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. 2022. hal-03629573

HAL Id: hal-03629573
https://inria.hal.science/hal-03629573
Preprint submitted on 4 Apr 2022

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.
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, 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.

There is a large litterature 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.
The second difficulty comes from the fact that the definition of graph rewriting itself is problematic. Contrarily to term rewriting for which the definition is essentially fixed by the algebraic structure of terms (again), 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, possibly 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. 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 know phrasal construction “He saw a man with a telescope” with two clear readings. To sum up, none of the approaches applies, at least in a straightforward way.

Some hypotheses in Natural Language Processing transformations are rather unusual compared to standard computations. For instance, we may suppose 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 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. 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 [16]). 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 [17] 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 which 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 and it seems quite new, we could not relate it (at least directly) to previous propositions.

Both 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 [18].

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 the huge work they did and for their kind and helpful comments. We are really grateful for this careful and precise reading.

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 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, $x_1 \succ x_2 \succ \cdots$.

A multi-set relative to some set $X$ is a function $m : X \rightarrow \mathbb{N}$. $m(x)$ is called the multiplicity of $x$. Let $\mathcal{M}(X)$ be the set of all multisets on $X$. Given a multiset $s \in \mathcal{M}(X)$, we define its support to be the set $s = \{ x \in X \mid s(x) \neq 0 \}$. Finally, given two multisets $s$ and $t$ in $\mathcal{M}(X)$, $s + t$ denote the multiset in $\mathcal{M}(X)$ defined by $(s + t)(x) = s(x) + t(x)$.

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

Definition 1 (Multiset induced order). The multiset induced order of $\prec$, next denoted $\preceq_{\text{mul}}$, is the smallest partial order on multisets such that for any multisets $s$ and $t$, the two following statements hold:

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

We write $s \preceq_{\text{mul}} t$ when $s \preceq_{\text{mul}} t$ and $s \neq t$. Finally, when the context is clear, we will drop the subscript "mul" and $s \prec t$ stands for $s \preceq_{\text{mul}} t$.

2.2. (Regular) languages, automata and transducers. For this section about Language Theory, we refer the reader to the book of Sakarovitch [19] for justifications or proofs of the statements we mention. We took in large parts his notations.

Given an alphabet $\Sigma$, the set of words (finite sequences of elements of $\Sigma$) is denoted by $\Sigma^*$. 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 $1\Sigma$ or, when clear from the context, simply by $1$. Note that $(\Sigma^*, 1, \cdot )$ constitutes a monoid. Given two alphabets $\Sigma$ and $\Gamma$, a word homomorphism is a function $\phi : \Sigma^* \rightarrow \Gamma^*$ respecting concatenation. That is: $\phi(w \cdot w') = \phi(w) \cdot \phi(w')$ and $\phi(1) = 1$. Actually, $\phi$ is uniquely described by its values on $\Sigma$.

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

Finally, given some language \( L \subseteq \Sigma^* \), let \( L^* = 1 + L + L^2 + \cdots \).

Given some alphabet \( \Sigma \), Regular Expressions are built on the following grammar:

\[
E ::= a \in \Sigma \mid (E + E) \mid (E \cdot E) \mid (E^*).
\]

To any regular expression corresponds a language built according to the definition above. Languages that correspond to regular expressions form the rational languages.

An automaton on some alphabet \( \Sigma \) is a 4-tuple \( \langle Q, q_0, F, \delta \rangle \) with \( \delta \subseteq Q \times \Sigma \times Q \). The set \( Q \) is the set of states, \( q_0 \) being the initial state and \( F \subseteq Q \) being the set of final states. The relation \( \delta \) can be extended to \( Q \times \Sigma^* \times Q \) via the equations:

\[
(q, 1, q) \in \delta^* \\
(q, w \cdot a, q') \in \delta^* \text{ if } (q, w, q'') \in \delta^* \text{ and } (q'', a, q') \in \delta
\]

An automaton \( A = \langle Q, q_0, F, \delta \rangle \) defines a language \( L(A) = \{w \in \Sigma^* \mid (q_0, w, q_f) \in \delta^* \text{ with } q_f \in F\} \). It is Kleene’s famous Theorem that languages recognized by automata are rational languages. Automata will be presented by drawings. The input arrow indicate the initial state while the output arrows indicate the final states. The language associated to the following automaton is \( L = A(B \cdot A)^* + (A \cdot B)^* \):

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. Thus, all along, we suppose that any state in an automaton is accessible and co-accessible.

Given two alphabets \( \Gamma \) and \( \Delta \), a finite state transducer, or transducer in short, 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 extends as above:

\[
(q, 1, 1, q) \in \delta^* \\
(q, v \cdot v', w \cdot w', q') \in \delta^* \text{ if } (q, v, w, q'') \in \delta^* \text{ and } (q'', v', w', q') \in \delta
\]

The transducer \( \tau \) 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 qualified as rational.

