

# Sufficient Criteria for Applicability and Non-Applicability of Rule Sequences

Leen Lambers<sup>a,1</sup> Hartmut Ehrig<sup>a,2</sup> Gabi Taentzer<sup>b,3</sup>

<sup>a</sup> *Institut für Softwaretechnik und Theoretische Informatik  
Technische Universität Berlin, Germany*

<sup>b</sup> *Fachbereich Mathematik und Informatik  
Philipps-Universität Marburg, Germany*

---

Abstract

In several rule-based applications using graph transformation as underlying modeling technique the following questions arise: How can one be sure that a specific sequence of rules is applicable (resp. not applicable) on a given graph? Of course, it is possible to use a trial and error strategy to find out the answer to these questions. In this paper however, we will formulate suitable sufficient criteria for applicability and other ones for non-applicability. These criteria can be checked in a static way i.e. without trying to apply the whole rule sequence explicitly. Moreover if a certain criterion is not satisfied, then this is an indication for reasons why rule sequences may or may not be applicable. Consequently it is easier to rephrase critical rule sequences. The results are formulated within the framework of double pushout (DPO) graph transformations with negative application conditions (NACs).

*Keywords:* graph transformation, analysis, applicability of rules

---

## 1 Introduction

When analyzing integrated specifications of rules with control structures [4,7] for consistency, a considerable amount of rule sequences is to be checked for applicability (resp. non-applicability). Hence, it is not appropriate to check the applicability or non-applicability of each rule sequence explicitly. Therefore static analysis techniques are desirable. Statically means that it is not necessary to use a trial and error strategy and therefore backtracking can be avoided. In the following, we present sufficient criteria for the applicability and for the non-applicability of a rule sequence to a graph. These criteria can be checked in a static way, since they are based mainly on the dependency or independency of rules. Moreover the non-satisfaction of one of the criteria gives a hint to the reason for a rule sequence to be applicable or

---

<sup>1</sup> Email: [leen@cs.tu-berlin.de](mailto:leen@cs.tu-berlin.de)

<sup>2</sup> Email: [ehrig@cs.tu-berlin.de](mailto:ehrig@cs.tu-berlin.de)

<sup>3</sup> Email: [taentzer@mathematik.uni-marburg.de](mailto:taentzer@mathematik.uni-marburg.de)

inapplicable. Applicability criteria have been studied also in [9] for simple digraphs using matrix graph grammars.

As example we consider a simple Mutual Exclusion Algorithm implemented by graph transformation rules with a start graph  $G$  and type graph presented in Fig. 1. The algorithm enables two processes to access a resource according to the mutual exclusion principle. The purpose of this algorithm is to control the usage of the resource such that at most one process holds the resource at time (safety property). Furthermore, if a process demonstrates a request for the resource it should be served eventually (liveness property). In the following, we will check safety (resp. liveness) by checking if certain rule sequences are not applicable (resp. applicable) on the start graph.

The paper is structured as follows: Section 2 presents preliminaries. At first we repeat the main definitions for rule-based transformations with means of double pushout (DPO) graph transformations [1] with negative application conditions (NACs) [3]. Then we define asymmetric dependency and independency on the level of transformations. Therefrom we can deduce the concept of asymmetric independency for rules. In Section 3 and 4 we define sufficient criteria for applicability and non-applicability of rule sequences. Section 5 takes up the example again and shows how to check the criteria presented in Section 3 and 4 with some support of the graph transformation tool AGG [8].

## 2 Independency and Dependency of Rules

### 2.1 Graph Transformation with NACs

We repeat the basic definitions for double pushout graph transformation with negative application conditions (NACs). A graph rule holding a NAC  $n$  can be applied on a graph  $G$  only if the forbidden structure expressed by  $n$  is not present in  $G$ .

**Definition 2.1** [graph, graph morphism, rule] A *graph*  $G = (G_E, G_V, s, t)$  consists of a set  $G_E$  of edges, a set  $G_V$  of vertices and two mappings  $s, t : G_E \rightarrow G_V$ , assigning to each edge  $e \in G_E$  a source  $q = s(e) \in G_V$  and target  $z = t(e) \in G_V$ . A *graph morphism* (short *morphism*)  $f : G_1 \rightarrow G_2$  between two graphs  $G_i = (G_{i,E}, G_{i,V}, s_i, t_i)$ , ( $i = 1, 2$ ) is a pair  $f = (f_E : G_{E,1} \rightarrow G_{E,2}, f_V : G_{V,1} \rightarrow G_{V,2})$  of mappings, such that  $f_V \circ s_1 = s_2 \circ f_E$  and  $f_V \circ t_1 = t_2 \circ f_E$ . A morphism  $f : G_1 \rightarrow G_2$  is injective (resp. surjective) if  $f_V$  and  $f_E$  are injective (resp. surjective) mappings. A graph transformation *rule*  $p : L \xleftarrow{l} K \xrightarrow{r} R$  consists of a rule name  $p$  and a pair of injective morphisms  $l : K \rightarrow L$  and  $r : K \rightarrow R$ . The graphs  $L, K$  and  $R$  are called the left-hand side (LHS), the interface, and the right-hand side (RHS) of  $p$ , respectively.

**Definition 2.2** [rule and transformation with NACs, applicability of rule with NACs]

- A *negative application condition* or  $NAC(n)$  on  $p$  for a rule  $p : L \xleftarrow{l} K \xrightarrow{r} R$  ( $l, r$  injective) is an arbitrary morphism  $n : L \rightarrow N$ . A morphism  $g : L \rightarrow G$  satisfies  $NAC(n)$  on  $L$ , written  $g \models NAC(n)$ , if and only if  $\exists q : N \rightarrow G$  injective such

