



HAL
open science

On termination of Graph Rewriting Systems through language theory

Guillaume Bonfante, Miguel Couceiro

► **To cite this version:**

Guillaume Bonfante, Miguel Couceiro. On termination of Graph Rewriting Systems through language theory. 2020. hal-02478696

HAL Id: hal-02478696

<https://inria.hal.science/hal-02478696v1>

Preprint submitted on 14 Feb 2020

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.

1 On Termination of Graph Rewriting Systems 2 through Language Theory

3 **Guillaume Bonfante**

4 University of Lorraine
5 LORIA, Nancy, France
6 guillaume.bonfante@univ-lorraine.fr.fr

7 **Miguel Couceiro**

8 University of Lorraine
9 LORIA, Nancy, France
10 miguel.couceiro@loria.fr

11 — Abstract —

12 The termination issue we tackle is rooted in natural language processing where graph rewriting
13 systems (GRS) may contain a large number of rules, often in the order of thousands. Decidable
14 concepts thus become mandatory to verify the termination of such systems. The notion of graph
15 rewriting consider does not make any assumption on the structure of graphs (they are not “term
16 graphs”, “port graphs” nor drags). The lack of algebraic structure in our setting led us to proposing
17 two orders on graphs inspired from language theory: the matrix multiset-path order and the rational
18 embedding order. We show that both are stable by context, which we then use to obtain the main
19 contribution of the paper: under a suitable notion of “interpretation”, a GRS is terminating if and
20 only if it is compatible with an interpretation.

21 **2012 ACM Subject Classification** Theory of computation → Rewrite systems

22 **Keywords and phrases** Graph Rewriting, Termination, Orders, Natural Language Processing

23 **Digital Object Identifier** 10.4230/LIPIcs...

24 **1** Introduction

25 Computer linguists rediscovered few years ago that graph rewriting is a good model of
26 computation for rule-based systems. They used traditionally terms, see for instance Chomsky’s
27 Syntagmatic Structures [3]. But usual phenomena such as anaphora do not fit really well
28 within such theories. In such situations, graphs behave much better. For examples of graph
29 rewriting in natural language processing, we refer the reader to the parsing procedure by
30 Guillaume and Perrier [12] or the word ordering modeling by Kahane and Lareau [14]. The
31 first named author with Guillaume and Perrier designed a graph rewriting model called
32 GREW [2] that is adapted to natural language processing.

33 The rewriting systems developed by the linguists often contain a huge number of rules, e.g.,
34 those synthesized from lexicons (e.g. some rules only apply to transitive verbs). For instance,
35 in [12], several systems are presented, some with more than a thousand of rules. Verifying
36 properties such as termination by hand thus becomes intractable. This fact motivates our
37 framework for tackling the problem of GRS termination.

38 Following the tracks of term rewriting, for which the definition is essentially fixed by the
39 algebraic structure of terms, many approaches to graph rewriting emerged in past years. Some
40 definitions (here meaning semantics) are based on a categorical framework, e.g., the double
41 pushout (DPO) and the single pushout (SPO) models, see [21]. To make use of algebraic
42 potential, some authors make some, possibly weak, hypothesis on graph structures, see for
43 instance the main contribution by Courcelle and Engelfriet [4] where graph decompositions,
44 graph operations and transformations are described in terms of monadic second-order logics



© Guillaume Bonfante and Miguel Couceiro;
licensed under Creative Commons License CC-BY
Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

45 (with the underlying decidability/complexity results). In this spirit, Ogawa describes a graph
46 algebra under a limited tree-width condition [18].

47 Another line of research follows from the seminal work by Lafont [15] on interaction
48 nets. The latter are graphs where nodes have some extra structure: nodes have a label
49 related to some arity and co-arity. Moreover, nodes have some "principal gates" (ports) and
50 rules are actionned via them. One of the main results by Lafont is that rewriting in this
51 setting is (strongly) confluent. This approach has been enriched by Fernandez, Kirchner
52 and Pinaud [11], who implemented a fully operational system called PORGY with strategies
53 and related semantics. Also, it is worth mentioning the graph rewriting as described by
54 Dershowitz and Jouannaud [8]. Here, graphs are seen as a generalization of terms: symbols
55 have a (fixed) arity, graphs are connected via some sprouts/variables as terms do. With such
56 a setting, a good deal of term rewriting theory also applies to graphs.

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

71 Some hypotheses are rather unusual for standard computations, e.g., fixed number of
72 nodes. Indeed, nodes are usually related to words or concepts (which are themselves closely
73 related to words). A paraphrase may be a little bit longer than its original version, but
74 its length can be easily bounded by the length of the original sentence up to some linear
75 factor. In GREW, node creations are restricted. To take into account the rare cases for which
76 one needs extra nodes, a "reserve" is allocated at the beginning of the computation. All
77 additional nodes are taken from the reserve. Doing so has some efficiency advantages, but
78 that goes beyond the scope of the paper. Also, node and edge labels, despite being large,
79 remain finite sets: they are usually related to some lexicons. These facts together have an
80 important impact on the termination problem: since there are only finitely many graphs of a
81 given size, rewriting only leads to finitely many outcomes. Thus, deciding termination for a
82 *particular input graph* is decidable. However, our problem is to address termination in the
83 class of *all* graphs. The latter problem is often referred to as *uniform termination*, whereas
84 the former is referred to as *non-uniform*. For word rewriting, uniform termination of non
85 increasing systems constituted a well known problem, and was shown to be undecidable by
86 Sénizergues in [24].

87 This paper proposes a novel approach for termination of graph rewriting. In a former
88 paper [1], we proposed a solution based on label weights. Here, the focus is on the description
89 (and the ordering) of paths within graphs. In fact, paths in a graph can be seen as regular
90 languages. The question of path ordering thus translates into a question of regular language
91 orderings. Accordingly, we define the *graph multi-set path ordering* that is related to that in
92 [6]. Dershowitz and Jouannaud, in the context of drag rewriting, consider a similar notion of

93 path ordering called GPO (see [7]). Our definitions diverge from theirs in that our graph
 94 rewriting model is quite different: here, we do not benefit (as they do) from a good algebraic
 95 structure. Our graphs have no heads, tails nor hierarchical decomposition. In fact, our
 96 ordering is not even well founded! Relating the two notions is nevertheless interesting and
 97 left for further work. Plump [20] also defines path orderings for term graphs, but those
 98 behave like sets of terms.

99 One of our graph orderings will involve matrices, and orderings on matrices. Nonetheless,
 100 as far as we see, there is no relationship with matrix interpretations as defined by Endrullis,
 101 Waldmann and Zantemma [10].

