Fundamenta Informaticae 33(2), 1998, 201–209
IOS Press

Termination of graph rewriting is undecidable[*]

Detlef Plump
Fachbereich Mathematik und Informatik
Universität Bremen
28334 Bremen, Germany
det@informatik.uni-bremen.de

Abstract. It is shown that it is undecidable in general whether a graph rewriting system (in the "double pushout approach") is terminating. The proof is by a reduction of the Post Correspondence Problem. It is also argued that there is no straightforward reduction of the halting problem for Turing machines or of the termination problem for string rewriting systems to the present problem.

Keywords: graph rewriting, termination, Post Correspondence Problem

1. Introduction

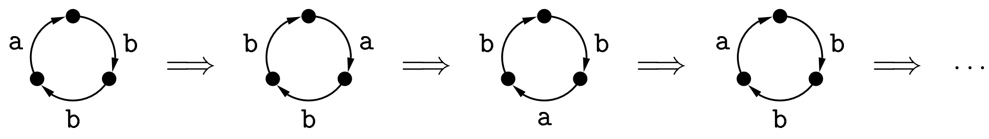
In 1978, Huet and Lankford showed that it is undecidable in general whether a term rewriting system is terminating, that is, whether every computation of a system eventually halts [4]. Moreover, their proof implies that termination of string rewriting systems is undecidable. In the present paper, it is shown that termination of graph rewriting systems—in the so-called double pushout approach [2, 1]—is undecidable, too.

Huet and Lankford simulated Turing machines by term rewriting systems such that a given machine halts on all inputs if and only if the corresponding term rewriting system terminates for all terms. Thus, they obtained a reduction of the uniform halting problem for Turing machines to the termination problem for term rewriting systems. In the framework of graph rewriting, however, a straightforward reduction of the halting problem or of the termination problem for string rewriting is not possible. In both cases, the problem is caused by cyclic graphs. Consider, for instance, the string rewrite rule `ab` $\to$ `ba` which may be translated into the following graph rewrite rule for edge-labelled graphs:

$$\begin{tikzcd}
\bullet \arrow[r, "\mathtt{a}"] & \bullet \arrow[r, "\mathtt{b}"] & \bullet & \Longrightarrow & \bullet \arrow[r, "\mathtt{b}"] & \bullet \arrow[r, "\mathtt{a}"] & \bullet
\end{tikzcd}$$

Clearly, the string rewriting system $\{$`ab` $\to$ `ba`$\}$ is terminating, that is, its rule can be applied to a string only finitely often. In contrast, the graph rewriting system consisting of the above rule does not terminate as it admits the following infinite rewrite sequence of cyclic graphs:

[*]Part of this research was performed while the author was on leave at CWI, Amsterdam, supported by a grant of the HCM network EXPRESS.



The same problem arises when Turing machines are simulated by graph rewriting. An example is a Turing machine doing nothing else than moving its head one cell to the right after reading the symbol a. This machine obviously terminates for all input strings. But when translated into a graph rewriting system (analogously to the translation of string rewriting systems sketched above), a cyclic "tape" labelled with a's will issue an infinite rewriting.

The halting problem for Turing machines easily can be reduced to the termination problem for graph rewriting *on well-formed graphs* representing Turing machine configurations. Likewise, there is a straightforward reduction of the termination problem for string rewriting to the termination problem for graph rewriting on "string graphs". Undecidability of termination for these restricted graph classes, however, does not imply that termination is undecidable for general graph rewriting, that is, for graph rewriting on arbitrary graphs.

Therefore, in this paper, the Post Correspondence Problem (PCP) is reduced to the problem of deciding whether a given graph rewriting system is terminating. The basic idea of the encoding of the PCP is similar to the idea behind reductions of the PCP to the termination problem for term rewriting systems, as given by Lescanne [5], Zantema [7] and Ferreira [3]. But the encoding by graph rewriting is more involved as the non-linear rules used in those papers do not have counterparts in graph rewriting systems.

The rest of this paper is organized as follows: Section 2 contains a brief review of graph rewriting systems, in Section 3 the main result is proved, and Section 4 concludes by stating an undecidability result following from the proof of the main result.

2. Graph rewriting

Below the double pushout approach to graph rewriting is briefly reviewed. For a comprehensive survey, the reader may consult [1] or [2].