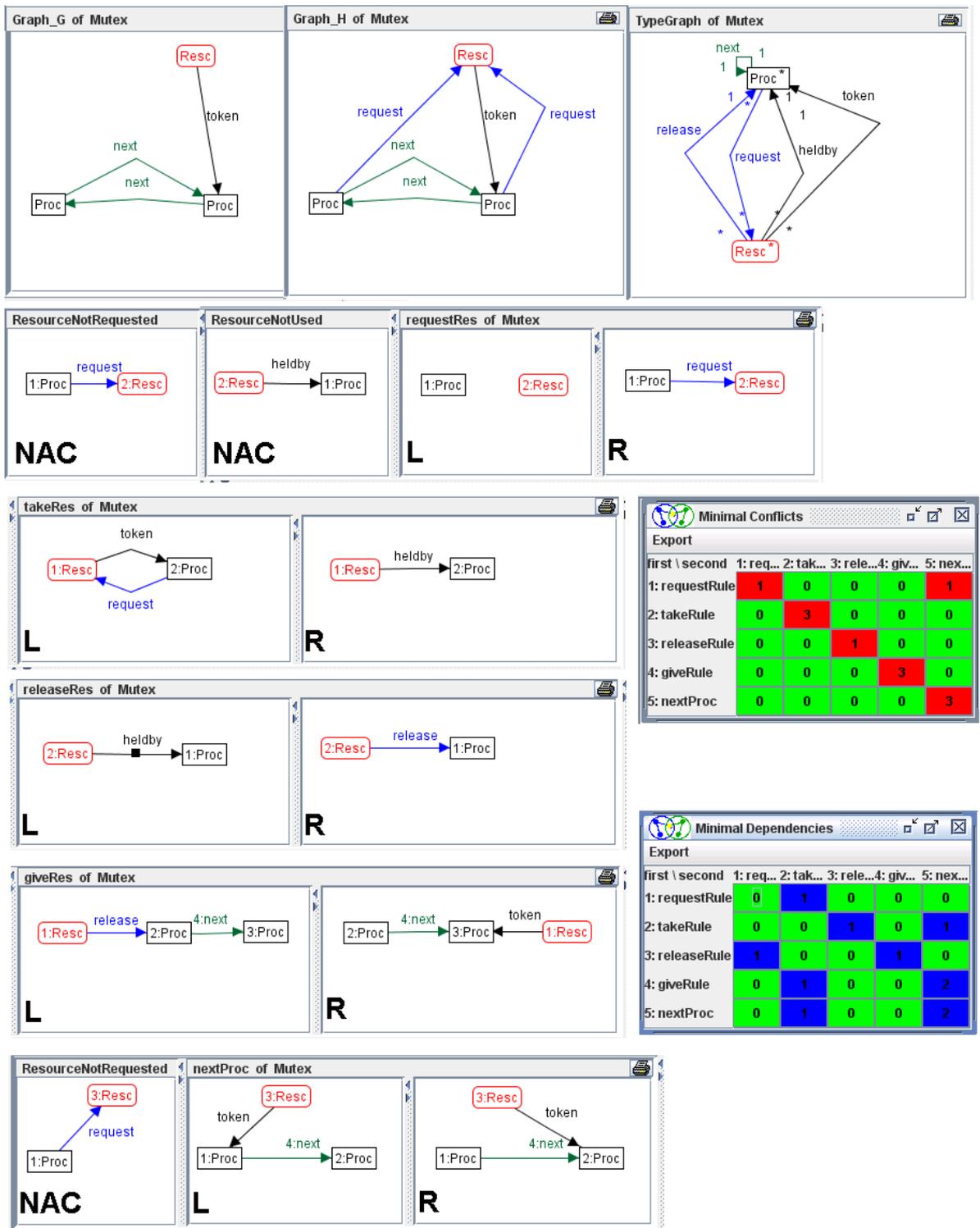


Figure 1. Start graph G, graph H and type graph of Mutual Exclusion - Rules *requestRes*, *takeRes*, *releaseRes*, *giveRes* and *nextProc* - Dependency and Conflict Matrix in AGG

that  $q \circ n = g$ .

$$\begin{array}{ccc} L & \xrightarrow{n} & N \\ \downarrow g & & \downarrow q \\ G & \xleftarrow{X_q} & N \end{array}$$

A set of NACs on  $p$  is denoted by  $NAC_p = \{NAC(n_i) | i \in I\}$ . A morphism  $g : L \rightarrow G$  satisfies  $NAC_p$  if and only if  $g$  satisfies all single NACs on  $p$  i.e.  $g \models NAC(n_i) \forall i \in I$ .

- A rule  $(p, NAC_p)$  with NACs is a rule with a set of NACs on  $p$ .
- A direct transformation  $G \xrightarrow{p, g} H$  via a rule  $p : L \leftarrow K \rightarrow R$  with  $NAC_p$  and a match  $g : L \rightarrow G$  consists of the double pushout [1] (DPO)

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ \downarrow g & & \downarrow & & \downarrow h \\ G & \longleftarrow & D & \longrightarrow & H \end{array}$$

where  $g$  satisfies  $NAC_p$ , written  $g \models NAC_p$ . Since pushouts along injective morphisms always exist, the DPO can be constructed if the pushout complement of  $K \rightarrow L \rightarrow G$  exists. If so, we say that the match  $g$  satisfies the gluing condition of rule  $p$ . If there exists a morphism  $g : L \rightarrow G$  which satisfies the gluing condition and  $g \models NAC_p$  we say that rule  $p$  is *applicable* on  $G$  via the match  $g$ .

**Example 2.3** Consider the graph transformation rules of the Mutual Exclusion Algorithm presented in Fig. 1. Rule *requestRes* expresses a request of a process to access the resource if this process has not requested the resource yet or is using the resource already. Rule *takeRes* enables a process to start using the resource if this process possesses a token. *releaseRes* can release the resource again and *giveRes* passes the token to the other process after releasing. Finally rule *nextProc* can pass a token from one process to the other one as long as the first one has not expressed a request.

In [6] it is proven that to each rule  $p$  with  $NAC_p$  there exists an inverse rule  $p^{-1}$  with  $NAC_{p^{-1}}$  such that each transformation can be inverted. The following definition shows how to construct from  $NAC_p$  on rule  $p$  equivalent NACs  $NAC_{p^{-1}}$  on the inverse rule  $p^{-1}$  [2]:

**Definition 2.4** [construction of NACs on inverse rule] For each  $NAC(n_i)$  with  $n_i : L \rightarrow N_i$  on  $p = (L \leftarrow K \rightarrow R)$ , the equivalent NAC  $R_p(NAC(n_i))$  on  $p^{-1} = (R \leftarrow K \rightarrow L)$  is defined in the following way:

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ \downarrow n_i & (1) & \downarrow & (2) & \downarrow n'_i \\ N_i & \longleftarrow & Z & \longrightarrow & N'_i \end{array}$$

- If the pair  $(K \rightarrow L, L \rightarrow N_i)$  has a pushout complement, we construct  $(K \rightarrow Z, Z \rightarrow N_i)$  as the pushout complement (1). Then we construct pushout (2) with the morphism  $n'_i : R \rightarrow N'_i$ . Now we define  $R_p(NAC(n_i)) = NAC(n'_i)$ .

- If the pair  $(K \rightarrow L, L \rightarrow N_i)$  does not have a pushout complement, we define  $R_p(NAC(n_i)) = true$ .

For each set of NACs on  $p$ ,  $NAC_p = \cup_{i \in I} NAC(n_i)$  we define the following set of NACs on  $p^{-1}$ :  $NAC_{p^{-1}} = R_p(NAC_p) = \cup_{i \in I'} R_p(NAC(n_i))$  with  $i \in I'$  if and only if the pair  $(K \rightarrow L, L \rightarrow N_i)$  has a pushout complement.

With means of this construction we can formulate the following fact [2,6]:

**Fact 2.5 (inverse direct transformation with NACs)** *For each direct transformation with NACs  $G \Rightarrow H$  via a rule  $p : L \leftarrow K \rightarrow R$  with  $NAC_p$  a set of NACs on  $p$ ,  $H \Rightarrow G$  is a direct transformation with NACs via the inverse rule  $p^{-1}$  with  $NAC_{p^{-1}}$ .*

**Example 2.6** Consider rule *requestRes*. The inverse rule of *requestRes* deletes the *request* edge between a process and the resource if there is no other *request* edge nor a *heldby* edge between this process and the resource.

## 2.2 Independency and Dependency of Rules

The (non-)applicability of a rule sequence on a graph  $G$  is affected by dependencies and independencies between the rules in the rule sequence. In particular it is affected by asymmetric dependencies as explained in this section.

Two direct transformations are in conflict if they are not parallel independent. In [5] a Conflict Characterization is given in which different reasons for conflicting transformations become clear. It is possible that one transformation deletes (resp. produces) a structure which is used (resp. forbidden) by the other one and the other way round. In particular we can deduce that two direct transformations are in conflict if *at least* one transformation depends on the other one. This asymmetric parallel dependency and on the contrary asymmetric parallel independency are expressed by the following definition.