Clearly, the identity relation \( I_{\Delta^*} \) in \( \Sigma^* \) 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 \( \Sigma = \{A, B\} \):

\[
\begin{array}{c}
A \\
B \\
A \\
\end{array} \\
\begin{array}{c}
B \quad B \\
\end{array}
\]
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. It is due to Nivat [20] (Prop. 4, §3). A relation \(S \subseteq \Gamma^* \times \Delta^*\) 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 \(S = \{(w, w') \mid \exists t \in L \land w = \phi(t) \land \psi(t) = w'\}\). The 3-tuple \((\phi, L, \psi)\) 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 \(\Delta^*\)). 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 \([\tau] = \{(w, w') \mid \exists t \in L \land w = \phi(t) \land w' = \psi(t)\}\). Then the image \(\text{Im}(\tau) = \psi(L)\). 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 \(\tau\), 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 \([\tau]^\dagger_{L''} : L' \cap L'' \to [\tau]'(w)\) 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 \((\phi, L_0, \psi)\) 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 \([\tau]^\dagger_{L''}\). 3) is a rereading of 1) with an enlarged domain. \(\square\)

**Corollary 1.** Given a rational language \(L\), the identity function \(\text{Id}_L : L \to L\) is a rational function.

**Proof.** Indeed, \(\text{Id}_L = (\text{Id}_{\Sigma^*})|_{L}'\). \(\square\)

A transducer \(\tau\) 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 verifies \(|[\tau](w)| \leq |w|\) for all \(w\) on which \([\tau]\) is defined.

We say that a transition of a transducer \(\tau\) 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 \(\tau\), then \(|[\tau](w)| < |w|\).

### 3. Graph rewriting

There are several definitions of graph rewriting in the literateature. The one we propose 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 $\Sigma_N$ of node labels, a (finite) set $\Sigma_E$ of edge labels and we define graphs accordingly. A graph is a triple $G = (N, E, \ell)$ with $E \subseteq N \times \Sigma_E \times N$ and $\ell : N \to \Sigma_N$ 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 $e \in \Sigma_E$. 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 $N_G$, $E_G$ and $\ell_G$ 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 $m \in N_G$ and $m \xrightarrow{e} n \in E_G$ when the context is clear.

Furthermore, in a drawing $\bigcirc \xrightarrow{\downarrow} \bigcirc$, $a, b$ denote nodes, $\bigcirc, \bigtriangledown$ are their respective node labels and $A$ is the edge label (here between $a$ and $b$).

The set of graphs on node labels $\Sigma_N$ and edge labels $\Sigma_E$ is denoted by $\mathcal{G}\Sigma_N \Sigma_E$ or $\mathcal{G}$ in short. Two graphs $G$ and $G'$ are said to share their nodes when $N_G = N_{G'}$. Given two graphs $G$ and $G'$ such that $N_G \subseteq N_{G'}$, set $G \shoveleft G'$ to be the graph $\langle N_{G'}, E_{G'} \cup E_G, \ell \rangle$ with $\ell(n) = \ell_G(n)$ if $n \in N_G$ and $\ell(n) = \ell_{G'}(n)$, otherwise.

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

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 $\{\mu(n) \mid n \in N_P\}$, of the edges $\{\mu(m) \xrightarrow{e} \mu(n) \mid m \xrightarrow{e} n \in P\}$ and node labels $\mu(n) \mapsto \ell(n)$.

A pattern is a pair $P = \langle P_0, \nu \rangle$ 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 $P = \langle P_0, \nu \rangle$ and a graph $G$, a pattern morphism is a basic pattern morphism $\mu : P_0 \to G$ for which there is no morphism $\xi$ such that $\mu = \xi \circ \nu$. This is the case of a negative condition.

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

![Diagram of a pattern morphism](image)

The pattern $P = \langle P_0, [\nu] \rangle$ with $\nu$ defined by $\bigcirc \xrightarrow{a} \bigtriangledown$ prevents the application of the morphism above. Indeed, $\xi = [b_0 \mapsto g_0, b_1 \mapsto g_1]$ is such that $\xi \circ \nu = \mu$.

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 $\bigcirc \xrightarrow{a} \bigcirc$ that...
we hope is self-explanatory. The negative pattern \( \circ \xrightarrow{\alpha} \bullet \xrightarrow{\beta} \circ \xrightarrow{\gamma} \bullet \) is an other way to prevent the application of \( \mu \) on \( G \).

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

**Definition 2** (The command language). There are three basic commands: label\((p, \alpha)\) for node renaming, del\(\_edge(p, e, q)\) for edge deletion and finally add\(\_edge(p, e, q)\) for edge creation. In these basic commands, \( p \) and \( q \) are nodes, \( \alpha \) is some node label and \( e \) is some edge label. A pattern \( (P_0, \bar{\nu}) \) is compatible with a command whenever all nodes involved by the command belong to \( P_0 \).

