Towards a Systematic Method for Proving Termination of Graph Transformation Systems

H. J. Sander Bruggink[1,2]

Institut für Informatik und Interaktive Systeme, Universität Duisburg–Essen
Duisburg, Germany

Abstract

We describe a method for proving the termination of graph transformation systems. The method is based on the fact that infinite reductions must include infinite 'creation chains', that is chains of edges in different graphs of the reduction sequence, such that each edge is involved in creating the next edge. In our approach, the length of such creation chains is recorded by associating with each edge label a creation depth, witch denotes the minimal length of a creation chain from an edge in the initial graph to that edge. We develop an algorithm which can prove the absence of such infinite chains (and therefore termination), analyse problems of the approach and propose possible solutions.

1 Introduction

Proving termination of graph transformation systems (GTSs) has applications in model transformation, program analysis and modelling dynamic systems. This has brought about emerging interest in finding termination techniques for GTSs [4,3,6,16,10].

In term and string rewriting, proving termination has historically attracted much more attention. Many general techniques for proving termination have been devised, even to the point that many of them can be used to automatically obtain termination proofs of specific rewrite systems. The most succesful of these techniques make use of the fact that infinite reduction sequences must have infinite 'creation chains', that is infinite chains of symbols $f_1, f_2, f_3, \ldots$ in different terms of the reduction sequence, such that each symbol of the sequence is involved in creating the next symbol [12,1,8,5]. Proving that such chains do not exist is often easier than proving termination of a rewrite system directly.

[1] Partially supported by DFG project SANDS.

[2] E-mail: sander.bruggink@uni-due.de

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Although termination techniques for term rewriting have been transferred to the area of term graph rewriting (see e.g. [13] for an overview), transferring proof techniques to the general form of graph rewriting is often not directly possible, because they make use of the inherent hierarchical nature of terms and strings. Still, we think it is a worthwile area of research to see how much of the theory can be ported to a graph rewriting setting.

In this paper we describe ongoing research to develop a method for proving the termination of GTSs based on the "absence of infinite creation chains" approach. We do not focus on proving non-existence of infinite reduction sequences starting from one particular source graph, such as most other termination results for GTSs, but instead on proving non-existence of infinite reduction sequences starting from arbitrary members of *infinite* classes of graphs, which we will call *regular graph languages*. Additionally, we describe an algorithm which can find termination proofs for some graph transformation systems automatically. Finally, we analyse problems of the approach and propose possible solutions.

2 Preliminaries

We work with edge-labelled directed graphs and employ the double pushout (DPO) approach to graph transformation. We refer to the standard literature for definitions and discussion, e.g. [7,15]. Here, we just recapitulate the most important concepts that are needed in some more detail later.

Given a set Σ of labels, a Σ-*graph* is given by $G = \langle V_G, E_G, \mathrm{src}_G, \mathrm{tgt}_G, \mathrm{lab}_G \rangle$, where: V_G is the set of *nodes*, E_G is the set of *edges*, $\mathrm{src}, \mathrm{tgt} : E_G \to V_G$ are resp. the *source* and *target* function, and $\mathrm{lab}_G : E_G \to \Sigma$ is the *labelling* function. We restrict our attention to *finite* graphs, i.e. we assume that V_G and E_G are finite sets.

Let a graph morphism be defined as usual. A Σ-production is then a pair of injective graph morphisms $L \leftarrow K \rightarrow R$, where L is called the left-hand side, R the right-hand side and K the interface of the production. Throughout the paper we assume that each production is *discrete* (i.e. K contains only nodes) and *edge-consuming* (i.e. L contains at least one edge). Graph productions are represented graphically by drawing the left-hand side and the right-hand side in which some nodes are named. The interface can be reconstructed by taking only the named nodes, and the morphisms will be just the identities.

We define a GTS to be a set of graph productions, i.e. we do not fix a start graph as usual. Instead, we define termination relative to a set of possible start graphs:

- $\mathcal{R}$ is terminating on a Σ-graph G, if no infinite reduction sequence from G exists. In this case, we also say that G is $\mathcal{R}$-terminating.
- $\mathcal{R}$ is terminating on a set of Σ-graphs $\mathbb{L}$, if it is terminating on each $G \in \mathbb{L}$. In this case, we also say that $\mathbb{L}$ is $\mathcal{R}$-terminating.
- $\mathcal{R}$ is terminating, if it is terminating on the set of all Σ-graphs.

