, if for any word, there is at most one successful computation, the transducer outputs at most one word, the relation becomes functional. In that case, we denote the function it computes by $[\tau]$.

Definition 6 (Rational Embedding Order)**.** Given two regular languages L and L' on Σ, write $L \lesssim L'$ if:

- there is an injective function $f : L' \to L$ that is computed by a transducer τ and
- such that $|f(w)| \leq |w|$, for every $w \in L'$. Such functions (and the corresponding transducers) are said to be *decreasing* (in [22]).

The transducer τ is said to be a *witness* of $L \lesssim L'$.

We say that a transition of a transducer is *deleting* when it is of the form $a \mid 1$ for some $a \in \Sigma$. A transducer whose transitions are of the form $X \mid Y$, with $|Y| \leq |X|$, is itself decreasing. If a path corresponding to an input w passes through a deleting transition, then $|\tau(w)| < |w|$.

In the sequel we will make use of the following results that are direct consequences of Nivat's Theorem [23] (Prop. 4, §3).

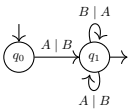
Proposition 3. Let $[\tau] : L \to L'$ be computed by a transducer τ, and let L'' be a regular language. Then the following assertions hold.

1. The restriction $\tau_{|L''} : L'' \cap L \to L'$ mapping $w \mapsto \tau(w)$ is computable by a transducer.
2. The co-restriction $\tau^{|L''} : L \to L' \cap L''$ mapping $w \mapsto \tau(w)$ if $\tau(w) \in L''$ and otherwise undefined, is computable by a transducer.
3. The function $\tau' : L \to L'$ defined by $\tau'(w) = \tau(w)$ if $w \in L''$ and otherwise undefined, is computable by a transducer.

Observe that the identity on Σ^* is computed by a transducer (made of a unique initial/final state with transitions $a \mid a$ for all $a \in \Sigma$). Then, the identity on L is obtained by Proposition 3(1,2). Thus we have that $\lesssim$ is reflexive. Also, it is well known that both transducers and injective functions can be composed. Hence, we also have that $\lesssim$ is transitive. Thus, $\lesssim$ is a preorder.

However, we do not have anti-reflexivity in general. For instance, we have

$$L_1 = A \cdot (A + B)^* \lesssim L_2 = B \cdot (A + B)^* \lesssim L_1.$$

To see this, consider the following transducer (whose initial state is indicated by an in-arrow, whereas the final one by an out-arrow): 

. This shows that $L_1 \lesssim L_2$. Swap 'A' and 'B', to see that the reversed relation also holds.

It is worth noting that there is a simple criterion to ensure a strict inequality.

Proposition 4. Suppose $L_1 \lesssim L_2$ has a witness $\tau : L_2 \to L_1$. If τ contains one (accessible and co-accessible) deleting transition, then the relation is strict.

Proof. As before, set $L_1 < L_2$ whenever $L_1 \lesssim L_2$ but not $L_2 \lesssim L_1$. Ad absurdum, suppose $L_1 \lesssim L_2 \lesssim L_1$ with a transducer $\theta : L_1 \to L_2$ and τ as above. Then $\theta \circ \tau$ (the composition of the two transducers) defines an injective function. Let w be the smallest input word from the initial state to a final state through the transition $a \mid 1$ in τ. Define the set

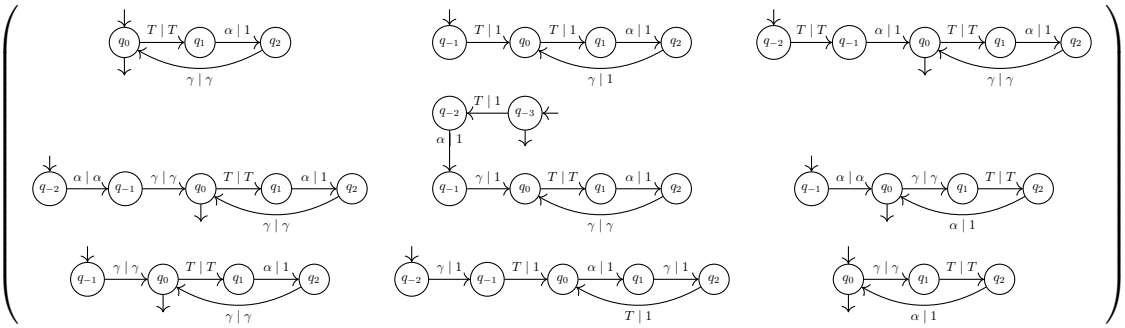
$$M^{<|w|} = \{u \in L_2 \mid |u| < |w|\}.$$