**Definition 3** (Operational semantics). Given a pattern \( P = (P_0, \bar{\nu}) \) compatible with some command \( c \), and some pattern matching \( \mu : P \rightarrow G \) where \( G \) is the graph on which the transformation is applied, we have the following possible cases: \( c = \text{label}(p, \alpha) \) turns the label of \( \mu(p) \) into \( \alpha \), \( c = \text{del\_edge}(p, e, q) \) removes \( \mu(p) \rightarrow \alpha \rightarrow \mu(q) \) if it exists, otherwise does nothing, and \( c = \text{add\_edge}(p, e, q) \) adds the edge \( \mu(p) \rightarrow e \rightarrow \mu(q) \) if it does not exist, otherwise does nothing. The graph obtained after such an application is denoted by \( G'\mu c \). Given a sequence of commands \( \bar{c} = (c_1, \ldots, c_n) \), let \( G'\mu \bar{c} \) be the resulting graph, i.e., \( G'\mu \bar{c} = (\cdots ((G'\mu c_1)\mu c_2)\mu \cdots c_n) \).

**Definition 4** (Rule). A rule is a pair \( R = (P, \bar{\nu}) \) made of a pattern and a (compatible) sequence of commands.

Such a rule \( R \) applies to a graph \( G \) via a pattern morphism \( \mu : P \rightarrow G \). Let \( G' = G \cdot \mu \bar{c} \), then we write \( G \rightarrow_{R, \mu} G' \).

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

Given a GRS \( R \), we define the relation \( G \rightarrow G' \) whenever there is a rule \( R \) and a pattern morphism \( \mu \) such that \( G \rightarrow_{R, \mu} G' \).

A derivation is a (possibly infinite) sequence \( G_0 \rightarrow G_1 \rightarrow \cdots \). A normal form is a graph \( G \) for which there is no \( G' \) such that \( G \rightarrow G' \). 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 \( \mu_1, \mu_2 \) (also possibly the same) with \( G \rightarrow_{R_1, \mu_1} G'' \) and \( G \rightarrow_{R_2, \mu_2} G''' \). Both choices are explored. To sum up, a rewriting system \( R \) computes the relation \([R] \) in \( \mathcal{G} \) defined by: \( (G, G') \in [R] \) if and only if there is a (finite) derivation \( G \rightarrow \cdots \rightarrow G' \) and \( G' \) is a normal form.

### 3.1. The main example.

Let \( \Sigma_N = \{ A \} \) and \( \Sigma_E = \{ \alpha, \beta, T \} \). 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 \( \beta \) between node \( n \) and node \( 1 \) each time we find a maximal chain: \( 1 \xrightarrow{\alpha} 2 \xrightarrow{\alpha} 3 \xrightarrow{\alpha} 4 \) within a graph \( G \).

To perform that task, let us consider the basic pattern \( P_{init} = \circ \xrightarrow{\gamma} \bullet \) together with its two negative conditions \( \nu_1 = \bullet \xrightarrow{\alpha} \circ \xrightarrow{\gamma} \bullet \) that ensures node \( P \) is at the begining of the chain and \( \nu_2 = \bullet \xrightarrow{\beta} \circ \xrightarrow{\gamma} \bullet \) that verifies the computation has not been already performed. We consider three rules:
Init: $\langle \langle \text{Init}, (\nu_1, \nu_2) \rangle, (\text{add\_edge}(p, T, q)) \rangle$ which fires the transitive closure.

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

the chain.

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

procedure.

Actually, to prevent all pathological cases (e.g., when the edge $\beta$ 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 $G = \langle 1^\alpha 2^\alpha 3 \rangle$. By applying succesively

'Init', 'Follow' and 'End', $G$ rewrites as: $\langle 1^\alpha 2^\alpha 3 \rightarrow 1^\alpha 2^\alpha 3 \rightarrow 1^\alpha 2^\alpha 3 \rightarrow \rangle$

as expected by the requirements.

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 $(\ast)$ and $(\circ)$. Second, $(\circ)$ is "baddly" 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.

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 related to the

pattern image and its rewritten part. We shall tackle this issue with Proposition 2.

Self-application. Let $R = \langle P, \bar{c} \rangle$ be the rule made of a pattern $P = \langle P_0, \bar{\nu} \rangle$ and a

sequence of commands $\bar{c}$. The identity morphism $1_{P_0} : P_0 \rightarrow P_0$ is then a pattern

matching and thus we can apply rule $R$ on $P_0$ itself, that is, $P_0 \rightarrow_{R,1_{P_0}} P_0 = P_0^{1_{P_0}} \bar{c}$.

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 $R = \langle P, \bar{c} \rangle$, a graph $G$ and a pattern morphism $\mu : P \rightarrow G$.