A *label alphabet* $\Sigma = \langle \Sigma_{\mathrm{V}}, \Sigma_{\mathrm{E}} \rangle$ is a pair of finite sets of *vertex labels* and *edge labels*. A *graph* over Σ is a system $G = \langle \mathrm{V}_G, \mathrm{E}_G, \mathrm{s}_G, \mathrm{t}_G, \mathrm{l}_G, \mathrm{m}_G \rangle$ consisting of two finite sets V_G and E_G of *vertices* (or *nodes*) and *edges*, two *source* and *target functions* $\mathrm{s}_G, \mathrm{t}_G \colon \mathrm{E}_G \to \mathrm{V}_G$, and two *labelling functions* $\mathrm{l}_G \colon \mathrm{V}_G \to \Sigma_{\mathrm{V}}$ and $\mathrm{m}_G \colon \mathrm{E}_G \to \Sigma_{\mathrm{E}}$.

Given two graphs G and H, G is a *subgraph* of H, denoted by $G \subseteq H$, if V_G and E_G are subsets of V_H and E_H, respectively, and if s_G, t_G, l_G and m_G are restrictions of the corresponding functions in H. A *graph morphism* $f \colon G \to H$ between two graphs G and H consists of two functions $f_{\mathrm{V}} \colon \mathrm{V}_G \to \mathrm{V}_H$ and $f_{\mathrm{E}} \colon \mathrm{E}_G \to \mathrm{E}_H$ preserving sources, targets and labels, that is, $\mathrm{s}_H \circ f_{\mathrm{E}} = f_{\mathrm{V}} \circ \mathrm{s}_G$, $\mathrm{t}_H \circ f_{\mathrm{E}} = f_{\mathrm{V}} \circ \mathrm{t}_G$, $\mathrm{l}_H \circ f_{\mathrm{V}} = \mathrm{l}_G$ and $\mathrm{m}_H \circ f_{\mathrm{E}} = \mathrm{m}_G$. The morphism f is an *isomorphism* if f_{V} and f_{E} are bijective. In this case G and H are *isomorphic*, which is denoted by $G \cong H$.

A *rule* $r = (L \supseteq K \subseteq R)$ consists of three graphs L, K and R such that K is a subgraph of both L and R. The graphs L and R are the *left-* and *right-hand side* of r, and K is the *interface*. Given a graph G, a graph morphism $f \colon L \to G$ satisfies the *gluing condition* if the following holds:

Dangling condition. No edge in $G - f(L)$ is incident to any node in $f(L) - f(K)$.
Identification condition. For all distinct items $x, y \in L$, $f(x) = f(y)$ only if $x, y \in K$.[1]

Given two graphs G and H, and a set of rules $\mathcal{R}$, there is a *direct derivation*[2] from G to H based on $\mathcal{R}$, denoted by $G \Rightarrow_{\mathcal{R}} H$, if there is a rule $r = (L \supseteq K \subseteq R)$ in $\mathcal{R}$ and a graph morphism $f \colon L \to G$ satisfying the gluing condition, such that H is isomorphic to the graph M constructed as follows:

[1]This condition is understood to hold separately for nodes and edges.
[2]See [1, 2] for an equivalent definition by a "double pushout" of graph morphisms.

1. Remove all nodes and edges in $f(L) - f(K)$ to obtain a subgraph D of G.
2. Add disjointly to D all nodes and edges in $R - K$ to obtain M, where all added items keep there labels and where the source of an edge e in $R - K$ is defined by

$$\mathrm{s}_M(e) = \begin{cases} f_{\mathrm{V}}(\mathrm{s}_R(e)) & \text{if } \mathrm{s}_R(e) \in \mathrm{V}_K, \\ \mathrm{s}_R(e) & \text{otherwise.} \end{cases}$$

   The target $\mathrm{t}_M(e)$ is defined analogously.