3 Finite graph automata

3.1 Regular graph languages

The following notion of a finite graph automaton is a generalization of the notion of finite (string) automaton. If a string is encoded as a graph in the usual way, that is, as a linear graph with the string's symbols as edge labels, a finite automaton accepts the string if and only if a graph morphism exists from the string into the automaton (taking care that the first symbol of the string is mapped to an edge leaving a start state, and the last symbol to an edge entering an accepting state). This definition can be straightforwardly generalized to graphs in the following way:

Definition 3.1 A finite graph automaton (FGA) $\mathfrak{A}$ is a finite graph. A graph G is accepted by a finite graph automaton $\mathfrak{A}$ if there exists a morphism $f : G \to \mathfrak{A}$. The language accepted by a FGA $\mathfrak{A}$ is defined as:

$$\mathrm{L}(\mathfrak{A}) := \{G \mid \exists f : G \xrightarrow{f} \mathfrak{A}\}$$

FGAs $\mathfrak{A}$ and $\mathfrak{B}$ are *equivalent* if $\mathrm{L}(\mathfrak{A}) = \mathrm{L}(\mathfrak{B})$.

A set of graphs $\mathbb{L}$ is a *regular graph language* if $\mathbb{L} = \mathrm{L}(\mathfrak{A})$ for some FGA $\mathfrak{A}$.

We will use fraktur uppercase letters for FGAs, while we use 'normal' uppercase letters for other graphs.

The class of regular graph languages as defined here is incomparable to the class of graph languages which can be recognized by context free (edge replacement) graph grammars [9]. On the one hand, the language of *all* graphs over a certain label set is regular because it is accepted by the final graph over that label set; it is however not context free (cf. Theorems IV.3.3, IV.3.4 and IV.3.6 of [9]). On the other hand, the language of all finite trees is context free but not regular.

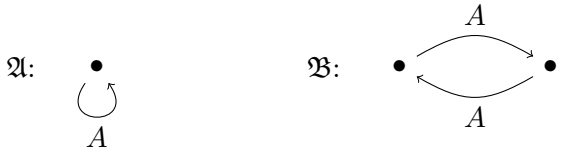
The definition of finite graph automaton is similar to the definition of a type graph: a regular graph language is the set of graphs of a specific type. However, where a type graph is usually meant to be a static structure which is set up in advance, our aim is the other way around: given an untyped graph transformation system, we want to generate an automaton which accepts all the reachable graphs. For this reason, we prefer to maintain different terminology.

3.2 Minimization of finite graph automata

Lemma 3.2 *Let two* FGA*s* $\mathfrak{A}$ *and* $\mathfrak{B}$ *be given.* $\mathfrak{A}$ *and* $\mathfrak{B}$ *are equivalent if and only if there exists a morphism* $\mathfrak{A} \to \mathfrak{B}$ *and a morhpism* $\mathfrak{B} \to \mathfrak{A}$*.*

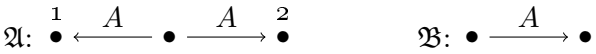
Proof. Directly from the definitions and the fact that the composition of two morphisms is a morphism again. □

Example 3.3 Consider the following FGAs:



In many notions of equivalence of graphs the (nodes of the) above two FGAs are considered equivalent because they represent the same (regular) tree; see e.g. [11]. However, here they are not equivalent because they do not accept the same regular graph language: the A-loop is accepted by $\mathfrak{A}$ but not by $\mathfrak{B}$.

Example 3.4 Consider the following FGAs:



$\mathfrak{A}$ and $\mathfrak{B}$ are equivalent (accept the same graph language) because there exists a morphism from $\mathfrak{A}$ to $\mathfrak{B}$ (so every graph in $\mathrm{L}(\mathfrak{A})$ is also in $\mathrm{L}(\mathfrak{B})$) and also a morphism from $\mathfrak{B}$ to $\mathfrak{A}$ (so every graph in $\mathrm{L}(\mathfrak{B})$ is also in $\mathrm{L}(\mathfrak{A})$).

In Example 3.4, the left automaton is clearly more "complex" than the right one: either the left-most or the right-most node of the left graph (marked 1 and 2 in the figure, respectively) is superfluous: for every morphism which maps a node to node 1, there is an alternative morphism which maps the same node to 2. So, node 1 and the edge leading to it could be removed without affecting the accepted graph language. This observation gives rise to the question of whether or not we can find a *minimal* FGA among an equivalence class of FGAs. We define a minimal FGA to be an FGA which does not contain redundancy in the way illustrated above.

Definition 3.5 A FGA $\mathfrak{A}$ is *minimal* if every morphism $f : \mathfrak{A} \to \mathfrak{A}$ is an isomorphism, i.e. $\mathfrak{A}$ has no non-trivial automorphisms.

Definition 3.6 Let a FGA over the signature Σ, $\mathfrak{A} = \langle V, E, \mathrm{src}, \mathrm{tgt}, \mathrm{lab}\rangle$, and an automorphism $f : \mathfrak{A} \to \mathfrak{A}$ be given. The *lessening* of $\mathfrak{A}$ over f is defined as follows:

$$\mathsf{less}_f(\mathfrak{A}) = \langle V', E', \mathrm{src}', \mathrm{tgt}', \mathrm{lab}'\rangle$$

where:

- $V' = V \restriction \mathcal{R}ng(f)$
- $E' = E \restriction \mathcal{R}ng(f)$
- $\mathrm{src}' = \mathrm{src} \restriction (V' \times E')$
- $\mathrm{tgt}' = \mathrm{tgt} \restriction (V' \times E')$
- $\mathrm{lab}' = \mathrm{lab} \restriction (E' \times \Sigma)$

Lemma 3.7 *Let $\mathfrak{A}$ be an* FGA.