Let $P = \langle P_0, \bar{\nu} \rangle$. We define $R_\mu$ to be the rule obtained by renaming nodes $p$ in $P_0$

to $\mu(p)$ (and their references within $\bar{\nu}$ and $\bar{c}$). For instance, the rule 'Follow' can be

rewritten as $\text{Follow}_\mu = \langle 1^\alpha 2^\alpha 3, (\text{add\_edge}(1, T, 3), \text{del\_edge}(1, T, 2)) \rangle$

\footnote{We implicitly use the fact that the negative conditions are not isomorphic here.}
where $\mu$ denotes the pattern morphism used to apply 'Follow' in the derivation of Example 2. Observe that: (i) the basic pattern of $R_\mu$ is actually $\mu(P_0)$, which is a subgraph of $G$, (ii) $i : \mu(P_0) \rightarrow G$ mapping $n \mapsto n$ is a pattern matching, and (iii) applying rule $R_\mu$ with $i$ is equivalent to applying rule $R$ with $\mu$. In other words, $G \rightarrow R_\mu G'$ if (and only if) $G \rightarrow R \mu, i G'$. 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: $A_p \alpha T^\beta$, and the result is the graph itself: $A_p \alpha T^\beta$. 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': $A_p \alpha T^\beta \times$. 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 [5] for a precise definition of uniformity. Nevertheless, from the definition of uniformity, we will take the benefit of two facts that are justified in [5]§7.

First, 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.

Second, the following property holds for uniform rules.

**Proposition 2.** Suppose that $G \rightarrow R_\mu G'$ 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 $G = P_0 \triangleright C$ and $G' = P'_0 \triangleright C$ with $P'_0$ being the self-application of the rule. Moreover, $\mathcal{E}_C \cap \mathcal{E}_{P_0} = \emptyset$ and $\mathcal{E}_C \cap \mathcal{E}_{P'_0} = \emptyset$.

The uniform versions of the rule "Init", "Follow" and "End" are:

**Init:** $\langle \langle P_\text{init}, (\nu_1, \nu_2, \nu_3) \rangle, (\text{add}_\text{edge}(p, T, q)) \rangle$ with $\nu_3 = \alpha T^\beta \times$.

**Follow:** $\langle \langle A_p \alpha T^\beta, \alpha T^\beta \times, A_p \alpha T^\beta \beta \times \rangle, (\text{add}_\text{edge}(p, T, r), \text{del}_\text{edge}(p, T, q)) \rangle$ which follows the chain.

**End:** $\langle \langle A_p \alpha T^\beta, (\text{del}_\text{edge}(p, T, q), \text{add}_\text{edge}(q, \beta, p)) \rangle, (\text{del}_\text{edge}(p, T, q), \text{add}_\text{edge}(q, \beta, p)) \rangle$ which stops the procedure.

Observe that the negative condition on the "Follow" rule prevents its application as in (♦). Then, the system has no longer such infinite derivations. The two paths to the normal form are both finite.

4. Termination of Graph Rewriting Systems

A GRS $R$ is said to be terminating if the relation $\rightarrow$ is well-founded, that is there is no infinite derivations $G_1 \rightarrow G_2 \rightarrow \cdots$.

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 1. Suppose that we are given a strict partial order \( \succ \), not necessarily well founded. If \( G \rightarrow G' \) implies \( G \succ G' \) for all graphs \( G \) and \( G' \), then the system is terminating. Indeed, suppose it is not the case, let \( G_1 \rightarrow G_2 \rightarrow \cdots \) 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 \( G_i \rightarrow \cdots \rightarrow G_j \) 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 [21] 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 [22]. We refer the reader to this publication for proofs and formal definitions. That said, rules such as 'Init' and 'End' are "simple": we put a weight on edge labels \( \omega : \Sigma_E \rightarrow \mathbb{R} \) and we say that the weight of a graph is the sum of the weights of its edge labels:

\[
\omega(G) = \sum_{e \in G} \omega(e).
\]

Let us set \( \omega(\alpha) = 0, \omega(\beta) = -2 \) and \( \omega(T) = -1 \) and we say \( G \succ G' \) if and only if \( \omega(G) > \omega(G') \).

Then, observe that for \( P_{\text{Init}} = \begin{array}{c}
\circ \\
\circ
\end{array} \rightarrow \begin{array}{c}
\circ \\
\circ
\end{array} = P'_{\text{Init}}, \) we have \( \omega(P_{\text{Init}}) = 0 > -1 = \omega(P'_{\text{Init}}) \), thus \( P_{\text{Init}} \succ P'_{\text{Init}} \). More generally, due to Proposition 2 for any rule application \( G \rightarrow_{\text{Init},\mu} G' \), we have \( G = P_{\text{Init}} \upharpoonright C \) with \( E_C = E_{P_{\text{Init}}} \cup E_C \). Again, by Proposition 2 we know that \( E_{P_{\text{Init}}} \cap E_C = \emptyset \), thus \( \omega(G) = \omega(P_{\text{Init}}) + \omega(C) \). For the same reason, \( \omega(G') = \omega(P'_{\text{Init}}) + \omega(C) \) so that \( \omega(G) = 0 + \omega(C) > -1 + \omega(C) = \omega(G') \).

To conclude, \( G \succ G' \).

To sum up, coming back to Remark 1 there is no infinite sequence \( G \rightarrow_{\text{Init},\mu_1 \cdots \mu_n} \).