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

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

- 112 1. We propose two orders on graphs inspired from language theory, and we show that both
 113 are monotonic and stable by context.
- 114 2. We introduce a notion of graph interpretation, and show that terminating GRSs are
 115 exactly those compatible with such interpretations.

116 2 Notations and Graph Rewriting

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

121 A *language* on Σ is some subset $L \subseteq \Sigma^*$. The set of all languages on Σ is $\mathcal{P}(\Sigma^*)$. The
 122 addition of two languages $L, L' \subseteq \Sigma^*$ is defined by $L + L' = \{w \mid w \in L \vee w \in L'\}$. The empty
 123 language is denoted by 0 and $\langle \mathcal{P}(\Sigma^*), +, 0 \rangle$ is also a monoid. Given some word $w \in \Sigma^*$, we
 124 will also denote by w the language made of the singleton $\{w\} \in \mathcal{P}(\Sigma^*)$. Given two languages
 125 $L, L' \subseteq \Sigma^*$, their concatenation is defined by $L \cdot L' = \{w \cdot w' \mid w \in L \wedge w' \in L'\}$. In this way,
 126 $\langle \mathcal{P}(\Sigma^*), \cdot, 1, \cdot \rangle$ is also a monoid.

127 A *preorder* on a set X is a binary relation $\preceq \subseteq X^2$ that is reflexive ($x \preceq x$, for all $x \in X$)
 128 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*
 129 *order* if it is anti-symmetric (if $x \preceq y$ and $y \preceq x$, then $x = y$, for all $x, y \in X$). A preorder is
 130 an *equivalence relation* if it is symmetric ($x \preceq y \Rightarrow y \preceq x$). Observe that each preorder \preceq
 131 induces an equivalence relation \sim : $a \sim b$ if $a \preceq b$ and $b \preceq a$. The strict part of \preceq is then
 132 the relation: $x \prec y$ iff $x \preceq y$ and $\neg(x \sim y)$. We also mention the “dual” preorder \succeq of \preceq
 133 defined by: $x \succeq y$ iff $y \preceq x$. A preorder \preceq is said to be *well-founded* if there is no infinite
 134 chain $\dots \prec x_2 \prec x_1$ or, equivalently, $x_1 \succ x_2 \succ \dots$.

135 The remainder of this section may be found in [2] and we refer the reader to it for an
 136 extended presentation. We suppose given a (finite) set Σ_N of *node labels*, a (finite) set Σ_E
 137 of *edge labels* and we define graphs accordingly. A graph is a triple $G = \langle N, E, \ell \rangle$ with

XX:4 On Termination of Graph Rewriting Systems through Language Theory

138 $E \subseteq N \times \Sigma_E \times N$ and $\ell : N \rightarrow \Sigma_N$ is the labeling function of nodes. Note that there may be
 139 more than one edge between two nodes, but at most one is labeled with some $e \in \Sigma_E$. In the
 140 sequel, we use the notation $m \xrightarrow{e} n$ for an edge $(m, e, n) \in E$.

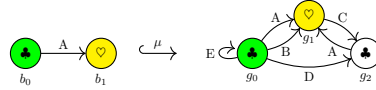
141 Given a graph G , the sets \mathcal{N}_G , \mathcal{E}_G and ℓ_G denote respectively the sets of nodes, edges and
 142 its labeling function. We will also (abusively) use the notation $m \in G$ and $m \xrightarrow{e} n \in G$
 143 instead of $m \in \mathcal{N}_G$ and $m \xrightarrow{e} n \in \mathcal{E}_G$ when the context is clear. Furthermore, in $\textcircled{\clubsuit}_a \xrightarrow{A} \textcircled{\heartsuit}_b$,
 144 a, b are nodes, \clubsuit, \heartsuit are the respective node labels and A is the edge label (here between a
 145 and b).

146 The set of graphs on node labels Σ_N and edge labels Σ_E is denoted by $\mathcal{G}_{\Sigma_N, \Sigma_E}$ or \mathcal{G}
 147 in short. Two graphs G and G' are said to share their nodes when $\mathcal{N}_G = \mathcal{N}_{G'}$. Given two
 148 graphs G and G' such that $\mathcal{N}_G \subseteq \mathcal{N}_{G'}$, set $G \blacktriangleleft G' = \langle \mathcal{N}_{G'}, \mathcal{E}_G \cup \mathcal{E}_{G'}, \ell \rangle$ with $\ell(n) = \ell_G(n)$ if
 149 $n \in \mathcal{N}_G$ and $\ell(n) = \ell_{G'}(n)$, otherwise.

150 A graph morphism μ between a source graph G and a target graph H is a function $\mu : \mathcal{N}_G \rightarrow \mathcal{N}_H$
 151 that preserves edges and labelings, that is, for all $m \xrightarrow{e} n \in G$, $\mu(m) \xrightarrow{e} \mu(n) \in G'$
 152 holds, and for any node $n \in G$: $\ell_G(n) = \ell_{G'}(\mu(n))$. A *basic pattern* is a graph, and a
 153 *basic pattern matching* is an injective morphism from a basic pattern P to some graph G .
 154 Given such a morphism $\mu : P \rightarrow G$, we define $\mu(P)$ to be the sub-graph of G made of
 155 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
 156 $\mu(n) \mapsto \ell_G(\mu(n))$.

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

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



163
 164 The pattern $P = \langle P_0, [\nu] \rangle$ with ν defined by $\textcircled{\heartsuit}_{b_1} \xrightarrow{A} \textcircled{\heartsuit}_{b_1} \xrightarrow{\nu} \textcircled{\heartsuit}_{b_0} \xrightarrow{A} \textcircled{\heartsuit}_{b_1}$ prevents the application
 165 of the morphism above. Indeed, $\xi = b_0 \mapsto g_0, b_1 \mapsto g_1$ is such that $\xi \circ \nu = \mu$. When there is
 166 only one negative condition, we represent the pattern by crossing nodes and edges which *are*
 167 *not* within the basic pattern. For instance, the pattern P above looks like $\textcircled{\heartsuit}_{b_0} \xrightarrow{A} \textcircled{\heartsuit}_{b_1}$ that we
 168 hope is self-explanatory.