For any word u, we have $|\theta \circ \tau(u)| \leq |u|$. Thus $\theta \circ \tau(M^{<w}) \subseteq M^{<w}$. Since $M^{<w}$ is a finite set and $\theta \circ \tau$ is injective, it is actually bijective when restricted to $M^{<w}$. However, $|\theta \circ \tau(w)| \leq |\tau(w)| < w$ implies $\theta \circ \tau(w) \in M^{<w}$. By the Pigeon-hole Principle, there is one word in $M^{<w}$ that has two pre-images via $\theta \circ \tau$. Thus, $\theta \circ \tau$ cannot be injective, which yields a contradiction. □

Remark 3. From Proposition 3 it follows that if two regular languages L and L' are such that $L \subseteq L'$, then $L \lesssim L'$.

Definition 7. The rational embedding order extends to matrices by pointwise ordering: Let M and N with dimension $P \times P$, and write $M \lesssim N$ if for every $i, j \in P \times P$, we have $M_{i,j} \lesssim N_{i,j}$.

Recall the modified version of 'Follow' (Example 3). The following transducers show that all entries strictly decrease.



In the following, to compare two graphs by means of the rational embedding order, we transform graphs into matrices as follows. Given a graph G, let M'_G be the matrix of dimension $\mathcal{N}_G \times \mathcal{N}_G$ such that $(M'_G)_{i,j} = T_1^{i,j} + T_2^{i,j} + \cdots T_\ell^{i,j}$ with $T_1, \ldots, T_\ell$ the labels of the edges from i to j. In other words, we "decorate" the labels with the source and target nodes. Then, $G \lesssim G'$ whenever $M'_G \lesssim M'_{G'}$.

3.6 Stable orders on matrices

A matrix on E is said to be *finite* whenever all its entries are finite. Two matrices M and M' (of same dimension) on E are said to be disjoint if for every i, j, $M_{i,j} \cdot M'_{i,j} = \mathsf{0}$.

Definition 8. Let M be a matrix of dimension $P \times P$ and $P \subseteq G$. The extension of M to dimension $G \times G$ is the matrix $M^{\uparrow G}$ defined by:

$$(M^{\uparrow G})_{i,j} = \begin{cases} M_{i,j} & \text{if } i, j \in P \\ \mathsf{0} & \text{otherwise} \end{cases}$$

The notation $M^{\uparrow G}$ is shortened to $M^{\uparrow}$ when G is clear from the context.

Fact 1. *Let M be a matrix of dimension $P \times P$, with $P \subseteq G$. Then $(M^{\uparrow G})^* = (M^*)^{\uparrow G}$.*

Definition 9. We say that a partial order $\preceq$ on E is *stable by context* if for every $P \subseteq G$, all matrices L and R of dimension $P \times P$, and every C of dimension $G \times G$, the following assertions hold.

1. If L, R, C are finite, L being disjoint from C, R being disjoint from C and $R^* \prec L^*$, then $(R + C)^* \prec (L + C)^*$;
2. If $R \prec L$, then $R^{\uparrow G} \prec L^{\uparrow G}$.

Lemma 1. Let $\preceq$ be a partial order stable by context and consider finite matrices L, R of dimension $P \times P$ and let C be a finite matrix of dimension $G \times G$ with $P \subseteq G$. Then, $R^* \prec L^*$ implies $(R^{\uparrow} + C)^* \prec (L^{\uparrow} + C)^*$.

Proof. If $R^* \prec L^*$, then we have $(R^*)^{\uparrow} \prec (L^*)^{\uparrow}$ by Definition 9.2. By Lemma 1, it follows that $(R^{\uparrow})^* \prec (L^{\uparrow})^*$. Clearly, $R^{\uparrow}$ and $L^{\uparrow}$ are finite, and from Definition 9.1, we have $(R^{\uparrow} + C)^* \prec (L^{\uparrow} + C)^*$ □

Theorem 1. Let $\preceq$ be a partial order stable by context. Suppose that for every rule $R = \langle P, \vec{c} \rangle$ with $P = \langle P_0, \vec{\nu} \rangle$ and P'_0 the self-application of R, we have $(P'_0)^* \prec (P_0)^*$. Then the corresponding GRS is terminating.