Actually, for the "End" rule, we have the same observation:

\[
\begin{array}{c}
\circ \\
\circ
\end{array} \rightarrow \begin{array}{c}
\circ \\
\circ
\end{array} \quad \omega(\cdot) = -1 > 0
\]

Again, the weight decrease whatever the context. Thus, the system \( \{\text{Init}, \text{End}\} \) is terminating.

But how do we handle rule 'Follow'? No weights as above can work. Indeed, whatever the choice of \( \omega \), applying \( \omega \) on the pattern of "Follow" and its self application lead to \( \omega(\begin{array}{c}
\circ \\
\circ
\end{array}) = \omega(\begin{array}{c}
\circ \\
\circ
\end{array}) \) 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 \( G \rightarrow G' \) 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 \( \Sigma_E \) 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 \( \Sigma_E \). 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: \( G \rightarrow G' \). Any word to state \( r \) that goes through the transitions \( p \rightarrow q \rightarrow r \) can be mapped to a shorter one in \( G' \) via the transition \( p \rightarrow r \). 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 expression with multiplicity. See, e.g., Sakarovitch’s book [19] for
an introduction and justifications of the upcoming constructions. We introduce here
the basic ideas.

4.3. Formal series. A formal series on $\Sigma$ (with coefficients in $\mathbb{N}$) is a (total) function
$s : \Sigma^* \to \mathbb{N}$. That is, it is a multiset on $\Sigma^*$. Given $n \in \mathbb{N}$, let $n$ be the series defined
by $n(w) = 0$, if $w \neq 1$, and $n(1) = n$, where 1 denotes the empty word. The empty
language is $\emptyset$, the language made of the empty word is $\mathbb{1}$. 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 $s \cdot t(w) = \sum_{u,v = w} s(u)t(v)$. The
star operation is defined by $s^* = 1 + s + s^2 + \cdots$. The monoid $\Sigma^*$ 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 $\Sigma$ is built upon the grammar [23]:

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

Thus, given the constructions mentioned in the previous paragraph, any $\mathbb{N}$-
rational expression $E \in E$ denotes some formal series. To each $\mathbb{N}$-rational expression
the correct ordering of the upcoming constructions. We introduce here
that is, the structure of matrices over $\mathbb{N}$.

4.4. The language matrix. Let us suppose given an edge label set $\Sigma_E$. Let $E$
denote the $\mathbb{N}$-rational expressions over $\Sigma_E$. A matrix $M$ of dimension $P \times P$
for some (finite) set $P$ is an array $(M_{i,j})_{i,j \in P}$ 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: $(M + N)_{i,j} = M_{i,j} + N_{i,j}$
and $M \times N$ also of dimension $P \times P$ with $(M \times N)_{i,j} = \sum_{k P} M_{i,k} N_{k,j}$.

Given a graph $G$, we define the matrix $M_G$ of dimension $\mathcal{N}_G \times \mathcal{N}_G$ as follows:
$M_G(i,j) = T_1 + \cdots + T_\ell$ with $T_1, \ldots, T_\ell \in \Sigma_E$, whether $\ell = 0$ (in which case $M_{i,j} = 0$)
or not, $1 \notin M_{i,j}$.

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

\footnote{If $|\cdot| : \Sigma^* \to \mathbb{N}$ such that $|\mathbb{1}| = 0$ and for any $w \neq 1$, $|w| > 0$
and for all $w, w'$, $|w \cdot w'| = |w| + |w'|$ holds.}
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 \( M_L^p \) and \( M_R^p \) that are given by:

\[
M_L^p = \begin{pmatrix}
1 & T & T \cdot \alpha \\
0 & 1 & \alpha \\
0 & 0 & 1
\end{pmatrix} \quad M_R^p = \begin{pmatrix}
1 & 0 & T \\
0 & 1 & \alpha \\
0 & 0 & 1
\end{pmatrix}.
\]

Note that any word within \( M_R^p \)'s entries is a sub-word of the corresponding entry in \( M_L^p \).

**Example 3.** Consider now a variation of ‘Follow’ that is made of the pattern \( \langle \alpha \beta \gamma \delta \rangle \) and commands \((\text{add\_edge}(p, T, r), \text{del\_edge}(p, T, q))\). By setting \( L' \) as its pattern and \( R' \) as its self-application, we get the following matrices:

\[
M_L' = \begin{pmatrix}
(T\alpha\gamma)^* & T(\alpha\gamma T)^* & T\alpha(\gamma T\alpha)^* \\
\alpha\gamma(T\alpha\gamma)^* & (\alpha\gamma T)^* & \alpha(\gamma T\alpha)^* \\
\gamma(T\alpha\gamma)^* & \gamma(T\alpha T)^* & (\gamma T\alpha)^*
\end{pmatrix} \quad M_R' = \begin{pmatrix}
(T\gamma)^* & 0 & T(\gamma T)^* \\
\alpha\gamma(T\gamma)^* & 1 & \alpha(\gamma T\alpha)^* \\
\gamma(T\gamma)^* & 0 & (\gamma T\alpha)^*
\end{pmatrix}.
\]

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 [15] or with Plum’s [17]), the definition is a bit less direct. Here, we do not consider an order on letters as it is done for terms.