**Definition 2.7** [asymmetrically parallel dependent and independent transformations] A direct transformation  $G \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  is *asymmetrically parallel dependent* on  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  if:

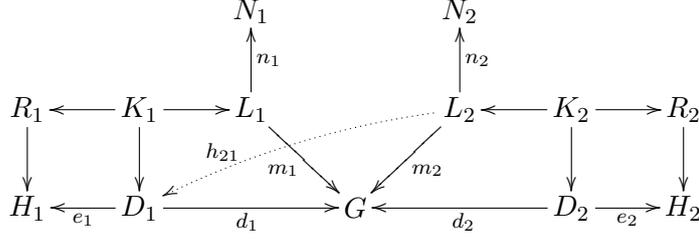
- (i)  $\nexists h_{21} : L_2 \rightarrow D_1 : d_1 \circ h_{21} = m_2$  (*delete-use-conflict*)

OR

- (ii) there exists a unique  $h_{21} : L_2 \rightarrow D_1 : d_1 \circ h_{21} = m_2$ , but  $e_1 \circ h_{21} \not\equiv NAC_{r_2}$  (*produce-forbid-conflict*).

A direct transformation  $G \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  is *asymmetrically parallel independent* on  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  if  $G \xrightarrow{(r_2, m_2)} H_2$  is not asymmetrically parallel dependent on  $G \xrightarrow{(r_1, m_1)} H_1$ . This means in particular that  $\exists h_{21} : L_2 \rightarrow D_1 :$

$d_1 \circ h_{21} = m_2$  such that  $e_1 \circ h_{21} \models NAC_{r_2}$ .



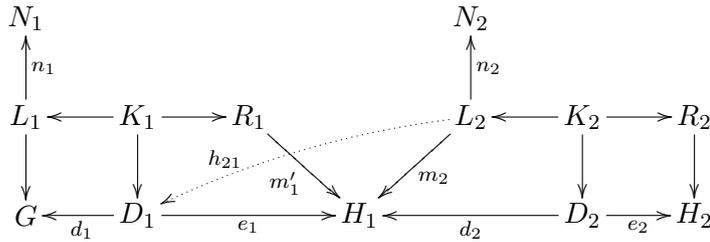
If a sequence of two direct transformations is not sequentially independent this is because either the second transformation depends on the first one or the other way round. The case in which the second transformation depends (resp. is independent) on the first one is described by asymmetric sequential dependency (resp. independency) as in the following definition. This case occurs in particular when the first transformation produces a structure which is used by the second one or the first transformation deletes a structure forbidden by the second one. In other words the first transformation triggers the second one.

**Definition 2.8** [asymmetrically sequential dependent and independent transformations] A direct transformation  $H_1 \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  is *asymmetrically sequential dependent* on  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  if

(i)  $\nexists h_{21} : L_2 \rightarrow D_1 : e_1 \circ h_{21} = m_2$  (*produce-use-dependency*)  
OR

(ii) there exists a unique  $h_{21} : L_2 \rightarrow D_1 : e_1 \circ h_{21} = m_2$ , but  $d_1 \circ h_{21} \not\models NAC_{r_2}$  (*delete-forbid-dependency*).

A direct transformation  $H_1 \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  is *asymmetrically sequential independent* on  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  if  $H_1 \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  is not asymmetrically sequential dependent on  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$ . This means in particular that  $\exists h_{21} : L_2 \rightarrow D_1 : e_1 \circ h_{21} = m_2$  such that  $d_1 \circ h_{21} \models NAC_{r_2}$ .



Now we can define asymmetric parallel (resp. sequential) independency for rules by demanding that the corresponding transformations are asymmetrically independent. Analogously it is possible to define asymmetric parallel (resp. sequential) dependency for rules.

**Definition 2.9** [asymmetrically parallel independent rules] The rule  $r_2$  is *asymmetrically parallel independent* on  $r_1$  if every transformation  $G \xrightarrow{(r_2, m_2)} H_2$  via  $r_2$

with  $NAC_{r_2}$  is asymmetrically parallel independent on any other transformation  $G \xrightarrow{(r_1, m_1)} H_1$  via  $r_1$  with  $NAC_{r_1}$ .

**Definition 2.10** [asymmetrically sequential independent rules] A rule  $r_2$  is *asymmetrically sequential independent* on  $r_1$  if every transformation  $H_1 \xrightarrow{(r_2, m_2)} H_2$  via  $r_2$  with  $NAC_{r_2}$  is asymmetrically sequential independent on any other transformation  $G \xrightarrow{(r_1, m_1)} H_1$  via  $r_1$  with  $NAC_{r_1}$ .

**Example 2.11** Rule *nextProc* is asymmetric parallel dependent on *requestRes*, since rule *requestRes* produces a *request* edge which is forbidden by rule *nextProc*. On the contrary, rule *requestRes* is asymmetric parallel independent on *nextProc*, since rule *nextProc* neither deletes anything what can be used by *requestRes* nor produces anything forbidden by *requestRes*. Rule *requestRes* is asymmetric sequentially dependent on rule *releaseRes*, since *releaseRes* deletes a *heldby* edge which is forbidden by *requestRes*. On the contrary, rule *releaseRes* is asymmetric sequentially independent on rule *requestRes*, since *requestRes* neither produces anything what can be used by *releaseRes* nor deletes anything forbidden by *requestRes*.

For the criteria defined in the next section we need a special case of asymmetric sequential dependency for the case without NACs. It is possible namely that a rule  $r_1$  produces everything which is needed by  $r_2$  regardless of what is already present in the corresponding transformations.

**Definition 2.12** [purely sequential dependent rules] A rule  $r_2 : L_2 \leftarrow K_2 \rightarrow R_2$  is *purely sequential dependent* on  $r_1 : L_1 \leftarrow K_1 \rightarrow R_1$  if  $r_2$  is a rule without NACs and there exists an injective morphism  $l_{21} : L_2 \rightarrow R_1$ .

**Remark 2.13**  $r_2$  is asymmetrically sequential dependent on  $r_1$ , if  $r_2$  is purely sequential dependent on  $r_1$  and the following mild assumptions are satisfied: a morphism  $k_{21} : L_2 \rightarrow K_1$  does not exist such that  $r_1 \circ k_{21} = l_{21}$ ,  $r_2$  is non-deleting on nodes and  $id : R_1 \rightarrow R_1 \models NAC_{r_1^{-1}}$ .

**Example 2.14** Rule *releaseRes* is purely sequential dependent on rule *takeRes*, because its LHS can be embedded completely into the RHS of *takeRes*. Rule *takeRes* is asymmetric sequentially, but not purely dependent on rule *requestRes*, since *requestRes* does not produce a *token* edge.

We can analogously derive from the usual definition of sequential (resp. parallel) independence of transformations with NACs [5] (Def. A.1 in Appendix A) the definition for sequential (resp. parallel) independency on rules by demanding that all existing transformations via these rules are sequentially (resp. parallel) independent. Recall that for each pair of parallel independent transformations with NACs  $H_1 \xleftarrow{r_1, m_1} G \xrightarrow{r_2, m_2} H_2$ , there are an object  $G'$  and direct transformations  $H_1 \xrightarrow{r_2, m'_2} G'$  and  $H_2 \xrightarrow{r_1, m'_1} G'$  such that  $G \xrightarrow{r_1, m_1} H_1 \xrightarrow{r_2, m'_2} G'$  and  $G \xrightarrow{r_2, m_2} H_2 \xrightarrow{r_1, m'_1} G'$  are sequentially independent. The transformations can thus be performed in any order with the same result (Theorem A.3 in Appendix A). Note that we have the following relationship between parallel and sequential independency:  $G \xrightarrow{r_1} H_1 \xrightarrow{r_2} H_2$  are sequentially independent iff  $G \xleftarrow{r_1^{-1}} H_1 \xrightarrow{r_2} H_2$  are parallel independent. In the next

section we need sequential independency of a rule pair in order to be able to switch adjacent rules in a rule sequence.

