Conflict Detection for Graph Transformation with Negative Application Conditions

Leen Lambers[1], Hartmut Ehrig[2], and Fernando Orejas[3]

[1] Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin
Germany
leen@cs.tu-berlin.de,
[2] ehrig@cs.tu-berlin.de
[3] Dept. L.S.I. - Technical University Catalonia
Barcelona, Spain
orejas@lsi.upc.edu

Abstract. This paper introduces a new theory needed for the purpose of conflict detection for graph transformation with negative application conditions (NACs). Main results are the formulation of a conflict notion for graph transformation with NACs and a conflict characterization derived from it. A critical pair definition is introduced and completeness of the set of all critical pairs is shown. This means that for each conflict, occuring in a graph transformation system with NACs, there exists a critical pair expressing the same conflict in a minimal context. Moreover a necessary and sufficient condition is presented for parallel independence of graph transformation systems with NACs. In order to facilitate the implementation of the critical pair construction for a graph transformation system with NACs a correct construction is formulated. Finally, it is discussed how to continue with the development of conflict detection and analysis techniques in the near future.

1 Introduction

Several applications using graph transformation need or already use negative application conditions (NACs) to express that certain structures at a given time are forbidden, e.g., [1,2,3,4,5]. In order to allow conflict detection and analysis for these applications, the theory already worked out for graph transformation systems (gts) without NACs should be generalized to gts with NACs. The notion of critical pairs is central in this theory, allowing for conflict detection and analysis. It was developed at first in the area of term rewriting systems (e.g., [6]) and, later, introduced in the area of graph transformation for hypergraph rewriting [7,8] and then for all kinds of transformation systems fitting into the framework of adhesive high-level replacement categories [9].

This paper now generalizes the critical pair notion and some first important related results to gts with NACs. We tailored the theory presented in this paper for gts with NACs and not on other kind of constraints or application conditions, since NACs are already widely used in practice. It would be subject of future

A. Corradini et al. (Eds.): ICGT 2006, LNCS 4178, pp. 61–76, 2006.

work to develop also a critical pair theory for graph transformation with other kind of constraints as presented in [9]. Subject of future work as well and more directly related to the subject of this paper is the formulation of a critical pair lemma which gives a sufficient condition for local confluence of a gts with NACs.

The structure of this paper is as follows. In the first paragraph we repeat the necessary definitions for graph transformation in the double pushout approach [10] with NACs. Then we explain carefully what new types of conflicts can occur because of the NACs by means of a new conflict characterization. This conflict characterization leads in the next paragraph to a critical pair definition for gts with NACs. A critical pair describes a conflict in a minimal context. Since now there occur new types of conflicts we also distinguish other types of critical pairs. Afterwards we show completeness for this critical pair definition i.e. each conflict is expressed at least by one critical pair. Moreover we demonstrate, that if there are no critical pairs at all in the graph transformation system with NACs then this system is locally confluent or, more exactly, each pair of direct transformations is parallel independent. In the conclusion and outlook we explain how to continue with the development of critical pair theory to enable manageable conflict detection and analysis techniques for gts with NACs.

2 Graph Transformation with NACs

Definition 1 (graph and graph morphism). *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* $s, t : G_E \to G_V$*, 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 $f : G_1 \to G_2$ *between two graphs* $G_i = (G_{i,E}, G_{i,V}, s_i, t_i)$*,* $(i = 1, 2)$ *is a pair* $f = (f_E : G_{E,1} \to G_{E,2}, f_V : G_{V,1} \to G_{V,2})$ *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 graph morphism* $f : G_1 \to G_2$ *is injective (resp.surjective) if* f_V *and* f_E *are injective (resp. surjective) mappings. Two graph morphisms* $m_1 : L_1 \to G$ *and* $m_2 : L_2 \to G$ *are* jointly surjective *if* $m_{1,V}(L_{1,V}) \cup m_{2,V}(L_{2,V}) = G_V$ *and* $m_{1,E}(L_{1,E}) \cup m_{2,E}(L_{2,E}) = G_E$*. A pair of jointly surjective morphisms* (m_1, m_2) *is also called an* overlapping *of* L_1 *and* L_2*. The category having graphs as objects and graph morphisms as arrows is called* **Graph**.

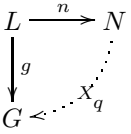
Definition 2 (rule). *A graph transformation* rule $p : L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ *consists of a rule name* p *and a pair of injective graph morphisms* $l : K \to L$ *and* $r : K \to R$*. The graphs* L, K *and* R *are called the left-hand side (lhs), the interface, and the right-hand side (rhs) of* p*, respectively.*

Definition 3 (match). *Given a rule* $p : L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ *and a graph* G*, one can try to apply* p *to* G *if there is an occurence of* L *in* G *i.e. a graph morphism, called* match $m : L \to G$*.*

A negative application condition or NAC as introduced in [11] forbids a certain graph structure to be present before or after applying the rule.

Definition 4 (negative application condition)
Let M be the set of all injective graph morphisms.

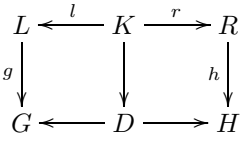
- *A* negative application condition *or $NAC(n)$ on L is a graph morphism $n : L \to N$. A graph morphism $g : L \to G$* satisfies *$NAC(n)$ on L i.e. $g \models NAC(n)$ if and only if $\nexists\ q : N \to G \in M$ such that $q \circ n = g$.*



- *A $NAC(n)$ on L (resp. R) for a rule $p : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ is called* left *(resp.* right*) NAC on p. $NAC_{p,L}$ (resp. $NAC_{p,R}$) is a set of left (resp. right) NACs on p. $NAC_p = (NAC_{p,L}, NAC_{p,R})$, consisting of a set of left and a set of right NACs on p is called a set of NACs on p.*

Definition 5 (graph transformation with NACs)

- *A* graph transformation system with NACs *is a set of rules where each rule $p : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ has a set $NAC_p = (NAC_{p,L}, NAC_{p,R})$ of NACs on p.*
- *A* direct graph transformation *$G \overset{p,g}{\Rightarrow} H$ via a rule $p : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ with $NAC_p = (NAC_{p,L}, NAC_{p,R})$ and a match $g : L \to G$ consists of the double pushout [10] (DPO)*



where g satisfies each NAC in $NAC_{p,L}$, written $g \models NAC_{p,L}$, and $h : R \to H$ satisfies each NAC in $NAC_{p,R}$, written $h \models NAC_{p,R}$. Since pushouts in **Graph** *always exist, the DPO can be constructed if the pushout complement of $K \to L \to G$ exists. If so, we say that, the match m satisfies the gluing condition of rule p. A* graph transformation*, denoted as $G_0 \overset{*}{\Rightarrow} G_n$ is a sequence $G_0 \Rightarrow G_1 \Rightarrow \cdots \Rightarrow G_n$ of direct graph transformations.*