Let \( \preceq \) be the word embedding on \( \Sigma^* \), that is, the smallest partial order such that \( 1 \preceq w \), and if \( u \preceq v \), then \( u \cdot w \preceq v \cdot w \) and \( w \cdot u \preceq w \cdot v \), for all \( u, v, w \in \Sigma^* \). This order \( \preceq \) can be extended to formal series, that is, the multiset-path ordering, see Dershowitz and Manna [24] or Huet and Oppen [25]. We still denote it by \( \preceq \).

**Proposition 3.** Addition and product are monotonic with respect to the multiset-path order. Moreover, addition is strictly monotonic with respect to \( \preceq \), 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 1). Suppose that there is \( w \in \mathcal{S} \) such that for all \( v \in \mathcal{R} \) we have \( v \triangleright w \), 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 \( r_0 \leq s_0 \) and \( r_1 \leq s_1 \). One of the two inequalities must be strict (otherwise \( r = s \)). Suppose \( r_0 \triangleleft s_0 \). By definition, observe that \( r_1 + s_1 \leq s_1 + t \). 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 \( r \leq s \) and let \( t \) be some series. We prove \( r \cdot t \leq s \cdot t \); the other inequality \( t \cdot r \leq t \cdot s \) is similar. Again, we proceed by induction on Definition [1]

- Suppose there is \( w \in \mathcal{S} \) such that for all \( v \in \mathcal{R} \cdot v \preceq w \). By induction on \( t \), if \( t = 0 \), \( r \cdot t = 0 \leq 0 = s \cdot t \). Otherwise, \( t = t_0 + v_0 \) for a word \( v_0 \). Observe that \( r \cdot v_0 = \sum_{v \in \mathcal{L}} r(v) v \cdot v_0 \). Since for all \( v \in \mathcal{R} \cdot v \cdot v_0 \preceq w \cdot v_0 \), we have \( r \cdot v_0 \preceq w \cdot v_0 \). 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, \( r \cdot t_0 \leq s \cdot t_0 \) and since \( r \cdot v_0 \preceq s \cdot v_0 \), 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 \( r \preceq s \) and again proceed by case analysis. Suppose that there is some \( w \in \mathcal{S} \) such that for all \( v \in \mathcal{R} \cdot v \preceq w \). Since \( t \neq 0 \), it contains at least one word \( v_0 \) such that \( t = t_0 + v_0 \). By \( r \preceq s \), \( r \cdot v_0 = \sum_{v \in \mathcal{L}} r(v) v \cdot v_0 \preceq \sum_{v \in \mathcal{S}} s(v) v \cdot v_0 = s \cdot v_0 \). Now, \( r \cdot t_0 \leq s \cdot t_0 \) by monotonicity. Thus \( r \cdot t = r \cdot t_0 + r \cdot v_0 \preceq s \cdot t_0 + s \cdot v_0 = s \cdot t \) where the strict inequality is due to strict monotonicity of addition.

\[ \square \]

**Definition 6** (Matrix multiset-path order). Let \( M \) and \( M' \) be two matrices with dimension \( P \times P \). Write \( M \preceq M' \) if for all \( k \geq |P| \) and for all \((i,j) \in P \times P \), we have \( M_{i,j}^{\preceq k} \leq M'_{i,j}^{\leq k} \).

**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. \( \square \)

### 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 \( \Sigma \), write \( L \preceq L' \) if there is an injective function \( f : L' \to L \) that is computed by a decreasing transducer \( \tau \).

The transducer \( \tau \) is said to be a *witness* of \( L \preceq L' \).

Recall that the identity on \( L \) is computed by a decreasing transducer (see Corollary [1]). Thus \( \preceq \) is reflexive. Also, it is well known that both transducers and injective functions can be composed. Hence, we also have that \( \preceq \) is transitive. Thus, \( \preceq \) is a preorder.

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

\[ L_1 = A \cdot (A + B)^* \preceq L_2 = B \cdot (A + B)^* \preceq L_1. \]

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

\[
\begin{array}{c}
\mathcal{R}_A \leftarrow \\
\mathcal{R}_B
\end{array}
\]

This shows that \( L_1 \preceq L_2 \). 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 \( L_1 \preceq L_2 \) has a witness \( \tau : L_2 \to L_1 \). If \( \tau \) 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 $L_1 \preceq L_2$ but not $L_2 \preceq L_1$. Ad absurdum, suppose $L_1 \preceq L_2 \preceq L_1$ with a transducer $\theta : L_1 \rightarrow L_2$ and $\tau$ 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 $\tau$. 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 define $L < L'$ if $\tau$ 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:

![Diagram of a transducer](image)

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

Definition 8. The rational embedding order extends to matrices by pointwise ordering: Let $M$ and $N$ with dimension $P \times P$, and write $M \preceq N$ if for every $i, j \in P \times P$, we have $M_{i,j} \preceq N_{i,j}$.