Proof. Let $\preceq$ be a partial order on graphs and consider the corresponding order on matrices: $G \prec G'$ if and only if $M^*_G \prec M^*_{G'}$. We show that for every rule, we have $G \to G'$ implies $G' \prec G$. So let R be a graph rewriting rule and let μ be a morphism such that $G \to_{R,\mu} G'$. By the discussion in the beginning of Section 3, without loss of generality, we

can suppose that μ is actually the inclusion of pattern P_0 in G. Now, let P_0 and P_0' be respectively the basic pattern and the self-application of R. Define C to be the graph made of the nodes of G without edges in P_0. By Proposition 1, $M_G = M_{P_0}^{\uparrow} + M_C$ and $M_{G'} = M_{P_0'}^{\uparrow} + M_C$. Moreover, $M_{P_0}, M_{P_0'}$ and M_C are finite, M_{P_0} is disjoint from M_C, and $M_{P_0'}$ is disjoint from M_C. From Lemma 1 it thus follows that $M_{G'}^* = (M_{P_0'}^{\uparrow} + M_C)^* \prec (M_{P_0}^{\uparrow} + M_C)^* = M_G^*$. □

3.7 Stability of the orderings

We can now prove the two announced stability results.

Proposition 5. The multiset path ordering is stable by context.

Proof. We first verify that condition 2 of Definition 9 holds. Suppose that $R \lhd L$ with R, L of dimension $P \times P$. Then, for all $(i,j) \notin P \times P$, $R_{i,j}^{\uparrow G} = 0 \trianglelefteq 0 = L_{i,j}^{\uparrow G}$. Now, for all $k \geq |G| \geq |P|$ and for all $(i,j) \in P \times P$, we have $(R^{\uparrow G})_{i,j}^{\leq k} = R_{i,j}^{\leq k} \trianglelefteq L_{i,j}^{\leq k} = (L^{\uparrow G})_{i,j}^{\leq k}$. To verify that condition 1 also holds, let $G \times G$ be the dimension of L, R and C. Take $k \geq |G|$. On the one side we have

$$(R + C)^{*\leq k} = \sum_{(A_1,\ldots,A_\ell) \in \{R,C\}^*,\, \ell \leq k} \prod_{i \leq \ell} A_i$$

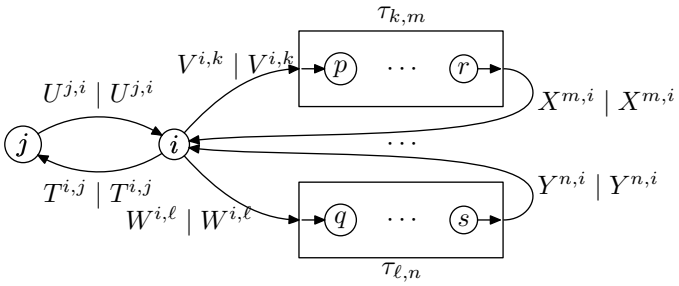
and on the other side

$$(L + C)^{*\leq k} = \sum_{(A_1,\ldots,A_\ell) \in \{R,C\}^* | \ell \leq k} \prod_{i \leq \ell} A_i\{R \leftarrow L\},$$

where $A_i\{R \leftarrow L\} = L$ if $A_i = R$, and C otherwise. As the product and the addition are (strictly) monotonic, the result follows. □

Proposition 6. The rational embedding order is stable by context.

Proof. Since we use a component-wise ordering, it is easy to verify that condition 2 of Definition 9 holds. To verify that condition 1 also holds, let $P \times P$ be the shared dimension of L, R and C of dimension $G \times G$ with $P \subseteq G$. Since $R < L$, there are decreasing transducers $\tau_{i,j} : L_{i,j} \to R_{i,j}$ with at least one of them deleting. We build the family of transducers $(\theta_{p,q})_{p,q \in G \times G}$ as follows. The family of transducers will share the major part of the construction. They only differ by their initial and terminal states. First, we make a copy of all transducers $(\tau_{i,j})_{i,j}$. Then, we add as states all the elements of G outside P. Given a non null entry $T = C_{i,j}$, we set a transition $i \xrightarrow{T^{i,j}|T^{i,j}} j$. That is the transducer "copies" the paths within C. For a an entry $T = C_{i,j}$ with $i \notin P, j \in P$, we set the transitions: $i \xrightarrow{T^{i,j}|T^{i,j}} q_n$ for all q_n initial state of the transducer $\tau_{j,n}, n \in P$. Similarly, for any entry $T = C_{i,j}$ with $i \in P, j \notin P$, we set the transitions: $r_n \xrightarrow{T^{i,j}|T^{i,j}} j$ for each terminal state r_n of the transducer $\tau_{n,i}, n \in P$. This construction can be represented as follows:



where U, T, W, X, Y range over the edge labels. Take $k, \ell \notin P$. Any path from state k to state ℓ describes a path in $C + L$ on the input side and a path in $C + R$ on the output side. Indeed, transitions within C are simply copied and the transducers $\tau_{i,j}$ transform paths in L into paths in R.

It remains to specify initial and final states of $\theta_{p,q}$ with $(p,q) \in G \times G$. Given some entry $p, q \in G$, if $p \notin P$, we set the initial state to be p. Otherwise, we introduce a new state ι which is set to be initial, and we add a transition $\iota \xrightarrow{1|1} i$ for any state i initial in $\tau_{p,r}$ for some r. If $q \notin P$, then, q is the final state. Otherwise, any state j within some $\tau_{r,q}$, $r \in P$, is final.

Consider some pair $p, q \in G$. We prove that the transducer $\theta_{p,q}$ is injective. Consider a path w in $C + L$. It can be decomposed as follows: $w = w_1\ell_1 \cdots w_k\ell_k$ where the ℓ_i's are the sub-words within L (that is the w_i's have the shape $v_i a_i$ where a_i is a transition from C to L). Consider a second word $w' = w'_1\ell'_1 \cdots w'_{k'}\ell'_{k'}$ such that the transducer $\theta_{p,q}(w) = \theta_{p,q}(w') = u$. Given the construction of $\theta_{p,q}$, consider the word $u = u_1r_1 \cdots u_kr_k$ with $r_1, \ldots, r_k$ some path within R. Indeed, only a letter within L can produce a letter within R. Consider the case where r_k is non empty. When the transducer reaches the first letter in ℓ_k, it is in a state $\tau_{k,m}$ for some m. Actually, $m = q$ since only $\tau_{k,q}$ contains a final state. Thus, the path is fixed within $\tau_{k,p}$ and then the injectivity of $\tau_{k,p}$ applies. So, $\ell'_{k'} = \ell_k$. We can go back within w_k. On this part of the word, the transitions have the shape $T^{i,j} \mid T^{i,j}$. Thus, $w_k = w'_{k'}$. We can continue this process up to the beginning of w and w'. □

3.8 Termination with node creation

Up to here, rewriting did not involve node creation. In this subsection, we will adress this issue informally, and we leave for future work the definitive formalization.

So consider a new command $\mathbf{add\_node}(\alpha)$ with α a label. This command adds a new node to the graph with label α. As in standard settings, this node is not related to others at creation time.

Actually, we see node creation as follows. Let us consider a fixed denumerable set U of nodes. For any finite graph, one may suppose without loss of generality that its nodes belong to U. More precisely, we see graphs as having U as an underlying node set, among which only finitely many are "under focus". The nodes in this finite set have a label within the set Σ_{N} and they may be related via edges to some other nodes, all the others are isolated and their label is $\perp$, a trash label. In other words, there is never some node creation, but some may be added to the focus. Notice that at each step, only finitely many nodes may be added to a graph via rewriting. In finitely many steps, starting from a finite graph, the denumerable set U is a sufficiently large container.

If we come back to our termination issue, there are a few changes. In the one hand, matrices have infinite dimension, that is, $U \times U$. But on the other hand, at each step of computation, only finitely many entries are non null. Thus, all established results still apply to this more general framework and the method is still correct.

As a consequence, if we still find some ordering on language matrices, termination will follow. However, there is one important point that must be discussed in detail. We said that the ordering on matrices did not need to be well founded to show termination. And for that, we needed the fact that all graphs met during computation have a fixed size (so that there are only finitely many of them). In the present context, this hypothesis is not reasonable in general. We thus propose two ways of overcoming this apparent limitation.

