Electronic Communications of the EASST
Volume 30 (2010)

International Colloquium on Graph and Model Transformation On the occasion of the 65th birthday of Hartmut Ehrig (GraMoT 2010)

A Termination Criterion for Graph Transformations with Negative Application Conditions

Paolo Bottoni, Francesco Parisi Presicce

13 pages

Guest Editors: Claudia Ermel, Hartmut Ehrig, Fernando Orejas, Gabriele Taentzer
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

A Termination Criterion for Graph Transformations with Negative Application Conditions

Paolo Bottoni, Francesco Parisi Presicce

Dipartimento di Informatica, "Sapienza" Università di Roma, Italy

Abstract: Termination of graph transformations is in general undecidable, but it is possible to prove it for specific systems by checking for sufficient conditions. In the presence of rules with negative application conditions, the difficulties increase. In this paper we propose a different approach to the identification of a (sufficient) criterion for termination, based on the construction of a labelled transition system whose states represent overlaps between the negative application condition and the right hand side that can give rise to cycles.

Keywords: graph transformations, termination, labelled transition system

1 Introduction

Model transformations are an essential component of the model-driven approach to software development. Graphs are a natural and intuitive way to describe models (e.g., class diagrams in UML) and graph transformations provide a rule-based approach to their modifications. Sometimes a particular transformation needs to be applied to the target graph/model as long as matchings of its left hand side can be found. In such cases, it is necessary to be able to determine that such a repeated application will eventually reach a state where the transformation is no longer applicable. More generally, the term *termination* refers to the problem of determining whether a set of rules can generate a graph/model to which none of the rules is still applicable. Ad hoc methods have been applied to show termination of specific rewriting systems (e.g. [KHE03]).

Termination properties can be (and have been) studied for specific rewriting systems, following the classical approach – given by Dershowitz and Manna in [DM79] – of proving termination by constructing a monotone measure function on some multiset associated to the object to be rewritten, and showing that the value of this function decreases with each application of the rule. Further termination criteria use polynomial orderings, recursive path orderings, etc. [Der87].

In a previous paper [BHPT05], we have identified an abstract notion of termination criterion for high-level replacement (HLR) systems, i.e. algebraic rewriting systems operating on objects and morphisms in adhesive HLR categories [EPPH06], in which rewriting is guided by control expressions. The approach is based on a generic measure function $\mathscr{F} : G \to \mathbb{N}$, called a *termination criterion* if it satisfies the property $\mathscr{F}(A +_C B) = \mathscr{F}(A) + \mathscr{F}(B) - \mathscr{F}(C)$ for morphisms $C \to A$ and $C \to B$ in a specific subclass $\mathscr{M}$. A termination criterion for a rule $p : L \leftarrow K \to R$ is such a function with $\mathscr{F}(\mathrm{L}) > \mathscr{F}(\mathrm{R})$.
However, we have subsequently shown in [BHP06] how the extension of this notion to rules with negative application conditions (NACs) encounters several difficulties. In particular, we have presented examples of pairs of rules for which no application criterion can differentiate between a terminating and a non-terminating rule.

In this paper, we propose a different approach to the identification of a (sufficient) termination criterion for the repeated application of a single rule with a NAC, based on the construction of a Labelled Transition System, where the states correspond to classes of matches of a rule with respect to all the possible intermediate graphs between the left-hand side of a rule and the negative application condition.

In the following, we give formal definitions for the adopted model of graphs in Section 2, and present motivations for the approach through a number of cases in Section 3. Section 4 presents the main result of the paper, showing the construction of the transition system, and Section 5 discusses related work. Finally, Section 6 draws conclusions and points to future work.

2 Formal Background

We use the DPO (Double PushOut) approach to graph transformation [EEPT06]

A graph $G = (V,E,s,t)$ consists of a finite set of *nodes* $V = V(G)$, a finite set of *edges* $E = E(G)$, a *source* and a *target* total functions, $s,t : E \to V$. In a *type graph* $TG = (V_T,E_T,s^T,t^T)$, V_T and E_T are sets of node and edge types, while the functions $s^T : E_T \to V_T$ and $t^T : E_T \to V_T$ define source and target node types for each edge type. A typed graph on $TG = (V_T,E_T,s^T,t^T)$ is a graph $G = (V,E,s,t)$ equipped with a graph morphism $type\colon G \to TG$, composed of two functions $type_V : V \to V_T$ and $type_E : E \to E_T$, preserving the *source* s^T and the *target* t^T functions, i.e. $type_V(s(e)) = s^T(type_E(e))$ and $type_V(t(e)) = t^T(type_E(e))$.