In the example in Fig. 1 a pair of direct transformations via the rules $p_1 : L_1 \leftarrow K_1 \rightarrow R_1$, $p_2 : L_2 \leftarrow K_2 \rightarrow R_2$ and matches m_1 resp. m_2 is depicted. The match m_1 fullfills the negative application condition $NAC(n_1)$ since there is no ingoing edge into node 1 in graph G. The morphism $e_2 \circ m_2$ though doesn't fullfill the negative application condition $NAC(n_1)$ since now there is an edge from node 7 to node 1 in graph H_2.
Remark: From now on we consider only gts with rules having an empty set of right NACs. This is without loss of generality, because each right NAC can be translated into an equivalent left NAC as explained in [9], where Theorem 7.17 can be specialized to NACs.

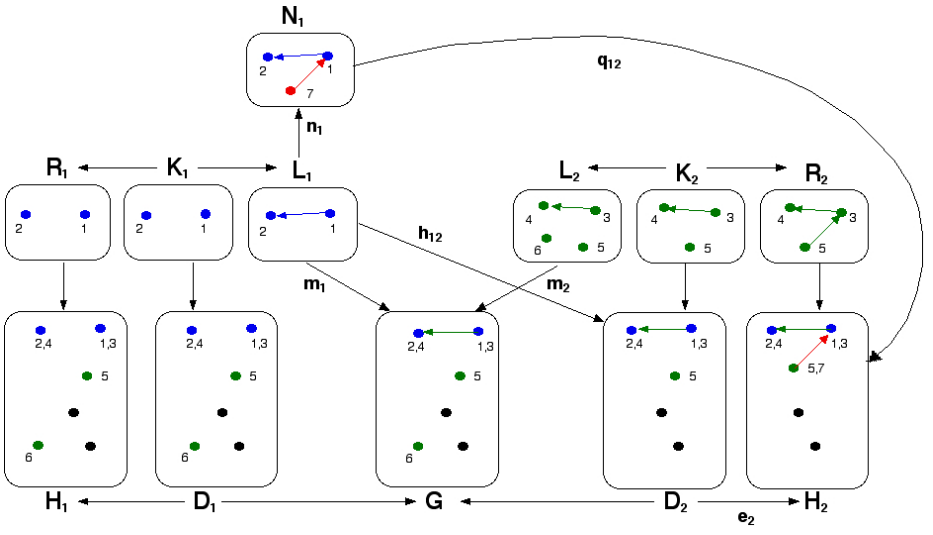
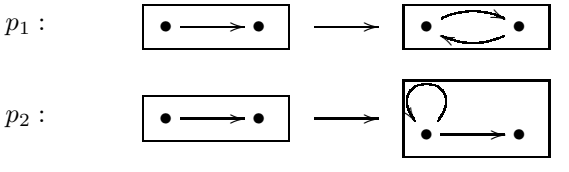


Fig. 1. forbid-produce/delete-use-conflict

3 Conflicts for Graph Transformation with NACs

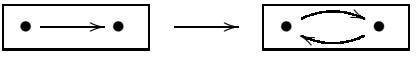
Confluence conflicts in term rewriting or graph transformation can typically occur when two rules are applied to the same term or graph in such a way that the corresponding redexes (i.e. for graph transformation the images of the corresponding matches) overlap. In particular, the conflict appears when one of the rules can *delete* part of the redex of the other rule. We call these conflicts *delete-use* (or *use-delete*) conflicts. As a consequence, this kind of conflicts are detected by computing the critical pairs of the given system, i.e., such delete-use or use-delete conflicts induced by the overlappings between any two rules. However, when dealing with graph transformation with NACs some new forms of conflict may be present. For instance, an otherwise harmless overlapping (e.g., if no deletion happens) may cause a conflict as the following example shows (for simplicity, in the examples below we will display the rules just in terms of their left and right-hand sides, leaving the context implicit). Suppose that we have two rules p_1 and p_2 with exactly the same left-hand side:



It should be clear that if we apply the rule p_1 to a given graph, then we can apply afterwards the rule p_2 at the same redex. And, conversely, if we apply p_1

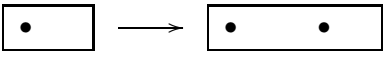
we can apply afterwards p_2 at the same redex. However, suppose that the rule p_2 has a left NAC which coincides with the right-hand side of the rule p_1. That is, suppose that the NAC of rule p_2 is defined by the inclusion:

$NAC(p_2)$:

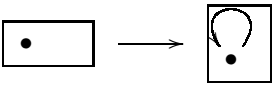


then, obviously, after applying the rule p_1 we would be unable to apply the rule p_2 at the same location because the associated NAC would forbid it. The problem here is that the application of the first rule produces some additional structure that is forbidden by the NAC of the second rule. For this reason we call these new kind of conflicts *produce-forbid* (or *forbid-produce*) conflicts. Actually, these new conflicts may arise even when the possible application of two rules do not overlap. For instance suppose that p_1 and p_2 are the rules below:

p_1 :

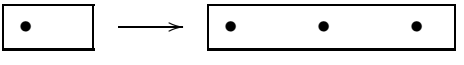


p_2 :



and suppose that the rule p_2 includes the left NAC:

$NAC(p_2)$:



meaning that the rule p_2 cannot be applied to a graph including at least three nodes. Now, suppose that we have a graph G including just two nodes a and b. Obviously, we can apply rule p_1 to G at node a and rule p_2 at node b without any overlapping. However, if we first apply rule p_1 this causes the creation of a new node that would now forbid the application of rule p_2 at node b.

In what follows, we will first look at the concept of parallel independence of two direct transformations with NACs, which expresses the condition to be fulfilled in order to apply two different rules to the same graph in any order with the same result. This is proven in Theorem 1, the Local Church-Rosser Theorem with NACs. Afterwards we will provide the conflict notion for gts with NACs and a characterization of the conflicts as described above.