(i) There exists a morphism from $\mathfrak{A}$ to $\mathsf{less}_f(\mathfrak{A})$.

(ii) There exists a morphism from $\mathsf{less}_f(\mathfrak{A})$ *to* $\mathfrak{A}$.

(iii) $\mathrm{L}(\mathfrak{A}) = \mathrm{L}(\mathsf{less}_f(\mathfrak{A}))$.

(iv) $\mathfrak{A} = \mathsf{less}_f(\mathfrak{A})$ *if and only* f *is an isomorphism.*

Proof. (i) Because the nodes and edges of $\mathsf{less}_f(\mathfrak{A})$ are exactly the ones which are in the domain of f, f itself functions as a morphism from $\mathfrak{A}$ to $\mathsf{less}_f(\mathfrak{A})$.

(ii) By construction, $\mathsf{less}_f(\mathfrak{A})$ is a subgraph of $\mathfrak{A}$, and therefore the identity functions as a morphism from $\mathsf{less}_f(\mathfrak{A})$ to $\mathfrak{A}$.

(iii) Suppose $g : G \to \mathfrak{A}$. Then by (i), $(g \,;\, f) : G \to \mathsf{less}_f(\mathfrak{A})$. Inversely, suppose $g : G \to \mathsf{less}_f(\mathfrak{A})$. Then by (ii), $(g \,;\, \mathsf{id}) : G \to \mathfrak{A}$.

(iv) Because the domain and range of automorphisms are the same (and finite), surjectiveness coincides with injectiveness and thus with being an isomorphism. By construction, f is surjective if and only if $\mathfrak{A} = \mathsf{less}_f(\mathfrak{A})$. □

In view of the above Lemma, it is easy to construct an equivalent minimal FGA from a given one. We now show that every equivalence class of FGAs has a unique (up to isomorphism) minimal element:

Lemma 3.8 *If* $\mathfrak{M}$ *and* $\mathfrak{N}$ *are minimal* FGA*s such that* $\mathrm{L}(\mathfrak{M}) = \mathrm{L}(\mathfrak{N})$*, then* $\mathfrak{M}$ *and* $\mathfrak{N}$ *are isomorphic.*

Proof. Since $\mathfrak{M}$ and $\mathfrak{N}$ accept the same language, there must be morphisms $f : \mathfrak{M} \to \mathfrak{N}$ and $g : \mathfrak{N} \to \mathfrak{M}$. By definition of minimality, $(f \,;\, g)$ and $(g \,;\, f)$ are isomorphisms. From this it follows that f and g are isomorphisms as well. □

4 Annotating GTSs with creation heights

We record *creation heights* of edges by annotating the edges of a graph with a natural number representing the creation height. The creation heights of the edges in the source graph of a reduction sequence are initialized to 0, and in each step the creation height of edges in the target of the step is equal to the least creation height of the edges involved in creating it plus one. Thus, the creation height of an edge represents the *minimal* length of the 'creation chains' to that edge. The absence of infinite creation chains is now equivalent to the creation heights in a reduction being bounded by some natural number.

First we define functions which translate annotated and non-annotated graph into each other: $\mathrm{lift}_n(G)$ annotates each edge of G with the creation height n, while $\mathrm{proj}(G)$ removes all annotations from the annotated graph G.

Definition 4.1 Let Σ be a set of labels. We define the maps $\mathrm{proj}(\cdot)$ and $\mathrm{lift}_n(\cdot)$ between Σ-graphs and $(\Sigma \times \mathbb{N})$-graphs as follows:

- Let G be a Σ-graph. We define: $\mathrm{lift}_n(G) := \langle V_G, E_G, \mathrm{src}_G, \mathrm{tgt}_G, \mathrm{lab}' \rangle$ where $\mathrm{lab}'(x) := \langle \mathrm{lab}(x), n \rangle$.
- Let G be a $(\Sigma \times \mathbb{N})$-graph. We define: $\mathrm{proj}(G) := \langle V_G, E_G, \mathrm{src}_G, \mathrm{tgt}_G, \mathrm{lab}_G \,;\, \pi_1 \rangle$ where π_1 is the projection function: $\pi_1(\langle x, y \rangle) := x$.