A DPO rule consists of three graphs, called left- and right-hand side (L and R), and interface graph K. Two injective morphisms[1] $l\colon K \to L$ and $r\colon K \to R$ model the embedding of K (containing the elements preserved by the rule) into L and R. Figure 1 shows a DPO direct derivation diagram. Square (1) is a pushout modeling the deletion from G of the elements of L not in K, while pushout (2) models the addition to G of the elements present in R but not in K. If $L = K$ the rule is called *non-deleting*, while if $K = R$ the rule is called *deleting*.

Figure 1 illustrates the notion of *negative application condition* (NAC), of the form $n\colon L \to N$ that a match $m\colon L \to G$ must satisfy. A rule is applicable with match $m\colon L \to G$ if there is no morphism $q\colon N \to G$ such that $q \circ n = m$.

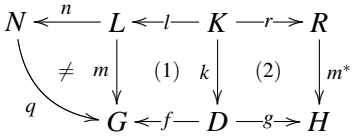

Figure 1: DPO Direct Derivation Diagram for rules with NAC.

3 A naive approach

In this section we analyze a few examples of simple rules which exhibit different behaviors despite appearing very similar. The same examples will be used later on to illustrate the different

[1] In this paper, except for the typing morphisms $type\colon G \to TG$, all morphisms are total and injective.

cases of our main result.

We consider here only **non-deleting rules**: the case of the repeated application of a deleting rule, i.e., a rule where L has at least one more item (node or edge) than K and $K = R$, is easier to handle. If the original graph is finite, and every application removes at least one item, eventually a graph is produced where the rule is no longer applicable.

In non-deleting rules, we omit the K component of rules and write a rule with a single NAC as $p : N \leftarrow L \rightarrow R$, (not to be confused with a generic rule with L as interface).

When dealing with rules equipped with NACs, the first (only partially incorrect) thought that comes to mind is that the rule cannot be applied again if it produces, in the RHS, the NAC, i.e., if $N \subset R$. In fact, Figure 2 shows a rule for which the existence of an injection of N into R is sufficient to prevent the repeated application of the rule (on the same pair of nodes).

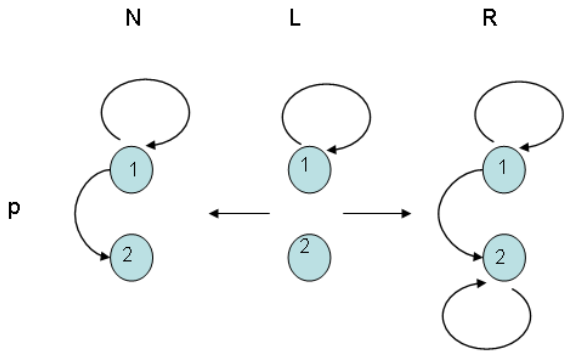

Figure 2: A terminating rule, with $N \subset R$.

However, the existence of an injection of N into R is not sufficient to guarantee the non-applicability of the same rule and to discriminate between a terminating and a non-terminating rule, as the example in Figure 3 shows.

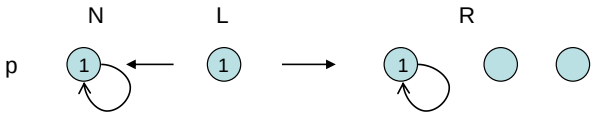

Figure 3: A rule which does not terminate, with $N \subset R$.

It is important to observe how for the examples in Figure 2 and Figure 3 no single function F from graphs to natural numbers, which is a termination criterion for rules without NACs, could be used to discriminate between the two cases. The difference between the two cases is in fact that in the rule of Figure 3, the number of matches increases, whereas in Figure 2 it decreases.

Maybe we should reverse the direction of the inclusion, and consider rules where $R \subset N$. Again, this condition is neither sufficient nor necessary for termination, as the following two examples show.

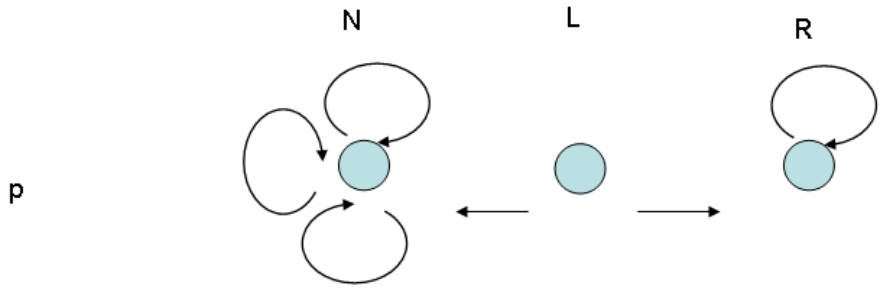

Figure 4: A terminating rule, with $R \subset N$.