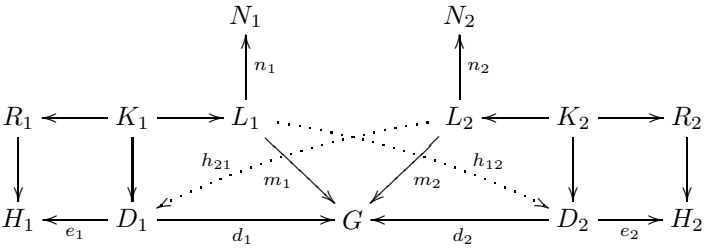
Definition 6 (parallel independence). *Two direct transformations* $G \overset{(p_1,m_1)}{\Longrightarrow} H_1$ *with* NAC_{p_1} *and* $G \overset{(p_2,m_2)}{\Longrightarrow} H_2$ *with* NAC_{p_2} *are parallel independent if*

$$\exists h_{12} : L_1 \to D_2 \text{ s.t. } (d_2 \circ h_{12} = m_1 \text{ and } e_2 \circ h_{12} \models NAC_{p_1})$$

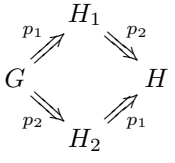
and

$$\exists h_{21} : L_2 \to D_1 \text{ s.t. } (d_1 \circ h_{21} = m_2 \text{ and } e_1 \circ h_{21} \models NAC_{p_2})$$

as in the following diagram:



Theorem 1 (Local Church-Rosser Theorem with NACs). *If a pair of direct transformations $H_1 \overset{p_1}{\Leftarrow} G \overset{p_2}{\Rightarrow} H_2$ with NACs is parallel independent, then there are two direct transformations $H_1 \overset{p_2}{\Rightarrow} H$ and $H_2 \overset{p_1}{\Rightarrow} H$ with NACs s.t.*



Proof. Because of the Local Church-Rosser Theorem for rules without NACs all necessary pushouts in $H_1 \overset{p_2}{\Rightarrow} H$ and $H_2 \overset{p_1}{\Rightarrow} H$ can be constructed and moreover the matches $e_2 \circ h_1$ and $e_1 \circ h_2$ satisfy the NACs of rule p_1 resp. p_2 by the definition of parallel independence for graph transformation with NACs.

The following lemma describes that, if a match for the potential second transformation exists, it is unique. Moreover this lemma will allow an elegant conflict characterization in Lemma 2.

Lemma 1 (unique match). *Given two direct transformations $G \overset{(p_1,m_1)}{\Longrightarrow} H_1$ with NAC_{p_1} and $G \overset{(p_2,m_2)}{\Longrightarrow} H_2$ with NAC_{p_2}, then the following holds:*

- *if $\exists h_{12} : L_1 \to D_2$ s.t. $d_2 \circ h_{12} = m_1$ then h_{12} is unique*
- *if $\exists h_{21} : L_2 \to D_1$ s.t. $d_1 \circ h_{21} = m_2$ then h_{21} is unique.*

Proof. Since each rule consists of two injective morphisms and pushouts are closed under injective morphisms, d_1 and d_2 are injective morphisms as well. If there would exist $h'_{12} : L_1 \to D_2 : d_2 \circ h'_{12} = m_1$ then because of d_2 injective and $d_2 \circ h'_{12} = d_2 \circ h_{12} = m_1$ it follows that $h'_{12} = h_{12}$. Analogously one can prove that h_{21} is unique.

Definition 7 (conflict). *Two direct transformations $G \overset{(p_1,m_1)}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{(p_2,m_2)}{\Rightarrow} H_2$ with NAC_{p_2} are* in conflict *if they are not parallel independent i.e. if*

$$\nexists h_{12} : L_1 \to D_2 \text{ s.t. } (d_2 \circ h_{12} = m_1 \text{ and } e_2 \circ h_{12} \models NAC_{p_1})$$

or

$$\nexists h_{21} : L_2 \to D_1 \text{ s.t. } (d_1 \circ h_{21} = m_2 \text{ and } e_1 \circ h_{21} \models NAC_{p_2}).$$

The following lemma characterizes this conflict notion for graph transformation with NACs s.t. the difference with the conflict notion for graph transformation without NACs becomes more clear. As described in the introduction of this section new types of conflicts can occur and the lemma in fact characterizes four different types of conflicts that can occur partly simultaneously.

Two direct transformations $G \overset{(p_1,m_1)}{\Rightarrow} H_1$ and $G \overset{(p_2,m_2)}{\Rightarrow} H_2$ are in delete-use-conflict (resp.use-delete-conflict) if rule p_1 (resp. p_2) deletes part of the graph G, which is used by rule p_2 (resp.p_1) in the second (resp. first) direct transformation. This kind of conflict occurs also in gts without NACs [12]. In contrast a produce-forbid-conflict (resp. forbid-produce-conflict) occurs only in gts with NACs. Namely, if rule p_1 (resp.p_2) produces a graph structure which is forbidden by the NAC of rule p_2 (resp. p_1).

Lemma 2 (conflict characterizaton). *Two direct transformations* $G \overset{(p_1,m_1)}{\Rightarrow} H_1$ *with* NAC_{p_1} *and* $G \overset{(p_2,m_2)}{\Rightarrow} H_2$ *with* NAC_{p_2} *are* in conflict *if and only if:*

1. (a) $\nexists h_{12} : L_1 \to D_2 : d_2 \circ h_{12} = m_1$ *(use-delete-conflict)*
   or
   (b) *there exists a unique* $h_{12} : L_1 \to D_2 : d_2 \circ h_{12} = m_1$, *but* $e_2 \circ h_{12} \not\models NAC_{p_1}$ *(forbid-produce-conflict)*

   or
2. (a) $\nexists h_{21} : L_2 \to D_1 : d_1 \circ h_{21} = m_2$ *(delete-use-conflict)*
   or
   (b) *there exists a unique* $h_{21} : L_2 \to D_1 : d_1 \circ h_{21} = m_2$, *but* $e_1 \circ h_{21} \not\models NAC_{p_2}$ *(produce-forbid-conflict).*

Proof. $G \overset{(p_1,m_1)}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{(p_2,m_2)}{\Rightarrow} H_2$ with NAC_{p_2} are *in conflict* if

$$\nexists h_{12} : L_1 \to D_2 \; s.t. \; (d_2 \circ h_{12} = m_1 \; and \; e_2 \circ h_{12} \models NAC_{p_1})$$

or

$$\nexists h_{21} : L_2 \to D_1 \; s.t. \; (d_1 \circ h_{21} = m_2 \; and \; e_1 \circ h_{21} \models NAC_{p_2})$$