Given some $n \geq 0$, a *derivation* of length n from G to H based on $\mathcal{R}$ is a sequence of the form $G \cong G_0 \Rightarrow_{\mathcal{R}} G_1 \Rightarrow_{\mathcal{R}} \ldots \Rightarrow_{\mathcal{R}} G_n = H$. The relation $\Rightarrow^n_{\mathcal{R}}$ is defined as follows: $G \Rightarrow^n_{\mathcal{R}} H$ if there exists a derivation of length n from G to H based on $\mathcal{R}$. The relation $\Rightarrow^*_{\mathcal{R}}$ ($\Rightarrow^+_{\mathcal{R}}$) is defined by: $G \Rightarrow^*_{\mathcal{R}} H$ ($G \Rightarrow^+_{\mathcal{R}} H$) if there is some $n \geq 0$ ($n > 0$) such that $G \Rightarrow^n_{\mathcal{R}} H$.

A *graph rewriting system* $\mathcal{G} = (\Sigma, \mathcal{R})$ consists of a label alphabet Σ and a finite set $\mathcal{R}$ of rules with graphs over Σ. The system $\mathcal{G}$ is *terminating* if there does not exist an infinite rewrite sequence of the form $G_0 \Rightarrow_{\mathcal{R}} G_1 \Rightarrow_{\mathcal{R}} \ldots$

3. Undecidability of termination

This section is devoted to the proof of the main result, which is stated next.

Theorem 3.1. *It is undecidable in general whether a graph rewriting system is terminating.*

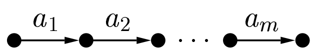
In what follows, Theorem 3.1 is proved by reducing the Post Correspondence Problem (PCP) to the problem of deciding whether a given graph rewriting system is terminating. Every instance of the PCP will be encoded as a graph rewriting system that is non-terminating if and only if the given instance has a solution.

Recall that the PCP is the problem to decide, given a nonempty list

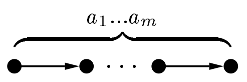
$$\mathcal{L} = \langle(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\rangle$$

of pairs of words over some finite alphabet Γ, whether there exists a sequence $i_1, \ldots, i_k$ of indices such that $\alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k}$. The list $\mathcal{L}$ is an *instance* of the PCP, and a sequence $i_1, \ldots, i_k$ with the above property is a *solution* of this instance. It is well-known that it is undecidable in general whether an instance of the PCP has a solution [6].

In the following encoding of the PCP, a string $a_1 \ldots a_m$ (with $m \geq 0$) will be encoded as a graph consisting of m consecutive edges labelled by $a_1, \ldots, a_m$:



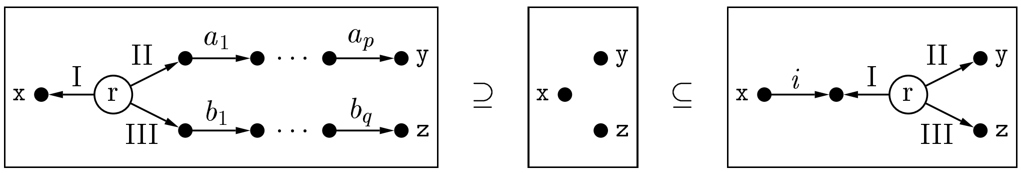
Such a graph will be depicted also as follows:



Let now $\mathcal{L} = \langle(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\rangle$ be an arbitrary instance of the PCP. It is assumed that for each pair (α_i, β_i), not both α_i and β_i are empty words. (The PCP with this restriction clearly remains undecidable.) Construct the graph rewriting system $\mathcal{G}(\mathcal{L}) = (\Sigma, \mathcal{R})$ as follows: $\Sigma_{\mathrm{V}} = \{\bullet, \mathrm{l}, \mathrm{r}\}$, $\Sigma_{\mathrm{E}} = \Gamma + \{1, \ldots, n\} + \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$[3] and $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 \cup \mathcal{R}_4$, where $\mathcal{R}_1, \ldots, \mathcal{R}_4$ are defined below.

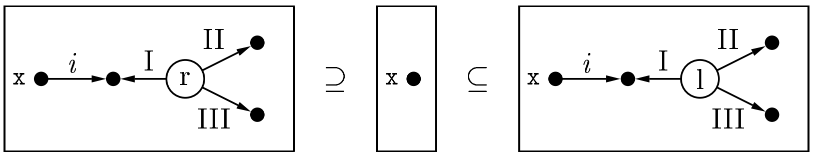
[3]Here + denotes the disjoint union of sets.