A rule which must terminate is presented in Figure 4, where no more than two loops can be added to each node in the graph.
This is not the case for the rule in Figure 5, where again $R \subset N$, but the rule may terminate or not, depending on whether we choose a different match after applying the rule or the same match (the NAC N does not prohibit several loops on the same node).

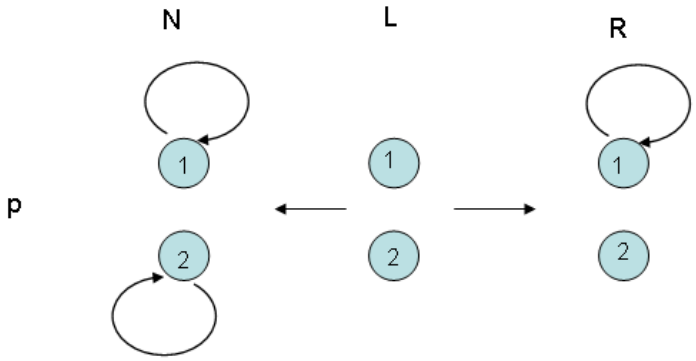

Figure 5: A rule which may terminate, with $R \subset N$.

While the examples presented here are simple enough that an ad-hoc analysis is sufficient to determine whether we have termination or not, for the general case we need criteria that will distinguish the cases seen above. In the next section we show how to extract this information from labelled transition systems associated with these rules.

4 A Termination Criterion

We study here the termination of single non-deleting rules with a single NAC; we will mention in the last section how to extend the results to the case of one rule with multiple NACs, of a set of rules with NACs, and of rule sequences.

Consider the simple example in Figure 6. The rule is a non-deleting rule, so it is not clear how to apply the standard approach based on the 'consumption' of some finite quantity. Nevertheless,

the rule can only be applied a finite number of times to a finite graph (the rule can be applied no more than twice to each pair of nodes of G). What decreases after each application is, in a sense, the "distance" between the left hand side L of the rule and the negative application condition N.

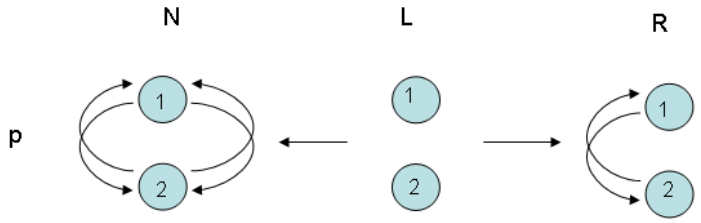

Figure 6: A simple terminating rule.

Consider now the slightly different rule in Figure 7 and notice that it is no longer true that its application must always terminate. After the first application, if the roles of the 2 nodes are reversed in the matching, the remaning part of the negative application condition is generated. But it is also possible to continue adding edges from node 1 to node 2, without ever generating the (rest of the) NAC to prevent further applications. Notice that these additional edges do not affect the applicability of the rule.

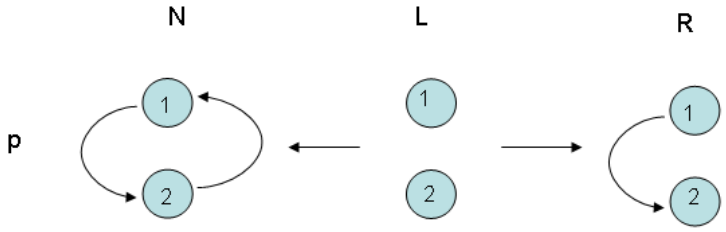

Figure 7: A simple non terminating rule.

In both cases, the rule generates a graph 'between' the left hand side and the NAC. We now abstract from the specific examples to describe this 'approaching' the NAC as a path on a labelled transition system.