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

170 ► **Definition 2 (The command language).** *There are three basic commands: `label(p, alpha)` for*
 171 *node renaming, `del_edge(p, e, q)` for edge deletion and `add_edge(p, e, q)` for edge creation.*
 172 *In these basic commands, p and q are nodes, α is some node label and e is some edge label.*
 173 *A pattern $\langle P_0, \vec{\nu} \rangle$ is compatible with a command whenever p and q are nodes in P_0 .*

174 ► **Definition 3 (Operational semantics).** *Given a pattern $P = \langle P_0, \vec{\nu} \rangle$ compatible with some*
 175 *command c , and some pattern matching $\mu : P \rightarrow G$ where G is the graph on which the*
 176 *transformation is applied, we have the following possible cases: $c = \text{label}(p, \alpha)$ turns the*
 177 *label of $\mu(p)$ into α , $c = \text{del_edge}(p, e, q)$ removes $\mu(p) \xrightarrow{e} \mu(q)$ if it exists, otherwise*
 178 *does nothing, and $c = \text{add_edge}(p, e, q)$ adds the edge $\mu(p) \xrightarrow{e} \mu(q)$ if it does not exist,*
 179 *otherwise does nothing. The graph obtained after such an application is denoted by $G \cdot_\mu c$.*

180 Given a sequence of commands $\vec{c} = (c_1, \dots, c_n)$, let $G \cdot_{\mu} \vec{c}$ be the resulting graph, i.e.,
 181 $G \cdot_{\mu} \vec{c} = (\dots((G \cdot_{\mu} c_1) \cdot_{\mu} c_2) \cdot_{\mu} \dots c_n)$.

182 **► Definition 4.** A rule is a pair $R = \langle P, \vec{c} \rangle$ made of a pattern and a (compatible) sequence of
 183 commands. Such a rule R applies to a graph G when there is a pattern morphism $\mu : P \rightarrow G$.
 184 Let $G' = G \cdot_{\mu} \vec{c}$, then we write $G \rightarrow_{R, \mu} G'$. We define $G \rightarrow G'$ whenever there is a rule R
 185 and a pattern morphism μ such that $G \rightarrow_{R, \mu} G'$.

186 2.1 The main example

187 Let $\Sigma_N = \{A\}$ and $\Sigma_E = \{\alpha, \beta, T\}$. For the discussion, we suppose that T is a working label,
 188 that is not present in the initial graphs. We want to add a new edge β between node n
 189 and node 1 each time we find a maximal chain: $\textcircled{A}_1 \xrightarrow{\alpha} \textcircled{A}_2 \xrightarrow{\alpha} \textcircled{A}_3 \xrightarrow{\alpha} \dots \xrightarrow{\alpha} \textcircled{A}_n$ within a graph

190 G . Consider the basic pattern $P_{init} = \textcircled{A}_p \xrightarrow{\alpha} \textcircled{A}_q$ together with its two negative conditions

191 $\nu_1 = \textcircled{X} \xrightarrow{\alpha} \textcircled{A}_p \xrightarrow{\alpha} \textcircled{A}_q$ and $\nu_2 = \textcircled{X} \xrightarrow{\beta} \textcircled{A}_p \xrightarrow{\alpha} \textcircled{A}_q$. We consider three rules:

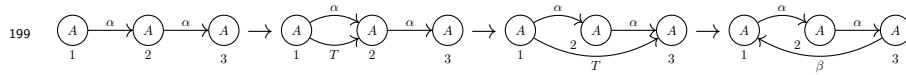
192 **Init:** $\langle \langle P_{init}, [\nu_1, \nu_2] \rangle, (\text{add_edge}(p, T, q)) \rangle$ which fires the transitive closure.

193 **Follow:** $\langle \textcircled{A}_p \xrightarrow{T} \textcircled{A}_q \xrightarrow{\alpha} \textcircled{A}_r, (\text{add_edge}(p, T, r), \text{del_edge}(p, T, q)) \rangle$ which follows the chain.

194 **End:** $\langle \textcircled{A}_p \xrightarrow{T} \textcircled{A}_q \xrightarrow{\alpha} \textcircled{X}, (\text{del_edge}(p, T, q), \text{add_edge}(q, \beta, p)) \rangle$ which stops the processus.

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

198 **► Example 5.** Take $\textcircled{A}_1 \xrightarrow{\alpha} \textcircled{A}_2 \xrightarrow{\alpha} \textcircled{A}_3$. By applying 'Init', 'Follow' and 'End', it rewrites as:



200 2.2 Three technical facts about Graph Rewriting

201 It is well known that the main issue with graph rewriting definitions is the way the context
 202 is related to the pattern image and its rewritten part. We shall tackle this issue with
 203 Proposition 6.

204 Self-application

205 Let $R = \langle P, \vec{c} \rangle$ be the rule made of a pattern $P = \langle P_0, \vec{v} \rangle$ and a sequence of commands \vec{c} .
 206 There is the identity morphism $1_{P_0} : P_0 \rightarrow P_0$, and thus we can apply rule R on P_0 itself,
 207 that is, $P_0 \rightarrow_{R, 1_{P_0}} P'_0 = P_0 \cdot_{1_{P_0}} \vec{c}$. We call this latter graph the *self-application* of R .

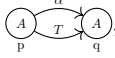
208 Rule node renaming

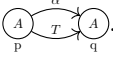
209 To avoid heavy notation, we will use the following trick. Suppose that we are given a
 210 rule $R = \langle P, \vec{c} \rangle$, a graph G and a pattern morphism $\mu : P \rightarrow G$. Let $P = \langle P_0, \vec{v} \rangle$.
 211 We define R_{μ} to be the rule obtained by renaming nodes p in P_0 to $\mu(p)$ (and their
 212 references within \vec{c}). For instance, the rule 'Follow' can be rewritten as $Follow_{\mu} =$
 213 $\langle \textcircled{A}_1 \xrightarrow{T} \textcircled{A}_2 \xrightarrow{\alpha} \textcircled{A}_3, (\text{add_edge}(1, T, 3), \text{del_edge}(1, T, 2)) \rangle$ where μ denotes the pattern morphism
 214 used to apply 'Follow' in the derivation. Observe that: (i) the basic pattern of R_{μ} is
 215 actually $\mu(P_0)$, which is a subgraph of G , (ii) $\iota : \mu(P_0) \rightarrow G$ mapping $n \mapsto n$ is a pattern

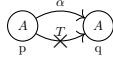
XX:6 On Termination of Graph Rewriting Systems through Language Theory

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

219 Uniform rules

220 Let us consider rule 'Init' above. It applies on: , and the result is the graph itself:

221 . Indeed, we cannot add an already present edge (relative to a label) within a
 222 graph. Thus, depending on the graph, the rule will or will not append an edge. Such an

223 unpredictable behavior can be easily avoided by modifying the pattern of 'Init' to: .

224 The same issue may come from edge deletions. A *uniform* rule is one for which commands
 225 apply (that is, modify the graph) for each rule application. Since this is not the scope of the
 226 paper, we refer the reader to [2] for a precise definition of uniformity. We will only observe
 227 two facts.

