HAL open science

On graph rewriting systems termination through language theory

Guillaume Bonfante, Miguel Couceiro

▶ To cite this version:

Guillaume Bonfante, Miguel Couceiro. On graph rewriting systems termination through language theory. 2024. hal-03629573v2

HAL Id: hal-03629573

https://inria.hal.science/hal-03629573v2

Preprint submitted on 12 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

ON GRAPH REWRITING SYSTEMS: TERMINATION THROUGH LANGUAGE THEORY

GUILLAUME BONFANTE AND MIGUEL COUCEIRO

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. Thus algorithms become mandatory 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.

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 dealing with sharing or references, graphs behave much better. Graph Rewriting may be used for several purposes such as for example parsing procedures as described by Guillaume and Perrier [2] or the word ordering modeling by Kahane and Lareau [3] or corpus transformation by Gerdes, Guillaume and Kahane [4]. The first named author with Guillaume and Perrier designed a graph rewriting model called grew [5] that is adapted to natural language processing. This is an open-source project available at `http:\\grew.fr`.

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, several systems are presented in [2], 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.

There is an extensive literature on the termination of *term-rewriting* including the surveys [6, 7] and a famous competition `http://termination-portal.org/wiki/Termination_Portal`. Accordingly, we will refer to term rewriting as a background. However, extensions from term to graph rewriting is not easy at all: this is due to the fact that termination methods for term rewriting are based on the underlying algebraic structure of terms, a property that does not hold in general for graphs. Thus, there is not a simple shift from terms to graphs for this kind of problems.

Date: Submitted March 12, 2024.

Also, the definition of graph rewriting itself is problematic. Contrarily to term rewriting for which the definition is essentially fixed (again) 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 [8]. To make use of algebraic potential, some authors make some, potentially weak, hypotheses on graph structures, see for instance the main contribution by Courcelle and Engelfriet [9] 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 [10].

Another line of research follows from the seminal work by Lafont [11] 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 [12], 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 [13]. 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 graph rewriting.

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. But, at the same time, termination is a mandatory property for computations, thus our main concern.

Furthermore, 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. To sum up, none of the approaches applies, at least in a straightforward way.

Some hypotheses in Natural Language Processing (NLP) transformations are rather unusual compared to standard computations. Firstly, actual rewriting systems are quite often non confluent. For example, if one computes the semantics of a sentence out of its grammatical analysis, it is quite common that there are multiple solutions. For instance, the well know phrasal construction "He saw a man with a telescope" has two clear readings. Secondly, we may suppose that there is a 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 helps largely the implementation of graph rewriting since we do not need to cope with graph isomorphism, which 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 (size) increasing systems constituted by a well known problem, and it was shown to be undecidable by Sénizergues in [14].

This paper proposes a novel approach for termination of graph rewriting. In a former paper [15], we proposed a solution based on label weights, see also [16]. Here, the focus is on the description (and the ordering) of paths within graphs. In fact, paths in a graph can be structured 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 [6]. Dershowitz and Jouannaud, in the context of drag rewriting, consider a similar notion of path ordering called GPO (see [17]). 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 [18] also defines path orderings for term graphs, but those behave like sets of terms.

We introduce a second ordering on languages. It is based on a transducer technique that can be seen as a uniform way of transforming paths of the left hand side of a rule to its right hand side. It is called the *rational embedding ordering*. To our knowledge, this is a new order, and we could not relate it (at least directly) to previous propositions. Both orderings will involve matrices, and orderings on matrices. Nonetheless, there is no relationship with matrix interpretations as defined by Endrullis, Waldmann and Zantema [19].

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 4 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 4.5) and the rational embedding order (Subsection 4.6). We also propose the notion of stability by context (Subsection 4.7) and show that both orderings are stable under this condition (Subsection 4.8). In Section 5 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 introduce a notion of graph interpretation, and show that GRSs that are terminating are exactly those that are compatible with such interpretations.

Acknowledgment. The authors thank the anonymous referees of an earlier version of this work for their kind and helpful comments. We are grateful for their careful and thorough reviews.

2. Background in Set Theory and Languages

In this section we recall some general definitions and notation that are used in the sequel. Let us begin with relations and orders.

2.1. **Orders and multisets.** 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 will also refer to 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 $\cdots \prec x_2 \prec x_1$.

A *multi-set* on a set X is a function $m : X \to \mathbb{N}$. For $x \in X$, the value $m(x)$ is called the *multiplicity* of x. We denote by $\mathcal{M}(X)$ the set of all multisets on X. Given a multiset $s \in \mathcal{M}(X)$, its *support* is the set $\underline{s} = \{x \in X \mid s(x) \neq 0\}$. Given two multisets s and t in $\mathcal{M}(X)$, $s + t$ denotes the multiset in $\mathcal{M}(X)$ defined by $(s + t)(x) = s(x) + t(x)$.

Any order $\preceq$ on X induces an order on $\mathcal{M}(X)$ as follows.

Definition 1 (Multiset induced order)**.** The *multiset induced order* of $\preceq$, denoted $\preceq_{mul}$, is the smallest partial order on multisets such that for any $s, t \in \mathcal{M}(X)$, the two following statements hold:

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

We write $s \prec_{mul} t$ when $s \preceq_{mul} t$ and $s \neq t$. When the context is clear, we will drop the subscript "mul" and simply write $s \preceq t$ instead of $s \preceq_{mul} t$.

2.2. **Regular languages, automata and transducers.** Throughout this section about Language Theory, we refer the reader to the book by Sakarovitch [20] for justifications or proofs of the statements we mention.

Given an alphabet Σ, the set of words (finite sequences of elements of Σ) is denoted by Σ^*. The size of a word w, that is, its length, is denoted $|w|$. 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. Given two alphabets Σ and Γ, a word homomorphism is a function $\phi : \Sigma^* \to \Gamma^*$ that preserves concatenation, that is, $\phi(w \cdot w') = \phi(w) \cdot \phi(w')$, and preverses the empty word, that is, $\phi(1) = 1$. Note that ϕ is uniquely determined by its values on Σ.