Let $p : N \xleftarrow{n} L \xrightarrow{r} R$ be a rule. Let $\mathscr{H}^p = \{H_1^p, \ldots, H_k^p\}$ be the set of all graphs (up to isomorphism w.r.t. the image of L)and $\mathscr{M}^p = \{h_j^i : H_i^p \to H_j^p\}$ be the set of associated morphisms (if they exist) s.t. the following are jointly verified:

- for each $i = 1, \ldots, k$, there exist morphisms $L \xrightarrow{h_i^L} H_i^p \xrightarrow{h_N^i} N$.
- for each $i, j = 1, \ldots, k$, if h_j^i exists, then $h_j^L = h_i^L \circ h_j^i$ and $h_i^N = h_j^i \circ h_j^N$.

The indexing of the set $\mathscr{H}^p$ is irrelevant, but we can assume that it is compatible with the

partial order induced by morphisms, so that $L = H_1^p$ and $N = H_k^p$. This also indicates that this set is not empty. The set $\mathscr{M}^p$ is not empty either, as it contains at least $h_k^1 = n$ and all the identity morphisms. Moreover, both sets are finite since our morphisms are all total and injective, including $n\colon L \to N$. (We will indicate in Section 6 how to relax this condition and allow non-injective $n\colon L \to N$ while keeping the set finite).

Let V_L be the set of nodes in L, and $M_L = \{m_1, \ldots, m_r\}$ the set of matches $m_i : V_L \to V_L$ of V_L into itself, including the identity $id_{V_L} = m_1$. We construct a Labelled Transition System $\mathscr{L}^p = (S, \Lambda, \longrightarrow)$ as follows:

1. S contains a state s_i for each graph $H_i^p \in \mathscr{H}^p$. Each s_i induces a classification function c_i for matches of p such that $c_i(m_l) = true$ iff m_l can be extended to a match on H_i^p, but not to a match on any other $H_j^p \in \mathscr{H}_p \setminus (\{H_1^p, H_i^p\} \cup \{H_t^p \mid \exists h_i^t : H_t^p \to H_i^p\})$, i.e. H_i^p is the biggest graph to which m_l can be extended.

2. Λ contains a label p^i, $i = 1, \ldots, r$ for each morphism in M_L.

3. the transition relation $\longrightarrow \subset S \times \Lambda \times S$ is such that $(s_i, p^l, s_j) \in \longrightarrow$ (denoted by $s_i \xrightarrow{p^l} s_j$) if applying p with match m_l on graph H_i^p produces a graph for which $c_j(m_l) = true$.

Going back to the examples earlier in this section, the labelled transition system of the rules in Figure 6 and Figure 7 constructed as above are illustrated in Figure 8 and Figure 9, respectively (ignore, for now, the labels on the arcs of the transition systems).

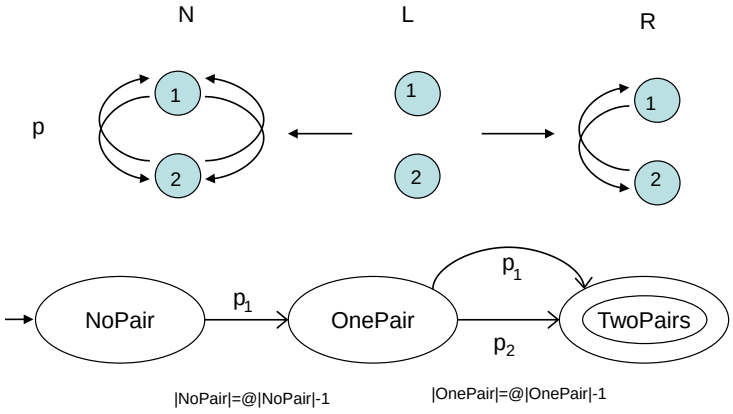

Figure 8: A terminating rule.

We say that p:

- *should terminate simply* if there exists a chain from s_1 to s_k in $\mathscr{L}^p$ with all transitions labelled with p_1.

- *may terminate simply* if for all chains from s_1 to s_k in $\mathscr{L}^p$, at least one label is different from p_1.

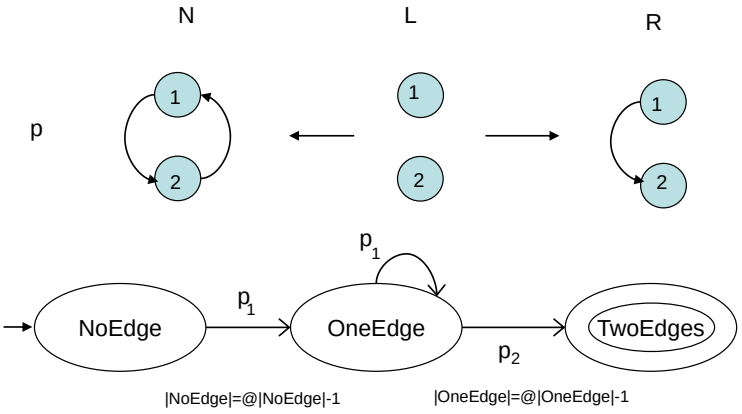

Figure 9: A rule which may terminate.