**Definition 2.15** [parallel and sequential independent rules] Rules  $r_1$  and  $r_2$  are *parallel independent* if each pair of transformations  $G \xrightarrow{(r_1, m_1)} H_1$  via  $r_1$  with  $NAC_{r_1}$  and  $G \xrightarrow{(r_2, m_2)} H_2$  via  $r_2$  with  $NAC_{r_2}$  is parallel independent. The pair of rules  $(r_1, r_2)$  is *sequentially independent* if each sequence of transformations  $G \xrightarrow{(r_1, m_1)} H_1$  via  $r_1$  with  $NAC_{r_1}$  and  $H_1 \xrightarrow{(r_2, m_2)} G'$  via  $r_2$  with  $NAC_{r_2}$  is sequentially independent.

**Remark 2.16** Note that the following correspondences between asymmetric parallel (resp. sequential) independency and parallel (resp. sequential) independency exists. Rules  $r_1$  and  $r_2$  are parallel independent if and only if  $r_1$  is asymmetrically parallel independent of  $r_2$  and  $r_2$  is asymmetrically parallel independent of  $r_1$ . The rule pair  $(r_1, r_2)$  is sequentially independent if and only if  $r_2$  is asymmetrically sequential independent on  $r_1$  and  $r_1^{-1}$  is asymmetrically sequential independent on  $r_2^{-1}$ .

**Example 2.17** The rule pair  $(requestRes, nextProc)$  is sequentially independent. Note that the NAC of rule  $nextProc$  forbids a process to shift the token if this process expressed a request. Thus whenever  $(requestRes, nextProc)$  can be applied in this order the request is expressed by the process to which the token is shifted. This is equivalent to first shifting the token to this process which then expresses a request.

### 3 Applicability of Rule Sequences

#### 3.1 Applicability Criteria

Let  $s : r_1 r_2 \dots r_n$  be a sequence of  $n$  rules and  $G_0$  a graph on which this sequence should be applied. The criteria defined in the following definition guarantee this applicability. The initialization criterion is trivial, since it just requires the first rule being applicable to graph  $G_0$ . The no node-deleting rules criterion avoids dangling edges. The third criterion ensures that the applicability of a rule  $r_i$  is not impeded by one of the predecessor rules  $r_j$  of  $r_i$ . Criterion 4a will be satisfied if rule  $r_i$  is purely sequential dependent from a rule  $r_j$  occurring somewhere before  $r_i$  in the sequence  $s$ . In this case  $r_j$  triggers the applicability of  $r_i$  regardless of what is present already in the start graph  $G_0$ . As soon as the sequential dependencies are not pure though this criterion is not satisfiable. Therefore we have also a more general criterion 4b. It ensures the applicability of a rule  $r_i$  needing some subgraph of a direct predecessor rule together with parts of the start graph  $G_0$ . This is expressed by the fact that a concurrent rule  $r_c$  of  $r_{i-1}$  and  $r_i$  exists which is applicable on the start graph  $G_0$ . The construction of a concurrent rule with NACs is explained in [6]. The correctness of the criteria is described in Theorem 3.2 which is proven in the Appendix B.

**Definition 3.1** [applicability criteria] Given a sequence  $s : r_1 r_2 \dots r_n$  of  $n$  rules and a graph  $G_0$ . Then we define the following applicability criteria for  $s$  on  $G_0$ :

- (i)  $r_1$  is applicable on  $G_0$  via the injective match  $m_1 : L_1 \rightarrow G_0$  (*initialization*)

- (ii) each rule occurring in  $s$  is non-deleting on nodes (*no node-deleting rules*)
- (iii)  $\forall r_i, r_j$  in  $s$  with  $1 \leq j < i \leq n$ ,  $r_i$  is asymmetrically parallel independent on  $r_j$  (*no impeding predecessors*)
- (iv)  $\forall r_i$  in  $s$  with  $1 < i \leq n$  which are not applicable on  $G_0$  via an injective match
  - (a) there exists a rule  $r_j$  in  $s$  which is applicable via an injective match on  $G_0$  with  $1 \leq j < i \leq n$  and  $r_i$  is purely sequential dependent on  $r_j$  (*pure enabling predecessor*), which especially means that  $r_i$  has no NACs  
OR
  - (b) there exists a concurrent rule  $r_c$  of  $r_{i-1}$  and  $r_i$  such that  $r_c$  is applicable via an injective match on  $G_0$  and  $r_c$  is asymmetrically parallel independent on  $r_j$  for all  $j < (i - 1)$  and  $r_j$  is asymmetrically parallel independent on  $r_c$  for all  $i < j \leq n$ . (*direct enabling predecessor*)

**Theorem 3.2 (correctness of applicability criteria)** *Given  $s : r_1 r_2 \dots r_n$  a sequence of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 3.1 are satisfied for rule sequence  $s$  and graph  $G_0$ , then this rule sequence is applicable on  $G_0$  with injective matching i.e. there exists a graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$  with injective matching.*

### 3.2 Applicability of Summarized Rule Sequences

The 4th criterion in Def. 3.1 will be satisfied only, if the rule  $r_i$  is purely sequential dependent from one single rule  $r_j$  occurring before  $r_i$  in the sequence or if it is asymmetrically sequential dependent on the rule  $r_{i-1}$ . In some transformations though a rule needs not only a subgraph from one single predecessor, but from several ones. For these cases the criteria could then be satisfied by a sequence in which exactly these rules are summarized to a concurrent rule. Note that the correctness of the following theorem is proven in the Appendix B.

**Theorem 3.3 (summarized rule sequences)** *Given  $s : r_1 r_2 \dots r_n$  a sequence of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 3.1 are satisfied for a rule sequence  $s' : r'_1 r'_2 \dots r'_m$  with  $m < n$  in which neighbored rules in  $s$  can be summarized by a concurrent rule, then the original rule sequence  $s$  is applicable on  $G_0$  with injective matching i.e. there exists a graph transformation  $G_0 \xrightarrow{r'_1} G_1 \dots G_{n-1} \xrightarrow{r'_n} G_n$  with injective matching.*

### 3.3 Applicability of Shift-Equivalent Rule Sequences

If it is not possible to satisfy criterion 4b in Def. 3.1 for a rule sequence  $s : r_1 r_2 \dots r_n$ , then it is still possible to exploit shift-equivalence. This is because criterion 4b could be satisfiable for a rule sequence  $s'$  in which a normal predecessor is shifted to be a direct one. Note that a rule  $r_j$  in  $s$  can be switched with a rule  $r_{j+1}$  only if the pairs of rules  $(r_j, r_{j+1})$  and  $(r_{j+1}, r_j)$  is sequentially independent. If the criteria then hold though for rule sequence  $s'$  in which some rules have been shifted, they hold also for the original rule sequence  $s$ .

**Definition 3.4** [shift-equivalent rule sequences] A rule sequence  $s'$  is *shift-equivalent* with a rule sequence  $s : r_1 r_2 \dots r_m$  if  $s'$  can be obtained by switching

rules  $r_j$  with  $r_{j+1}$  and the switching is allowed only if  $(r_j, r_{j+1})$  and  $(r_{j+1}, r_j)$  are sequentially independent according to Def. 2.15.