$\mathcal{R}_1$ contains the rules[4]



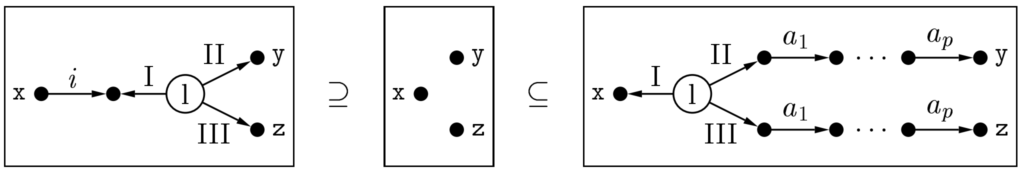
for $i = 1, \dots, n$, where $\alpha_i = a_1 \dots a_p$ and $\beta_i = b_1 \dots b_q$.

$\mathcal{R}_2$ contains the rules



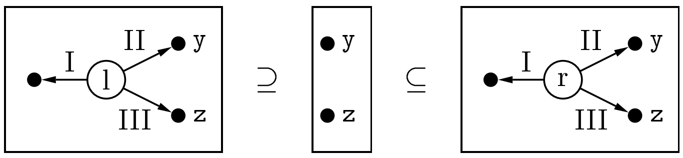
for $i = 1, \dots, n$.

$\mathcal{R}_3$ contains the rules



for $i = 1, \dots, n$, where $\alpha_i = a_1 \dots a_p$.

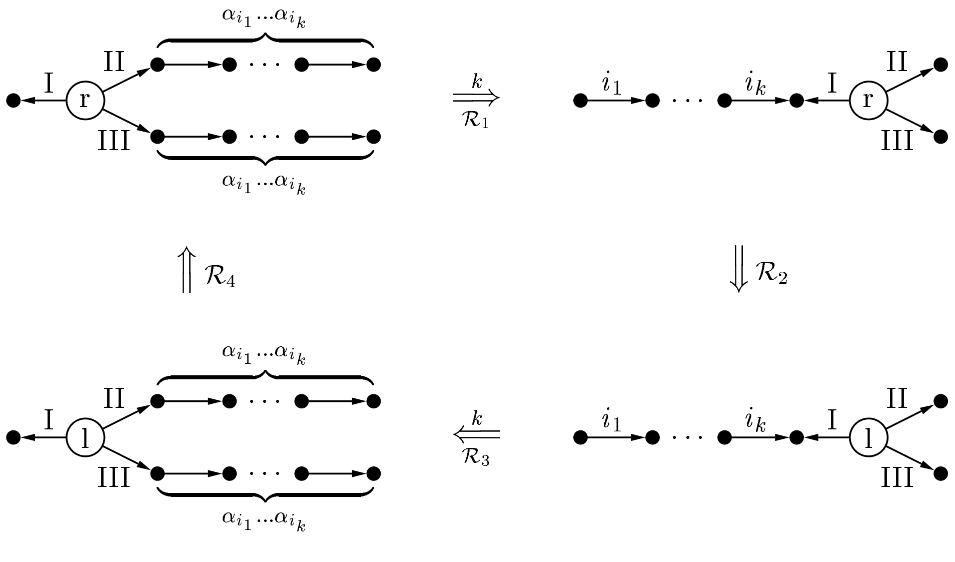
$\mathcal{R}_4$ contains the following rule:



Now the task is to show that the instance $\mathcal{L}$ has a solution if and only if $\mathcal{G}(\mathcal{L})$ is not terminating. The "only if"-direction, which is the easier part, is given by the next lemma. Three further lemmas will be needed to show the converse.

Lemma 3.1. *If $\mathcal{L}$ has a solution, then $\mathcal{G}(\mathcal{L})$ is not terminating.*

Proof:
If $i_1, \dots, i_k$ is a solution of $\mathcal{L}$, then $\mathcal{G}(\mathcal{L})$ admits the following cyclic derivation:


$\alpha_{i_1} \dots \alpha_{i_k}$
I
r
II
III
$\alpha_{i_1} \dots \alpha_{i_k}$
$\overset{k}{\Longrightarrow}_{\mathcal{R}_1}$
i_1
i_k
I
r
II
III
$\Downarrow \mathcal{R}_2$
$\Uparrow \mathcal{R}_4$
$\alpha_{i_1} \dots \alpha_{i_k}$
I
l
II
III
$\alpha_{i_1} \dots \alpha_{i_k}$
$\overset{k}{\Longleftarrow}_{\mathcal{R}_3}$
i_1
i_k
I
l
II
III

□

[4]In the following pictures, x, y and z are node names which are used to depict subgraph inclusions.