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

230 Second, the following property holds for uniform rules (see [2]§7 for a proof).

231 ► **Proposition 6.** *Suppose that $G \rightarrow_{R,\iota} G'$ with $R = \langle P, \vec{c} \rangle$ and $P = \langle P_0, \vec{v} \rangle$ (the basic pattern
 232 P_0 being a subgraph of G). Let C be the graph obtained from G by deleting the edges in P_0 .
 233 Then $G = P_0 \blacktriangleleft C$ and $G' = P'_0 \blacktriangleleft C$ with P'_0 being the self-application of the rule. Moreover,
 234 $\mathcal{E}_C \cap \mathcal{E}_{P_0} = \emptyset$ and $\mathcal{E}_C \cap \mathcal{E}_{P'_0} = \emptyset$.*

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

236 3 Termination of Graph Rewriting Systems

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

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

244 ► **Remark 7.** Suppose that we are given a strict partial order \succ , not necessarily well founded.
 245 If $G \rightarrow G'$ implies $G \succ G'$ for all graphs G and G' , then the system is terminating. Indeed,
 246 suppose it is not the case, let $G_1 \rightarrow G_2 \rightarrow \dots$ be an infinite reduction sequence. Since there
 247 are only finitely many graphs of size of G_1 , it means that there are two indices i and j such
 248 that $G_i \rightarrow \dots \rightarrow G_j$ with $G_i = G_j$. But then, since $G_i \succ G_{i+1} \succ \dots \succ G_j$, we have that
 249 $G_i \succ G_j = G_i$ which is a contradiction.

250 A similar argument was exhibited by Dershowitz in [5] in the context of term rewriting.
 251 For instance, it is possible to embed rewriting within real numbers rather than natural
 252 numbers to prove termination.

253 Let us try to prove the termination of our main example (see Subsection 2.1). Rules such
 254 as 'Init' and 'End' are "simple": we put a weight on edge labels $\omega : \Sigma_E \rightarrow \mathbb{R}$ and we say that
 255 the weight of a graph is the sum of the weights of its edges labels. Set $\omega(\alpha) = 0, \omega(\beta) = -2$
 256 and $\omega(T) = -1$. Then, rules 'Init' and 'End' decrease the weight by 1 and, since rule 'Follow'

257 keeps the weight constant, it means the two former rules can be applied only finitely many
 258 times. Observe that negative weights are no problem with respect to Remark 7.

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

260 3.1 A language point of view

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

266 Now, let us consider the effect of graph rewriting in terms of languages. Consider an
 267 application of the 'Follow' rule: $G \rightarrow G'$. Any word to state r that goes through the
 268 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
 269 languages corresponding to state r contain shorter words. The remainder of this section is
 270 devoted to formalizing this intuition into proper orders on graphs. For that, we will need to
 271 *count* the number of paths between any two states. Hence, we shall introduce \mathbb{N} -rational
 272 expressions, that is, *rational expression with multiplicity*. See, e.g., Sakarovitch's book [23]
 273 for an introduction and justifications of the upcoming constructions. We introduce here the
 274 basic ideas.

275 3.2 Formal series

276 A formal series on Σ (with coefficients in \mathbb{N}) is a (total) function $s : \Sigma^* \rightarrow \mathbb{N}$. Given a word
 277 w , $s(w)$ is the multiplicity of w . The set of words $\underline{s} = \{w \in \Sigma^* \mid s(w) \neq 0\}$ is the *support*
 278 of s . Given $n \in \mathbb{N}$, let \mathbf{n} be the series defined by $\mathbf{n}(w) = 0$, if $w \neq 1$, and $\mathbf{n}(1) = n$, where 1
 279 denotes the empty word. The empty language is $\mathbf{0}$, the language made of the empty word is
 280 $\mathbf{1}$. Moreover, for $a \in \Sigma$, the series a is given by $a(w) = 0$ if $w \neq a$ and $a(a) = 1$.

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

285 Given a series s , let $s^{\leq k}$ be its restriction to words of length less or equal to k , i.e.,
 286 $s^{\leq k}(w) = 0$ whenever $|w| > k$ and $s^{\leq k}(w) = s(w)$, otherwise.

287 An \mathbb{N} -rational expression on an alphabet Σ is built upon the grammar [22]:

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

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

292 3.3 The language matrix

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

XX:8 On Termination of Graph Rewriting Systems through Language Theory

298 Let 1_P be the unit matrix of dimension $P \times P$, that is $(1_P)_{i,j} = 0$ if $i \neq j$ else 1.
 299 From now on, we abbreviate the notation from 1_P to 1 if the context is clear. Then, let
 300 $M_G^* = 1 + M_G + M_G^2 + \dots$. Each component of M_G^* is actually an \mathbb{N} -regular expression (see
 301 Sakarovitch Ch. III, §4 for instance). The (infinite) sum is correctly defined since for all i, j ,
 302 $(M_G)_{i,j} = T_1 + \dots + T_\ell$. Thus, $1 \notin (M_G)_{i,j}$.

303 The question about termination can be reformulated in terms of matrices whose compon-
 304 ents are languages (with words counted with their multiplicity). To prove the termination
 305 of the rewriting system, it is then sufficient to prove that for any two graphs $G \rightarrow G'$,
 306 $M_G^* > M_{G'}^*$. To prove such a property in the infinite class of finite graphs, we will use the
 307 notion of “stable orders”.

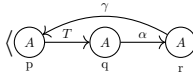
Recall the ‘Follow’ rule and consider the basic pattern L and the self-application R . Then,

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

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

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

308 Any word within M_R^* ’s components is a sub-word of the corresponding component in M_L^* .

► **Example 8.** Consider now a variation of ‘Follow’: 
 $(\text{add_edge}(p, T, r), \text{del_edge}(p, T, q))$.

By setting L' as the pattern and R' as the self-application, we get the following matrices:

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

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

3.4 The matrix multiset path order

311 The order we shall introduce in this section is inspired by the notion of *multiset path ordering*
 312 within the context of term rewriting (see for instance [6]). However, in the present context of
 313 graph rewriting (to be compared with Dershowitz and Jouannaud’s [7] or with Plump’s [20]),
 314 the definition is a bit more complicated. Here, we do not consider an order on letters as it is
 315 done for terms.

316 Let \leq be the word embedding on Σ^* , that is, the smallest partial order such that $1 \leq w$,
 317 and if $u \leq v$, then $(u \cdot w \leq v \cdot w$ and $w \cdot u \leq w \cdot v$, for all $u, v, w \in \Sigma^*$. This order \leq can be
 318 extended to formal series, that is, the multiset-path ordering, see Dershowitz and Manna [9]
 319 or Huet and Oppen [13].