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

$$L + L' = \{w \mid w \in L \text{ or } w \in L'\}.$$

The empty language is denoted by 0. Note that $\langle \mathcal{P}(\Sigma^*), +, 0 \rangle$ 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 $L \cdot L' = \{w \cdot w' \mid w \in L \text{ and } w' \in L'\}$. In this way, $\langle \mathcal{P}(\Sigma^*), 1, \cdot \rangle$ is also a monoid. Given the distributivity of $\cdot$ over $+$, the 5-tuple $\langle \mathcal{P}(\Sigma^*), +, 0, \cdot, 1 \rangle$ constitutes a semiring.

Moreover, for $L \subseteq \Sigma^*$, let $L^* = 1 + L + L^2 + \cdots$, where $L^{i+1} = L \cdot L^i$, for every integer $i \geq 0$.

Given an alphabet Σ, the set E of *regular expressions* over Σ is defined recursively by the following grammar:

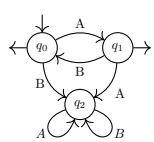
$$\mathsf{E} ::= a \in \Sigma \mid \mathsf{E} + \mathsf{E} \mid \mathsf{E} \cdot \mathsf{E} \mid \mathsf{E}^*.$$

To any regular expression corresponds a language built according to the grammar given above. Languages that correspond to regular expressions form the so called *rational* or *regular languages*.

An automaton on some alphabet Σ is a 4-tuple $\langle Q, q_0, F, \delta \rangle$ where Q is a set of states, q_0 is a particular state (called the *initial state*), F is the set *final* or *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is its transition relation. Note that $\delta \subseteq Q \times \Sigma \times Q$ can be thought of as a function $\delta \colon Q \times \Sigma \to Q$, usually called its *transition function*. The relation δ can be extended to $Q \times \Sigma^* \times Q$ via the equations:

$$\begin{aligned} (q, 1, q) &\in \delta^* \\ (q, w \cdot a, q') &\in \delta^* \text{ if } \quad (q, w, q'') \in \delta^* \text{and } (q'', a, q') \in \delta \end{aligned}$$

An automaton $\mathtt{A} = \langle Q, q_0, F, \delta \rangle$ defines a language $L(\mathtt{A}) = \{w \in \Sigma^* \mid (q_0, w, q_f) \in \delta^* \text{ with } q_f \in F\}$. Kleene's famous Theorem states that languages recognized by automata are rational languages. Automata are commonly represented by directed and labelled graphs. The input arrow indicates the initial state while the output arrows indicate the final states. The language associated to the following automaton is $\mathcal{L} = A(B \cdot A)^* + (A \cdot B)^*$:



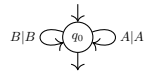
A state of an automaton is said to be accessible whenever there is a path from the initial state to it. It is co-accessible when there is a path from it to a final state. Removing a state that is not accessible or co-accessible does not change the recognized language. We thus suppose that any state in an automaton is accessible and co-accessible.

Given two alphabets Γ and Δ, a *finite state transducer* (or simply *transducer*) is a 4-tuple $\tau = \langle Q, q_0, F, \delta \rangle$ with Q a finite set, $q_0 \in Q$ its initial state, $F \subseteq Q$ its final states and $\delta \subseteq Q \times \Gamma^* \times \Delta^* \times Q$ its transition function. It can be naturally extended as above:

$$\begin{aligned} (q, 1, 1, q) &\in \delta^* \\ (q, v \cdot v', w \cdot w', q') &\in \delta^* \text{if } (q, v, w, q'') \in \delta^* \text{ and } (q'', v', w', q') \in \delta \end{aligned}$$

The tranducer τ defines a relation on $\Gamma^* \times \Delta^*$: $[\tau](w, w')$ holds for some words $w \in \Gamma^*, w' \in \Delta^*$ if and only if there is some state $q_f \in F$ such that $(q_0, w, w', q_f) \in \delta^*$. Such a relation will be referred to as *rational*.

Clearly, the identity relation Id_{Σ^*} in Σ^* is a rational relation. Take the transducer to have a unique state q which is both initial and final. And set $\delta = \{(q, a, a, q) \mid a \in \Sigma\}$. That definition is drawn as follows for $\Sigma = \{A, B\}$:



Rational relations have some nice closure properties. For instance, they are closed by union, intersection and composition. When the rational relation is actually a function, we mention it as a rational function. In that case, we use the functional notation: $[\tau] : \Gamma^* \to \Delta^*$.

There is a well known characterization of rational relations by Nivat [21] (Prop. 4, §3). A relation $\mathcal{S} \subseteq \Gamma^* \times \Delta^*$ is rational whenever there is a rational language $L \in \Sigma^*$ and two word homomorphisms $\phi : \Sigma^* \to \Gamma^*$ and $\psi : \Sigma^* \to \Delta^*$ such that $\mathcal{S} = \{(w, w') \mid \exists t \in L \wedge w = \phi(t) \wedge \psi(t) = w'\}$. The 3-tuple (ϕ, L, ψ) is known as a bimorphism.

Rational languages are closed by word homomorphism, that is if $L \subseteq \Sigma^*$ is a rational language and $\phi : \Sigma^* \to \Delta^*$ is a word homomorphism, then $\phi(L) = \{\phi(w) \mid w \in L\}$ is a rational language (within Δ^*). They are also closed by inverse homomorphism: $\phi^{-1}(L) = \{w \in \Sigma^* \mid \phi(w) \in L\}$ is a rational language.

Thus, the domain of definition and the image of rational relations are actually rational languages. Indeed, take $[\tau] = \{(w, w') \mid \exists t \in L \wedge w = \phi(t) \wedge w' = \psi(t)\}$. Then the image $Im([\tau]) = \psi(L)$. The same for the domain. In other words, a rational relation can be seen as a relation between rational languages.