We consider at first the first line of this disjunction. Let $A(h_{12}) := d_2 \circ h_{12} = m_1$, $B(h_{12}) := e_2 \circ h_{12} \models NAC_{p_1}$, $P(h_{12}) := (A(h_{12}) \wedge B(h_{12}))$ and M_{12} be the set of all morphisms from L_1 to D_2. Then the first line is equivalent to

$$\nexists h_{12} \in M_{12} : (A(h_{12}) \wedge B(h_{12})) \equiv \nexists h_{12} \in M_{12} : P(h_{12})$$

This is equivalent to

$$\forall h_{12} \in M_{12} : \neg P(h_{12}) \equiv (M_{12} = \emptyset) \vee (M_{12} \neq \emptyset \wedge \forall h_{12} \in M_{12} : \neg P(h_{12}))$$

Moreover $P \equiv A \wedge B \equiv A \wedge (A \Rightarrow B)$ and thus $\neg P \equiv \neg(A \wedge B) \equiv \neg(A \wedge (A \Rightarrow B)) \equiv \neg A \vee \neg(A \Rightarrow B) \equiv \neg A \vee \neg(\neg A \vee B) \equiv \neg A \vee (A \wedge \neg B)$. This implies that $(M_{12} = \emptyset) \vee (M_{12} \neq \emptyset \wedge \forall h_{12} \in M_{12} : \neg P(h_{12})) \equiv$

$$(M_{12} = \emptyset) \vee (M_{12} \neq \emptyset \wedge \forall h_{12} \in M_{12} : \neg A(h_{12}) \vee (A(h_{12}) \wedge \neg B(h_{12})))$$

Because of Lemma 1 and because the disjunction holding for each morphism in M_{12} is an exclusive one this is equivalent to

$$(M_{12}=\emptyset)\vee(M_{12}\neq\emptyset\wedge\forall h_{12}\in M_{12}:\neg A(h_{12}))\vee(\exists!h_{12}\in M_{12}:(A(h_{12})\wedge\neg B(h_{12})))$$

Now $(M_{12}=\emptyset)\vee(M_{12}\neq\emptyset\wedge\forall h_{12}\in M_{12}:\neg A(h_{12}))\equiv\forall h_{12}\in M_{12}:\neg A(h_{12})\equiv\nexists h_{12}\in M_{12}:A(h_{12})$. This implies finally that $\nexists h_{12}:L_1\to D_2$ *s.t.* $(d_2\circ h_{12}=m_1$ *and* $e_2\circ h_{12}\models NAC_{p_1})$ is equivalent to

$$(\nexists h_{12}\in M_{12}:d_2\circ h_{12}=m_1)\vee(\exists!h_{12}\in M_{12}:(d_2\circ h_{12}=m_1\wedge e_2\circ h_{12}\not\models NAC_{p_1}))$$

is equivalent to

1. (a) $\nexists h_{12}:L_1\to D_2:d_2\circ h_{12}=m_1$ (use-delete-conflict)
   or
   (b) there exists a unique $h_{12}:L_1\to D_2:d_2\circ h_{12}=m_1$, but $e_2\circ h_{12}\not\models NAC_{p_1}$ (forbid-produce-conflict)

Analogously we can proceed for the second part of the disjunction.

Note that a use-delete-conflict (resp. delete-use-conflict) cannot occur simultaneously to a forbid-produce-conflict (resp. produce-forbid-conflict), since $(1.a)\Rightarrow\neg(1.b)$ (resp. $(2.a)\Rightarrow\neg(2.b)$). The following types of conflicts can occur simultaneously though: use-delete/delete-use-, use-delete/produce-forbid-, forbid-produce/delete-use-, forbid-produce/produce-forbid-conflict. In the example in Fig. 1 a pair of direct transformations in forbid-produce/delete-use-conflict is shown. In this case the first rule forbids an additional edge pointing to node (1,3) which is added by the second rule and the first rule deletes the edge (1,3)-(2,4) which is used by the second rule. Note that the labels express how nodes and edges are mapped to each other.

4 Critical Pairs for Graph Transformation with NACs

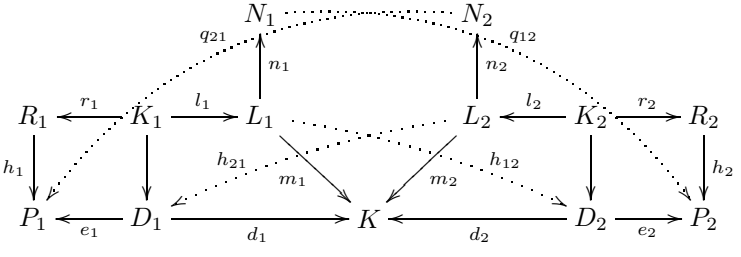
Now that we have a detailed conflict characterization we can look at a conflict in a minimal context i.e. a *critical pair*. Basically we exclude from the conflict all the graph parts that in no way can be responsible for the occurence of the conflict. In the case of a *delete-use-conflict* this would be the graph context which is not reached by any of the matches of the lhs's of the rules. This is because these graph parts can not be used nor deleted by any of the rules anyway. Therefore we consider only jointly surjective matches or overlappings of the lhs's of both rules in part $(1a)$ and $(2a)$ of the following critical pair definition. In the case of a *produce-forbid-conflict* we can leave out the graph parts which are not affected by any negative application condition of one rule and reached by a match of the rhs of the other rule. This is because these graph parts are not forbidden by a NAC of one rule and can not have been produced by the other rule anyway. Therefore we only consider overlappings or jointly surjective mappings of the NAC of one rule with the rhs of the other rule in part $(1b)$ and $(2b)$ of the

following critical pair definition. Thus in fact in the example in Fig. 1 we obtain the critical pair by ignoring all unlabelled graph nodes. Remember that in the following critical pair definition M is the set of all injective graph morphisms as defined in Def. 4.