320 ► **Definition 9** (Multiset path order). *The multiset path order is the smallest partial order on*
 321 *finite series such that*

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

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

325 ► **Proposition 10.** *Addition and product are monotonic with respect to the multiset-path
 326 order. Moreover, addition is strictly monotonic with respect to \trianglelefteq , and if $r \triangleleft s$, then $r \cdot t \triangleleft s \cdot t$
 327 and $t \cdot r \triangleleft t \cdot s$, whenever $t \neq 0$ (otherwise, we have equality).*

328 **Proof.** It is not difficult to see that addition is monotonic. So suppose that $r \triangleleft s$. We prove that
 329 $r + t \triangleleft s + t$, by induction (see Definition 9). Suppose that there is $w \in \underline{s}$ such that for all $v \in \underline{r}$
 330 we have $v \triangleleft w$, then $r \triangleleft s$. Since $r(w) = 0$, then $(r + t)(w) = t(w) < s(w) + t(w) = (s + t)(w)$,
 331 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
 332 the two inequalities must be strict (otherwise $r = s$). Suppose $r_0 \triangleleft s_0$. By definition, observe
 333 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
 334 induction on (r_0, s_0) . As addition is commutative, the result holds.

335 For the product, suppose that $r \trianglelefteq s$ and let t be some series. We prove $r \cdot t \trianglelefteq s \cdot t$; the
 336 other inequality $t \cdot r \trianglelefteq t \cdot s$ is similar. Again, we proceed by induction on Definition 9:

- 337 ■ Suppose there is $w \in \underline{s}$ such that for all $v \in \underline{r}$, $v \triangleleft w$. By induction on t , if $t = 0$,
 338 $r \cdot t = 0 \trianglelefteq 0 = s \cdot t$. Otherwise, $t = t_0 + v_0$ for a word v_0 . Observe that $r \cdot v_0 = \sum_{v \in \underline{r}} r(v)v \cdot v_0$.
 339 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, $r \cdot t = r \cdot (t_0 + v_0) = r \cdot t_0 + r \cdot v_0$
 340 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
 341 holds.
 342 ■ 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
 343 result then follows by induction.

344 To show strict monotonicity, suppose $r \triangleleft s$ and again proceed by case analysis. Suppose that
 345 there is some $w \in \underline{s}$ such that for all $v \in \underline{r}$, $v \triangleleft w$. Since $t \neq 0$, it contains at least one word
 346 v_0 such that $t = t_0 + v_0$. By $r \triangleleft s$, $r \cdot v_0 = \sum_{v \in \underline{r}} r(v)v \cdot v_0 \triangleleft \sum_{v \in \underline{s}} s(v)v \cdot v_0 = s \cdot v_0$. The
 347 result then follows by induction on the expansion of t and using the strict monotonicity of
 348 addition. ◀

349 ► **Definition 11** (Matrix multiset-path order). *Let M and M' be two matrices with dimension
 350 $P \times P$. Write $M \trianglelefteq M'$ if for all $k \geq |P|$ and for all $(i, j) \in P \times P$, we have $M_{i,j}^{\leq k} \trianglelefteq M'_{i,j}^{\leq k}$.*

351 ► **Corollary 12.** *The addition and the multiplication are monotonic with respect to the matrix
 352 multiset-path order.*

353 **Proof.** It follows from Proposition 10 since addition and product of matrices are defined as
 354 addition and product of their components. ◀

355 3.5 The Rational Embedding Order

356 Let Σ be some fixed alphabet. For a transducer τ , we denote the function it computes by $[\tau]$.

357 ► **Definition 13** (Rational Embedding Order). *Given two regular languages L and L' on Σ ,
 358 write $L \lesssim L'$ if:*

- 359 ■ there is an injective function $[\tau] : L' \rightarrow L$ and
 360 ■ $[\tau]$ can be computed by a transducer τ such that $|\tau(w)| \leq |w|$, for every $w \in L'$. Such
 361 transducers are said to be decreasing (in [16]).

XX:10 On Termination of Graph Rewriting Systems through Language Theory

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

363 We say that a transition of a transducer is *deleting* when it is of the form $a \mid 1$ for some
 364 $a \in \Sigma$. A transducer whose transitions are of the form $X \mid Y$, with $|Y| \leq |X|$, is itself
 365 decreasing. If a path corresponding to an input w passes through a deleting transition, then
 366 $|\tau(w)| < |w|$.

367 In the sequel we will make use of the following results that are direct consequences of
 368 Nivat's Theorem [17].

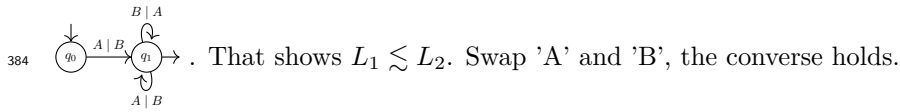
369 ► **Proposition 14.** *Let $[\tau] : L \rightarrow L'$ be computed by a transducer τ , and let L'' be a regular
 370 language. Then the following assertions hold.*

- 371 1. *The restriction $\tau|_{L''} : L'' \cap L \rightarrow L'$ mapping $w \mapsto \tau(w)$ is computable by a transducer.*
- 372 2. *The co-restriction $\tau|^{L''} : L \rightarrow L' \cap L''$ mapping $w \mapsto \tau(w)$ if $\tau(w) \in L''$ and otherwise
 373 undefined, is computable by a transducer.*
- 374 3. *The function $\tau' : L \rightarrow L'$ defined by $\tau'(w) = \tau(w)$ if $w \in L''$ and otherwise undefined, is
 375 computable by a transducer.*

376 Observe that the identity on Σ^* is computed by a transducer (made of a unique ini-
 377 tial/final state with transitions $a \mid a$ for all $a \in \Sigma$). Then, the identity on L is obtained
 378 by Proposition 14(1,2). Thus we have that \lesssim is reflexive. Also, it is well known that both
 379 transducers and injective functions can be composed. Hence, we also have that \lesssim is transitive.
 380 Thus, \lesssim is a preorder.