Definition 4.2 The set of heights of a $(\Sigma \times \mathbb{N})$-graph G is defined as:

$$\mathrm{hts}(G) := \{n \mid \langle l, n\rangle \in \mathcal{R}ng(\mathrm{lab}_G)\}$$

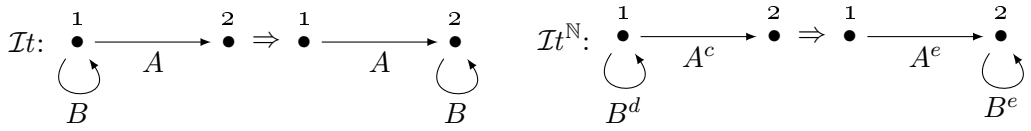
where for a function f we define $\mathcal{R}ng(f)$ to be the range of f, i.e.: $\mathcal{R}ng(f) := \{b \in B \mid f(a) = b \text{ for some } a \in A\}$. The set of heights of a reduction sequence of $(\Sigma \times \mathbb{N})$-graphs $\rho = G_0 \Rightarrow G_1 \Rightarrow G_2 \Rightarrow \cdots$ is defined as: $\mathrm{hts}(\rho) := \bigcup_{0 \le i \le |\rho|} \mathrm{hts}(G_i)$.

In the following, the set Σ of labels will be fixed. $(\Sigma \times \mathbb{N})$-graphs will be called *annotated* graphs. A label $\langle l, n\rangle$ of an annotated graph will be denoted by l^n. The same convention is used for productions, steps, reductions, etc.

Definition 4.3

- An annotated graph $G^\star$ is called an *annotation* of a graph G, if $\mathrm{proj}(G^\star) = G$. The set of annotations of a graph G is denoted by $\mathcal{A}nn(G)$.
- An annotated graph production $p^\star : L^\star \stackrel{l}{\leftarrow} K^\star \stackrel{r}{\rightarrow} R^\star$ is an *annotation* of a production $p : L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ if
  · $\mathrm{proj}(L^\star) = L$
  · $R^\star = \mathrm{lift}_c(R)$, where $c = \min(\mathrm{hts}(L^*)) + 1$ [3], and
  · $K^\star = K$.
  (Note that the assumption that all productions are discrete assures that l and r are morphisms from $K^\star$ to $L^\star$ and from $K^\star$ to $R^\star$, resp.)
  The set of annotations of a graph production p is denoted by $\mathcal{A}nn(p)$.
- If $\mathcal{R}$ is a GTS, then its annotation $\mathcal{R}^{\mathbb{N}}$ is defined as: $\mathcal{R}^{\mathbb{N}} = \bigcup_{p \in \mathcal{R}} \mathcal{A}nn(p)$.

Example 4.4 The annotation of the GTS $\mathcal{I}t$ on the left, consists of all rules of the form on the right, where $e = \min\{c, d\} + 1$.



The creation heights of the right-hand side are all strictly larger than the lowest creation height of the left-hand side.

Note that the annotation $\mathcal{R}^{\mathbb{N}}$ of a GTS $\mathcal{R}$ contains infinitely many rules. This is no problem in either theory or practice. Given an annotated graph $G^\star$ and a morphism m from the left-hand side of a (non-annotated) production $p : L \leftarrow K \rightarrow R$ into $\mathrm{proj}(G^\star)$, we can construct the unique annotated production $p^\star : L^\star \leftarrow K^\star \rightarrow R^\star$ such that m is a morphism of $L^\star$ into $G^\star$. In other words, the technique of this paper can be implemented by creating the needed annotated rules on the fly.

The following proposition follows from the fact that, for each L^*, there is a *unique* R^* such that $L^* \leftarrow K^* \rightarrow R^*$ is an annotation.

Proposition 4.5 *Let $\mathcal{R}$ be a* GTS *and $\mathcal{R}^{\mathbb{N}}$ its annotation.*

[3] Alternatively, we could define $c = \max(\mathrm{hts}(L^*)) + 1$. In this case, all the results of Sect. 4 would still hold. However, with *min* we obtain a stronger termination criterion in Sect. 5.

For each $\mathcal{R}$-step $G \Rightarrow_{p,m} H$ and annotated graph $G^\star \in \mathcal{A}nn(G)$, there exists a unique $\mathcal{R}^\mathbb{N}$-step $G^\star \Rightarrow_{p^\star,m} H^\star$ such that $p^\star \in \mathcal{A}nn(p)$. Moreover, it holds that $H^\star \in \mathcal{A}nn(H)$.

It follows from Prop. 4.5 that for each reduction sequence $\rho = G_0 \Rightarrow G_1 \Rightarrow \cdots$ there exists a unique annotated reduction sequence $\rho^\star = G_0^\star \Rightarrow G_1^\star \Rightarrow \cdots$ such that $G_0^\star = \mathrm{lift}_0(G_0)$ and $\mathrm{proj}(G_i^\star) = G_i$ for each G_i in the reduction sequence. In the following, this reduction sequence will be called the *canonical annotation* of ρ, and denoted by $\rho^\mathbb{N}$. In the following, we will use $\mathrm{hts}(\rho)$ as a synonym for $\mathrm{hts}(\rho^\mathbb{N})$.