- *does not terminate simply* if there is no chain from s_1 to s_k in $\mathscr{L}^p$.

In the definition above, by chain we mean a path which does not contain the same state twice.

Referring to the examples above, in the first case, there is a path from s_1 to s_k and the rule should terminate. In the second case, the rule may terminate, as the path from the initial to the final state presents different labels (corresponding to the changing of the roles between the two nodes). However, the presence of a loop on the middle node indicates that the application may not terminate (corresponding to the nodes maintaining their original roles in the match). In these figures, and in the following ones, we have only indicated the states reachable from H_1^p after applying the first match, assumed as m_1.

Unfortunately, the presence of a unique path as in Figure 8 allows only a "should" and not a "must", as the example in Figure 10 illustrates. This is the case of a rule which should terminate simply, as there is a path which consumes the possibility of rule application on the original match, but does not terminate, as the number of matches for the rule increases at each application of p.

In order to consider the variation on the number of possible matches induced by the application of p on the minimal context for a given state, let $@\,|\cdot|\colon S \to \mathbb{N}$ be a function which associates with each state s_i the number of matches for p on the graph H_i^p prior to application of p and with $|\cdot|\colon S \to \mathbb{N}$ the function defining the number of matches on H_i^p after the application of p.

We use these functions to identify the effect of each transition from a path in $\mathscr{L}^p$ on the number of matches in the resulting graph and obtain in Theorem 1 the first result on termination of p starting on a graph isomorphic to L, i.e. on the minimal context in which p is applicable.

Theorem 1 ***[Termination on minimal context.]***
Given a graph G isomorphic to L, we have the following:

1. *A sufficient condition for the termination of* ***asLongAsPossible*** *p* ***end*** *starting on G is that p is of type* ***should terminate simply*** *AND on all transitions $s_i \xrightarrow{p_\downarrow} s_j$ the number of matches classified by c_i decreases, the number of matches classified by c_j can increase at most of*

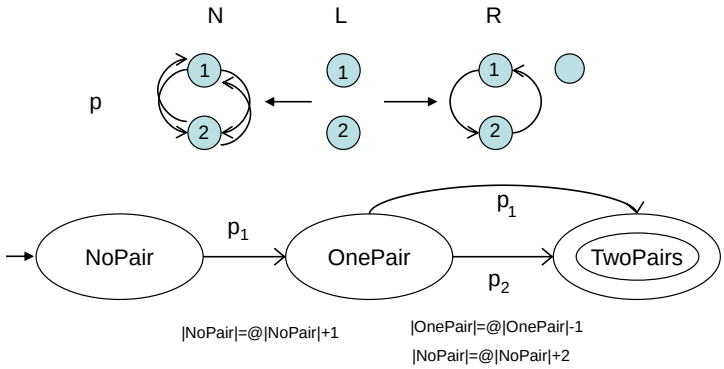

Figure 10: A rule which should terminate simply, but does not terminate.

1, and the number of matches classified by c_l, $l \neq i, j$ does not increase.

2. *A sufficient condition for the non-termination of* ***asLongAsPossible*** *p* ***end*** *starting on G is that p is of type* ***does not terminate simply*** *OR that for each path from s_1 to s_k there is at least one state s_i s.t. $\mid s_i \mid \geq @ \mid s_i \mid$, for a transition starting from state s_i, or a state s_j s.t. $\mid s_j \mid > @ \mid s_j \mid$ for a transition reachable from s_j.*

Proof. **[Sketch]** The proof derives from a straightforward counting argument, where the decreasing number of matches prevents the application of a rule on parts of the graph generated by the right hand side, while the 'simply terminating' condition prevents the repeated application using the same match. More formally, we have:

1. As G is isomorphic to L, we can assume w.l.o.g. that the first application of the rule is for match m_1. Moreover, for all states $s_i \neq s_1$, we have $\mid s_i \mid = 0$. Hence, the first application will be associated with a transition $s_1 \stackrel{p_1}{\rightarrow} s_j$, while the difference in the number of matches will follow the laws $\mid s_1 \mid < @ \mid s_1 \mid$, $\mid s_j \mid \leq @ \mid s_j \mid +1$, $\mid s_i \mid = 0$, for $s_i \neq s_1, s_j$. At each subsequent application on the same match, this match will follow the whole path from s_i to s_k, without ever creating new matches for other states. Hence, any chosen match will eventually be forbidden by the NAC, while no new matches are created in any intermediate step. As a consequence, p can be applied at most $r \times k$ times.