So, for a rational function, restricting it to its domain of definition and its image, it becomes legitimate to write it as: $[\tau] : \phi(L) \to \psi(L)$.

Proposition 1. 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.

Proof. For 1), let (ϕ, L_0, ψ) be the bimorphism computing $[\tau]$. Then, $(\phi, L_0 \cap \phi^{-1}(L''), \psi)$ computes $[\tau_{|L''}]$. For 2), $(\phi, L_0 \cap \psi^{-1}(L''), \psi)$ computes the function $[\tau]^{|L''}]$. 3) is a rereading of 1) with an enlarged domain. □

Corollary 1. Given a rational language L, the identity function $Id_L : L \to L$ is a rational function.

Proof. Indeed, $Id_L = (Id_{\Sigma^*})^{|L}_{|L}$. □

A transducer τ is decreasing whenever for all transition (q, w_1, w_2, q'), we have $|w_2| \leq |w_1|$. In that case, it is clear that the function $[\tau]$ computed by such a transducer satisfies $|[\tau](w)| \leq |w|$ for all w on which $[\tau]$ is defined.

We say that a transition of a transducer τ is *deleting* when it is of the form $(q, a, 1, q')$ for some $a \in \Sigma$, $q, q' \in Q$. If a path corresponding to an input w passes through a deleting transition of a decreasing transducer τ, then $|[\tau](w)| < |w|$.

3. Graph rewriting

There are several definitions of graph rewriting in the literature. The one we consider here corresponds to GREW, our graph rewriting tool. Extensions of our results on termination to other frameworks is beyond the scope of the paper. The content of this section may be found in [5] where we discuss the definition in more details. The definition was actually driven by the applications of graph rewriting we implemented in the context of Natural Language Processing such as semantics translations, grammatical parsing, corpus construction and correction and so on.

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 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 drawing $\underset{a}{\clubsuit} \xrightarrow{\mathrm{A}} \underset{b}{\heartsuit}$, a, b denote nodes, $\clubsuit, \heartsuit$ are their respective node labels and A is the edge label (here between a and b).

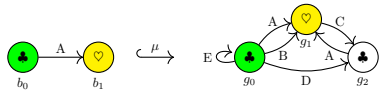
The set of graphs on node labels Σ_{N} and edge labels Σ_{E} is denoted by $\mathcal{G}_{\Sigma_{\mathrm{N}}, \Sigma_{\mathrm{E}}}$ or $\mathcal{G}$ in short. Two graphs G and G' are said to share their nodes when $\mathcal{N}_G = \mathcal{N}_{G'}$. Given two graphs G and G' such that $\mathcal{N}_G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

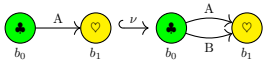
A graph morphism μ between a source graph G and a target graph H is a function $\mu : \mathcal{N}_G \to \mathcal{N}_H$ that preserves edges and labelings, that is, for all $m \xrightarrow{e} n \in G$, $\mu(m) \xrightarrow{e} \mu(n) \in H$ holds, and for any node $n \in G$: $\ell_G(n) = \ell_H(\mu(n))$.

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 $\mu : P \to G$, we define $\mu(P)$ to be the sub-graph of G made of the nodes $\{\mu(n) \mid n \in \mathcal{N}_P\}$, of the edges $\{\mu(m) \xrightarrow{e} \mu(n) \mid m \xrightarrow{e} n \in P\}$ and node labels $\mu(n) \mapsto \ell_P(n)$.

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*[1]. 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 a basic pattern morphism $\mu : P_0 \to G$ for which there is no morphism $\xi_i : N_i \to G$ such that $\mu = \xi_i \circ \nu_i$.

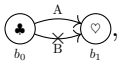
Example 1. Consider the basic pattern morphism $\mu : P_0 \to G$ (colors define the mapping):

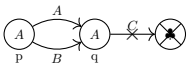


The pattern $P = \langle P_0, [\nu] \rangle$ with ν defined by 

prevents the application of the pattern. Indeed, $\xi = [b_0 \mapsto g_0, b_1 \mapsto g_1]$ is such that $\xi \circ \nu = \mu$.

[1]A negative condition is not supposed to be an isomorphism. Otherwise, there is no pattern matching. Indeed, $\mu = (\mu \circ \nu^{-1}) \circ \nu$ which prevents the application as defined below.

Without loss of generality, we can suppose that for a negative condition, the injection morphism is the identity on the basic pattern. Then, we can represent a negative condition by crossing nodes and edges which *are not* within the basic pattern. For instance, the negative condition above is represented by 

, which we hope to be self-explanatory. The negative pattern 

is an other way to prevent the application of μ on G.

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

Definition 2 (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 finally $\texttt{add_edge}(p, e, q)$ 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 involved by the command belong to P_0.

Definition 3 (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.*, $G \cdot_\mu \vec{c} = (\cdots((G \cdot_\mu c_1) \cdot_\mu c_2) \cdot_\mu \cdots c_n)$.

Definition 4 (Rule)**.** 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 $G \to_{R,\mu} G'$.

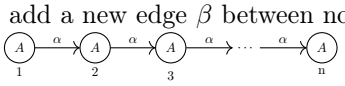
Definition 5 (Graph Rewriting System)**.** A graph rewriting system (GRS) $\mathbf{R}$ is a (finite) set of rules.

Given a GRS $\mathbf{R}$, we define the relation $G \to G'$ whenever there is a rule R and a pattern morphism μ such that $G \to_{R,\mu} G'$.

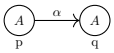
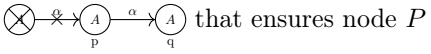
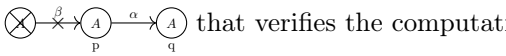
A *derivation* is a (possibly infinite) sequence $G_0 \to G_1 \to \cdots$. A *normal form* is a graph G for which there is no G' such that $G \to G'$. A computation operates by successive applications of rules until no rule can be applied.