First, if the ordering on matrices is well-founded, there are no more issues. Termination will hold directly. Second, if the ordering is not well-founded, then we will need an additional ingredient. Let us suppose that the ordering $\prec$ is stable by edge contraction: that is, if G' is obtained from G by contracting some edge $e \in G$, then $M_{G'} \prec M_G$. Suppose furthermore that for any steps, $G \to G'$, we have $M_{G'} \preccurlyeq M_G$, then the system is terminating. Indeed, due to Robertson and Seymour's Theorem (see [24],), any infinite sequence $M_{G_1} \succ M_{G_2} \succ \cdots$ will contain two indices for which M_{G_j} is a (directed) minor of M_{G_k} for some $j < k$. That is G_j is obtained from G_k by finitely many edge contractions. But, since the order is stable by edge contraction, then, $M_{G_j} \prec M_{G_k}$ which leads to the contradiction. Thus, the result.

The notion of minors for the directed case may be discussed further, see for instance [25]. We leave that exploration for some other day.

4 Interpretations for Graph Rewriting Termination

Interpretation methods are well known in the context of term rewriting, see for instance Dershowitz and Jouannaud's survey on rewriting [13]. Their usefulness comes from the fact that they belong to the class of simplification orderings, i.e., orderings for which if $t \trianglelefteq u$, then $t \preceq u$. In the context of graphs, we introduce a specific notion of "interpretation", that we will still call interpretation.

Definition 10. A graph *interpretation* is a triple $\langle X, \prec, \phi\rangle$ where $\langle X, \prec\rangle$ is a partially ordered set and $\phi : \mathcal{G} \to X$ is such that given two graphs P and P' having the same set of nodes and C disjoint of P and P', if $\phi(P) \prec \phi(P')$, then $\phi(P + C) \prec \phi(P' + C)$.

An interpretation $\Omega = \langle X, \prec, \phi\rangle$ is *compatible* with a rule R if $\phi(P'_0) \prec \phi(P_0)$ where P_0 is the basic pattern of R and P'_0 its self-application. Similarly, an interpretation is compatible with a GRS if it is compatible with all of its rules.

Theorem 2. Every GRS compatible with an interpretation Ω is terminating.

The theorem being a more abstract form of Theorem 1, its proof follows exactly the same steps.

Proof. Suppose that $G \prec G'$ if and only if $\phi(G) \prec \phi(G')$. We prove that for each rule R of the GRS, $G \to G'$ implies $G' \prec G$. Indeed, suppose that $G \to_{R,\mu} G'$. Let P_0 and P_0' be respectively the basic pattern and the self-application of R. Then, there is a graph C such that $G = P_0 + C$, $G' = P_0' + C$, such that P_0 and P_0' are disjoint from C. Since $\phi(P_0') \prec \phi(P)$, we then have $\phi(G') \prec \phi(G)$. □

Example 4. The triple $\langle \mathfrak{M}, \trianglelefteq, (M_{(-)})^* \rangle$ is an interpretation for 'Follow'.

Example 5. Let us come back to the weight analysis. Define $\overline{\omega}(G) = \sum_{p \xrightarrow{e} q \in G} \omega(e)$ with $\omega(\alpha) = 0, \omega(T) = -1, \omega(\beta) = -1$. Then, $\langle \mathbb{R}, <, \overline{\omega}(-) \rangle$ is an interpretation for 'Init' and 'End'.

Example 6. Let $\langle X_1, \prec_1, \phi_1 \rangle$ be an interpretation for a set of rules $\mathcal{R}_1$, and let $\langle X_2, \prec_2, \phi_2 \rangle$ be an interpretation for a set of rules $\mathcal{R}_2$. Suppose that for every rule R in $\mathcal{R}_2$, $G \to_{R,\mu} G'$ implies $G' \preceq_1 G$ (that is without strict inequality). Then the lexicographic ordering on $X_1 \times X_2$ defined by $(x_1, x_2) \prec_{1,2} (y_1, y_2)$ if and only if $x_1 \prec_1 y_1$, or $x_1 \preceq_1 y_1$ and $x_2 \prec_2 y_2$, constitutes an interpretation $\langle X_1 \times X_2, \prec_{1,2}, \phi_1 \times \phi_2 \rangle$ for $\mathcal{R}_1 \cup \mathcal{R}_2$.

Thus, combining Example 4 and Example 5, we have a proof of the termination of the Main Example (Subsection 2.1).

Corollary 2. The GRS given in Subsection 2.1 is terminating.

Example 7. Let $\mathcal{R}$ be a terminating GRS. Then there is an interpretation that "justifies" this fact. Indeed, take $\langle \mathcal{G}, \prec, 1_{\mathcal{G}} \rangle$ with $\prec$ defined to be the transitive closure of the rewriting relation $\to$. The termination property ensures that the closure leads to an irreflexive relation. The compatibility of $\prec$ with respect to $1_{\mathcal{G}}$ is immediate.