2. If p is of type **does not terminate simply**, any path starting from s_1 will eventually reach a state s_l for which there is a loop, i.e. a transition $s_l \stackrel{p_1}{\rightarrow} s_l$. Moreover, if there is a state s_i as described in the hypothesis, each application of the rule will create new matches. As such a state must be reached on any path from s_1 to s_k, new matches will be formed at each application of the rule on a match on this state. □

We now generalise this analysis so that it can be applied to any arbitrary graph G, as shown in Theorem 2. In particular, we need to consider the possibility of completing partial matches. Hence, we extend the construction of $\mathscr{L}^p$ to include states corresponding to the intermediate states between the empty graph and the state corresponding to L. We label these states as s_i^a

to distinguish them from those in the original set. We can then integrate the definition of the transitions on the original $\mathscr{L}^p$, with the study of the variations in the number of matches for these new states. We now say that p:

- *must terminate* if it should terminate simply and for all transitions $s_i \xrightarrow{p_l} s_j$ and all states $s_h \neq s_i, s_j, s_k$, we have $\mid s_i \mid < @ \mid s_i \mid, \mid s_j \mid \leq @ \mid s_j \mid +1$ and $\mid s_h \mid \leq @ \mid s_h \mid$.

- *may terminate* if it may terminate simply and for all states on the path the same condition on matches as above applies.

- *does not terminate* if (it does not terminate simply AND for all states from which s_k is reachable, there is a state for which the number of matches increases for some transition leading to s_k increases) OR (it should or may terminate simply, but there is at least one state s_i on a path from s_1 to s_k for which $\mid s_i \mid \geq @ \mid s_i \mid$, for a transition reachable from state s_i).

Theorem 2 ***[Main Result: Termination on arbitrary graphs.]***
Let $@ \mid \cdot \mid : S \to \mathbb{N}$ and $\mid \cdot \mid : S \to \mathbb{N}$ be counting functions as defined above, let p be a rule and G be a finite graph. Then the following holds:

1. *If rule p is of type* **must terminate***, then the application of* ***asLongAsPossible*** *p* ***end*** *on the starting graph G terminates after a finite number of steps.*

2. *If p is of type* ***does not terminate*** *then the application of* ***asLongAsPossible*** *p* ***end*** *on the starting graph G does not terminate.*

Proof. **[Sketch]**

1. Let p be a rule of type **must terminate** and suppose that the iteration of p starting on G does not terminate. This can happen only if there is a chain of transformations $G \Rightarrow G_1 \Rightarrow_r \cdots \Rightarrow_r G_n$ such that $\mid \{m : L(r) \to G_n\} \geq \{m : L(r) \to G\}$ But now this cannot happen, as each match $m : L(r) \to G$ can be used only at most k times because of the condition that the rule should terminate simply. Moreover, the second condition states that the number of matches in the state for which the match was chosen can only decrease, so that only the matches originally contained in G can be used. Moreover, no new match can be created, as the only way the number of matches in a state $s_j \neq s_k$ can increase is by transferring a match from a state s_i following a transition $s_i \to s_j$. Finally, if matches are created in s_k, they are immediately forbidden.

2. If the rule is of type **does not terminate simply**, then the final state s_k can be reached only from states not reachable from s_1. Hence, starting from a match in a state reachable from s_1, we will reach a state where the application of the rule can be iterated indefinitely on the same match. In the case that G hosts matches classified in states from which s_k can be reached, the remaining sub-conditions in the definition of *does not terminate*, the application of p on such matches will eventually lead to the creation of new matches. □

This result explains the behavior of the examples seen earlier in this section. In particular, it shows why the rules in Figures 8 and 10 behave differently.

We are now able to formalize the observations made in Section 3 on those simple rules. Figures 11–13 show the labelled transition systems for the rules in Figures 2–4 (repeated here for convenience), respectively.