As stated, a computation may be nondeterministic, it may happen that given some graph G, there are two rules (possibly the same) R_1, R_2 and two morphisms μ_1, μ_2 (also possibly the same) with $G \to_{R_1,\mu_1} G'$ and $G \to_{R_2,\mu_2} G''$. Both choices are explored. To sum up, a rewriting system $\mathbf{R}$ computes the relation $[\mathbf{R}]$ in $\mathcal{G}$ defined by: $(G, G') \in [\mathbf{R}]$ if and only if there is a (finite) derivation $G \to \cdots \to G'$ and G' is a normal form.

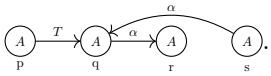
3.1. **The main example.** Let $\Sigma_{\mathrm{N}} = \{A\}$ and $\Sigma_{\mathrm{E}} = \{\alpha, \beta, T\}$. For the discussion, we suppose that T is a working label, that is not present in the initial graphs.

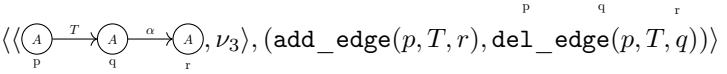
Suppose we want to add a new edge β between node n and node 1 each time we find a maximal chain: 

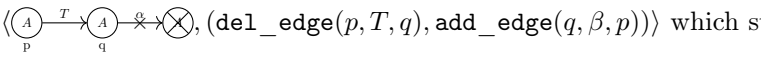
within a graph G.

To perform that task, let us consider the basic pattern $P_{init} =$  together with its two negative conditions $\nu_1 =$  that ensures node P is at the begining of the chain and $\nu_2 =$  that verifies the computation has not been already performed. We consider three rules:

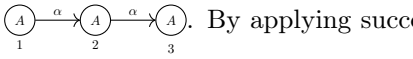
Init: $\langle\langle P_{init}, (\nu_1, \nu_2)\rangle, (\texttt{add_edge}(p, T, q))\rangle$ which fires the transitive closure.

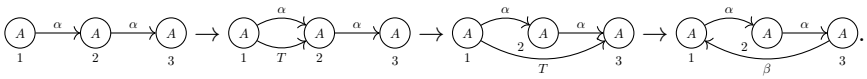
For the next rule, we need a negative condition, ν_3:  .

Follow: $\langle\langle$  $, \nu_3\rangle, (\texttt{add_edge}(p, T, r), \texttt{del_edge}(p, T, q))\rangle$ that follows the chain.

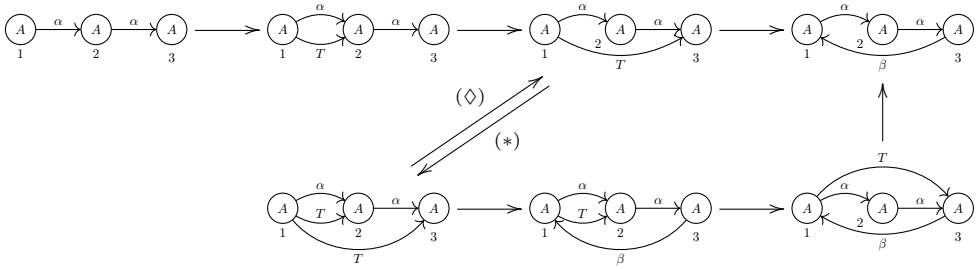
End: $\langle$  $, (\texttt{del_edge}(p, T, q), \texttt{add_edge}(q, \beta, p))\rangle$ which stops the processus.

Actually, to prevent all pathological cases (*e.g.*, when the edge β is misplaced, when two chains are crossing, and so on), we should 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 successively "Init", "Follow" and "End", G rewrites as follows:



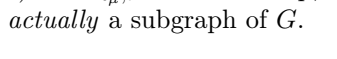
Actually, even if there are more than one derivation starting from G, all lead to the same normal form. This is known as confluence:

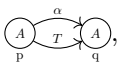
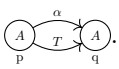
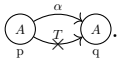
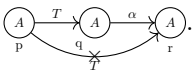


Two remarks about these graph rewriting steps. First, we see that there is an infinite derivation due to the steps $(*)$ and $(\Diamond)$. Second, $(\Diamond)$ is "badly" formed. Indeed, it is supposed to transport the T edge between 1 and 2 to 1 and 3. However, such an arc is already present, and consequently, the rule application just deletes the edge. To avoid such behaviors, we will enforce our assumptions about rules. This is the topic of the next section where we introduce the notion of uniform rules and full matching.

3.2. **Three technical facts about Graph Rewriting.** It is well known that the main issue with graph rewriting definitions is the way the context is connected to the pattern image and its rewritten part. We shall tackle this issue with Proposition 2.

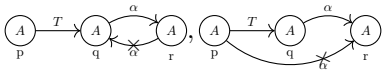
Self-application. Let $R = \langle P, \vec{c} \rangle$ be the rule made of a pattern $P = \langle P_0, \vec{\nu} \rangle$ and a sequence of commands $\vec{c}$. The identity morphism $1_{P_0} : P_0 \to P_0$ is then a pattern matching[2], and thus we can apply rule R on P_0 itself, that is, $P_0 \to_{R,1_{P_0}} P_0' = P_0 \cdot_{1_{P_0}} \vec{c}$. We call this latter graph the *self-application of* R.

Rule node renaming. To avoid heavy notations, 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 $\vec{\nu}$ and $\vec{c}$). 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 of Example 2. 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 adding a negative condition to "Init":  . The same issue applies to "Follow", and we introduce accordingly a new negative condition (N):  .

A *uniform* rule is one for which any commands operates (that is, modifies the graph) for each rule application. Since this is not the scope of the paper, we refer the reader to [5] for a precise definition of uniformity. Roughly speaking, any deleted edge must be present in the basic pattern and any inserted edge must be absent from the image. The technical difficulty rises from partial applications of commands (deleting twice the same edge for instance). Nevertheless, as justified in [5]§7, any rule can be replaced by a finite set of uniform rules (using negative conditions as above) that operate identically. Thus, from now on, we always suppose that rules are uniform.