In the following, a node labelled with l or r is called a *control node*. As each rule in $\mathcal{R}$ has a unique control node in its left- and right-hand side, a direct derivation $G \Rightarrow_{\mathcal{R}} H$ will be considered as an application of a rule "to a control node", and the images of the left- and right-hand control node in G and H will be considered as "the same" node (although the labels may be different).

Lemma 3.2. *Every infinite rewrite sequence over $\mathcal{G}(\mathcal{L})$ contains a control node to which the rule in $\mathcal{R}_4$ is applied infinitely often.*

Proof:
It suffices to show that the system $\mathcal{G}^{\ominus} = (\Sigma, \mathcal{R} - \mathcal{R}_4)$ is terminating. For then every infinite rewrite sequence over $\mathcal{G}(\mathcal{L})$ contains infinitely many applications of the rule in $\mathcal{R}_4$. As no rule in $\mathcal{R}$ increases the number of control nodes, it follows that there is a control node to which the rule in $\mathcal{R}_4$ is applied infinitely often.

To show that $\mathcal{G}^{\ominus}$ is terminating, suppose the contrary. Call an edge a Γ-edge if it is labelled with a symbol from Γ.

Claim: In every rewrite sequence over $\mathcal{G}^{\ominus}$, no Γ-edge produced by $\mathcal{R}_3$ is ever removed.

Proof: Given a step $G \Rightarrow_{\mathcal{R}_3} H$, there is no directed path in H from a node labelled with r to the source node of any Γ-edge produced by the step. Hence none of these edges can be removed by applying a rule to H. Moreover, it is easy to see that all rules in $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$ preserve the absence of a directed path from a node labelled with r to a fixed Γ-edge. Thus none of the Γ-edges produced by $G \Rightarrow_{\mathcal{R}_3} H$ can be removed in a future step.

By the claim and the fact that the rules in $\mathcal{R}_1 \cup \mathcal{R}_2$ do not produce Γ-edges, all Γ-edges removed by an $\mathcal{R}_1$-step in a rewrite sequence over $\mathcal{G}^{\ominus}$ have been present already in the start graph of the sequence. It follows that every infinite rewrite sequence over $\mathcal{G}^{\ominus}$ contains only a finite number of $\mathcal{R}_1$-steps: each of these steps removes some Γ-edges while neither $\mathcal{R}_1$ nor $\mathcal{R}_2$ produces Γ-edges. Hence, if $\mathcal{G}^{\ominus}$ is not terminating, the system $(\Sigma, \mathcal{R}_2 \cup \mathcal{R}_3)$ must admit an infinite rewrite sequence. But this is impossible since all rules in $\mathcal{R}_2 \cup \mathcal{R}_3$ decrease the number of nodes and edges with label in $\{\mathrm{r}\} \cup \{1, \ldots, n\}$. Thus $\mathcal{G}^{\ominus}$ is terminating. This concludes the proof of Lemma 3.2. □

Lemma 3.3. *If an infinite rewrite sequence over $\mathcal{G}(\mathcal{L})$ starts from a connected graph, then all graphs in the sequence contain exactly one control node.*

Proof:
For every graph G, call a subgraph C an *index chain* of length k, $k \geq 0$, if C has the form