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

![Diagram of transducers](image)

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 $N_G \times N_G$ such that $(M_G')_{i,j} = T_1^{i,j} + \cdots + T_t^{i,j}$ with $T_1, \ldots, T_t$ 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 $\overline{M}_G = (M_G')^* < (M_G')^* = \overline{M}_{G'}$. 

ON GRAPH REWRITING SYSTEMS TERMINATION THROUGH LANGUAGE THEORY 15
Suppose that $G < G'$, that is $\overline{M}_G = (M'_G)^* < (M'_G')^* = \overline{M}_{G'}$. Consider a path $p_1 T_1 \rightarrow p_2 T_2 \cdots p_k$ in $G$. It is mapped bijectively to the word $T_1^{p_1} p_2 T_2 \cdots T_n^{p_n} p_{n+1}$. 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 $G$, $T \in \Sigma_E$ and ends with a letter of the shape $U^{n,s}$ for some $n \in G$, $U \in \Sigma_E$.

Suppose that we have some rule involving some pattern $P = L$ and its self-application $P \cdot \vec{c} = R$. Suppose $\overline{R} < \overline{T}$. Then, there are decreasing transducers such that any word $T^{r_1,s_1} \cdots T^{r_k,s_k}$ in $L$ read by $\tau_{r_1,s_k}$ 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.** 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 $M^G$ defined by:

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

The notation $M^G$ 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 $(M^G)^* = (M^*)^G$.

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

**Definition 10** (Context stability). We say that a partial order $\leq$ (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 $R^G \prec L^G$.

**Lemma 1.** Let $\preceq$ be a partial order stable by context and consider finite matrices $L, R, C$ 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 [10] 2. By Lemma 5 it follows that $(R^\uparrow)^* \prec (L^\uparrow)^*$. Clearly, $R^\uparrow$ and $L^\uparrow$ are finite, and from Definition 10, we have $(R^\uparrow + C)^* \prec (L^\uparrow + C)^*$. \[ \square \]

**Theorem 1.** Let $\preceq$ be a partial order stable by context. Suppose that for every rule $R = \langle P, \vec{c} \rangle$ with $P = \langle P_0, \vec{v} \rangle$ and $P_0'$ the self-application of $R$, we have $M'_0 \prec M'_0$. Then the corresponding GRS is terminating.
Proof. Let $\preceq$ be a partial order on graphs and consider the corresponding order on matrices: $G \preceq G'$ if and only if $M_G^2 \preceq M_{G'}^2$. We show that for every rule, we have $G \rightarrow G'$ implies $G' \preceq G$. As justified at the beginning of the section, this is sufficient for proving termination.

So let $R$ be a graph rewriting rule and let $\mu$ be a morphism such that $G \rightarrow_{R,\mu} G'$. By the discussion in the beginning of Section 3, without loss of generality, we can suppose that $\mu$ 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 $M_G = M_{P_0} + M_C$ and $M_{G'} = M_{P'_0} + M_C$. 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 $M_{G'}^* = (M_{P'_0} + M_C)^* \preceq (M_{P_0} + M_C)^* = M_G^*$. \hfill $\square$

4.8. Stability of the orderings. We can now prove the two announced 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 $R \triangleleft L$ with $R, L$ of dimension $P \times P$. Then, for all $(i, j) \notin P \times P$, $R_{i,j}^G = 0 \leq 0 = L_{i,j}^G$.

Now, for all $k \geq |G| \geq |P|$ and for all $(i, j) \in P \times P$, we have $(R^G_{i,j})_{i,j}^k = R_{i,j}^k \subseteq L_{i,j}^k = (L^G_{i,j})_{i,j}^k$. 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

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

and on the other side

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

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

**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 $R \triangleleft L$, there are decreasing transducers $\tau_{p,q} : L_{p,q} \rightarrow R_{p,q}$ 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 $p \xrightarrow{T_{p,q}} q$. 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: $p \xrightarrow{T_{p,q}} n$ for any $n \in P$ with $n \geq$ 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: $f_{n,p} \xrightarrow{T_{p,q}} q$ for...
any terminal state $f_{n,p}$ of the transducer $\tau_{n,p}, n \in P$. This construction can be represented as follows:

$$
\begin{array}{c}
\tau_{k,m} \\
\uparrow \\
X_{m,i} | X_{m,i} \\
V_{i,k} | V_{i,k} \\
T^{j,i} | T^{j,i} \\
W_{i,\ell} | W_{i,\ell} \\
U_{j,i} | U_{j,i} \\
\uparrow \\
\circ \\
\rho \\
\theta \\
\tau_{\ell,n} \\
Y_{n,i} | Y_{n,i} \\
\end{array}
$$

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 $p \not\in P$, we set the initial state to be $p$ itself. Otherwise, we introduce a new state $i_p$ which is set to be initial, and we add a transition $\tau_p \xrightarrow{11} i_{p,n}$ for any $n \in P$ with $i_{p,n}$ the initial state in $\tau_{p,n}$. If $q \not\in P$, then, $q$ is the final state. Otherwise, any terminal state $f_{n,s}$ within some $\tau_{n,s}, n \in P$ 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 $T_{r}^{r,m}$ and ends with $U_{n,s}^{m}$. 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 $w = w_1\ell_1w_2\cdots w_k\ell_k$. We let $w_1 = T_{1}^{r_1,s_1}\cdots T_{t_1}^{r_{t_1},s_{t_1}}$ and $\ell_1 = S_{u_1,v_1}\cdots S_{u_{s_1},v_{s_1}}$.