We will also need an extra hypothesis that we call full-matching. We have to ensure that any pattern matching image is a subgraph full isomorphism. That is, in the image, there are no edges outside those in the pattern for the matched nodes. For instance, for the rule "Follow", we need extra negative conditions:



and all subsequent combinations. Whenever rules are uniform and pattern matching is full, the following Proposition holds (justified in [5]§7).

Proposition 2. 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

[2]We implicitly use the fact that the negative conditions are not isomorphic here.

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 $\mathcal{E}_C \cap \mathcal{E}_{P'_0} = \emptyset$.

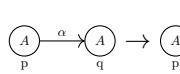
In what follows, we consider that basic patterns of rules "Init", "Follow" and "End" have been completed with the adequate negative conditions so that they are uniform, full-matching rules. Observe that the negative condition (N) on the "Follow" rule prevents its application as in ($\Diamond$). Then, the system has no longer such infinite derivations. The two paths to the normal form **HERE!** are both finite.

4. Termination of Graph Rewriting Systems

A GRS **R** is said to be *terminating* if the relation $\to$ is well-founded, that is there is no infinite derivations $G_1 \to G_2 \to \cdots$.Since there is no node creation (nor node deletion) in our notion of rewriting, any derivation starting from a graph G will produce 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).

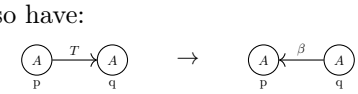
Remark 1. Suppose that we are given a strict partial order $\succ$, not necessarily well founded. If $G \to G'$ implies $G \succ G'$ for all graphs G and G', then the system is terminating. Indeed, suppose it is not the case, let $G_1 \to G_2 \to \cdots$ be an infinite reduction sequence. Since there 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 $G_i = G_j$. 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 [22] in the context of term rewriting. For instance, it is possible to embed the rewriting relation within the order on real numbers rather than the one on natural numbers to prove termination.

4.1. **Weight functions.** Let us try to prove the termination of our main example (see Subsection 3.1). The notion of weight functions was introduced in [23]. We refer the reader to this publication for proofs and formal definitions. The idea of a weight function is already presented by Bruggink, König, and Zantema in the context of double pushout transformations in [16]. That said, for rules such as '$Init$' and 'End' this is 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: $\omega(G) = \sum_{x \xrightarrow{e} y \in G} \omega(e)$. Let us set $\omega(\alpha) = 0, \omega(\beta) = -2$ and $\omega(T) = -1$ and we write $G \succ G'$ if and only if $\omega(G) > \omega(G')$. Observe that for $P_{Init} =$  $= P'_{Init}$, we have $\omega(P_{Init}) = 0 > -1 = \omega(P'_{Init})$, and thus $P_{Init} \succ P'_{Init}$.

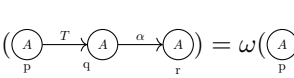
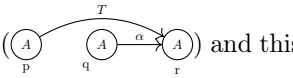
More generally, by Proposition 2, for any rule application $G \to_{Init,\mu} G'$, we have $G = P_{Init} \blacktriangleleft C$ with $\mathcal{E}_G = \mathcal{E}_{P_{Init}} \cup \mathcal{E}_C$. Again, by Proposition 2, we know that $\mathcal{E}_{P_{Init}} \cap \mathcal{E}_C = \emptyset$, and thus $\omega(G) = \omega(P_{Init}) + \omega(C)$. For the same reason, $\omega(G') = \omega(P'_{Init}) + \omega(C)$ so that $\omega(G) = 0 + \omega(C) > -1 + \omega(C) = \omega(G')$, which entails $G \succ G'$.

Observe that there is no infinite sequence $G \to_{Init,\mu_1} G_1 \to_{Init,\mu_2} \cdots$. In fact, for the "End" rule, we also have:



$$\omega(.) = -1 \qquad > \qquad \omega(.) = -2$$

Again, the weight decreases whatever the context, and thus, by Remark 1, the system $\{Init, End\}$ is terminating.

But how to handle rule "Follow"? No weights as above can work. Indeed, whatever the choice of ω, applying ω on the pattern of "Follow" and its self application lead to $\omega($  $) = \omega($  $)$ and this equality holds for any application of "Follow". It is now time to introduce some new ingredients.

4.2. **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 graph 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 in G' compared to G. 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 expressions with multiplicity.* See, *e.g.*, Sakarovitch's book [20] for an introduction and justifications of the upcoming constructions.

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

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* $s \cdot t$ is given 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[3], 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 [24]:

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

Thus, given the constructions above, 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]Id est there is a notion of length $|-| : \Sigma^* \to \mathbb{N}$ such that $|1| = 0$ and for any $w \neq 1$, $|w| > 0$ and for all w, w', $|w \cdot w'| = |w| + |w'|$ holds.

4.4. **The language matrix.** Let Σ_{E} be an edge label set, and 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 two matrices M and N sharing dimension $P \times P$, let the matrix $M + N$ of dimension $P \times P$ defined by: $(M + N)_{i\in P, j\in P} = M_{i,j} + N_{i,j}$ and $M \times N$ also of dimension $P \times P$ with $(M \times N)_{i,j} = \sum_{k\in P} M_{i,k}N_{k,j}$.

Given a graph G, we define the matrix M_G of dimension $\mathcal{N}_G \times \mathcal{N}_G$ as follows:

$$M_{Gi,j} = T_1 + \cdots + T_\ell,$$

where $T_1, \ldots, T_\ell$ are the 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 $M^* = 1 + M + M^2 + \cdots$. Each entry of M^* is actually an $\mathbb{N}$-regular expression (see, *e.g.*, Sakarovitch [20], Ch. III, §4). The (infinite) sum is correctly defined if $M = M_G$ for some graph G. Indeed, since for all i, j, we have the equality $M_{i,j} = T_1 + \cdots + T_\ell$ for some $T_1, \ldots, T_\ell \in \Sigma_{\mathrm{E}}$, whether $\ell = 0$ (in which case $M_{i,j} = 0$) or not, $1 \notin \underline{M_{i,j}}$.