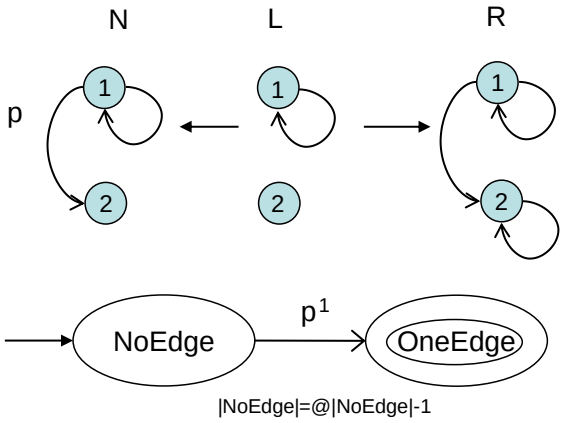

Figure 11: A terminating rule, with $N \subset R$.

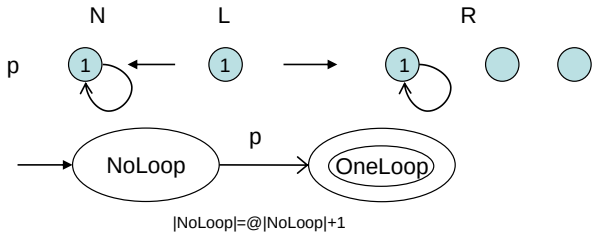

Figure 12: A rule which should terminate simply, but does not terminate, with $N \subset R$.

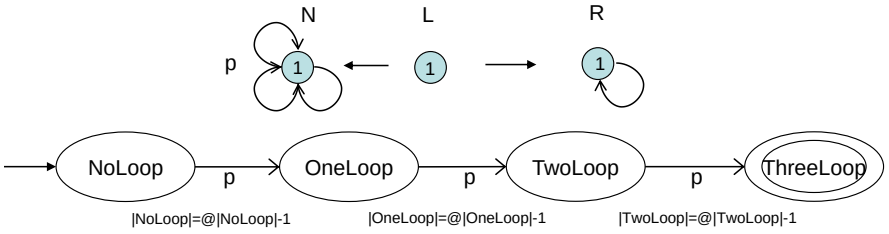

Figure 13: A terminating rule, with $R \subset N$.

There are several cases between the two extremes of the Main Result, for rules which are of type **may terminate**. For these rules, depending on the classification of the matches hosted by a graph G, and the choice of the matches to use the application of the rule, one can statically define if a certain sequence of matches will make the rule terminate or not.

This is the case of the rule in Figure 5 whose labelled transition system is shown, with labels indicating the number of matches, in Figure 14.

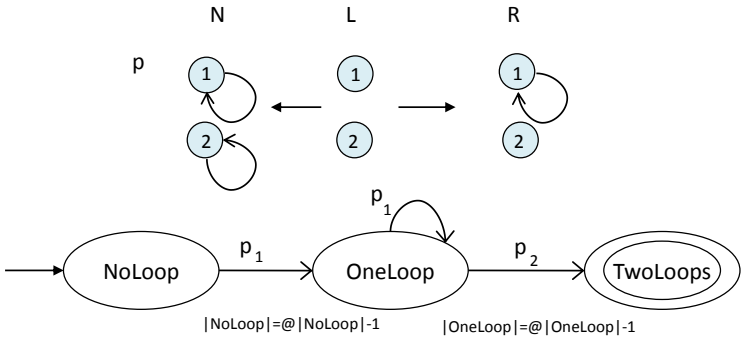

Figure 14: A rule which may terminate, with $R \subset N$.

5 Related Work

Termination of (string) rewriting systems has been studied for over 30 years (see [DM79] for example). Much more recent is the interest in termination of graph transformation systems.

One of the earliest applications is to program optimization and can be found in [Ass00], (submitted for publication a few years earlier) where termination criteria are defined for 2 specific types of rules. One kind is a deleting rule, which must remove at least one item from a specific subgraph: since graphs are finite, the removal must eventually stop. The other kind is a non-deleting rule that must add at least one edge incident to a node with a specific label: since no pair of nodes can have more than one edge with the same label, the addition must eventually stop and so is the applicability.

The general problem of termination for graph rewriting has been tackled by Detlef Plump in [Plu98], where he proves that it is an undecidable problem. Although the framework deals only with 'plain' transformation rules (i.e., without application conditions), we expect the result to hold in general, for example by using trivial conditions always satisfied.

Ad hoc sufficient conditions have been analyzed for special cases. In layered graph transformation systems [EEd+06] the different types of rules are grouped, establishing an application order. In each of the 2 kinds of layers (deleting and non-deleting) there are no infinite derivation sequences with injective matchings. Each rule in a deletion layer must delete at least one item, but not a newly created one. Each rule in a non-deletion layer cannot delete items, cannot be applied twice with the same match and cannot use a newly created item for the match. A finite number of layers and a finite initial graphs guarantee termination.