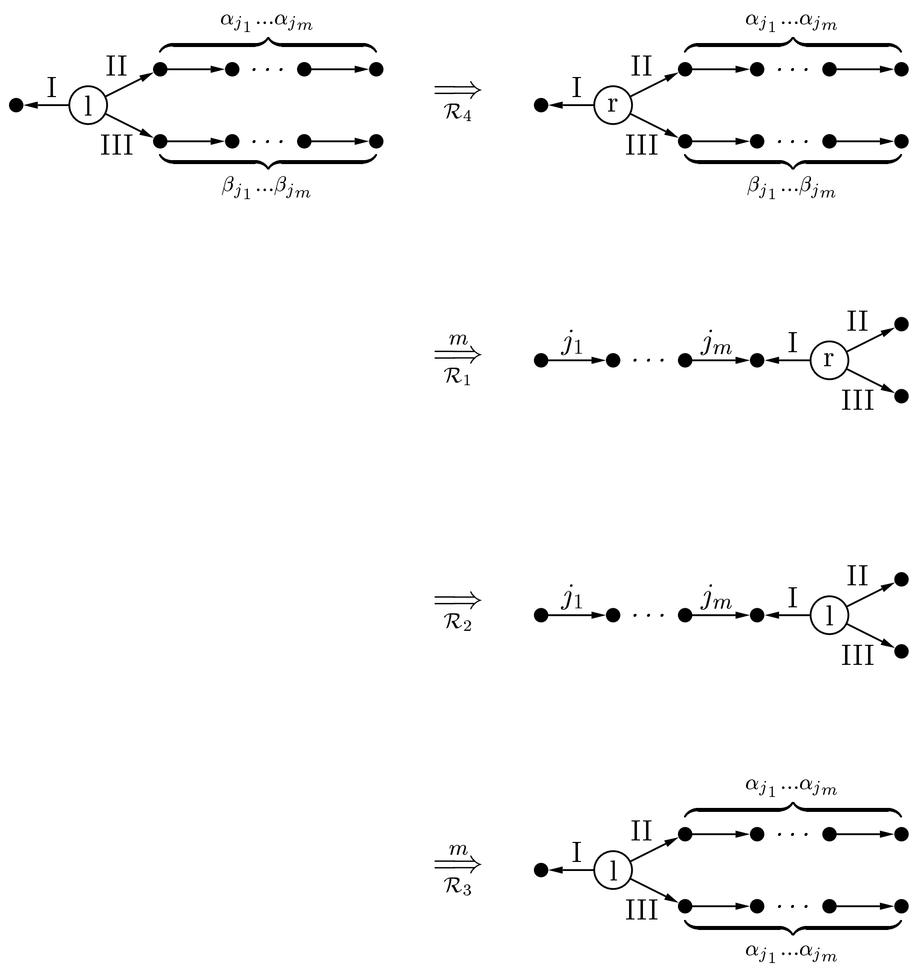
$$\begin{tikzcd} \bullet \arrow[r, "i_1"] & \bullet & \cdots & \bullet \arrow[r, "i_k"] & \bullet & \boxed{\mathrm{r}} \arrow[l, "\mathrm{I}"'] \end{tikzcd}$$

where $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and where only the control node may be incident to edges not belonging to C. The following claim follows immediately from the shape of the rules in $\mathcal{R}$.

Claim: Let $G \Rightarrow_{\mathcal{R}} H$ be a direct derivation and C be an index chain of length k in G. Then
(1) C is (up to isomorphism) also an index chain in H, or
(2) C is transformed into an index chain of length $k+1$, or
(3) $G \Rightarrow_{\mathcal{R}} H$ is an application of a rule from $\mathcal{R}_2$ to the control node in C.

Let now $G_0 \Rightarrow_{\mathcal{R}} G_1 \Rightarrow_{\mathcal{R}} \ldots$ be an infinite derivation such that G_0 is connected. Since all rules in $\mathcal{R}$ preserve the number of control nodes, it suffices to show that there is some graph in the rewrite sequence that contains exactly one control node. By Lemma 3.2, there is a control node c to which the rule in $\mathcal{R}_4$ is applied infinitely often. Hence there are $0 \leq s < t$ such that $G_s \Rightarrow_{\mathcal{R}_4} G_{s+1} \Rightarrow_{\mathcal{R}}^{+} G_t \Rightarrow_{\mathcal{R}_4} G_{t+1}$, where $G_s \Rightarrow_{\mathcal{R}_4} G_{s+1}$ and $G_t \Rightarrow_{\mathcal{R}_4} G_{t+1}$ are applications of $\mathcal{R}_4$ to c and where $G_{s+1} \Rightarrow_{\mathcal{R}}^{+} G_t$ does not contain an application of $\mathcal{R}_4$ to c. By the shape of the rule in $\mathcal{R}_4$, c belongs to an index chain of length 0 in G_{s+1}. Moreover, there must be some s' with $s < s' < t$ such that $G_{s'} \Rightarrow_{\mathcal{R}_2} G_{s'+1}$ is an application of $\mathcal{R}_2$ to c. Hence, by the above observation, c belongs to an index chain in $G_{s'}$. By the shape of the rules in $\mathcal{R}_2$, if c is connected with another control node d in $G_{s'}$, then all nodes of the index chain containing c are connected with d via a path that does not contain c. But this contradicts the definition of an index chain. That is, c cannot be connected with another control node. Since G_0 is connected and all rules in $\mathcal{R}$ preserve connectedness, it follows that c is the only control node in $G_{s'}$. □