**Theorem 3.5 (checking shift-equivalent rule sequences)** *If the criteria in Def. 3.1 are satisfied for a rule sequence  $s : r_1 r_2 \dots r_n$  and a graph  $G_0$ , then all shift-equivalent rule sequences as defined in Def. 3.4 are applicable on  $G_0$  with injective matching as well with the same result.*

**Proof** This follows directly from Def. 3.4, the Local Church-Rosser Theorem with NACs [5] (Theorem A.3 in Appendix A), Def. 2.15 and Theorem 3.2.  $\square$

**Remark 3.6** Note that a somewhat weaker version of this theorem holds as well. Namely, suppose that we want to prove applicability of a rule sequence  $s : r_1 r_2 \dots r_n$  to a graph  $G_0$ . Then it is sufficient to show the satisfaction of the criteria in Def. 3.1 for a rule sequence  $s'$  which can be deduced from  $s$  by switching forward rule  $r_{i+1}$  with  $r_i$  only if rule pair  $(r_{i+1}, r_i)$  is sequentially independent.

## 4 Non-Applicability of Rule Sequences

Let  $s : r_1 r_2 \dots r_n$  be a sequence of  $n$  rules and  $G_0$  a graph. The satisfaction of the following criteria for  $s$  and  $G_0$  guarantee that the sequence  $s$  will not be applicable on  $G_0$ . Criterion 1 is trivial, since it just requires the first rule being non-applicable to graph  $G_0$ . Criterion 2 checks if predecessors for a rule  $r_i$  which is not applicable already on  $G_0$  are present in the sequence such that they can trigger the applicability of  $r_i$ . If not,  $r_i$  will not be applicable and therefore neither the rule sequence will be applicable. Note that the correctness of the following criteria is proven in Appendix B.

**Definition 4.1** [non-applicability criteria] Given  $s : r_1 r_2 \dots r_n$  a sequence of  $n$  rules and a graph  $G_0$ . Then we define the following non-applicability criteria for  $s$  on  $G_0$ :

- (i)  $r_1$  is not applicable on  $G_0$  (*initialization error*)  
OR
- (ii)  $\exists r_i$  in  $s$  with  $1 < i \leq n$  such that  $r_i$  is not applicable on  $G_0$  but for all rules  $r_j$  in  $s$  with  $1 \leq j < i \leq n$ ,  $r_i$  is asymmetrically sequential independent on  $r_j$  *no enabling predecessor*.

**Theorem 4.2 (correctness of non-applicability criteria)** *Given a sequence  $s : r_1 r_2 \dots r_n$  of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 4.1 are satisfied for rule sequence  $s$  and graph  $G_0$ , then this rule sequence is not applicable on  $G_0$  i.e. there exists no graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$ .*

Analogously to Theorem 3.5 we can formulate the following theorem expressing that shift-equivalent-rule sequences are not applicable to a graph  $G_0$  if one of the sequences satisfies the non-applicability criteria.

**Theorem 4.3 (checking shift-equivalent rule sequences)** *If the criteria in Def. 4.1 are satisfied for a rule sequence  $s : r_1 r_2 \dots r_n$  and a graph  $G_0$ , then all shift-equivalent rule sequences as defined in Def. 3.4 are not applicable on  $G_0$  either.*

**Proof** This follows directly from Def. 3.4, the Local Church-Rosser Theorem with NACs [5] (Theorem A.3 in Appendix A), Def. 2.15 and Theorem 4.2.  $\square$

**Remark 4.4** Note that a somewhat weaker version of this theorem holds as well. Namely, suppose that we want to prove non-applicability of a rule sequence  $s : r_1 r_2 \dots r_n$  to a graph  $G_0$ . Then it is sufficient to show the satisfaction of the criteria in Def. 4.1 for a rule sequence  $s'$  which can be deduced from  $s$  by switching forward rule  $r_{i+1}$  with  $r_i$  only if rule pair  $(r_i, r_{i+1})$  is sequentially independent.

## 5 Checking the Criteria

### 5.1 Checking for Sequential and Parallel Dependency of Rules in AGG

To check asymmetric sequential and parallel dependency of rules, we compute all corresponding critical pairs by AGG [8]. A critical pair represents the parallel (resp. sequential) dependency of rules in a minimal context. The minimal conflicts (i.e. asymmetric parallel dependencies) are represented in a conflict matrix. The minimal dependencies (i.e. asymmetric sequential dependencies) are represented in a dependency matrix. The entry numbers within these matrices indicate how many minimal conflicts and dependencies, resp. have been found. For the example they are shown in Fig. 1. More precisely, entry  $(r_j, r_i)$  (row, column) in the conflict matrix in AGG describes all  $G \xrightarrow{(r_i, m_i)} H_i$  which are asymmetrically parallel dependent on  $G \xrightarrow{(r_j, m_j)} H_j$  in a minimal context. Entry  $(r_j, r_i)$  in the dependency matrix in AGG describes all  $G \xrightarrow{(r_j, m_j)} H_j \xrightarrow{(r_i, m_i)} G'$  such that the second transformation is asymmetrically sequential dependent on the first one in a minimal context. Note that it is not possible yet to check with AGG if a pair of rules  $(r_j, r_i)$  is sequentially independent as defined in Def. 2.15. It is part of current work though to enrich the dependency matrix with more information and thus enable this possibility.

### 5.2 Checking Applicability Criteria on Mutual Exclusion

We check applicability (i.e. liveness) of the following rule sequences:

***requestRes, takeRes, releaseRes*** should be applicable to the right process in the graph  $G$  shown in Figure 1. We thus check for this rule sequence and graph  $G$  that the applicability criteria in Def. 3.1 are satisfied.

- (i) Rule *requestRes* is applicable to  $G$ .
- (ii) Rule *takeRes* is asymmetrically parallel independent of rule *requestRes*. Furthermore, rule *releaseRes* is asymmetrically parallel independent of *requestRes* as well as *takeRes*.
- (iii) Rules *takeRes, releaseRes, giveRes* are not applicable to  $G$ . Thus, we have to show that their application is enabled by the rule applications performed before. Rule *takeRes* is asymmetrically sequential dependent on *requestRes*, and the concurrent rule of *takeRes* and *requestRes* is applicable on  $G$ . It expresses how a resource can be requested and taken in one step. *requestRes* is thus a direct enabling predecessor for *takeRes*. Moreover rule *releaseRes* is purely sequential dependent on rule *takeRes* and therefore *takeRes* is a pure

enabling predecessor for *releaseRes*.

*requestRes, takeRes, releaseRes, giveRes* is a slightly longer sequence and should still be applicable to  $G$ . It is not possible though to fulfill in particular criterion 4 in Def. 3.1 for this sequence. However, it is still possible to fulfill the applicability criteria for a summarized sequence as proven in Theorem 3.3. In this case, it is possible to satisfy the criteria for the summarized sequence only. It consists of one concurrent rule *requestTakeReleaseGiveRes* which is equal to the rule *nextProc*. *nextProc* is applicable on  $G$  and therefore the original sequence is applicable on  $G$  as well. Thus sometimes an applicable rule sequence as given in this example might be hard to detect. This is because *giveRes* needs some subgraph of the start graph  $G$  which is not needed by the other rules and in addition all rules (except for the first one) are asymmetrically sequential dependent of the previous rule.