We thus have the following corollary.

Corollary 3. A GRS is terminating if and only if it is compatible with some interpretation.

5 Conclusion

We proposed a new approach based on the theory of regular languages to decide the termination of graph rewriting systems, which does not account for node additions but settles the uniform termination problem for these GRS. We think that there is room to reconsider some old results of this theory under the new light. In particular, we think of profinite topology [26], is a powerful tool that could give us some insight on the underlying structure of the orders. In the two cases, we can extend the orders to take into account orders on the edge labels.

As the next natural step, we intend to explore more systematically graph rewriting with node creations and that take into account node labels. Moreover, in the experiments mentioned in the introduction about natural language processing, in principle, these two orders should still be sufficient to ensure termination. However, we need to implement these new results for an extensive and complete evaluation.

Acknowledgment. The authors are thankful to the anonymous reviewers for their careful reading of our manuscript and for the constructive remarks and useful suggestions, that were of great help when revising this manuscript.

References

[1] Noam Chomsky. *Syntactic Structures*. The Hague: Mouton, 1957.

[2] Bruno Guillaume and Guy Perrier. Dependency parsing with graph rewriting. In *Proceedings of the 14th International Conference on Parsing Technologies, IWPT 2015, Bilbao, Spain, July 5-7, 2015*, pages 30–39, 2015.

[3] Sylvain Kahane and François Lareau. Word ordering as a graph rewriting process. In *Formal Grammar - 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers. FG 2016, Bozen, Italy, August 2016, Proceedings*, volume 9804, pages 216–239. Springer, 2016.

[4] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. *Application of Graph Rewriting to Natural Language Processing*. Logic, Linguistic and Computer Science. Wiley, 2018.

[5] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.

[6] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2012.

[7] Mizuhito Ogawa. A note on algebraic structure of tree decomposition of graphs. In *The First Asian Workshop on Programming Languages and Systems, APLAS 2000, National University of Singapore, Singapore, December 18-20, 2000, Proceedings*, pages 223–229, 2000.

[8] Yves Lafont. Interaction nets. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990*, pages 95–108, 1990.

[9] Maribel Fernández, Hélène Kirchner, and Bruno Pinaud. Strategic port graph rewriting: an interactive modelling framework. *Mathematical Structures in Computer Science*, 29(5):615–662, 2019.

[10] Nachum Dershowitz and Jean-Pierre Jouannaud. Drags: A compositional algebraic framework for graph rewriting. *Theor. Comput. Sci.*, 777:204–231, 2019.

[11] Géraud Sénizergues. Some undecidable termination problems for semi-Thue systems. *Theoretical Computer Science*, 142:257–276, 1995.

[12] Guillaume Bonfante and Bruno Guillaume. Non-simplifying graph rewriting termination. In *Proceedings 7th International Workshop on Computing with Terms and Graphs, TERMGRAPH 2013, Rome, Italy, 23th March 2013*, pages 4–16, 2013.

[13] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. 1990.

[14] Nachum Dershowitz and Jean-Pierre Jouannaud. Graph path orderings. In *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, pages 307–325, 2018.

[15] Detlef Plump. Simplification orders for term graph rewriting. In Igor Prívara and Peter Ružička, editors, *Mathematical Foundations of Computer Science 1997*, pages 458–467, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[16] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2):195–220, Mar 2008.

[17] Nachum Dershowitz. A note on simplification orderings. *Information Processing Letters*, pages 212–215, 1979.

[18] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[19] Jan Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1):1 – 53, 2003.

[20] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8), 1979.

[21] Gerard Huet and Derek C. Oppen. Equations and rewrite rules: a survey. In *In formal language Theory: perspective and open problems*. Academic Press, 1980.

[22] Jeannine Leguy. Transductions rationnelles décroissantes. *RAIRO*. Informatique théorique, 15(2):141–148, 1981.

[23] Maurice Nivat. Transducteurs des langages de Chomsky. *Ann. Inst. Fourier, Grenoble*, 18:339–455, 1968.

[24] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

[25] Shiva Kintali and Qiuyi Zhang. Forbidden directed minors and Kelly-width. *Theoretical Computer Science*, 662:40 – 47, 2017.

[26] Jean-Eric Pin. Profinite methods in automata theory. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 31–50. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.