Definition 8 (critical pair). *A critical pair is a pair of direct transformations* $K \overset{(p_1,m_1)}{\Rightarrow} P_1$ *with* NAC_{p_1} *and* $K \overset{(p_2,m_2)}{\Rightarrow} P_2$ *with* NAC_{p_2} *such that:*

1. (a) $\nexists h_{12} : L_1 \to D_2 : d_2 \circ h_{12} = m_1$ *and* (m_1, m_2) *jointly surjective (use-delete-conflict)*
   or
   (b) *there exists a* $h_{12} : L_1 \to D_2$ *s.t.* $d_2 \circ h_{12} = m_1$*, but for one of the NACs* $n_1 : L_1 \to N_1$ *of* p_1 *there exists a morphism* $q_{12} : N_1 \to P_2 \in M$ *s.t.* $q_{12} \circ n_1 = e_2 \circ h_{12}$ *and* (q_{12}, h_2) *jointly surjective (forbid-produce-conflict)*

   or
2. (a) $\nexists h_{21} : L_2 \to D_1 : d_1 \circ h_{21} = m_2$ *and* (m_1, m_2) *jointly surjective (delete-use-conflict)*
   or
   (b) *there exists a* $h_{21} : L_2 \to D_1$ *s.t.* $d_1 \circ h_{21} = m_2$*, but for one of the NACs* $n_2 : L_2 \to N_2$ *of* p_2 *there exists a morphism* $q_{21} : N_2 \to P_1 \in M$ *s.t.* $q_{21} \circ n_2 = e_1 \circ h_{21}$ *and* (q_{21}, h_1) *jointly surjective (produce-forbid-conflict)*



Remarks to related work: Note that the definition in this paper for parallel independence and conflict as well as the Local Church Rosser Theorem for graph transformation with NACs coincide with their equivalents as introduced in [11]. Moreover, if the gts doesn't hold any NAC, then the definition of parallel independence, conflict and critical pair as given in this paper correspond to the respective definition in the context of graph transformation without NACs [9]. Leadoff ideas to capture the critical pair notion for graph transformation with NACs were described in [13] and coincide with the formalization in this paper. Furthermore in [5] so-called critical conflict pairs for single pushout graph transformation with NACs are defined. The correspondence between this notion and the critical pair notion as introduced in this paper should be investigated in more detail.

Now we prove that Definition 8 of critical pairs leads to completeness. This means, that each occuring conflict in the graph transformation system with NACs can be expressed by a critical pair i.e. the same kind of conflict but in a minimal context. Therefore at first we need the following definition and lemma.

Definition 9 (extension diagram). *An* extension diagram *is a diagram (1),*

$$\begin{array}{ccc} G_0 & \overset{t}{\Longrightarrow}{}^{*} & G_n \\ {\scriptstyle k_0}\downarrow & (1) & \downarrow{\scriptstyle k_n} \\ G_0' & \overset{t'}{\Longrightarrow}{}^{*} & G_n' \end{array}$$

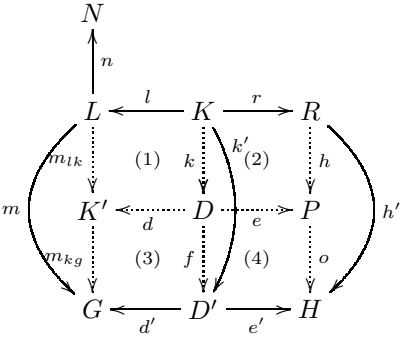
where, $k_0 : G_0 \to G_0'$ *is a morphism, called extension morphism, and* $t : G_0 \overset{*}{\Rightarrow} G_n$ *and* $t' : G_0' \overset{*}{\Rightarrow} G_n'$ *are transformations via the same productions* $(p_0, \cdots, p_{n-1})$ *and matches* $(m_0, \cdots, m_{n-1})$ *and* $(k_0 \circ m_0, \cdots, k_{n-1} \circ m_{n-1})$ *respectively, defined by the following DPO diagrams :*

p_i :

$$\begin{array}{ccccc} L_i & \xleftarrow[l_i]{} & K_i & \xrightarrow[r_i]{} & R_i \\ {\scriptstyle m_i}\downarrow & & {\scriptstyle j_i}\downarrow & & \downarrow{\scriptstyle n_i} \\ G_i & \xleftarrow[f_i]{} & D_i & \xrightarrow[g_i]{} & G_{i+1} \\ {\scriptstyle k_i}\downarrow & & {\scriptstyle d_i}\downarrow & & \downarrow{\scriptstyle k_{i+1}} \\ G_i' & \xleftarrow[f_i']{} & D_i' & \xrightarrow[g_i']{} & G_{i+1}' \end{array}$$

Remark: Since t and t' are transformations for a gts with NACs, the matches $(m_0, \cdots, m_{n-1})$ and $(k_0 \circ m_0, \cdots, k_{n-1} \circ m_{n-1})$ have to satisfy the NACs of the rules $(p_0, \cdots, p_{n-1})$.

Lemma 3 (induced direct transformation). *Given a direct transformation* $G \overset{p}{\Rightarrow} H$ *with NACs via the rule* $p : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ *and match* $m : L \to G$ *and given an object* K' *with two morphisms* $L \overset{m_{lk}}{\rightarrow} K' \overset{m_{kg}}{\rightarrow} G$ *s.t.* $m = m_{kg} \circ m_{lk}$*, with* $m_{kg} \in M$*, then there exists a direct transformation, the so called* induced direct transformation $K' \overset{p}{\Rightarrow} P$ *via the same rule* p *and the match* m_{lk}*, satisfying the NACs of* p *as in the following diagram :*



Proof. Given $G \overset{p}{\Rightarrow} H$ with NAC n as shown above. Since $d' \in M$ we can take pullback (3) of m_{kg} and d'. Since $m_{kg} \circ m_{lk} \circ l = m \circ l = d' \circ k'$ then there

exists a morphism $k : K \to D$ with $k' = f \circ k$ and $d \circ k = m_{lk} \circ l$ because of the pullback property of (3). Because of the pushout-pullback-decomposition lemma [9], $l \in M$ and $m_{kg} \in M$ diagrams (1) and (3) are both pushouts. Now we can construct pushout (2) of $D \leftarrow K \rightarrow R$ because of $r \in M$. Since $e' \circ f \circ k = e' \circ k' = h' \circ r$ there exists a morphism $o : P \to H$ with $o \circ h = h'$ and $o \circ e = e' \circ f$ because of the pushout-property of (2). Because of the pushout-decomposition property also diagram (4) is a pushout.