Lemma 3.4. *If $\mathcal{G}(\mathcal{L})$ is not terminating, then there is an infinite rewrite sequence that starts with a derivation of the form*



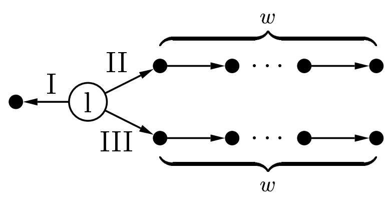
$\alpha_{j_1} \cdots \alpha_{j_m}$
I
II
III
l
$\beta_{j_1} \cdots \beta_{j_m}$
$\Longrightarrow_{\mathcal{R}_4}$
$\alpha_{j_1} \cdots \alpha_{j_m}$
I
II
III
r
$\beta_{j_1} \cdots \beta_{j_m}$
$\overset{m}{\Longrightarrow}_{\mathcal{R}_1}$
j_1
j_m
I
r
II
III
$\Longrightarrow_{\mathcal{R}_2}$
j_1
j_m
I
l
II
III
$\overset{m}{\Longrightarrow}_{\mathcal{R}_3}$
$\alpha_{j_1} \cdots \alpha_{j_m}$
I
II
III
l
$\alpha_{j_1} \cdots \alpha_{j_m}$

where $m \geq 1$ and $j_1, \ldots, j_m \in \{1, \ldots, n\}$.

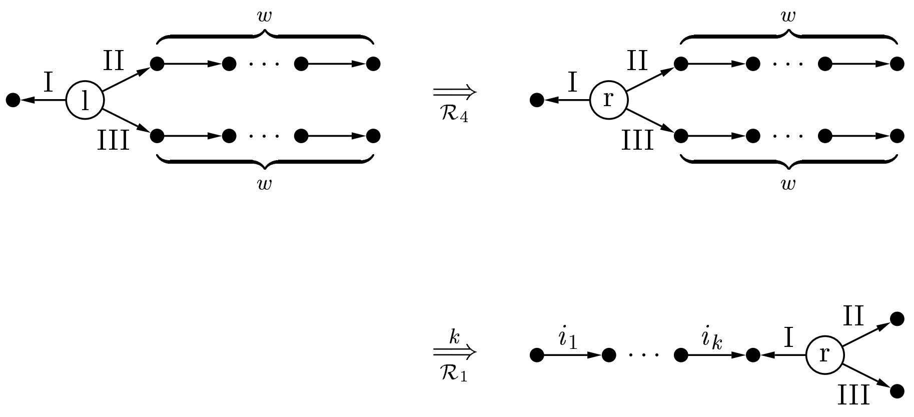
Proof:
Since the left- and right-hand sides of all rules in $\mathcal{R}$ are connected, every direct derivation $G \Rightarrow_{\mathcal{R}} H$ takes place within some connected component C of G and transforms C into a connected component of H. Therefore every infinite rewrite sequence over $\mathcal{G}(\mathcal{L})$ contains a connected component which is subject to infinitely many rule applications. By filtering out all rule applications to this component and restricting all graphs to this component, one obtains an infinite rewrite sequence of connected graphs. By Lemma 3.2, from some point on this sequence has the form $G_1 \Rightarrow_{\mathcal{R}_4} H_1 \Rightarrow^*_{\mathcal{R}^\ominus} G_2 \Rightarrow_{\mathcal{R}_4} H_2 \Rightarrow^*_{\mathcal{R}^\ominus} \ldots$ with $\mathcal{R}^\ominus = \mathcal{R} - \mathcal{R}_4$. By Lemma 3.3, all graphs in this sequence contain exactly one control node. Thus, taking into account the shape of the rules in $\mathcal{R}$, the derivation $G_1 \Rightarrow_{\mathcal{R}_4} H_1 \Rightarrow^*_{\mathcal{R}^\ominus} G_2$ must be of the form $G_1 \Rightarrow_{\mathcal{R}_4} H_1 \Rightarrow^m_{\mathcal{R}_1} P_1 \Rightarrow_{\mathcal{R}_2} Q_1 \Rightarrow^m_{\mathcal{R}_3} G_2$, where $m \geq 1$ and where G_1, H_1, P_1, Q_1 and G_2 are as in the above picture. (The control node to which $\mathcal{R}_4$ is applied in G_1 can be subject to an $\mathcal{R}_4$-application in G_2 only if $H_1 \Rightarrow^*_{\mathcal{R}^\ominus} G_2$ is as depicted. Note that, by the dangling condition for direct derivations, nodes being removed are incident only to edges having a preimage in the left-hand side of the applied rule.) □