More recent research [VVE+06] uses a similar idea to the one presented here. A Graph Transformation System is abstracted by ignoring certain structure in a graph and used to define a Petri Net to represent the number of elements of a certain type. Transitions correspond to rule application with 'consumption' of elements (and reduction of tokens). Termination of the GTS corresponds then to the Petri Net exhausting its tokens.

6 Concluding Remarks

We have discussed an approach to analyze termination properties of specific kinds of graph transformations. We have focused on the termination of a single rule p given by an expression of the form **asLongAsPossible** p **end**, for a non-deleting rule p. Termination of plain transformation rules (i.e., rules without application conditions) usually depends upon a function which measures the consumption of a finite commodity and whose value decreases at each application of the rule. When application conditions are present, we can also measure the (hopefully decreasing) distance between the context and the negative application condition. This is what the steps in the labelled transition system represent.

The only morphisms used in this paper are total and injective. We can relax this condition, especially for the NAC $n\colon L \to N$ by allowing a non-injective one, and then by requiring that for each $i = 1,\ldots,k$, the morphisms $L \overset{h_i^L}{\to} H_i^p \overset{h_N^i}{\to} N$ are such that h_i^L is injective and h_N^i satisfies the Gluing Condition with respect to h_i^L. This allows us to avoid taking into account all the intermediate graphs which differ only by an arbitrary number of copies of spurious elements, generated at each application and then collapsed in N.

The examples presented in this paper are necessarily small. What we have not investigated (yet) is the feasibility of the approach to real problems, and in particular the complexity of the labelled transition system relatively to the size of the negative application conditions and a systematic way to construct it.

The next step is to extend the approach to multiple negative application conditions. Some preliminary results are encouraging and we expect to be able to adapt the approach to several rules, each with a single NAC. The case of a rule with several NACs can then be reduced to that of a set of rules, all sharing the same morphism but with different NACs. We are also investigating the termination problem for rule sequences using the interaction of the components.

Although the discussion and the examples are stated in terms of graphs, no specific properties of graphs are used, but only (mono)morphisms and their extensions. The approach can easily be extended to model transformations in high-level replacement (HLR) systems, i.e. algebraic rewriting systems operating on objects and morphisms in adhesive HLR categories [EPPH06].

Bibliography

[Ass00] U. Assmann. Graph rewrite systems for program optimization. *ACM Trans. Program. Lang. Syst.* 22(4):583–637, 2000. doi:http://doi.acm.org/10.1145/363911.363914

[BHP06] P. Bottoni, K. Hoffmann, F. Parisi-Presicce. Termination of Algebraic Rewriting with Inhibitors. In Karsai and Taentzer (eds.), *Proc. GraMoT 2006*. ECEASST 4. 2006.

[BHPT05] P. Bottoni, K. Hoffmann, F. Parisi-Presicce, G. Taentzer. High-Level Replacement Units and their Termination Properties. *Journal of Visual Languages and Computing* 16:485–507, 2005.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation* 3(1& 2):69–115, 1987. Corrigendum: 4,3 (Dec. 1987), 409-410.

[DM79] N. Dershowitz, Z. Manna. Proving termination with multiset orderings. *Commun. ACM* 22(8):465–476, 1979.
doi:http://doi.acm.org/10.1145/359138.359142

[EEd+06] H. Ehrig, K. Ehrig, J. deLara, G. Taentzer, D. Varro, S. Varro-Gyapay. Termination Criteria for Model transformation. In *Proc. FASE 2005*. LNCS 3442, pp. 49–63. Springer, 2006.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, 2006.

[EPPH06] H. Ehrig, J. Padberg, U. Prange, A. Habel. Adhesive High-Level Replacement Systems: A New Categorical Framework for Graph Transformation. *Fundamenta Informaticae* 74(1):1–29, 2006.

[KHE03] J. M. Küster, R. Heckel, G. Engels. Defining and validating transformations of UML models. In *Proc. HCC 2003*. Pp. 145–152. IEEE Computer Society, 2003.

[Plu98] D. Plump. Termination of graph rewriting is undecidable. *Fundamenta Informaticae* 33(2):201–209, 1998.

[VVE+06] D. Varro, S. Varro-Gyapay, H. Ehrig, U. Prange, G. Taentzer. Termination analysis of Model transformations by Petri Nets. In *Proc. ICGT 2006*. LNCS 4178, pp. 260–274. Springer, 2006.