It remains to show that m_{lk} satisfies the NACs of p. Suppose that m_{lk} doesn't fullfill some $NAC(n)$ of p, then there exists a morphism $q : N \to K' \in M$ s.t. $q \circ n = m_{lk}$, but this implies $m_{kg} \circ q \circ n = m_{kg} \circ m_{lk} = m$ with $m_{kg} \circ q \in M$ and this is a contradiction.

Theorem 2 (completeness of critical pairs). *For each pair of direct transformations $H_1 \overset{(p_1,m_1')}{\Leftarrow} G \overset{(p_2,m_2')}{\Rightarrow} H_2$ in conflict there is a critical pair with extension diagrams (1) and (2) and $m \in M$.*

$$\begin{array}{ccccc} P_1 & \Longleftarrow & K & \Longrightarrow & P_2 \\ \downarrow & & \downarrow m & & \downarrow \\ H_1 & \Longleftarrow & G & \Longrightarrow & H_2 \end{array} \qquad (1) \qquad (2)$$

Proof. According to Lemma 2 the following reasons are responsible for a pair of direct transformations $G \overset{(p_1,m_1')}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{(p_2,m_2')}{\Rightarrow} H_2$ with NAC_{p_2} to be *in conflict*:

1. (a) $\nexists h_{12}' : L_1 \to D_2' : d_2' \circ h_{12}' = m_1'$ (use-delete-conflict)
   or
   (b) there exists a unique $h_{12}' : L_1 \to D_2' : d_2' \circ h_{12}' = m_1'$, but $e_2' \circ h_{12}' \not\models NAC_{p_1}$ (forbid-produce-conflict)

   or
2. (a) $\nexists h_{21}' : L_2 \to D_1 : d_1' \circ h_{21}' = m_2'$ (delete-use-conflict)
   or
   (b) there exists a unique $h_{21}' : L_2 \to D_1 : d_1' \circ h_{21}' = m_2'$, but $e_1' \circ h_{21}' \not\models NAC_{p_2}$ (produce-forbid-conflict)

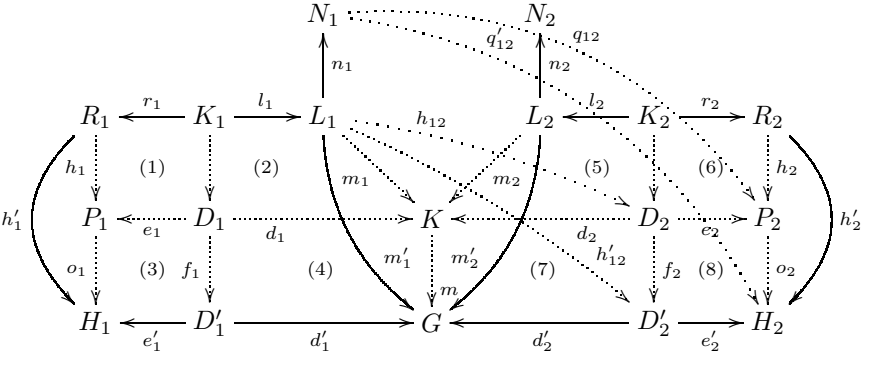
It is possible, that (1.b) and (2.b) are both false. In this case, (1.a) or (2.a) have to be true which corresponds to the usual use-delete-conflict (resp. delete-use-conflict) and in [9] it is described how to embed a critical pair into this pair of direct transformations. In the other case (1.b) or (2.b) are true. Let at first (1.b) be true. This means that there exists a unique $h_{12}' : L_1 \to D_2' : d_2' \circ h_{12}' = m_1'$, but $e_2' \circ h_{12}' \not\models NAC_{p_1}$. Thus for one of the NACs $n_1 : L_1 \to N_1$ of p_1 there exists a morphism $q_{12}' : N_1 \to H_2 \in M$ such that $q_{12}' \circ n_1 = e_2' \circ h_{12}'$. For each pair of graph morphisms with the same codomain, there exists an $E - M$ pair factorization [9] with E the set of all jointly surjective morphisms and M the set of injective graph morphisms as defined in Def. 4. Thus for $q_{12}' : N_1 \to H_2$ and $h_2' : R_2 \to H_2$ we obtain an object P_2 and morphisms $h_2 : R_2 \to P_2$,

$q_{12} : N_1 \to P_2$ and $o_2 : P_2 \to H_2$ with (h_2, q_{12}) jointly surjective and $o_2 \in M$ such that $o_2 \circ h_2 = h'_2$ and $o_2 \circ q_{12} = q'_{12}$. Because of Lemma 3 pushouts (5) - (8) can be constructed, if we consider the fact that also $H_2 \Rightarrow G$ is a direct transformation via the inverse rule of p_2. Since $o_2 \in M$ and (7) and (8) are pushouts also $f_2 \in M$ and $m \in M$. Because of the same argumentation as in Lemma 3, since m'_2 fullfills all the NACs of p_2 also m_2 fullfills them. Now we have the first half $K \Rightarrow P_2$ of the critical pair under construction.

We still have to check if this critical pair is in forbid-produce-conflict. Since (8) is a pullback and $o_2 \circ q_{12} \circ n_1 = q'_{12} \circ n_1 = e'_2 \circ h'_{12}$ there exists a morphism $h_{12} : L_1 \to D_2$, with $e_2 \circ h_{12} = q_{12} \circ n_1$ and $f_2 \circ h_{12} = h'_{12}$. Because $q'_{12} = o_2 \circ q_{12} \in M$ and $o_2 \in M$ we have $q_{12} \in M$. This means, that $e_2 \circ h_{12}$ doesn't fullfill the NAC $n_1 : L_1 \to N_1$.

Now we can start constructing the second half of the critical pair. Let m_1 be the morphism $d_2 \circ h_{12}$, then the following holds $m \circ m_1 = m \circ d_2 \circ h_{12} = d'_2 \circ f_2 \circ h_{12} = d'_2 \circ h'_{12} = m'_1$.

Because of Lemma 3 and $m \in M$ pushouts (1) - (4) can be constructed and m_1 satisfies the NACs of p_1. Thus finally we obtain a critical pair according to Def. 8 of type (1.b) because we have h_{12} with $d_2 \circ h_{12} = m_1$. Moreover there is $q_{12} \in M$ with (q_{12}, h_2) jointly surjective and $e_2 \circ h_{12} = q_{12} \circ n_1$.



We can proceed analogously for the case of (2.b) being true leading to a critical pair of type (2.b) according to Def. 8.