Using Lemma 3.1 and 3.4, it is straightforward to prove the main result.

Proof of Theorem 3.1:
By the undecidability of the Post Correspondence Problem, it suffices to show that the instance $\mathcal{L}$ has a solution if and only if the graph rewriting system $\mathcal{G}(\mathcal{L})$ is not terminating. Lemma 3.1 shows the "only if"-direction of this equivalence. For the converse, suppose that $\mathcal{G}(\mathcal{L})$ is not terminating. Then, by Lemma 3.4, there is an infinite rewrite sequence containing a graph of the form



where w is a some word over Γ. By the shape of the rules in $\mathcal{R}$, the next steps in the sequence have the form


w
I
II
l
III
w
$\underset{\mathcal{R}_4}{\Longrightarrow}$
w
I
II
r
III
w
$\overset{k}{\underset{\mathcal{R}_1}{\Longrightarrow}}$
i_1
i_k
I
r
II
III

where $k \geq 1$ and $i_1, \ldots, i_k \in \{1, \ldots, n\}$. By the shape of the rules in $\mathcal{R}_1$, this implies $\alpha_{i_1} \ldots \alpha_{i_k} = w = \beta_{i_1} \ldots \beta_{i_k}$. Thus, $i_1, \ldots, i_k$ is a solution of $\mathcal{L}$. $\square$

4. Conclusion

By a reduction of the Post Correspondence Problem, it has been shown that termination of graph rewriting is undecidable in general. Somewhat surprisingly, the possible presence of cycles in graphs prevents a straightforward reduction of the halting problem for Turing machines or of the termination problem for string rewriting systems to the present problem.

It is worth noting that the given reduction of the PCP also shows the undecidability of the following problem. Call a graph rewriting system *cyclic* if it admits a derivation in which some graph occurs twice. (So every cyclic system is non-terminating, but the converse does not hold in general.) From the proof of Theorem 3.1 one obtains the following result.

Theorem 4.1. *It is undecidable in general whether a graph rewriting system is cyclic.*

For, the proof of Lemma 3.1 shows that the system $\mathcal{G}(\mathcal{L})$ is cyclic whenever an instance $\mathcal{L}$ of the PCP has a solution, and the premise of Lemma 3.4—which requires that $\mathcal{G}(\mathcal{L})$ is not terminating—clearly holds true if $\mathcal{G}(\mathcal{L})$ is cyclic.

Acknowledgement. The author wishes to thank Jürgen Müller for discussions about termination of graph rewriting, and Annegret Habel for comments on a previous version of this paper.

References

[1] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation — Part I: Basic concepts and double pushout approach. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, chapter 3, pages 163–245. World Scientific, 1997.

[2] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Proc. Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer-Verlag, 1979.

[3] Maria C.F. Ferreira. Termination of term rewriting. Dissertation, Universiteit Utrecht, Faculteit Wiskunde en Informatica, 1995.

[4] Gérard Huet and Dallas Lankford. On the uniform halting problem for term rewriting systems. Report no. 283, INRIA Rocquencourt, 1978.

[5] Pierre Lescanne. On termination of one rule rewrite systems. *Theoretical Computer Science*, 132:395–401, 1994.

[6] Grzegorz Rozenberg and Arto Salomaa. *Cornerstones of Undecidability.* Prentice Hall, 1994.

[7] Hans Zantema. Total termination of term rewriting is undecidable. *Journal of Symbolic Computation*, 20:43–60, 1995.