We can now show that a reduction sequence is infinite if and only if there is no bound on its creation heights. The property depends on the assumption that at least one annotation (in particular the minimal one) is replaced by an (arbitrary amount of) strictly bigger annotations, which is assured by the assumptions that interfaces are discrete and each left-hand side contains at least one label.

Theorem 4.6 *A reduction sequence ρ is finite if and only if there is a $b \in \mathbb{N}$ such that for every $c \in \mathrm{hts}(\rho)$ it holds that $c < b$.*

Proof. ($\Rightarrow$) Trivial.

($\Leftarrow$) Assume that b is the bound on the creation heights of ρ. We associate with each graph in the reduction sequence an array $x_0, \ldots, x_b$ of natural numbers, where x_c denotes the number of edges in the graph that have annotation c. These sequences can be ordered lexicographically, yielding a well-founded ordering on such arrays. According to this ordering the graphs strictly decrease in every step of the sequence, therefore the sequence is finite. □

5 Proving termination

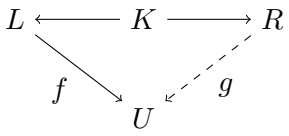
In this section we describe a method to prove termination of graph rewriting. In particular, we are interested in the question of whether or not a GTS is terminating on a regular graph language. However, for ease of presentation, we first consider the weaker question of whether or not an infinite reduction sequence exists from a single graph. It will turn out that the technique developed for graphs is easily extended to regular graph languages.

5.1 Termination on a single graph

In order to find out whether or not there is a bound on the creation heights of the graphs reachable from a specific graph, we use the notion of *unwinding*, which is essentially a simpler form of unfolding [2]: it encodes in an economical way the graphs which are reachable from a specific graph, so that some properties of the GTS (combined with the start graph) can be read from it.

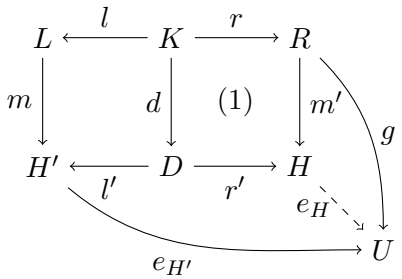
Definition 5.1 Let $\mathcal{R}$ be a GTS. A graph U is a $\mathcal{R}$-*unwinding* of a graph G, if there is a morphism $h : G \to U$, and for each production $L \leftarrow K \rightarrow R$ and morphism $f : L \to U$, there exists a morphism $g : R \to U$ such that the following diagram

commutes:



Lemma 5.2 *Let a* GTS $\mathcal{R}$ *and a graph* G *be given, and let* U *be a* $\mathcal{R}$*-unwinding of* G*. For each graph* H *such that* $G \Rightarrow^* H$ *there exists a morphism* $e_H : H \to U$*.*

Proof. By induction on the length of the reduction sequence $G \Rightarrow^* H$. If the length of the reduction is 0, then the lemma follows immediately. Otherwise, $G \Rightarrow^* H' \Rightarrow H$, where a production $p : L \leftarrow K \to R$ is responsible for the last step. By definition of unwinding, there exists a morphism $g : R \to U$, and by the induction hypothesis, there exists a morphism $e_{H'} : H' \to U$, so we have the following situation:



The existence of the dashed morphism e_H follows from the morphisms $(l' \,;\, e_{H'})$ and g, and the fact that the right square (marked (1) above) is a pushout by definition.□

Corollary 5.3 *Let a* GTS $\mathcal{R}$ *and a graph* G *be given. Furthermore, let* $U^\star$ *be a* $\mathcal{R}^\mathbb{N}$*-unwinding of* $\mathrm{lift}_0(G)$*.* G *is* $\mathcal{R}$*-terminating if and only if there is a bound on the creation heights of* $U^\star$*.*

Proof. Suppose there exists an infinite $\mathcal{R}$-reduction sequence from G. Then there exists an infinite $\mathcal{R}^\mathbb{N}$-reduction sequence from $\mathrm{lift}_0(G)$ by Prop. 4.5. By Theorem 4.6, the creation heights in this reduction sequence are unbounded, and therefore by Lemma 5.2 the creation heights of $U^\star$ are unbounded. □

This provides us with the following systematic method for proving that, given a GTS $\mathcal{R}$, no infinite $\mathcal{R}$-reduction sequence exists from a given graph G:

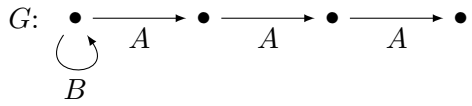
(i) "Construct" the annotation $\mathcal{R}^\mathbb{N}$ of $\mathcal{R}$.

(ii) Construct a $\mathcal{R}^\mathbb{N}$-unwinding U of $\mathrm{lift}_0(G)$. Constructing an unwinding will be the topic of Sect. 6.