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

382 ► **Example 15.** $L_1 = A \cdot (A + B)^* \lesssim L_2 = B \cdot (A + B)^* \lesssim L_1$. Consider the transducer (in
 383 the drawing, the initial state is shown with an in-arrow and the final one by an out-arrow):



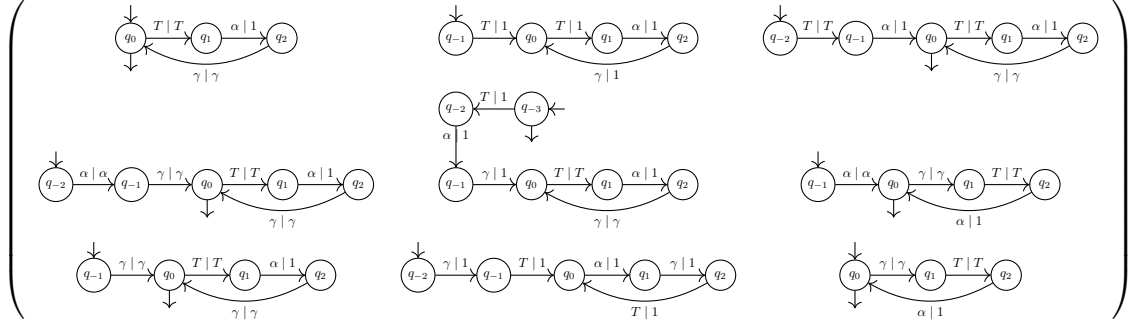
385 However, there is a simple criterion to ensure a strict inequality. Suppose $L_1 \lesssim L_2$ has
 386 a witness $\tau : L_2 \rightarrow L_1$. If τ contains one (accessible and co-accessible) deleting transition,
 387 then, the relation is strict.

388 As before, set $L_1 < L_2$ whenever $L_1 \lesssim L_2$ but not $L_2 \lesssim L_1$. Suppose $L_1 \lesssim L_2 \lesssim L_1$
 389 with a transducer $\theta : L_1 \rightarrow L_2$ and τ as above. Let w be the smallest input word from the
 390 initial state to a final state through the transition $a \mid 1$. Then $\theta \circ \tau$ (the composition of the
 391 two transducers) defines an injective function. Define the set $M^{<|w|} = \{u \in L_2 \mid |u| < |w|\}$.
 392 Generally speaking, for any word u , $|\theta \circ \tau(u)| \leq |u|$. Thus $\theta \circ \tau(M^{<w}) \subseteq M^{<w}$. Since $M^{<w}$
 393 is a finite set and $\theta \circ \tau$ is injective, it is actually bijective when restricted to $M^{<w}$. However,
 394 $|\theta \circ \tau(w)| \leq |\tau(w)| < w$ implies $\theta \circ \tau(w) \in M^{<w}$. By the Pigeon-hole Principle, there is one
 395 word in $M^{<w}$ that has two pre-images via $\theta \circ \tau$. Thus, $\theta \circ \tau$ cannot be injective, which yields
 396 a contradiction.

397 ► **Remark 16.** Observe that, when two regular languages verify $L \subseteq L'$, it follows from
 398 Proposition 14 that $L \lesssim L'$.

399 ► **Definition 17.** *The rational embedding order extends to matrices by pointwise ordering:
 400 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
 401 $M_{i,j} \lesssim N_{i,j}$.*

402 Recall the modified version of 'Follow' (Example 8). The following transducers show that
 403 all components strictly decrease.



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

409 3.6 Stable orders on matrices

410 A matrix on E is said to be *finite* whenever all its component are finite. Two matrices M
 411 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 18.** 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:

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

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

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

414 ► **Definition 19.** We say that a partial order \preceq on E is stable by context if for every $P \subseteq G$,
 415 all matrices L and R of dimension $P \times P$, and every C of dimension $G \times G$, the following
 416 assertions hold.

- 417 1. If L, R, C are finite, L being disjoint from C , R being disjoint from C and $R^* \prec L^*$, then
 418 $(R + C)^* \prec (L + C)^*$;
- 419 2. If $R \prec L$, then $R^{\uparrow G} \prec L^{\uparrow G}$.

420 ► **Lemma 20.** Let \preceq be partial order stable by context and consider finite matrices L, R
 421 of dimension $P \times P$ and let C be a finite matrix of dimension $G \times G$ with $P \subseteq G$. Then,
 422 $R^* \prec L^*$ implies $(R^\uparrow + C)^* \prec (L^\uparrow + C)^*$.

423 **Proof.** If $R^* \prec L^*$, then, $(R^*)^\uparrow \prec (L^*)^\uparrow$ by Definition 19.2. By Lemma 1, it follows
 424 that $(R^\uparrow)^* \prec (L^\uparrow)^*$. Clearly, R^\uparrow and L^\uparrow are finite, and from Definition 19.1, we have
 425 $(R^\uparrow + C)^* \prec (L^\uparrow + C)^*$ ◀

426 ► **Theorem 21.** Let \preceq be a partial order stable by context. Suppose that for every rule
 427 $R = \langle P, \vec{c} \rangle$ with $P = \langle P_0, \vec{v} \rangle$ and P_0 the self-application of R , we have $(P_0)^* \prec (P_0)^*$. Then
 428 the corresponding GRS is terminating.

XX:12 On Termination of Graph Rewriting Systems through Language Theory

429 **Proof.** Let \preceq be a partial order on graphs and consider the corresponding order on matrices:
 430 $G \prec G'$ iff $M_G^* \prec M_{G'}^*$. We show that for every rule, we have $G \rightarrow G'$ implies $G' \prec G$. So let
 431 R be a graph rewriting rule and let μ be a morphism such that $G \rightarrow_{R,\mu} G'$. By the discussion
 432 in the beginning of Section 3, without loss of generality, we can suppose that μ is actually
 433 the inclusion of pattern P_0 in G . Now, let P_0 and P'_0 be respectively the basic pattern and the
 434 self-application of R . Define C to be the graph made of the nodes of G without edges in P_0 .
 435 By Proposition 6, $M_G = M_{P_0}^\uparrow + M_C$ and $M_{G'} = M_{P'_0}^\uparrow + M_C$. Moreover, $M_{P_0}, M_{P'_0}$ and M_C
 436 are finite and M_{P_0} is disjoint from M_C and $M_{P'_0}$ is disjoint from M_C . Thus, we can apply
 437 Lemma 20, and we get $M_{G'}^* = (M_{P'_0}^\uparrow + M_C)^* \prec (M_{P_0}^\uparrow + M_C)^* = M_G^*$. ◀

438 3.7 Stability of the orderings

439 We can now prove the two announced stability results.

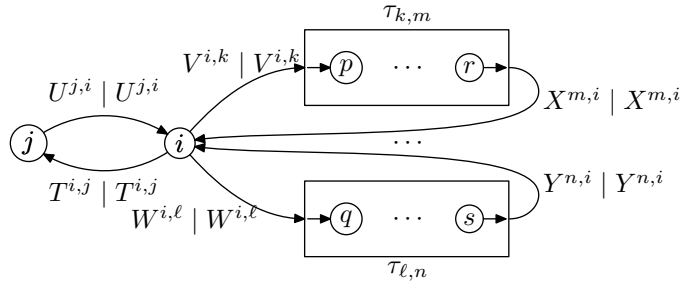
440 ▶ **Proposition 22.** *The multiset path ordering is stable by context.*

441 **Proof.** We first verify that condition 2 of Definition 19 holds. Suppose that $R \triangleleft L$ with
 442 R, L of dimension $P \times P$. Then, for all $(i, j) \notin P \times P$, $R_{i,j}^{\uparrow G} = 0 \leq 0 = L_{i,j}^{\uparrow G}$. Now, for all
 443 $k \geq |G| \geq |P|$ and for all $(i, j) \in P \times P$, we have $(R^{\uparrow G})_{i,j}^{\leq k} = R_{i,j}^{\leq k} \leq L_{i,j}^{\leq k} = (L^{\uparrow G})_{i,j}^{\leq k}$.