Let us start with a non empty $w_1$, that is $T_{1}^{r_1,s_1}$ is in $C$. Let us follow the computation on this word within the transducer. Up to $T_{t_1}^{r_{t_1},s_{t_1}}$, the transducer is deterministic, thus the unicity of the path up to $T_{t_1}^{r_{t_1},s_{t_1}}$. For the last letter, $T_{s_1}^{r_{s_1},s_{s_1}}$, there is a choice. One can reach any state $i_{s_1,m}$ that is initial within $\tau_{s_1,m}$. 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 $S^{m,s}$. Then, since it is a terminal state in $\tau_{s_1,m}$, that leads to $m = s$. 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 $T_{r}^{r,s}$ 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 beginning, again, there is a choice. One fires one of the transitions $\tau_p \xrightarrow{11} i_{p,m}$ 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 $\ell_i$’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 the outputs letters by the $\tau_{i,j}$’s), the word $u$ has the shape $u = w_1 r_1 \cdots w_k r_k = w'_1 r'_1 \cdots w_{k'} r'_{k'}$ with $r_1, \ldots, r_k, r'_1, \ldots, r'_{k'}$ some paths within $R$. Thus, $w_1$ and $w'_1$ have a common prefix, say $w_{1}''$. Suppose that $w_1$ is strictly shorter than $w'_1$. Its last letter has the shape $T^{n:s}$ with $s \in P$. That is not compatible with $w'_1$ (whose only letter of that sort is the last one). Thus, $w_1 = w'_1$. Then, $r_1 = r'_1$ since the $\tau_{r,s}$ 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 $t \succeq u$, 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 $\phi : G \rightarrow X$ 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 $\Omega$ 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, $G \rightarrow G'$ implies $G' \prec G$. Indeed, suppose that $G \rightarrow_{R,p} G'$. 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 $\phi(P_0') \prec \phi(P_0)$, we then have $\phi(G') \prec \phi(G)$.

**Example 4.** The triple $\langle M, \leq, (M(\cdot))' \rangle$ is an interpretation for ‘Follow’.

**Example 5.** Let us come back to the weight analysis. Define $\omega(G) = \sum_{p \rightarrow_{\eta} q \in G} \omega(\epsilon)$ with $\omega(\alpha) = 0$, $\omega(T) = -1$, $\omega(\beta) = -1$. Then, $\langle R, <, \omega(-) \rangle$ 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 $R_1$, and let $\langle X_2, \prec_2, \phi_2 \rangle$ be an interpretation for a set of rules $R_2$. Suppose that for every
rule $R$ in $R_2$, $G \rightarrow_{R,\mu} G'$ 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 $R_1 \cup R_2$.

Thus, 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 $R$ be a terminating GRS. Then there is an interpretation that “justifies” this fact. Indeed, take $(G, \prec, 1_G)$ with $\prec$ defined to be the transitive closure of the rewriting relation $\rightarrow$. The termination property ensures that the closure leads to an irreflexive relation. The compatibility of $\prec$ with respect to $1_G$ 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 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. 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 embedding, we can extend them to take into account partial orders on the edge labels and partial orders on the node labels. This is left for further work. We left that part to skip technical difficulties related to this step.

As the next natural step, we intend to explore more systematically graph rewriting with node creation. Let us say a few word about it. First, matrices would have infinite dimension. But, at the same time, since graphs are finite, only finitely many entries are non null. Here again, the technical step is not immediate, but not without hope.

Second, there is one slippery point that must be discussed in details. 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, the hypothesis cannot hold in general.

Third, suppose the ordering is not well-founded. We need an extra-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 $G_{G'} \prec M_G$. Suppose furthermore that for any steps, $G \rightarrow G'$, we have $M_{G'} \prec M_G$, then the system is terminating. Indeed, due to Robertson and Seymour’s Theorem (see [27], ), 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,
$M_G \prec M_G$, which leads to the contradiction. The notion of minors for the directed case may be discussed further, see for instance [28]. We leave that exploration for some other day.

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. Anyway, from the theory to the implementation, we may meet some surprises.

References


**Université de Lorraine, CNRS, LORIA**

*Email address*: guillaume.bonfante@univ-lorraine.fr

**Université de Lorraine, CNRS, Inria, LORIA**

*Email address*: miguel.couceiro@loria.fr