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".

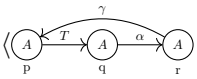
Recall the "Follow" rule and consider the corresponding basic pattern L and its self-application R. Their respective matrices are:

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

Let us consider an ordering on languages such that $L \subseteq L'$ implies $L < L'$. Then, observe that $(M_R)_{13} \not< (M_L)_{13}$. This matrix deals with edges/transitions. In order to consider paths, we need to compute M^*_L and M^*_R that are given by:

$$M^*_L = \begin{pmatrix} 1 & T & T\,\alpha \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix} \quad M^*_R = \begin{pmatrix} 1 & 0 & T \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix}.$$

Note that any word in M^*_R's entries is a sub-word of the corresponding entry in M^*_L.

Example 3. Consider now a variation of "Follow" that is 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 $M^*_{L'}$.

4.5. **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 [6]). However, in the present context of graph rewriting (to be compared with Dershowitz and Jouannaud's [17] or with Plump's [18]), the definition is a bit less direct. 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 [25] or Huet and Oppen [26]. We still denote it by $\trianglelefteq$.

Proposition 3. 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 $t \neq \mathsf{0}$ (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 1). 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 1:

- 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 $t \neq \mathsf{0}$, it contains at least one word v_0 such that $t = t_0 + v_0$. From $r \triangleleft s$, we have

$$r \cdot v_0 = \sum_{v \in \underline{r}} r(v) v \cdot v_0 \triangleleft \sum_{v \in \underline{s}} s(v) v \cdot v_0 = s \cdot v_0.$$

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 6 (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 $M_{i,j}^{\leq k} \trianglelefteq M'^{\leq k}_{i,j}$.

Corollary 2. The addition and the multiplication are monotonic with respect to the matrix multiset-path order.

Proof. It follows from Proposition 3 since addition and product of matrices are defined as addition and product of their entries. □

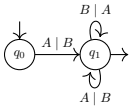
4.6. **The Rational Embedding Order.** We introduce a second order on regular languages.

Definition 7 (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 decreasing transducer τ. The transducer τ is said to be a *witness* of $L \lesssim L'$.

Recall that the identity on L is computed by a decreasing transducer (see Corollary 1). Thus $\lesssim$ 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. Let w be a word whose computation crosses the deleting transition. Then, following the definition, it is clear that $|[\tau](w)| < |w|$.

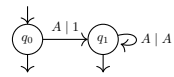
Now, as before, set $L_1 < L_2$ whenever $L_1 \lesssim L_2$ but not $L_2 \lesssim L_1$. Ad absurdum, suppose $L_1 \lesssim L_2 \lesssim L_1$ with a transducer $\theta : L_1 \to L_2$ and τ as above. Then $\theta \circ \tau$ (the composition of the two transducers) defines an injective function. Let w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

In the sequel, we write $L < L'$ if τ has a decreasing with one (accessible and co-accessible) deleting transition.

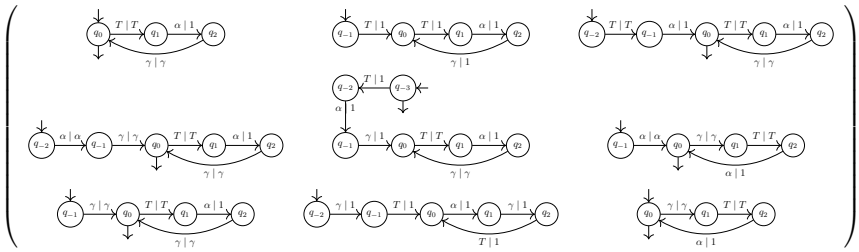
Remark 2. As the proof shows it, injectivity is necessary to ensure that decreasing automata with one deleting transition lead to a strict order. Here is a counter example showing that otherwise we would have $A^* < A^*$. Consider the transducer:



Remark 3. From Proposition 1 it follows that if two regular languages L and L' are such that $L \subseteq L'$, then $L \lesssim L'$.

Definition 8. 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 $(M'_G)_{i,j} = T_1^{i,j} + \cdots T_\ell^{i,j}$ 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, we define $G < G'$ whenever $\overline{M}_G = (M'_G)^* < (M'_{G'})^* = \overline{M}_{G'}$.

Suppose that $G < G'$, that is $\overline{M}_G = (M'_G)^* < (M'_{G'})^* = \overline{M}_{G'}$. Consider a path $p_1 \xrightarrow{T_1} p_2 \xrightarrow{T_2} \cdots p_k$ in G. It is mapped bijectively to the word $T_1^{p_1,p_2} \cdots T_n^{p_n,p_{n+1}}$. In the sequel, we do not make a distinction between the two forms of paths.

Remark 4. Consider a pair r, s of nodes in G. Then, the component $(\overline{M}_G)_{r,s}$ is the set of path from r to s in G. Thus, any such path begins necessarily with a first letter $T^{r,m}$ for some vertex m in G, $T \in \Sigma_{\mathrm{E}}$ and ends with a letter of the shape $U^{n,s}$ for some $n \in G$, $U \in \Sigma_{\mathrm{E}}$.

Suppose that we have some rule involving some pattern $P = L$ and its self-application $P \cdot \vec{c} = R$. Suppose $\overline{R} < \overline{L}$. Then, there are decreasing transducers such that any word $T^{r_1,s_1} \cdots T^{r_k,s_k}$ in L read by τ_{r_1,s_k} is transformed into a word $U^{r'_1,s'_1} \cdots U^{r'_{k'},s'_{k'}}$ in R. Consider an application of the rule within some graph G. We say that a letter $T^{r,s}$ is in C (the context) if $r \notin P$ or $s \notin P$. It is in P otherwise (that is both $r, s \in P$). A letter cannot be both in C and P. If a letter in P corresponds to a letter/edge in L, then we say it is in L. In the same way, it can be in R and possibly in both.