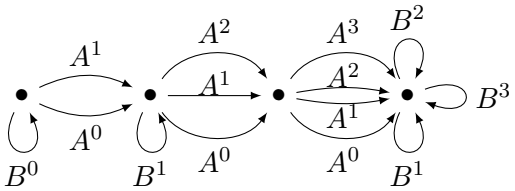
(iii) G is $\mathcal{R}$-terminating if and only if there is a bound on the creation heights of U. In practice, we will use the fact that a *finite* unwinding always has a bound on the creation heights.

Note that the method is not be total (it can't be, because termination of GTSs is non-decidable in general [14]), in particular the second step will loop forever if the GTS does not terminate.

Example 5.4 Consider the one-production GTS $\mathcal{I}t$ from Ex. 4.4 and the following graph G:



An unwinding of $\mathrm{lift}_0(G)$ is the following annotated graph:



The $\mathcal{I}t$-unwinding is finite, and thus the creation heights are bounded, so we conclude that no infinite $\mathcal{I}t$-reduction exists from G.

5.2 Termination on a regular graph language

Because of the way we have set things up, the results of the previous section are easily generalized from proving termination of a single graph to proving termination of a regular graph language. The following auxiliary result, which follows from the fact that the composition of two morphisms is a morphism again, is responsible for this.

Lemma 5.5 *Let $\mathcal{R}$ be a* GTS*, $\mathfrak{A}$ a* FGA *and $G \in \mathrm{L}(\mathfrak{A})$ a graph. Every $\mathcal{R}$-unwinding of $\mathfrak{A}$ is an $\mathcal{R}$-unwinding of G.*

From this lemma and the results of the previous section, the following result can be easily proved:

Lemma 5.6 *Let $\mathcal{R}$ be a* GTS*, $\mathfrak{A}$ be a* FGA *and $\mathfrak{U}$ be a $\mathcal{R}$-unwinding of $\mathfrak{A}$. For each $G \in \mathrm{L}(\mathfrak{A})$ and H such that $G \Rightarrow^* H$, it holds that there is a morphism $f : H \to \mathfrak{U}$.*

It is easy to see that, for an FGA $\mathfrak{A}$, we have $\mathrm{L}(\mathrm{lift}_0(\mathfrak{A})) = \{\mathrm{lift}_0(G) \mid G \in \mathrm{L}(\mathfrak{A})\}$, and thus we have the following corollary:

Corollary 5.7 *Let a* GTS *$\mathcal{R}$ and a* FGA *$\mathfrak{A}$ be given. Furthermore, let $\mathfrak{U}^\star$ be a $\mathcal{R}^{\mathbb{N}}$-unwinding of $\mathrm{lift}_0(\mathfrak{A})$. $\mathrm{L}(\mathfrak{A})$ is $\mathcal{R}$-terminating if and only if there is a bound on the creation heights of $\mathfrak{U}^\star$.*

This provides us with the following systematic method for proving that, given a GTS $\mathcal{R}$ and a FGA $\mathfrak{A}$, no infinite reduction sequence exists from any member of $\mathrm{L}(\mathfrak{A})$:

(i) Construct the annotation $\mathcal{R}^{\mathbb{N}}$ of $\mathcal{R}$.

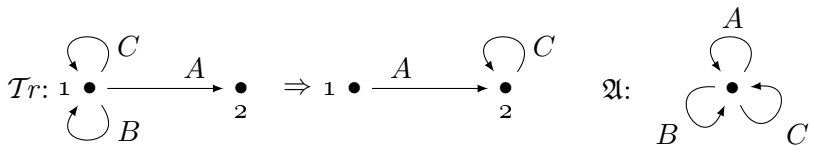
(ii) Construct an $\mathcal{R}^{\mathbb{N}}$-unwinding $\mathfrak{U}^\star$ of $\mathrm{lift}_0(\mathfrak{A})$.

(iii) $\mathrm{L}(\mathfrak{A})$ is $\mathcal{R}$-terminating if and only if there is a bound on the creation heights of $\mathfrak{U}^\star$.

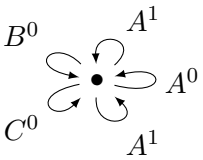
In the next section we will discuss an algorithm to construct an unwinding of a graph or FGA.

Example 5.8 Consider the FGA $\mathfrak{G} := G$, where G is the graph from Ex. 5.4, and let the GTS $\mathcal{I}t$ from the same example be given. The language $\mathrm{L}(\mathfrak{G})$ consists of all acyclic graphs with maximal path length 3, and because there is a bound on the creation heights in an unwinding of $\mathrm{lift}_0(\mathfrak{G})$, we conclude that $\mathrm{L}(\mathfrak{G})$ is $\mathcal{I}t$-terminating.