### 5.3 Checking Non-Applicability Criteria on Mutual Exclusion

We check non-applicability (i.e. safety) of the following rule sequences:

*requestRes, requestRes, nextProc* specifies a request of both processes and then a token transfer. This sequence should not be applicable on the start graph  $G$  in Fig. 1, since before transferring a token one of the requests should be processed. We check the non-applicability criteria in Def. 4.1 for this rule sequence on  $G$ .

- (i) *requestRes* is applicable on the start graph  $G$ .
- (ii) *nextProc* is not applicable on the start graph  $G$  and is sequentially independent on *requestRes*. Therefore the second criterion is fulfilled and there is no enabling predecessor.

*requestRes, giveRes* specifies a request and then a token shift because the resource has been released. This sequence should not be applicable to the start graph  $G$  in Fig. 1, since the resource is still unused. It is easy to verify that the criteria in Def. 4.1 are fulfilled. Moreover rule pair (*requestRes, giveRes*) and (*giveRes, requestRes*) are sequentially independent. Therefore due to Theorem 4.3 we can conclude directly that sequence *giveRes, requestRes* is not applicable either on  $G$ .

*takeRes, takeRes* specifies a take of the resource by both processes simultaneously. This sequence should not be applicable on the graph  $H$  in Figure 1 in which both processes request the resource. Thus we check the non-applicability criteria in Def. 4.1 for this sequence and graph  $H$ .

- (i) The first rule *takeRes* is applicable to graph  $H$  on the right process.
- (ii) The second rule is again *takeRes* and we just mentioned that it is applicable on the right process in  $H$ . We want to check though for safety reasons that it is impossible to apply the second *takeRes* on the other (i.e. *left*) process simultaneously. A matching of *takeRes* on the left process into  $H$  does not exist. For this kind of matching criterion 2 holds since in addition rule *takeRes* is asymmetrically sequentially independent from itself. This means in particular that the application of the first *takeRes* on the right process does not enable the application of the second *takeRes* on the left process and hence

such a rule application on  $H$  is not possible.

This example demonstrates that sometimes it is necessary to include information about the matches for the rule sequence to be checked for safety. Note moreover that if we would have taken a longer sequence consisting of *requestRes*, *requestRes*, *takeRes* and *takeRes*, we could not have checked the non-applicability of that sequence on the start graph  $G$  by Def. 4.1, since rule *takeRes* is asymmetrically sequentially dependent on rule *requestRes*. Thus it is crucial in this case to select the kernel sequence where the problem is conjectured and in particular this shows us that our criteria are sufficient but not necessary.

## 6 Conclusion and Future Work

In this paper sufficient criteria are formulated for the applicability (resp. non-applicability) of a rule sequence on a graph  $G$ . These criteria can be used to check in a static way applicability resp. non-applicability of rule sequences. Future work is concerned with formulating criteria for rules with attributes, further optimization of the criteria, efficiently checking the satisfaction of the criteria and implementing applicability checks in AGG. Furthermore, we like to evaluate the criteria in larger case studies such as [4,7].

## References

- [1] Corradini, A., U. Montanari, F. Rossi, H. Ehrig, R. Heckel and M. Löwe, *Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach*, in: G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*, World Scientific, 1997 pp. 163–245.
- [2] Ehrig, H., K. Ehrig, A. Habel and K.-H. Pennemann, *Constraints and application conditions: From graphs to high-level structures*, in: F. Parisi-Presicce, P. Bottoni and G. Engels, editors, *Proc. 2nd Int. Conference on Graph Transformation (ICGT'04)*, LNCS 3256 (2004), pp. 287–303.
- [3] Habel, A., R. Heckel and G. Taentzer, *Graph Grammars with Negative Application Conditions*, Special issue of *Fundamenta Informaticae* **26** (1996), pp. 287–313.
- [4] Lambers, L., H. Ehrig, L. Mariani and M. Pezzè, *Iterative model-driven development of adaptable service-based applications*, in: *proceedings of the International Conference on Automated Software Engineering*, 2007.
- [5] Lambers, L., H. Ehrig and F. Orejas, *Conflict Detection for Graph Transformation with Negative Application Conditions*, in: *Proc. Third International Conference on Graph Transformation (ICGT'06)*, LNCS **4178** (2006), pp. 61–76.
- [6] Lambers, L., H. Ehrig, F. Orejas and U. Prange, *Parallelism and concurrency in adhesive high-level replacement systems with negative application conditions*, in: H. Ehrig, J. Pfalzgraf and U. Prange, editors, *Workshop on Applied and Computational Category Theory (ACCAT'07)* (2007).
- [7] Mehner, K., M. Monga and G. Taentzer, *Interaction Analysis in Aspect-Oriented Models*, in: *Proc. 14th IEEE International Requirements Engineering Conference* (2006), pp. 66–75.
- [8] Taentzer, G., *AGG: A Graph Transformation Environment for Modeling and Validation of Software*, in: J. Pfaltz, M. Nagl and B. Boehlen, editors, *Application of Graph Transformations with Industrial Relevance (AGTIVE'03)*, LNCS 3062, Springer, 2004 pp. 446 – 456.  
URL [AGG-Homepage: http://tfs.cs.tu-berlin.de/agg](http://tfs.cs.tu-berlin.de/agg)
- [9] Velasco, P. and J. de Lara, *Using matrix graph grammars for the analysis of behavioural specifications: Sequential and parallel independence*, in: *Jornadas de Programacion y Lenguajes (PROLE2007)*, 2007.

## A Independency of Transformations

We repeat the definition of parallel and sequential independency for transformations and afterwards the Local Church-Rosser Theorem for independent transformations.

**Definition A.1** [parallel independent and sequential independent transformations]

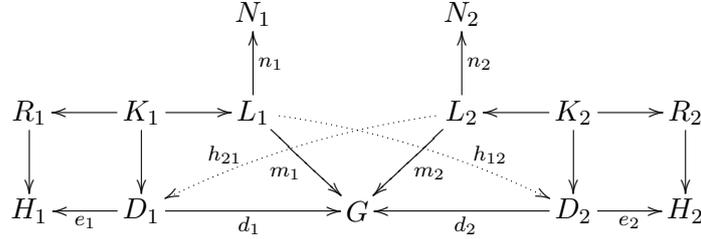
Two direct transformations  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  and  $G \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  are *parallel independent* if

$$\exists h_{12} : L_1 \rightarrow D_2 \text{ s.t. } (d_2 \circ h_{12} = m_1 \text{ and } e_2 \circ h_{12} \models NAC_{r_1})$$

and

$$\exists h_{21} : L_2 \rightarrow D_1 \text{ s.t. } (d_1 \circ h_{21} = m_2 \text{ and } e_1 \circ h_{21} \models NAC_{r_2})$$

as in the following diagram:



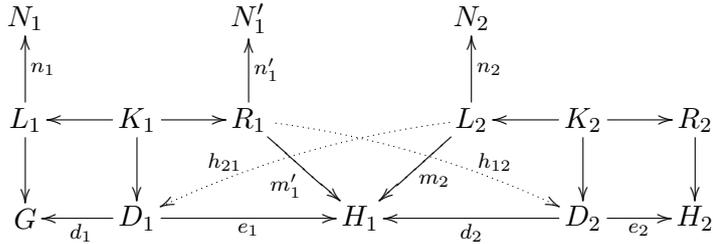
Two direct transformations  $G \xrightarrow{(r_1, m_1)} H_1$  with  $NAC_{r_1}$  and  $H_1 \xrightarrow{(r_2, m_2)} H_2$  with  $NAC_{r_2}$  are *sequentially independent* if

$$\exists h_{12} : R_1 \rightarrow D_2 \text{ s.t. } (d_2 \circ h_{12} = m'_1 \text{ and } e_2 \circ h_{12} \models NAC_{r_1^{-1}})$$

and

$$\exists h_{21} : L_2 \rightarrow D_1 \text{ s.t. } (e_1 \circ h_{21} = m_2 \text{ and } d_1 \circ h_{21} \models NAC_{r_2})$$

as in the following diagram:



Note that for  $n_1 : L_1 \rightarrow N_1$  in  $NAC_{r_1}$  we have the corresponding  $n'_1 : R_1 \rightarrow N'_1$  from  $NAC_{r_1^{-1}}$  according to Def. 2.4.