4.7. **Stable orders on matrices.** An element $e \in \mathsf{E}$ is finite if the set

$$\{w \in \Sigma^* \mid e(w) \neq 0\}$$

is finite. 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, $M_{i,j} \cdot M'_{i,j} = 0$.

Definition 9. 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:

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

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

Proposition 5. Let M be a matrix of dimension $P \times P$, with $P \subseteq G$. Then $(M^{\uparrow G})^* = (M^*)^{\uparrow G}$.

Proof. By induction on $k \in \mathbb{N}$, we prove that $(M^{\uparrow G})^k = (M^k)^{\uparrow G}$. The result follows. □

Definition 10 (Context stability)**.** We say that a partial order $\preceq$ (with respect to 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 $(R^{\uparrow G} + C)^* \prec (L^{\uparrow G} + C)^*$.

Proof. If $R^* \prec L^*$, then we have $(R^*)^{\uparrow G} \prec (L^*)^{\uparrow G}$ by Definition 10.2. By Lemma 5, it follows that $(R^{\uparrow G})^* \prec (L^{\uparrow G})^*$. Clearly, $R^{\uparrow G}$ and $L^{\uparrow G}$ are finite, and from Definition 10.1, we have $(R^{\uparrow G} + C)^* \prec (L^{\uparrow G} + C)^*$ □

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 $M^*_{P_0'} \prec M^*_{P_0}$. 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 $G' \prec G$. As justified at the begining of the section, this is sufficient to prove termination.

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 2, $M_G = M^{\uparrow}_{P_0} + M_C$ and $M_{G'} = M^{\uparrow}_{P_0'} + M_C$. 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 $M^*_{G'} = (M^{\uparrow}_{P_0'} + M_C)^* \prec (M^{\uparrow}_{P_0} + M_C)^* = M^*_G$. □

4.8. **Stability of the orderings.** We can now prove the two stability results.

Proposition 6. The multiset path ordering is stable by context.

Proof. We first verify that condition 2 of Definition 10 holds. Suppose that $R \lhd L$ with R, L of dimension $P \times P$. Then, for all $(i, j) \notin P \times P$, $R^{\uparrow G}_{i,j} = 0 \unlhd 0 = L^{\uparrow G}_{i,j}$. Now, for all $k \geq |G| \geq |P|$ and for all $(i, j) \in P \times P$, we have $(R^{\uparrow G})^{\leq k}_{i,j} = R^{\leq k}_{i,j} \unlhd L^{\leq k}_{i,j} = (L^{\uparrow G})^{\leq k}_{i,j}$. The inequality is strict for at least one pair $(i, j) \in P \times P$. Thus a strict inequality.

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

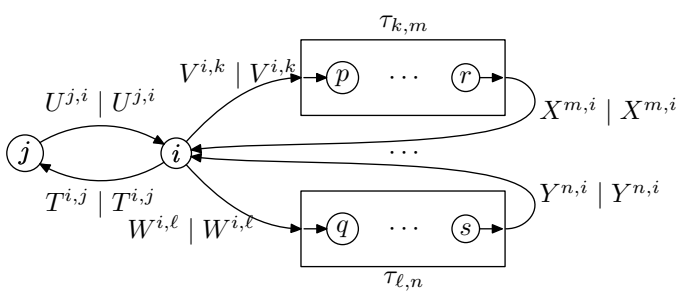
$$(L+C)^{*\leq k} = \sum_{(A_1,\ldots,A_\ell)\in\{R,C\}^*|\ell\leq k} \prod_{i\leq\ell} A_i\{R\leftarrow L\},$$

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 7. 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 10 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 $\tau_{p,q} : L_{p,q}\to R_{p,q}$ with at least one of them deleting for any $p,q\in P$. We extend the family to any $p,q\in G$. For that, 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 $(\tau_{p,q})_{(p,q)\in P\times P}$. Then, we add as states all the nodes of G outside P. Given a non null entry in the context $T = C_{p,q}$, that is $p,q\in C$, we set a transition $p \xrightarrow{T^{p,q}|T^{p,q}} q$. The transducer "copies" the paths within C. For a an entry $V = C_{p,q}$ with $p\notin P, q\in P$, we set a transitions: $p \xrightarrow{V^{p,q}|V^{p,q}} i_{p,n}$ for any $n\in P$ with $i_{p,n}$ the initial state of the transducer $\tau_{p,n}$. Similarly, for any entry $X = C_{p,q}$ with $p\in P, q\notin P$, we set the transitions: $f_{n,p} \xrightarrow{X^{p,q}|X^{p,q}} q$ for any terminal state $f_{n,p}$ of the transducer $\tau_{n,p}, n\in P$. This construction can be represented as follows:



where U, T, W, X, Y range over the edge labels.

It remains to specify initial and final states of the $\theta_{p,q}$ 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 itself. Otherwise, we introduce a new state ι_p which is set to be initial, and we add a transition $\iota_p \xrightarrow{1|1} i_{p,n}$ for any $n\in P$ with $i_{p,n}$ the initial state in $\tau_{p,n}$. If $q\notin P$, then, q is the final state. Otherwise, any terminal state $f_{n,q}$ within some $\tau_{n,q}$, $n\in P$ is set as final.

By construction, the transducers $\theta_{p,q}$'s are decreasing with at least one deleting transition (one within $\tau_{r,s}$ for some $r,s\in P$). We need to show that they are functional and injective.

Let us begin with some observations about the behavior of the transducers $\theta_{p,q}$.

Remark 5. We have seen in Remark 4 that a succesful path in $\tau_{r,s}$ necessarily begins with a letter $T^{r,m}$ and ends with $U^{n,s}$. Given their definition, the property holds for the $\theta_{p,q}$'s.