444 To verify that condition 1 also holds, let $G \times G$ be the dimension of L, R and C . Take
 445 $k \geq |G|$. We have on one side $(R + C)^{* \leq k} = \sum_{(A_1, \dots, A_\ell) \in \{R, C\}^*, \ell \leq k} \prod_{i \leq \ell} A_i$, and on the
 446 other side $(L + C)^{* \leq k} = \sum_{(A_1, \dots, A_\ell) \in \{R, C\}^*, \ell \leq k} \prod_{i \leq \ell} A_i \{R \leftarrow L\}$ where $A_i \{R \leftarrow L\} = L$ if
 447 $A_i = R$, and C otherwise. As the product and the addition are (strictly) monotonic, the
 448 result follows. ◀

449 ▶ **Proposition 23.** *The rational embedding order is stable by context.*

450 **Proof.** Since we use a component-wise ordering, it is easy to verify that condition 2 of
 451 Definition 19 holds. To verify that condition 1 also holds, let $G \times G$ be the shared dimension
 452 of L, R and C . Since $R < L$, there are decreasing transducers $\tau_{i,j} : L_{i,j} \rightarrow R_{i,j}$ with at least
 453 one of them deleting. Let P be the set of nodes corresponding to the pattern L . We build the
 454 family of transducers $(\theta_{p,q})_{p,q \in G \times G}$ as follows. The family of transducers will share the major
 455 part of the construction. First, we make a copy of all transducers $(\tau_{i,j})_{i,j}$. Then, we add as
 456 states all the nodes of C . Given an edge $i \xrightarrow{T} j \in C$, we set a transition $i \xrightarrow{T^{i,j} | T^{i,j}} j$. That is
 457 the transducer copies the paths within C . For a transition $i \xrightarrow{T} j$ with $i \notin P, j \in P$, we set
 458 the transitions: $i \xrightarrow{T^{i,j} | T^{i,j}} q_n$ for all q_n initial state of the transducer $\tau_{j,n}, n \in P$. Similarly,
 459 for any transition $i \xrightarrow{T} j$ with $i \in P, j \notin P$, we set the transitions: $r_n \xrightarrow{T^{i,j} | T^{i,j}} j$ for each
 460 terminal state r_n of the transducer $\tau_{n,i}, n \in P$. This construction can be represented as
 461 follows:



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

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

471 Consider some pair $p, q \in G$. We prove that the transducer $\theta_{p,q}$ is injective. Consider a
 472 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
 473 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).
 474 Consider a second word $w' = w'_1 \ell'_1 \cdots w'_k \ell'_k$ such that the transducer $\theta_{p,q}(w) = \theta_{p,q}(w') = u$.
 475 Given the construction of $\theta_{p,q}$, consider the word $u = u_1 r_1 \cdots u_k r_k$ with r_1, \dots, r_k some path
 476 within R . Indeed, only a letter within L can produce a letter within R . Consider the case
 477 where r_k is non empty. When the transducer reaches the first letter in ℓ_k , it is in a state
 478 $\tau_{k,m}$ for some m . Actually, $m = q$ since only $\tau_{k,q}$ contains a final state. Thus, the path is
 479 fixed within $\tau_{k,p}$ and then, the injection of $\tau_{k,p}$ applies. So, $\ell'_k = \ell_k$. We can go back within
 480 w_k . On this part of the word, the transitions have the shape $T^{i,j} | T^{i,j}$. Thus, $w_k = w'_k$.
 481 We can continue this process up to the beginning of w and w' . ◀

482 **4 Interpretations for Graph Rewriting Termination**

483 Interpretations methods are well known in the context of term rewriting, see for instance
 484 Dershowitz and Jouannaud's survey on rewriting [6]. Their usefulness comes from the fact
 485 that they belong to the class of simplification orderings, i.e., orderings for which if $t \leq u$,
 486 then $t \leq u$. In the context of graphs, we introduce a specific notion of "interpretation", that
 487 we will still call interpretation.

488 ▶ **Definition 24.** A graph interpretation is a triple $\langle X, \prec, \phi \rangle$ where $\langle X, \prec \rangle$ is a partially
 489 ordered set and $\phi : \mathcal{G} \rightarrow X$ is such that given two graphs P and P' having the same set of
 490 nodes and C disjoint of P and P' , if $\phi(P) \prec \phi(P')$, then $\phi(P + C) \prec \phi(P' + C)$.

491 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
 492 the basic pattern of R and P'_0 its self-application. Similarly, an interpretation is compatible
 493 with a GRS if it is compatible with all of its rules.

494 ▶ **Theorem 25.** Every GRS compatible with an interpretation Ω is terminating.

495 The theorem being a more abstract form of Theorem 21, its proof follows exactly the
 496 same steps.

497 **Proof.** Suppose that $G \prec G'$ iff $\phi(G) \prec \phi(G')$. We prove that for each rule R of the GRS,
 498 $G \rightarrow G'$ implies $G' \prec G$. Indeed, suppose that $G \rightarrow_{R,\mu} G'$. Let P_0 and P'_0 be respectively the
 499 basic pattern and the self-application of R . Then, there is a graph C such that $G = P_0 + C$,
 500 $G' = P'_0 + C$, such that P_0 and P'_0 are disjoint from C . Since $\phi(P'_0) \prec \phi(P_0)$, we then have
 501 $\phi(G') \prec \phi(G)$. ◀

502 ▶ **Example 26.** The triple $\langle \mathfrak{M}, \preceq, (M_{(-)})^* \rangle$ is an interpretation for 'Follow'.

503 ▶ **Example 27.** Let us come back to the weight analysis. Define $\bar{\omega}(G) = \sum_{p \xrightarrow{e} q \in G} \omega(e)$
 504 with $\omega(\alpha) = 0, \omega(T) = -1, \omega(\beta) = -1$. Then, $\langle \mathbb{R}, <, \bar{\omega}(-) \rangle$ is an interpretation for 'Init' and
 505 'End'.