**Remark A.2** Note that it follows directly from this definition that we have the following relationship between parallel and sequential independency:  $G \xrightarrow{r_1} H_1 \xrightarrow{r_2} H_2$  are sequentially independent iff  $G \xleftarrow{r_1^{-1}} H_1 \xrightarrow{r_2} H_2$  are parallel independent.

The following fact is proven in [6] and describes that independent transformations can be performed in any order with the same result.

**Fact A.3 (Local Church-Rosser Theorem with NACs)** *Given a graph transformation system with NACs and two parallel independent direct transformations with NACs  $H_1 \xleftarrow{r_1, m_1} G \xrightarrow{r_2, m_2} H_2$ , there are an object  $G'$  and direct transformations  $H_1 \xrightarrow{r_2, m'_2} G'$  and  $H_2 \xrightarrow{r_1, m'_1} G'$  such that  $G \xrightarrow{r_1, m_1} H_1 \xrightarrow{r_2, m'_2} G'$  and  $G \xrightarrow{r_2, m_2} H_2 \xrightarrow{r_1, m'_1} G'$  are sequentially independent. Vice versa, given two sequentially independent direct transformations with NACs  $G \xrightarrow{r_1, m_1} H_1 \xrightarrow{r_2, m'_2} G'$  there are an object  $H_2$  and sequentially independent direct transformations  $G \xrightarrow{r_2, m_2} H_2 \xrightarrow{r_1, m'_1} G'$  such that  $H_1 \xleftarrow{r_1, m_1} G \xrightarrow{r_2, m_2} H_2$  are parallel independent:*

$$\begin{array}{ccc}
 & H_1 & \\
 r_{1,m_1} \nearrow & & \searrow r_{2,m'_2} \\
 G & & G' \\
 r_{2,m_2} \searrow & & \nearrow r_{1,m'_1} \\
 & H_2 &
 \end{array}$$

## B Correctness of the Criteria

### Theorem B.1 (no impeding predecessors and pure or direct enabling predecessor)

*Given  $s : r_1 r_2 \dots r_n$  a sequence of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 3.1 are satisfied for rule sequence  $s$  and graph  $G_0$ , then this rule sequence is applicable on  $G_0$  with injective matching i.e. there exists a graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$  with injective matching.*

**Proof** Note that because of criterion 2 and the fact that only injective matching is allowed it is always possible to construct the pushout complement of  $G \xleftarrow{m} L \xleftarrow{l} K$  with  $m$  an injective match and  $l$  the left-hand side rule morphism. The gluing condition is thus fulfilled. This means in particular that as soon as a match  $m$  is found for a rule  $r$  into a graph  $G$ , this rule is applicable on  $G$  if the NAC is satisfied. Now we prove this theorem by induction over the number of rules in the rule sequence  $s$ .

- **(Basis.  $n = 1$ )** Criterion 1 ensures that a match  $m_1 : L_1 \rightarrow G_0$  exists. The fact that the gluing condition is always satisfied and  $m_1$  satisfies  $NAC_{r_1}$  allows us to construct therefore the direct transformation  $G_0 \xrightarrow{m_1} G_1$ .
- **(Induction Step.)** Consider the transformation  $G_0 \xrightarrow{r_1} G_1 \dots \xrightarrow{r_{n-1}} G_{n-1}$ . We are now looking for a match  $m_n : L_n \rightarrow G_{n-1}$  such that also the direct transformation  $G_{n-1} \xrightarrow{r_n} G_n$  exists. We have the following two cases.
  - (i) Rule  $r_n$  is applicable on  $G_0$  via an injective match  $m_n$ . We know because of criterion 3 in Def. 3.1 that  $r_n$  is asymmetrically parallel independent on each rule  $r_j$  in the sequence  $s$  with  $j < n$ . According to criterion 4b we know also that  $r_n$  is asymmetrically parallel independent on  $r_j = r_c$ . Therefore we have the following diagram where  $j = 1$  and there exists a morphism  $h_{n,1} : L_n \rightarrow D_1$

with  $d_1 \circ h_{n,1} = m_n$  (which implies that  $h_{n,1}$  is injective) and  $e_1 \circ h_{n,1} \models NAC_{r_n}$ :

$$\begin{array}{ccccccc}
 & & N_1 & & N_n & & \\
 & & \uparrow n_1 & & \uparrow n_n & & \\
 R_1 & \leftarrow & K_1 & \rightarrow & L_1 & & L_n \leftarrow K_n \rightarrow R_n \\
 \downarrow & & \downarrow & & \swarrow h_{n,1} & & \swarrow m_n \\
 G_1 & \xleftarrow{e_1} & D_1 & \xrightarrow{d_1} & G_0 & \xleftarrow{m_n} & D'_n \rightarrow H'_n \\
 & & & & & & \downarrow \\
 & & & & & & H'_n
 \end{array}$$

Considering  $e_1 \circ h_{n,1}$  as an injective match of  $r_n$  into  $G_1$  satisfying  $NAC_{r_n}$  we can construct the following diagram where  $j = 2$  and there exists a morphism  $h_{n,2} : L_n \rightarrow D_2$  with  $d_2 \circ h_{n,2} = e_1 \circ h_{n,1}$  and  $e_2 \circ h_{n,2} \models NAC_{r_n}$ .

$$\begin{array}{ccccccc}
 & & N_2 & & N_n & & \\
 & & \uparrow n_2 & & \uparrow n_n & & \\
 R_2 & \leftarrow & K_2 & \rightarrow & L_2 & & L_n \leftarrow K_n \rightarrow R_n \\
 \downarrow & & \downarrow & & \swarrow h_{n,2} & & \swarrow e_1 \circ h_{n,1} \\
 G_2 & \xleftarrow{e_2} & D_2 & \xrightarrow{d_2} & G_1 & \xleftarrow{m_n} & D'_n \rightarrow H'_n \\
 & & & & & & \downarrow \\
 & & & & & & H'_n
 \end{array}$$

Iteratively we can construct analog diagrams such that in the end the following diagram arises where  $j = n-1$  and there exists a morphism  $h_{n,n-1} : L_n \rightarrow D_{n-1}$  with  $d_{n-1} \circ h_{n,n-1} = e_{n-2} \circ h_{n,n-2}$  and  $e_{n-1} \circ h_{n,n-1} \models NAC_{r_n}$ .