Let us check that $\theta_{p,q}$ is functional. That is there at most one succeeding path within the automaton given some input word $w = w_1\ell_1 w_2 \cdots w_k\ell_k$. We let $w_1 = T_1^{r_1,s_1} \cdots T_{i_1}^{r_{i_1},s_{i_1}}$ and $\ell_1 = S_1^{u_1,v_1} \cdots S_{j_1}^{u_{j_1},v_{j_1}}$.

Let us start with a non empty w_1, that is $T_1^{r_1,s_1}$ is in C, and let us follow the computation on this word within the transducer. Up to $T_{i_1-1}^{r_{i_1-1},s_{i_1-1}}$, the transducer is deterministic, thus the unicity of the path up to $T_{i_1-1}^{r_{i_1-1},s_{i_1-1}}$. For the last letter, $T_{i_1}^{r_{i_1},s_{i_1}}$, there is a choice. One can reach any state $i_{s_{i_1},m}$ that is initial within $\tau_{s_{i_1},m}$. Two cases are under consideration. Either w_2 is empty (that is $k = 1$) or not.

After the last step, the computation will continue within $\tau_{s_1,m}$ reading letters within P. Since w_2 is empty, it will end within $\tau_{s_1,m}$ in a terminal state. But let the last letter of r_1 be $S^{n,s}$. Since it is a terminal state in $\tau_{s_1,m}$, it implies that $m = s$. Thus there is at most one succesful choice for m. Next, observe that the run in $\tau_{s_1,s}$ goes from one initial state to a terminal one. Functionality of $\tau_{s_1,m}$ applies. Thus w has only one image through $\theta_{p,q}$.

Otherwise, let $T^{r,s}$ be the first letter in w_2. The computation "leaves" $\tau_{s_1,m}$. But it must do it via a final state $f_{s_1,m}$ in $\tau_{s_1,m}$. The only possibility is $r = m$. Again, the choice vanishes. As above, inside $\tau_{s_1,m}$, the computation starts at some initial state, ends at a final state, thus is unique. In other words, there is a unique way to reach the second letter of w_2. This process can be followed up to the end of the input word.

Now suppose we begin within P. At the very begining, again, there is a choice. One fires one of the transitions $\iota_p \xrightarrow{1|1} i_{p,m}$ for some $m \in P$. Then, either w_2 is empty and the computation stays within the transducer. Again, the last letter determines which transducer we go through and again, we keep in mind that $\tau_{p,m}$ is functional. Otherwise, as above, it is the first letter of w_2 that fixes the value m. And we are back to a state in C as above.

Consider some pair $p, q \in G \times G$. 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 $\theta_{p,q}$, since letters in C are copied (and distinct from $\tau_{i,j}$'s output letters), the word u has the shape

$$u = w_1 r_1 \cdots w_k r_k = w'_1 r'_1 \cdots w_{k'} r'_{k'},$$

where $r_1, \ldots, r_k, r'_1, \ldots, r'_{k'}$ are some paths within R. Thus, w_1 and w'_1 have a common prefix, say w''_1. If w_1 is strictly shorter than w'_1, then its last letter is of the form $T^{r,s}$ for $s \in P$, which is not compatible with w'_1 (whose only letter of that sort is the last one). Thus $w_1 = w'_1$, and $r_1 = r'_1$, since the $\tau_{r,s}$'s are injective. The process continues up to k. □

5. 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 [6]. 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 11. 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 from 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 is a generalization of Theorem 1, but 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 $\mathcal{R}_1 \cup \mathcal{R}_2$.

Combining Example 4, Example 5 and Example 6, we have a proof of the termination of the Main Example (Subsection 3.1).

Corollary 3. The GRS given in Subsection 3.1 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 4. A GRS is terminating if and only if it is compatible with some interpretation.

6. 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 GRSs. We think that there is room to reconsider some old results of this theory under the new light.

Now, profinite topology [27] is a powerful tool that could give us some insights on the underlying structure of the orders. Indeed, if we are back to the multiset order,

we see that we look at the structure of languages with a stratification along the length of words. For both orders (multiset and rational embeddings) one perspective is to extend them to take into account partial orders on the edge labels and partial orders on the node labels. As the next natural step, we intend to explore more systematically graph rewriting with node creation. In particular, we intend to address and tackle the following issues.

Firstly, matrices should have infinite dimension. But at the same time, since graphs are finite, only finitely many entries are non null. Here, the technical step is not immediate, but it seems feasible.

Secondly, we saw that the ordering on matrices does 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). This hypothesis may not hold in general.

Thirdly, suppose the ordering is not well-founded. We need an extra-ingredient. Let us assume 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 $M_{G'} \prec M_G$. Furthermore, note that if $G \to G'$, then we have $M_{G'} \prec M_G$, and the system is terminating. Indeed, due to Robertson and Seymour's Theorem (see [28]), 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. Since the order is stable under edge contraction, $M_{G_j} \prec M_{G_k}$, which is a contradiction. The notion of minors for the directed case may be discussed further, see for instance [29].

Finally, concerning the practical aspect of the algorithm we designed, given the experiments mentioned in the introduction about natural language processing, in principle, these two orders should still be sufficient to ensure termination. For multiset-path ordering, the decision procedure is almost contained within the definition. This is not the case of the rational embedding for which we have to choose the transducer.

References

[1] N. 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] Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. Improving Surface-syntactic Universal Dependencies (SUD): surface-syntactic relations and deep syntactic features. In *TLT 2019 - 18th International Workshop on Treebanks and Linguistic Theories*, Paris, France, August 2019.

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

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

[7] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] H. J. Sander Bruggink, Barbara König, and Hans Zantema. Termination analysis for graph transformation systems. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2014.

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] Guillaume Bonfante and Bruno Guillaume. Non-size increasing graph rewriting for natural language processing. *Math. Struct. Comput. Sci.*, 28(8):1451–1484, 2018.

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

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

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

[27] 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.

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

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

Université de Lorraine, CNRS, LORIA
Email address: guillaume.bonfante@univ-lorraine.fr

Université de Lorraine, CNRS, LORIA and INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
Email address: miguel.couceiro@loria.fr