Example 5.9 Let the following GTS $\mathcal{T}r$ and FGA $\mathfrak{A}$ be given:



Note that $\mathrm{L}(\mathfrak{A})$ contains *all* graphs over the signature A, B, C. An unwinding of $\mathrm{lift}_0(\mathfrak{A})$ is the following:



Since the creation heights in the unwinding are bounded (because it is finite) we conclude that $\mathcal{T}r$ is terminating on the class of all $\{A, B, C\}$-graphs.

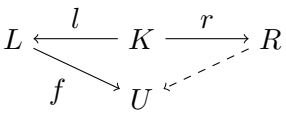
Note that in Ex. 5.8 and Ex. 5.9, we show termination of an *infinite* class of graphs, in the second case even of the class of *all* graphs (over the signature). Many other termination techniques for GTSs, on the other hand, focus on proving termination of a single source graph.

6 Constructing unwindings of graphs

The method described above depends on constructing an unwinding of the initial annotation of a graph. Since this unwinding is necessarily infinite if there is no bound on the creation heights, we cannot hope to find an algorithm which always terminates. However, usually the goal is to prove *termination* rather than *non-termination*, and thus a semi-decision procedure is already useful. We analyse the following algorithm:

Algorithm 1

***given**: a tuple $\langle \mathcal{R}, G \rangle$, where $\mathcal{R}$ is a GTS and G a graph;*
$U := G$;
do
$\mathcal{S} := \{(L \leftarrow K \rightarrow R) \in \mathcal{R} \mid$
there exists a morphism $f : L \to U$,
but there is no morphism $R \to U$ such that
the following diagram commutes:



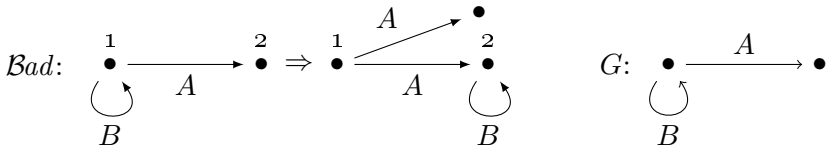
$\}$;
if** $\mathcal{S} \neq \varnothing$ **then

$(L \leftarrow K \rightarrow R) :=$ *an arbitrary element of* $\mathcal{S}$ *;*
Construct the following pushout:

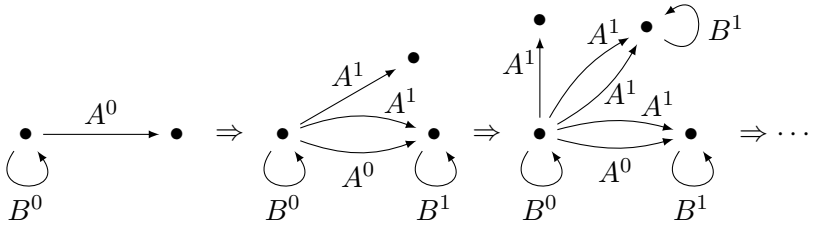
$$\begin{array}{ccc} K & \xrightarrow{r} & R \\ {\scriptstyle l}\downarrow & & \downarrow \\ L & & \\ {\scriptstyle f}\downarrow & & \\ U & \longrightarrow & U' \end{array}$$

$U := U'$*;*
endif*;*
until $\mathcal{S} = \varnothing$*;*
output*:* U*;*

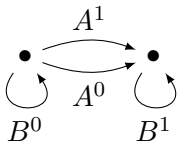
It is easy to show that if Algorithm 1 terminates on input $\langle \mathcal{R}, G \rangle$, then its output is an $\mathcal{R}$-unwinding of G. For example, the unwindings of Ex. 5.4 and Ex. 5.9 can be found by running the algorithm. However, the algorithm does not always terminate, not even in some cases in which a finite unwinding exists. Consider as a counter-example the following one-production GTS $\mathcal{B}ad$ and graph G:



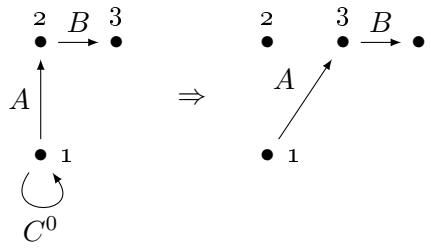
The GTS $\mathcal{B}ad$ is obviously terminating on finite graphs without A-cycles, or even more generally, on graphs without infinite A-paths. However, the following is now a run of the algorithm (starting from the initial annotation $\mathrm{lift}_0(G)$), which increases the number of A^1-edges in every step:



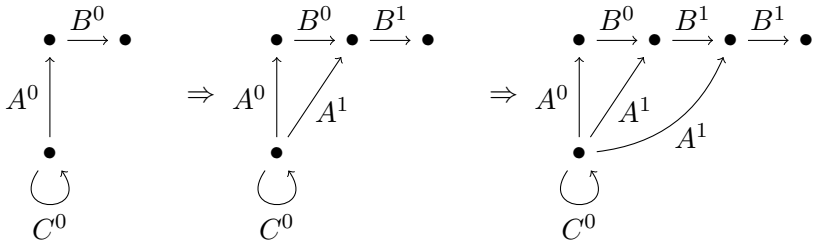
Note that the bound on the creation heights in graphs of the sequence (except the first) is 2. In the example above, the solution is to include a minimization step in the *while*-loop. In this case the second graph in the above sequence would be minimized to:



which is a finite unwinding of the initial annotation of G. However, minimization will *not* work in general. Consider the following GTS:



The following is (the beginning of) an infinite run of the algorithm with minimization:



Note that all graphs of the sequence are minimal, but that the highest creation height obtained is 1.