506 ▶ **Example 28.** Let $\langle X_1, \prec_1, \phi_1 \rangle$ be an interpretation for a set of rules \mathcal{R}_1 , and let $\langle X_2, \prec_2, \phi_2 \rangle$
 507 be an interpretation for a set of rules \mathcal{R}_2 . Suppose that for every rule R in \mathcal{R}_2 , $G \rightarrow_{R,\mu} G'$
 508 implies $G' \preceq_1 G$ (that is without strict inequality). Then the lexicographic ordering on
 509 $X_1 \times X_2$ defined by $(x_1, x_2) \prec_{1,2} (y_1, y_2)$ iff $x_1 \prec_1 y_1$, or $x_1 \preceq_1 y_1$ and $x_2 \prec_2 y_2$, constitutes
 510 an interpretation $\langle X_1 \times X_2, \prec_{1,2}, \phi_1 \times \phi_2 \rangle$ for $\mathcal{R}_1 \cup \mathcal{R}_2$.

511 Thus, combining Example 26 and Example 27, we have a proof of the termination of our
 512 main Example.

513 ▶ **Corollary 29.** *The GRS given in Subsection 2.1 is terminating.*

514 ▶ **Example 30.** Let \mathcal{R} be a terminating GRS. Then there is an interpretation that “justifies”
 515 this fact. Indeed, take $\langle \mathcal{G}, \prec, 1_{\mathcal{G}} \rangle$ with \prec defined to be the transitive closure of the rewriting
 516 relation \rightarrow . The termination property ensures that the closure leads to an irreflexive relation.
 517 The compatibility of \prec with respect to $1_{\mathcal{G}}$ is immediate.

518 Thus the following corollary.

519 ▶ **Corollary 31.** *A GRS is terminating iff it is compatible with some interpretation.*

520 **5 Conclusion**

521 We proposed a new approach based on the theory of regular languages to decide the
 522 termination of graph rewriting systems, which does not account for node additions but settles
 523 the uniform termination problem for these GRS. We think that there is room to reconsider
 524 some old results of this theory under the new light. In particular, we think of profinite
 525 topology [19], is a powerful tool that could give us some insight on underlying structure of
 526 the orders. In the two cases, we can extend the orders to take into account orders on the
 527 edge labels.

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

533 **References**

-
- 534 1 Guillaume Bonfante and Bruno Guillaume. Non-simplifying graph rewriting termination. In
 535 *Proceedings 7th International Workshop on Computing with Terms and Graphs, TERMGRAPH*
 536 *2013, Rome, Italy, 23th March 2013*, pages 4–16, 2013.
- 537 2 Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. *Application of Graph Rewriting to*
 538 *Natural Language Processing*. Logic, Linguistic and Computer Science. Wiley, 2018.
- 539 3 N. Chomsky. *Syntactic Structures*. The Hague: Mouton, 1957.
- 540 4 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*.
 541 Cambridge University Press, 2012.
- 542 5 Nachum Dershowitz. A note on simplification orderings. *Information Processing Letters*, pages
 543 212–215, 1979.
- 544 6 Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of Theoretical*
 545 *Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. 1990.
- 546 7 Nachum Dershowitz and Jean-Pierre Jouannaud. Graph path orderings. In *LPAR-22. 22nd*
 547 *International Conference on Logic for Programming, Artificial Intelligence and Reasoning,*
 548 *Awassa, Ethiopia, 16-21 November 2018*, pages 307–325, 2018.

- 549 8 Nachum Dershowitz and Jean-Pierre Jouannaud. Drags: A compositional algebraic framework
550 for graph rewriting. *Theor. Comput. Sci.*, 777:204–231, 2019.
- 551 9 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun.*
552 *ACM*, 22(8), 1979.
- 553 10 Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving
554 termination of term rewriting. *Journal of Automated Reasoning*, 40(2):195–220, Mar 2008.
- 555 11 Maribel Fernández, Hélène Kirchner, and Bruno Pinaud. Strategic port graph rewriting: an
556 interactive modelling framework. *Mathematical Structures in Computer Science*, 29(5):615–662,
557 2019.
- 558 12 Bruno Guillaume and Guy Perrier. Dependency parsing with graph rewriting. In *Proceedings*
559 *of the 14th International Conference on Parsing Technologies, IWPT 2015, Bilbao, Spain, July*
560 *5-7, 2015*, pages 30–39, 2015.
- 561 13 Gerard Huet and Derek C. Oppen. Equations and rewrite rules: a survey. In *In formal*
562 *language Theory: perspective and open problems*. Academic Press, 1980.
- 563 14 Sylvain Kahane and François Lareau. Word ordering as a graph rewriting process. In *Formal*
564 *Grammar - 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015,*
565 *Revised Selected Papers. FG 2016, Bozen, Italy, August 2016, Proceedings*, volume 9804, pages
566 216–239. Springer, 2016.
- 567 15 Yves Lafont. Interaction nets. In *Conference Record of the Seventeenth Annual ACM Symposium*
568 *on Principles of Programming Languages, San Francisco, California, USA, January 1990*,
569 pages 95–108, 1990.
- 570 16 Jeannine Leguy. Transductions rationnelles décroissantes. *RAIRO. Informatique théorique*,
571 15(2):141–148, 1981.
- 572 17 Maurice Nivat. Transducteurs des langages de Chomsky. *Ann. Inst. Fourier, Grenoble*,
573 18:339–455, 1968.
- 574 18 Mizuhito Ogawa. A note on algebraic structure of tree decomposition of graphs. In *The First*
575 *Asian Workshop on Programming Languages and Systems, APLAS 2000, National University*
576 *of Singapore, Singapore, December 18-20, 2000, Proceedings*, pages 223–229, 2000.
- 577 19 Jean-Eric Pin. Profinite methods in automata theory. In *26th International Symposium*
578 *on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg,*
579 *Germany, Proceedings*, volume 3 of *LIPICs*, pages 31–50. Schloss Dagstuhl - Leibniz-Zentrum
580 fuer Informatik, Germany, 2009.
- 581 20 Detlef Plump. Simplification orders for term graph rewriting. In Igor Prívvara and Peter
582 Ružička, editors, *Mathematical Foundations of Computer Science 1997*, pages 458–467, Berlin,
583 Heidelberg, 1997. Springer Berlin Heidelberg.
- 584 21 Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Trans-*
585 *formation: Volume I. Foundations*. World Scientific Publishing Co., Inc., River Edge, NJ,
586 USA, 1997.
- 587 22 Jan Rutten. Behavioural differential equations: a coinductive calculus of streams, automata,
588 and power series. *Theoretical Computer Science*, 308(1):1 – 53, 2003.
- 589 23 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
- 590 24 Géraud Sénizergues. Some undecidable termination problems for semi-thue systems. *Theoretical*
591 *Computer Science*, 142:257–276, 1995.