In the example in Fig. 1 the critical pair, obtained by ignoring all unlabelled nodes, can be embedded into the forbid-produce-delete-use-conflict depicted in this figure in a bigger context (i.e. two extra nodes).

Fact 3 (necessary and sufficient condition for parallel independence). *Each pair of direct transformations $H_1 \Leftarrow G \Rightarrow H_2$ in a gts with NACs is parallel independent if and only if there are no critical pairs for this gts with NACs. A gts with NACs is locally confluent if there are no critical pairs for this gts with NACs.*

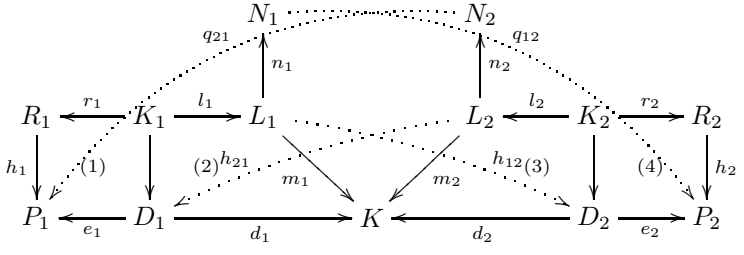
Proof. – Given a gts with NACs with an empty set of critical pairs and let $H_1 \Leftarrow G \Rightarrow H_2$ be a pair of non parallel independent direct graph transformations for this gts with NACs. This is a contradiction, since then there

would exist a critical pair which can be embedded into this pair of direct transformations as in Theorem 2.
- Given a gts with NACs with only parallel independent pairs of direct transformations $H_1 \Leftarrow G \Rightarrow H_2$. Then the set of critical pairs has to be empty, otherwise a critical pair would be a pair of non parallel independent direct transformations.
- If each pair of direct transformations $H_1 \Leftarrow G \Rightarrow H_2$ in a gts with NACs is parallel independent then each pair is also locally confluent and in consequence this gts with NACs is locally confluent.

5 Conflict Detection for Graph Transformation with NACs

5.1 Construction of Critical Pairs

Critical pairs allow for static conflict detection. Each conflict, occuring at some moment in the graph transformation, is represented by a critical pair. Thus it is possible to foresee each conflict by computing the set of all critical pairs before running the gts as implemented in the graph transformation tool AGG [14]. Each pair of rules of the gts induces a set of critical pairs. Computing this set for each pair of rules delivers us in the end the complete set of critical pairs for a gts. Here a straightforward construction is given to compute the set of critical pairs for a given pair of rules of the gts with NACs. Note that there exists already a more efficient construction for critical pairs in delete-use- or use-delete-conflict in step 1 described in [12]. For lack of space we only refer to it here and give instead the straightforward construction. Moreover we think that, using similar techniques, we could provide also a more efficient construction for the produce-forbid and forbid-produce critical pairs, but this is current work.



Given a pair of rules $(p_1 : L_1 \leftarrow K_1 \rightarrow R_1, p_2 : L_2 \leftarrow K_2 \rightarrow R_2)$ with NACs:

1. Consider any jointly surjective pair $(m_1 : L_1 \rightarrow K, m_2 : L_2 \rightarrow K)$.
   (a) Check gluing condition for (l_1, m_1) and (l_2, m_2). If it is satisfied then construct PO-complements D_1,D_2 in (2),(3) and PO's P_1,P_2 in (1) and (4).
   (b) Check if the pair of direct transformations $P_1 \Leftarrow K \Rightarrow P_2$ is in delete-use or use-delete-conflict, leading to critical pair $P_1 \Leftarrow K \Rightarrow P_2$.

2. Consider for each NAC $n_1 : L_1 \to N_1$ of p_1 any jointly surjective pair of morphisms $(h_2 : R_2 \to P_2, q_{12} : N_1 \to P_2)$ with q_{12} injective.
   (a) Check gluing condition for (h_2, r_2). If it is satisfied, then construct PO-complement D_2 in (4).
   (b) Construct PO K in (3) and abort, if $m_2 \not\models NAC_{p_2}$.
   (c) Check existence of $h_{12} : L_1 \to D_2$ s.t. $e_2 \circ h_{12} = q_{12} \circ n_1$ (e_2 injective implies uniqueness of h_{12}). If not existent, then abort.
   (d) Define $m_1 = d_2 \circ h_{12} : L_1 \to K$ and abort if $m_1 \not\models NAC_{p_1}$.
   (e) Check gluing condition for (m_1, l_1). If it is satisfied, then construct PO-complement D_1 in (2).
   (f) Construct P_1 as PO in (1) leading to critical pair $P_1 \Leftarrow K \Rightarrow P_2$.
3. Consider for each NAC $n_2 : L_2 \to N_2$ of p_2 any jointly surjective pair of morphisms $(h_1 : R_1 \to P_1, q_{21} : N_2 \to P_1)$ with q_{21} injective and continue analog to step 2.

5.2 Correctness of This Construction

The construction in the last paragraph is derived quite straightforwardly from Definition 8 and we are able to show that in fact it yields all critical pairs of a pair of rules of the gts with NACs.

Theorem 4. *The critical pair construction in paragraph 5.1 yields the set of all critical pairs for a pair of rules (p_1, p_2) of a gts with NACs.*

Proof.
- At first we prove that the pair of direct transformations constructed in steps 1,2 and 3 is really a critical pair. Step 1: Since the matches (m_1, m_2) of $P_1 \Leftarrow K \Rightarrow P_2$ are jointly surjective and this pair is in delete-use- or use-delete-conflict this is a critical pair. Step 2: Since there exists a morphism $h_{12} : L_1 \to D_2$ with $m_1 = d_2 \circ h_{12}$ and an injective morphism $q_{12} : N_1 \to P_2$ with $e_2 \circ h_{12} = q_{12} \circ n_1$ and (h_2, q_{12}) jointly surjective, this is a critical pair in forbid-produce-conflict. Step 3: Analog to Step 2.
- Secondly we prove that each critical pair is constructed by step 1, 2 or 3. Looking at Definition 8 there are three different types of critical pairs. Given a critical pair $P_1 \Leftarrow K \Rightarrow P_2$ of type 1a or 2a it is constructed by step 1. This is because the matches (m_1, m_2) are jointly surjective, (l_1, m_1) and (l_2, m_2) satisfy the gluing condition, because (2) and (3) are pushouts, (1) and (4) are also pushouts, pushouts are unique up to isomorphy and $P_1 \Leftarrow K \Rightarrow P_2$ are in delete-use- or use-delete-conflict. Given a critical pair $P_1 \Leftarrow K \Rightarrow P_2$ of type (1b) it is constructed by step 2. This is because (h_2, q_{12}) are jointly surjective, the gluing condition for (h_2, r_2) is satisfied because (4) is a pushout, (3) is a pushout, $m_2 \models NAC_{p_2}$, $h_{12} : L_1 \to D_2$ exists s.t. $e_2 \circ h_{12} = q_{12} \circ n_1$, $m_1 \models NAC_{p_1}$, the gluing condition for (m_1, l_1) holds since (2) is a pushout, (1) is a pushout and pushouts are unique up to ismorphy. Given a critical pair $P_1 \Leftarrow K \Rightarrow P_2$ of type (2b) it is constructed by step 3 analogously to a critical pair of type (1b).