7 Related Work

As mentioned in the introduction, there exist several other results in the area of GTSs. Many of the results, however, focus on proving termination from a single source graph, whereas our method proves termination from an infinite class of source graphs.

The most related termination technique is the approximation-based approach by Varró et al. [16]. Both results are not complete, in the sense that terminatiing GTSs exist for which the technique cannot prove termination. The approximations of [16] ignores the structure of the graph, and focusses on the number of occurrences of elements of certain type. In contrast, our approach can take the structure of the source graph(s) into account.

A more direct comparison of the two methods, for example with case studies, would be interesting, but is outside the scope of this paper.

8 Conclusion and future work

We describe a method to prove termination of GTS. The method works by showing that a reduction does not contain infinite creation chains, or equivalently, that the creation heights in a reduction are bounded by a natural number. This is done by encoding the graphs which are reachable from a given regular graph language in a so-called unwinding.

As it stands, the algorithm which constructs the unwinding does not always terminate, not even in the case of some reasobly simple *terminating* GTSs. However, restricting the algorithm to a certain number of steps, does yield an algorithm which either proves termination, or results in an "unkown" answer. Such an algorithm can be useful in praxis.

Also, the method shows termination of *infinite* classes of graphs. Most other termination results for GTSs focus on a single source graph.

For the above reasons, we feel the method is promising to investigate further. Possible improvements and ideas for further research include:

- *Obtaining similar results for more expressive classes of graph languages.* Regular graph languages are not expressive enough to recognize the class of acyclic finite graphs or even the class of finite trees, while non-cyclicity is can be an important property for proving termination. Combining creation heights with more expressive classes of graph languages may provide stronger results.
- *Extending the method to* GTS*s with negative application conditions.* This is a non-trivial extension because, where the creation heights of edges depend on other edges which are present, negative application conditions typically allow certain transformation steps only in the case that something is *not* present.
- *Extending the method also to non-termination proving.* It is also interesting to see whether conditions can be found on the basis of which non-termination can be concluded.

References

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

[2] Paolo Baldan. *Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 2000.

[3] Paolo Bottoni, Kathrin Hoffman, Francesco Parisi Presicce, and Gabriele Taentzer. High-level replacement units and their termination properties. *Journal of Visual Languages and Computing*, 2005.

[4] Paolo Bottoni, Manuel Koch, Francesco Parisi-Presicce, and Gabriele Taentzer. Termination of high-level replacement units with application to model transformation. *ENTCS*, 127, 2005.

[5] H. J. Sander Bruggink. A proof of finite family developments for higher-order rewriting using a prefix property. In *Proceedings of RTA '06*. Springer, 2006.

[6] Hartmut Ehrig, Karsten Ehrig, Juan de Lara, Gabriele Taentzer, Dániel Varró, and Szilvia Varró-Gyapay. Termination criteria for model transformation. In *Proceedings of FASE '05*. Springer, 2005.

[7] Hartmut Ehrig, Michael Pfender, and Hans Jürgen Schneider. Graph-grammars: An algebraic approach. In *FOCS*, 1973.

[8] Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-bounded string rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 15(3–4):149–171, 2004.

[9] Annegret Habel. *Hyperedge Replacement: Grammars and Languages.* Springer, 1992.

[10] Tihamér Levendovszky, Ulrike Prange, and Hartmut Ehrig. Termination criteria for DPO transformations with injective matches. In *Proceedings of the GT-VC workshop at CONCUR '06*. Elsevier, 2007. To appear.

[11] Laurent Mauborgne. An incremental unique representation for regular tree. *Nordic Journal of Computing*, 7(4):290–311, 2000.

[12] Vincent van Oostrom. Finite family developments. In *Proceedings of RTA '97*. Springer, 1997.

[13] Detlef Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars II*. World Scientific, 1997.

[14] Detlef Plump. Termination of graph rewriting is undecidable. *Fundementa Informaticae*, 33(2):201–209, 1998.

[15] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars*. World Scientific, 1997.

[16] Dániel Varró, Szilvia Varró-Gyapay, Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Termination analysis of model transformations by Petri Nets. In *Proceedings of ICGT '06*. Springer, 2006.