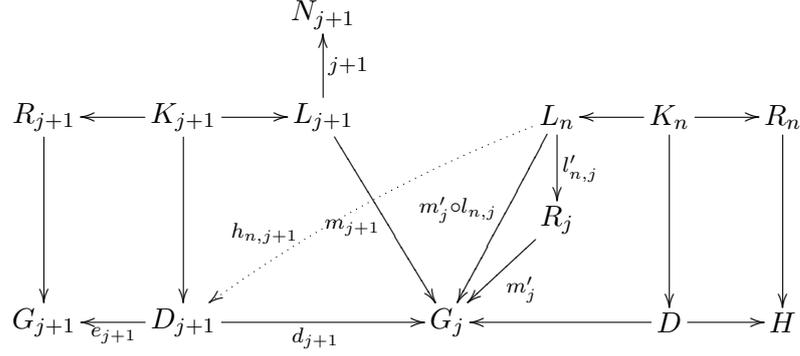
$$\begin{array}{ccccccc}
 & & N_{n-1} & & N_n & & \\
 & & \uparrow n_{n-1} & & \uparrow n_n & & \\
 R_{n-1} & \leftarrow & K_{n-1} & \rightarrow & L_{n-1} & & L_n \leftarrow K_n \rightarrow R_n \\
 \downarrow & & \downarrow & & \swarrow h_{n,n-1} & & \swarrow e_{n-2} \circ h_{n,n-2} \\
 G_{n-1} & \xleftarrow{e_{n-1}} & D_{n-1} & \xrightarrow{d_{n-1}} & G_{n-2} & \xleftarrow{m_n} & D \rightarrow H \\
 & & & & & & \downarrow \\
 & & & & & & H
 \end{array}$$

Considering  $e_{n-1} \circ h_{n,n-1}$  as an injective match of  $r_n$  into  $G_{n-1}$  we can apply now rule  $r_n$  to  $G_n$  and have the direct transformation  $G_n \xrightarrow{r_n} G_{n+1}$ .

(ii) There does not exist an injective match of rule  $r_n$  into  $G_0$ . Now we have two cases.

- Criterion 4a in Def. 3.1 holds. Then there exists a rule  $r_j$  in  $s$  such that  $1 \leq j < n$  and  $r_n$  is purely sequential dependent on  $r_j$ . This means that  $r_n$  does not have any NACs and we have an injective morphism  $l_{n,j} : L_n \rightarrow R_j$  according to Def. 2.12. Consider now the comatch  $m'_j : R_j \rightarrow G_j$  of the direct transformation  $G_{j-1} \xrightarrow{r_j} G_j$ . We then have an injective match  $m'_j \circ l_{n,j} : L_n \rightarrow G_j$  for the rule  $r_n$  into  $G_j$ . Now analogously to the previous case we can use

criterion 3 to construct the following diagram



and in the end the requested injective match  $e_{n-1} \circ h_{n,n-1} : L_n \rightarrow G_{n-1}$  and thus the direct transformation  $G_n \xrightarrow{r_n} G_{n+1}$  exists.

- Criterion 4b in Def. 3.1 holds. Then there exists a concurrent rule  $r_c$  of  $r_{n-1}$  and  $r_n$  with an injective match of  $r_c$  into  $G_0$  satisfying  $NAC_{r_c}$  and  $r_c$  is asymmetrically parallel independent on  $r_j$  for all  $j < (n-1)$ . (*direct enabling predecessor*) We now have a rule sequence  $s' : r_1 r_2 \dots r_{n-2} r_c$  for which the criteria in Def. 3.1 are fulfilled. This means that there exists a graph transformation sequence along the sequence  $s' : r_1 r_2 \dots r_c$  and therefore according to the Concurrency Theorem with NACs in [6] also along the rule sequence  $s : r_1 r_2 \dots r_{n-1} r_n$ .

□

**Theorem B.2 (no impeding predecessors and pure or direct enabling predecessors)**

3.3 Given a sequence  $s : r_1 r_2 \dots r_n$  of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 3.1 are satisfied for a rule sequence  $s' : r'_1 r'_2 \dots r'_m$  with  $m < n$  in which neighbored rules in  $s$  can be summarized by a concurrent rule, then the original rule sequence  $s$  is applicable on  $G_0$  with injective matching i.e. there exists a graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$  with injective matching.

**Proof** This follows directly from the Concurrency Theorem with NACs [6] and Theorem 3.2. □

**Theorem B.3 (no enabling predecessor)** Given a sequence  $s : r_1 r_2 \dots r_n$  of  $n$  rules and a graph  $G_0$ . If the criteria in Def. 4.1 are satisfied for rule sequence  $s$  and graph  $G_0$ , then this rule sequence is not applicable on  $G_0$  i.e. there exists no graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$ .

**Proof**

- If criterion 1 in Def. 4.1 is satisfied then it is obvious that a graph transformation sequence can not exist.
- Suppose that criterion 2 in Def. 4.1 is satisfied. Then  $\exists r_i$  in  $s$  with  $1 < i \leq n$  such that there does not exist a match into  $G_0$  satisfying  $NAC_{r_i}$  but for all rules  $r_j$  in  $s$  with  $1 \leq j < i \leq n$ ,  $r_i$  is asymmetrically sequential independent on  $r_j$ . Suppose that a graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$  exists. Then we

have the following diagram:

$$\begin{array}{ccccccc}
 & & N_{i-1} & & & & N_i \\
 & & \uparrow n_{i-1} & & & & \uparrow n_i \\
 L_{i-1} & \longleftarrow & K_{i-1} & \longrightarrow & R_{i-1} & & L_i \longleftarrow K_i \longrightarrow R_i \\
 & & \downarrow & & \downarrow & & \downarrow \\
 G_{i-2} & \xleftarrow{d_{i-1}} & D_{i-1} & \xrightarrow{e_{i-1}} & G_{i-1} & \xleftarrow{d_i} & D_i \xrightarrow{e_i} G_i \\
 & & \swarrow h_{i,i-1} & & \swarrow m_i & & \\
 & & & & & & 
 \end{array}$$

Because of the assumption and Def. 2.10 we know that  $h_{i,i-1}$  exists such that  $e_{i-1} \circ h_{i,i-1} = m_i$  and  $d_{i-1} \circ h_{i,i-1} \models NAC_{r_i}$ . Therefore we can now construct the following diagram:

$$\begin{array}{ccccccc}
 & & N_{i-2} & & & & N_i \\
 & & \uparrow n_{i-2} & & & & \uparrow n_i \\
 L_{i-2} & \longleftarrow & K_{i-2} & \longrightarrow & R_{i-2} & & L_i \longleftarrow K_i \longrightarrow R_i \\
 & & \downarrow & & \downarrow & & \downarrow \\
 G'_{i-3} & \xleftarrow{d'_{i-2}} & D'_{i-2} & \xrightarrow{e'_{i-2}} & G_{i-2} & \xleftarrow{d_i} & D_i \xrightarrow{e_i} G_i \\
 & & \swarrow h_{i,i-2} & & \swarrow d_{i-1} \circ h_{i,i-1} & & \\
 & & & & & & 
 \end{array}$$

Again because of the assumption and Def. 2.10 we know that  $h_{i,i-2}$  exists such that  $e'_{i-2} \circ h_{i,i-2} = d_{i-1} \circ h_{i,i-1}$  and  $d'_{i-2} \circ h_{i,i-2} \models NAC_{r_i}$ . Iteratively in the end we will find a match  $m \models NAC_{r_i} : L_i \rightarrow G_0$  and this is a contradiction. Therefore a graph transformation  $G_0 \xrightarrow{r_1} G_1 \dots G_{n-1} \xrightarrow{r_n} G_n$  can not exist.  $\square$