6 Conclusion and Outlook

We presented the first foundations for a critical pair theory for gts with NACs which in the end should lead to good conflict detection and analysis algorithms for all kinds of systems described with means of gts with NACs. Main results in this paper are a conflict notion and conflict characterization for gts with NACs. The definition of a critical pair for gts with NACs for which we could prove completeness. We provided a straightforward and correct construction of the set of all critical pairs.

The theory presented in this paper can be generalized to adhesive HLR systems [9] with NACs. It is subject of future work to reformulate in detail all results and proofs mentioned in this paper on this more abstract level. Note that we tuned most reasonings in this paper already for this generalization such that it will be a relatively straightforward step. Once formulated the theory for adhesive HLR systems with NACs it is possible to instantiate it in particular for typed attributed graph transformation systems with NACs. This more general kind of graph transformation technique is most significant for modeling and metamodeling in software engineering and visual languages.

The theory of critical pairs consists of an other important part not mentioned yet in this paper. In gts without NACs the critical pair lemma holds. It gives a sufficient condition for the gts to be confluent. This is the case if all critical pairs are strictly confluent [9]. Thus, the critical pair lemma enables us to infer confluence behaviour of the whole graph transformation system by investigating the confluence behaviour of the set of all critical pairs. A similar result should be obtained for gts with NACs. This is work in progress and we are confident to be on the right path to complete it with the critical pair definition presented in this paper.

Moreover the results in this paper build a necessary theoretical foundation to continue with investigations on how to design conflict detection and analysis for typed, attributed gts with NACs as manageable as possible. For gts without NACs in [12] a rule analysis was proposed in order to obtain a more efficient conflict detection as the straightforward one. In [15] this efficiency investigation was continued by designing the so-called essential critical pairs. They build a subset of all critical pairs and represent each conflict not only in a minimal context, but also in a unique way. It should be possible to formulate also for critical pairs with NACs such a subset of essential critical pairs, by analyzing the rules and defining the exact conflict reason for each conflict. Finally future work is not only concerned with optimizations for conflict detection, but also for conflict analysis or finding a manageable way to investigate the resolvability of each conflict.

References

1. Hausmann, J., Heckel, R., Taentzer, G.: Detection of Conflicting Functional Requirements in a Use Case-Driven Approach. In: Proc. of Int. Conference on Software Engineering 2002, Orlando, USA (2002)

2. Mens, T., Taentzer, G., Runge, O.: Detecting Structural Refactoring Conflicts using Critical Pair Analysis. In Heckel, R., Mens, T., eds.: Proc. Workshop on Software Evolution through Transformations: Model-based vs. Implementation-level Solutions (SETra'04), Satellite Event of ICGT'04), Rome, Italy, ENTCS (2004)
3. Taentzer, G., Ehrig, K., Guerra, E., de Lara, J., Lengyel, L., Levendovsky, T., Prange, U., Varro, D., Varro-Gyapay, S.: Model Transformation by Graph Transformation: A Comparative Study. In: Proc. Workshop Model Transformation in Practice, Montego Bay, Jamaica (2005)
4. Bottoni, P., Schürr, A., Taentzer, G.: Efficient Parsing of Visual Languages based on Critical Pair Analysis and Contextual Layered Graph Transformation. In: Proc. IEEE Symposium on Visual Languages. (2000) Long version available as technical report SI-2000-06, University of Rom.
5. Koch, M., Mancini, L., Parisi-Presicce, F.: Graph-based Specification of Acces Control Policies. In: JCSS 71. (2005) 1–33
6. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems. JACM **27,4** (1980) 797–821
7. Plump, D.: Hypergraph Rewriting: Critical Pairs and Undecidability of Confluence. In Sleep, M., Plasmeijer, M., van Eekelen, M.C., eds.: Term Graph Rewriting. Wiley (1993) 201–214
8. Plump, D.: Confluence of graph transformation revisited. In Middeldorp, A., van Oostrom, V., van Raamsdonk, F., de Vrijer, R., eds.: Processes, Terms and Cycles: Steps on the Road to Infinity:Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday. Volume 3838 of Lecture Notes in Computer Science. Springer (2005) 280,308
9. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theoretical Computer Science. Springer (2006)
10. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Algebraic approaches to graph transformation I : Basic Concepts and Double Pushout Approach. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations. World Scientific (1997) 163–245
11. Annegret Habel, R.H., Taentzer, G.: Graph grammars with negative application conditions. Fundamenta Informaticae **26** (1996) 287–313
12. Lambers, L., Ehrig, H., Orejas, F.: Efficient detection of conflicts in graph-based model transformation. In: Proc. International Workshop on Graph and Model Transformation (GraMoT'05). Electronic Notes in Theoretical Computer Science, Tallinn, Estonia, Elsevier Science (2005)
13. Schultzke, T.: Entwicklung und implementierung eines parsers für visuelle sprachen basierend auf kritischer paaranalyse. Master's thesis, Technische Universität Berlin (2001)
14. Taentzer, G.: AGG: A Graph Transformation Environment for Modeling and Validation of Software. In Pfaltz, J., Nagl, M., Boehlen, B., eds.: Application of Graph Transformations with Industrial Relevance (AGTIVE'03). LNCS 3062, Springer (2004) 446 – 456
15. Lambers, L., Ehrig, H., Orejas, F.: Efficient conflict detection in graph transformation systems by essential critical pairs. In: Proc. Workshop GTVMT. (2006)